ENERGY EFFICIENT CLUSTER HEAD SELECTION AND TDMA TIME SLOT ASSIGNMENT FOR WIRELESS SENSOR NETWORK

REJWANA SULTANA
(*B.Sc. Engg., NSTU*)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE IN
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY

2021

APPROVAL OF BOARD OF EXAMINERS

The thesis titled "ENERGY EFFICIENT CLUSTER HEAD SELECTION AND TDMA TIME SLOT ASSIGNMENT FOR WIRELESS SENSOR NETWORK", submitted by Rejwana Sultana, Roll No: 1014140004, Session: October-2014, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on 15 March 2021.

BOARD OF EXAMINERS

1. ----------
   Name: Dr Md. Mahbubur Rahman — Chairman (Supervisor)
   Designation: Professor
   Affiliation: Department of CSE, MIST, Dhaka

2. ----------
   Name: Brig Gen A B M Humayun Kabir, ndc, psc, te — Member (Ex- officio)
   Designation: Senior Instructor and Head of the Department
   Affiliation: Department of CSE, MIST, Dhaka

3. ----------
   Name: Lt Col Muhammad Nuzrul Islam, PhD — Member
   Designation: Instructor Class-A (Associate Professor)
   Affiliation: Department of CSE, MIST, Dhaka

4. ----------
   Name: Dr. Mohammed Shafiul Alam Khan — Member (External)
   Designation: Associate Professor
   Affiliation: Institute of Information Technology (IIT)
   University of Dhaka, Dhaka.

DECLARATION

It is hereby declared that this is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. The thesis (fully or partially) has not been submitted for any degree or diploma in any university or institute previously.

_____

Rejwana Sultana

Department of Computer Science and Engineering
Military Institute of Science and Technology
15 March 2021

ABSTRACT

The Wireless Sensor Network (WSN) is found to be a promising technology in the 21th century. WSN mainly comprises several small sensor nodes. Usually, these nodes function on limited battery power. Energy competence is, therefore, an important research topic in the WSN. Maximizing WSN's lifetime, a proper cluster head (CH) selection strategy is an essential research topic. Also, to increase the throughput, data transfer rate should be higher. To achieve this goal, an efficient data transfer mechanism is required. Clustering has become a common approach to conserve energy for the WSN. The lifespan of the network depends on the number of alive nodes which eventually increase the throughput of the network. Many studies have been done on clustering because clustering protocols can increase the network lifetime and throughput. In these study, the clustering based on different parameters, such as the position of CHs, the transmission power, the location of the nodes, the distance between nodes, number of neighbor nodes, the residual energy of the nodes, the one-hop neighbor information, etc. are examined to improve the effectiveness of clustering in WSN. Finally, five parameters such as neighbor node number, the distance between CH and BS, one-hop neighbor data, remaining energy, and sensing area of a sensor are selected to reduce the possibility of unnecessary CH selection, and adjusting the distance between two CHs along with their sensing area thereby minimizing the energy wastage in WSN. In addition to that, a dynamic Time Division Multiple Access (TDMA) time slot allocations based communication scheme is also proposed. In this scheme, the sensor device forwards the collected information to the CH using the TDMA schedule. In this intra-cluster communication, nodes with data to be sent have received a time frame from the BS, and other nodes remain silent. No broadcasting and requisitioning

procedures are required in this system, which further reduces the node's energy consumption.

In this research, the proposed protocol was tested and compared with the state-of-the-art protocols. The simulation was done by the OMNeT++ simulation tool. The proposed scheme depicts significant enhancement in terms of network lifespan and energy consumption. Simulation outcomes for efficient CH selection are compared with LEACH-C and EEBCA. For dynamic TDMA time slot assignment, simulation results are compared with LEACH-C and TAICC. The proposed protocol has significant improvement in network lifetime and it outperforms in terms of cumulative data packets sent, data frame sent versus rounds, and total data packet sent.

ACKNOWLEDGEMENT

All appreciations are for the Almighty Allah for making me such eligible to take the effort and complete this research work.

I would like to express my sincere appreciation to the admirable supervisor of the research, *Dr. Md. Mahbubur Rahman, Professor, Department of Computer Science and Engineering (CSE), Military Institute of Science and Technology (MIST)*, always supported me and gave the right guidance for my research to keep it going. Nothing can be compared to his in-depth suggestions and the independence he gave me to complete this research. I am very thankful to him for his supervision and assistance in completing my thesis work. Furthermore, I would like to express my sincere thanks to *Md. Nurul Islam Khan, M.Sc Student, BUET,* who helped me in simulating the proposed model.

I would like to give my heartfelt gratitude to *Brig Gen A B M Humayun Kabir, ndc, psc, te*, *Senior Instructor, Head of Department of Science and Engineering (CSE), Military Institute of Science and Technology (MIST)* for his valuable suggestions and guidelines, which inspire me a lot to complete these thesis. I would to like to commend all the board members of Examiners for their valuable time in recognizing my work and their precious observations. I want to thank all my friends and coworkers for their inspiration and advice. I am also very grateful and thankful to my beloved family because without their trust and confidence and support, I could not complete the work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS LIST

WSN	Wireless Sensor Network
CH	Cluster Head
ADC	Analog to Digital Converter
FND	First Node Death
TND	Tenth Node Death
LND	Last Node Death
ON_List	Ordinary Node List
TDMA	Time Division Multiple Access
MAC	Media Access Control
CSMA	Carrier Sense Multiple Access
BS	Base Station
LEACH	Low Energy Adaptive Clustering Hierarchy
LEACH-C	Low Energy Adaptive Clustering Hierarchy – Centralized
M-LEACH	Mobility Low Energy Adaptive Clustering Hierarchy
DB-LEACH	Distance Based LEACH
DBEA-LEACH	Distance Based Energy Aware LEACH
EECS	Energy Efficient Clustering Scheme
EESCA	Energy Efficient Structured Clustering Algorithm
I-LEACH	Improved LEACH
ECHSSPF	Efficient Cluster Head Selection Strategy for Provisioning Fairness
EEBCA	Energy Efficient Balanced Clustering Approach
CBICCT	CSMA Based Intra Cluster Communication Technique
TAICC	Traffic-Aware Intra Cluster Communication

SN	Sensor Node
CN	Cluster Node
MN	Member Node
WV	Weight Value

CHAPTER 1: INTRODUCTION

The Wireless sensor network (WSN) belongs to a collection of designated sensors for observing and collecting the physical conditions of the environment and organizing the data at a central location name BS. Sensor devices notice environmental conditions such as temperature, sound, levels of radiation, moisture, wind, waves, heat, sunlight, and so on. In WSN, sensor devices are used to record, monitor, and control various environmental conditions. WSN has become a demanding network technology due to its application in various wildlife sensing, detecting, surveillance, and monitoring environment [1], [2]. Within the network, the deployment of sensor nodes is random. Micro-electro-mechanical is the essence of the sensor. The size of the sensor node is tiny, and it runs on a battery. So when the sensor node runs out the energy, it will die. The node's role is to detect, capture, accumulate, and transmit the data from the environment to the BS via the radio connection. So, if the sensor node energy is finished, the network lifetime will finish. But ample resources like processing power, required energy, and storage are available in the BS. BS gathers the information and then transmits information for further processing to the real-world. WSN is commonly used in many applications, such as military reconnaissance, vehicular movements, the timing of volcanic earthquakes, weather forecasting, environment monitoring such as temperature, humidity, pressure, motion, etc. [3]–[6]. Sensor nodes in WSN are usually deployed in remote and hazardous areas [7]. So, if the sensor node's battery power is finished, it is difficult to replace, and even if all nodes are eventually dead, it leads to finish the lifetime of the network. The WSN's lifespan is therefore dependent upon this capacity of the sensor node [8]. These sensor devices are generally fitted with a radio transceiver, a microcontroller, a memory unit, and a set of transducers. For acquisition and processing the data from the deployed region, energy is the primary constraint of WSN. As sensor device operates with low power, therefore,

WSN's lifespan relies primarily on the effective use of its power [9]. Detecting physical or environmental conditions by independent sensor nodes and sending the collected data to the end-user, usually referred to as the data collection network [10]. The basic functionality of the wireless networks is shown in Figure 1.1.

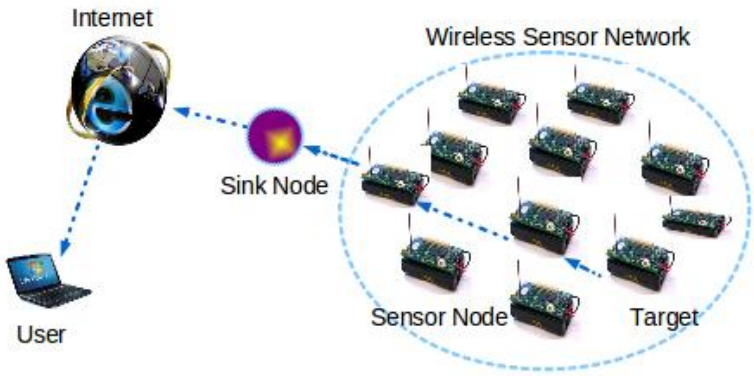


Figure 1.1: General structure of a wireless sensor network [11]

1.1 Challenges in Wireless Sensor Networks

In recent years, applications of WSN are increased a lot, but WSNs is becoming more challenging because of it consists of sensor nodes. WSN's sensor nodes are powered by a small battery in it. So owing to the limitation of resources such as the small battery, bandwidth, and processing capacity, there are some challenges faced in WSN. Few challenges are listed below [12], [13]:

- Energy efficiency
- Coverage
- Security
- Data aggregation

- Minimum delay
- Throughput
- Scalability

Energy Efficiency: Energy is WSN's primary concern since WSN's lifetime is entirely dependent on tiny sensor devices. Energy conservation is essential for WSN to prolong the lifetime of nodes. Energy efficiency is, therefore, a crucial challenge. Various operations are carried out by sensor nodes and need power for various operations, such as environmental sensor data, cumulative data, and data transmission to the CH or BS [14], [15]. As a sensor node works with limited resources, efficient energy uses will increase the network lifetime. The power supplied by the battery to the sensor node and WSN's sensors are deployed in a remote area. Though sensor nodes are inexpensive, replacing or recharging batteries is mostly impossible because of sensor nodes' deployment in the wild or remote area. Therefore, the lifetime of both the sensor node and the network is dependent on the optimized use of sensor nodes' energy. One of the biggest problems is making effective use of the resource. Convenient selection of CH can solve this problem and extend the lifespan of the network. Recently, efficient power or energy consumption is the main area for the researchers.

Data Aggregation: Data aggregation is done by the sensor node, whether it is a regular node or CH. Sensor devices gather the information from the surroundings, then accumulate the information using different techniques or features such as suppression (trying to find and removing duplicates), minimum, maximum, and mean [16]. There are many nodes in the network, where different nodes may collect the same data. Normal nodes send the collected information to the CH, and CH accumulates the information and reduce the possibility of data duplication. Data aggregation also reduces total number of transmissions,

which leads to less energy consumption. Then sensed data is transmitted to the BS. Because of the data aggregation mechanism, the possibility of data duplication in the transmission can be minimized.

Security: In this era, all types of communication requires strong security. The security issue is one of the challenges in WSNs. WSN may gather sensitive information related to malicious intrusions and attacks. If there is no proper security, it may cause a danger to the network and the data. So, WSN requires proper security with constraint resources [16].

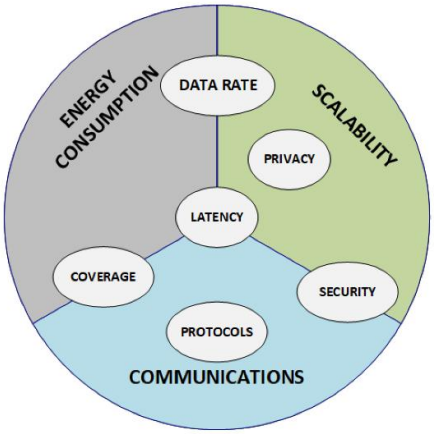


Figure 1.2: Overview of the challenges facing by WSN [17]

Throughput: In WSN, throughput is another factor that needs to be considered. Network throughput are related to both network life and energy consumption [16].

Minimum Delay: In all types of communication, the delay should be minimum to increase the throughput. It is one of the common challenges in all types of communication networks. Though data is collected from the surroundings, data must be sent to the CHs followed by the BS, so the delay should be minimal [18].

Scalability: Scalability means that if the number of nodes in the network is increased or the network size is modified, they will not affect the network's performance. The routing protocol in WSN needs to be scalable enough so that for increased or decreased nodes in the network will not affect the network's performance [18].

Coverage: Coverage in WSN defines how well or how much each point of a deployed network is under vigilance [19]. The sensing region for sensors needs to be considered in WSN. Coverage is another challenge for WSN because sensors' deployment is done randomly or in a self-organizing system. A suitable protocol needs to be used which covers the whole network. In WSN, the distance between the CHs, the distance between the CH and BS, must be considered. More energy will be consumed if the position of CHs is too far or too close.

1.2 Motivation

Nowadays, WSN plays a significant role in our day to day life. In every sector, we have found different applications of WSN. However, WSN consists of some small sensor nodes composed of memory, battery, transceiver, microcontroller, sensor, and ADC (Analog to Digital converter) [20]. These sensor nodes wholly depended on the battery for the power supply. So one of the most debated subjects in WSN is energy conservation or reducing energy usage. Many methods have been applied, many studies have been done to save energy [21]. One of the most excellent technique in WSN to reduce energy consumption by the sensor node is clustering. In grouping, the sensing region is split into a small area, called a cluster. In each cluster, one node is selected as a CH, acquiring information from other member nodes. CH gathers member node information, accumulates data, and then forwards the data to the BS. An effective clustering protocol can save substantial WSN

energy and extend the life of the network. Member nodes, other than BS, send data to CH. A large amount of energy is saved by this technique. So, in enhancing the network lifetime, the selection of CH plays a very crucial part. Often it creates a problem between the CH and BS due to distance. It leaves the cluster unbalanced if the distance is too close, and if the distance is too far, it takes more energy to transfer the data from CH to BS. Both cause more power consumption and decrease the lifespan of the network. Energy-efficient clustering in the WSN is also responsible for the size of the cluster. Other parameters like residual energy, number of neighbors, and one-hop neighbor information are also considered in many studies to reduce energy consumption. When these mentioned parameters are considered, and one hop information is used for selecting CH, there is a possibility of increasing the number of CH in the cluster. For that reason, energy consumption is increased by the sensor nodes, which reduces the network lifetime.

In WSN, much of the energy is also spent for communication purposes, i.e., primarily for data transmission and reception. In the steady-state phase, data transmission is occurred by intra-cluster communication. In many studies, it is found that data transmission also has an impact on energy efficiency. Different methods have been considered to maximize network lifetime. In some work, CSMA and TDMA both have been considered, sometimes different phases have been added, sometimes slot reservation technique has been considered to minimize the wastage of time slot and increase the throughput. Nevertheless, for slot reservation, lots of messages need to be broadcast. Because of the broadcasting message, it consumes more energy. So, to save the sensor node's energy, an appropriate intra-cluster communication mechanism requires to be suggested. Several problems that inspired this study to be carried out have been observed.

The above topic concerns the efficient selection of CHs to minimize energy usage and intra-cluster contact, resulting in lower consumption of energy in WSN. There are

several things to be considered, such as optimum CH selection, the distance from CH to CH, and an intra-cluster communication mechanism to develop a cluster-based algorithm. Therefore, we were inspired to establish an effective clustering strategy by agreeing with the aforementioned difficult problems.

1.3 Research Objectives

The objective of the research is to improve WSN's network lifespan and cluster-based routing performance. The following targets have been defined for mitigating the objectives:

- To create a new algorithm for CH selection to maximize the sensor node energy in WSN.
- To devise an efficient TDMA time slot assigning technique for intra-cluster communication to reduce energy loss.
- To analyze the proposed approach's efficiency by comparing it with the other existing methods correlated with it.

1.4 Organization of Thesis

The subsequent parts of the thesis are structured as follows:

Chapter 2 reviews some studies about CH selection and TDMA time slot assignment. Different methods have been mentioned in this part. Side by side, the problems of the existing methods have also been mentioned.

Chapter 3 describes about the proposed protocol. A new approach to CH selection and an intra-cluster communication (ICC) method is also explained.

Chapter 4 provides the performance analysis of the suggested clustering protocol through simulation results is discussed in this chapter. In terms of network lifetime, CH number, and throughput, better performance is analyzed and compared with existing protocols.

Chapter 6 summarizes the main idea and outcomes, also provides some directions for future work and limitations of the research work.

1.5 Summary

In this chapter we have discuss about the overview of WSN, challenges of WSN, motivation behind the study, research objectives and organization of the thesis.

CHAPTER 2: LITERATURE REVIEW

This chapter firstly discuss about the background of WSN and the related work of WSN. At the end of this chapter, a critical summary is presented to highlight the research gap and motivation of this research work.

2.1 Wireless Sensor Network

WSN is infrastructure less network which is used for monitoring physical and environmental conditions such as temperature, motion, vibration, sound, pressure, pollutants etc. and gathering data from environment. WSN comprises bunches of small, low controlled sensor nodes. Ordinarily, sensor nodes are sent in unsafe and dangerous territory. WSN mostly focuses on efficient energy use for maximizing network lifespan, and it is the most critical challenge [22]–[24]. Proficient energy utilization causes the least energy utilization, which prompts augmenting the network lifespan [25]. For efficient energy consumption, clustering plays a vital role in WSN. Efficient energy consumption increases the network lifetime as well as throughput. In WSN, intra cluster communication is also important because nodes in the cluster have to communicate with each other and time slot is required for this intra-cluster communication.

2.1.1 Clustering in WSN

In WSN, the sensing region is split into a small area-called a cluster to maximize the use of energy by the sensor node [26]. Grouping is one of the noticeable strategies for saving energy [27]. Effective CH choosing is one of the solutions for minimizing the use of node energy in clustering. Two types of clustering methods are in WSN- centralized and distributive. In centralized clustering, all tasks such as- cluster formation, CH selection are

done by a central node called sink node or BS. In distributive clustering, there is no central node or central control.

2.1.2 Cluster Head Selection

In the cluster there are several nodes. From these nodes few nodes are selected as CH and remaining nodes are member nodes. Member nodes gather data from the environment and transfer the data to the CH. Clustering and CH selection plays a very important role to minimize the energy wastage. So selection of appropriate CH is also very important. There are two approaches to select CH- probabilistic and deterministic.

2.1.3 Related Work on CH Selection

The main goals of WSN is to enhance of lifetime and throughput of the network. Proper CH selection helps to consume less energy, minimize energy usage. There are several CH selection techniques, some are select CH in probabilistic and some are select CH in deterministic manner. For CH selection, LEACH is one of the pioneer techniques. In [28], they suggested a new LEACH method, where CH is selected in a random probabilistic manner. As a result, low energy sensor nodes (SN) might be elected as CH, which can be dying out soon. So, the network lifetime may be decreased. In [29], authors have proposed LEACH-MAC an approach to control the LEACH protocol's randomness and restrict the CH advertisement. Authors of [30] introduce a feature in LEACH for mobile sensor nodes called Mobility-LEACH (M-LEACH), which reduces the network's resource consumption. However, LEACH usually supports static nodes. The CH-Leach protocol is proposed in [31], which uses the k-Means method to ensure a balanced distribution of energy across the entire network region. CHs is randomly selected, and those nodes are close to the centroid; they will become CH. Later on, LEACH-C protocol suggested in [32], where cluster

creation and chosen CH are all performed by BS in LEACH, CHs are chosen randomly by themselves, BS has centrally selected CHs, taking into account the residual energy of the nodes and the current node position. All nodes submit their position and available energy at the starting of each round to the BS. BS then determines all nodes' average capacity. In LEACH-C, when the node's energy is greater than that of the recently chosen CH node, this node is chosen as CH in the next round. CH has been chosen in a probabilistic manner in this article. Several studies focused on the CH selection based on only distance. If the only distance is considered a parameter for CH selection, then energy consumption is not solved. If the only distance is taken into account, it is possible to pick those nodes with less remaining energy as a CH so that CH will die very soon. If CH will die soon, it triggers the weakening of the network throughput. Two energy effective CH chosen strategies based on distance were proposed by the authors of [33]. They also consider the selection of CH based on higher energy. They proposed two clustering routing protocols named Distance Based (DB-LEACH) and Distance-Based Energy Aware (DBEA-LEACH). In DB-LEACH, the geometric distance between the candidate node and BS was taken into account, and DBEA-LEACH considered the distance and the residual energy of the node. Some other research works, CH selected based on remaining energy. More remaining energy holder nodes can be selected as CH, but though the distance is not considered a result if CH is far from BS, it consumes more energy. In [34], the authors proposed an algorithm for selecting effective CH, which adapts clusters and rotated CH positions to ensure that the energy load is evenly distributed among all nodes. In their proposed scheme, the probability of becoming CH was modified based on the residual energy. Authors of [35] proposed an Energy Efficient Clustering Scheme (EECS) where more residual energy contained node selected as a CH. They also introduced a method by which load among the CH can be distributed. It can balance the consumption of energy among CHs, but not for

the entire network. Authors of [36] suggested an Energy Efficient Structured Clustering Algorithm (EESCA) where hybrid CH selection has used. In this method, two parameters have been considered: location centrality and the node's lasting energy. In [37], authors have proposed Energy Efficient Balanced Clustering Approach (EEBCA) where two methods- one for CH selection and another for intra-cluster communication have considered. Using residual energy and neighbor node number, CHs are chosen in this paper. In [38] Improved LEACH (I-LEACH),CH is selected based on residual energy, neighbor node number, and distance data. It provides an energy-effective cluster but fails to improve the intended node balance. The authors of [39] Efficient CH Selection Strategy for Provisioning Fairness (ECHSSPF) used residual energy, neighbor node number, and one-hop neighbor communication. In this paper, the distance between CH and the BS is not studied. So, if a node is taken as a CH and away from the BS, it tends to cause extended-distance communication, leading to more power consumption. In [40], the authors considered the remaining energy, the neighbor node number, the one-hop neighbor communication, and the distance between CH and BS, these four parameters used to select CH. However, it suffers from many CH selections because here, one of neighbor information is used, and CH to CH distance has not been considered. Consider that a network is divided into some clusters, and using the four parameters-remaining energy, neighbor node number, one-hop neighbor information, and the distance between CH and the BS-a node from one of the clusters has been chosen as CH. Though it considers the one-hop neighbor information, the one-hop node will not be the CH candidate; the next node beside the one-hop node can be the CH candidate. In that way, there is a probability to select lots of CH in the network, which causes waste of energy. In view of these circumstances, an approach has been proposed where the distance between CH and CH is considered along with the four parameters-remaining energy, number of the neighbor node,

one-hop neighbor information, and distance between CH and the BS. Let us assume that the network is split into several clusters, and here the CH has one-hop communication details. Consider cluster node (CN1) is the one-hop neighbor of CH, so CN1 will not be selected as CH. However, CN2 and CN3 is not one-hop neighbor for CH. CH has no information about CN2 and CN3. That is why CN2 and CN3 will be the candidate node for CH, or they may be selected as CH in the future. So if one-hop neighbor is considered, then the number of CH will be increased in the cluster, which leads to the wastage of energy.

2.1 A brief overview of related works of CH selection

References	Method Name	CH selection method	Drawback of the method
[28]	LEACH	Select CH in a probabilistic manner	Low energy sensor node might be selected as CH
[29]	LEACH-MAC	Restrict CH advertisement	Does not distribute CHs evenly.
[30]	M-LEACH	Mobility of CHs and member nodes (MN)	If MN moves away from current CH then it causes to spend more energy
[31]	CH-LEACH	Nodes that are close to centroid select as a CH	It causes unequal cluster size, uneven load distribution in the network
[32]	LEACH-C	CH are selected centrally by BS, nodes with higher energy are selected as CH	Only energy of the node is considered

2.1 A brief overview of related works (cont.)

References	Method Name	CH selection method	Drawback of the method
[33]	DB-LEACH and DBEA-LEACH	Node that has less distance (between the BS and node) and higher remaining energy is selected as CH	Distance and residual energy are considered
[34]	Modified LEACH's stochastic CH selection algorithm	CH selected in probabilistic manner	Only residual energy is considered
[35]	EECS	Nodes have more residual energy selected as CH and based on probability each node becomes a candidate for CH	It produces more control overhead complexity because all nodes have to compete with each other for becoming CHs and Only remaining energy is used
[36]	EESCA	Nodes have less average communication distance and remaining energy lesser than 50% of the initial energy selected as CH	Hybrid CH selection using distance and remaining energy

2.1 A brief overview of related works (cont.)

References	Method Name	CH selection method	Drawback of the method
[37]	EEBCA	CH selection considering number of neighbor nodes, and remaining energy are used	Distance is not considered, if CH has long distance from the BS then it causes more energy consumption
[38]	I-LEACH	Residual energy, number of neighbors and distance between CH and BS are considered for CH selection	It creates non-uniform distribution of CH which consumed more energy in the network
[39]	ECHSSPF	Residual energy, number of neighbors and one-hop neighbor information are considered for CH selection	Due to back transmission it cannot create balanced clustering
[40]	CH selection using four parameters	Residual energy, number of neighbors, one-hop neighbor information and distance between CH and BS are considered for CH selection	Distance between CHs is not considered and for one-hop neighbor information number of CH in the cluster is increased

2.1.4 Intra-cluster Communication

Communication between the MNs within the cluster, known as intra-cluster communication. Almost all of the clustering protocols consist of several rounds. Each round work into two stages: the setup and the steady-state stage [41]. The network is divided into a small area called a cluster during the setup stage, then pick a CH [42]. Before

starting the steady-state phase, all nodes in the cluster get the TDMA schedule from BS. In the steady-state stage, MNs in each cluster send their observed data to their corresponding CH in a fixed TDMA slot.

2.1.5 Related Works on Intra-Cluster Communication

Traditionally in TDMA, at first, all nodes assign the time slot. It causes wastage of time slots. Different studies have been done to minimize the wastage of time slots. The authors of [43], proposed a CSMA based technique, where the preamble sampling technique is used. However, it only gives a better result for low traffic. Here, a CSMA based intra-cluster technique where member node initiative the sending process. SNS use the lengthy preamble to pay attention to CH if they want to send data. The method saves power utilization by the CH but diminishes the throughput. Moreover, this method raises latency. When all nodes in a cluster have data to send, TDMA offers the best outcome. However, all SNs inside the cluster does not have data to send in actual circumstances [44]. An efficient CH selection and an intra-cluster communication mechanism for various traffic patterns are proposed [37]. A classifier is used in this paper to fix the window size according to the traffic pattern dynamically. In [45], the authors proposed a hybrid method combining TDMA and CSMA for mobile nodes. CSMA is used for handling control messages, and the TDMA technique is used for data transfer. Synchronous, requisition, and schedule broadcast periods are considered for reserving the slot prior to the start of the data transmission stage. To minimize the wastage of unwanted slots, they use the slot requisition method before data transmission. Therefore, to know the requirements of the member nodes, CHs need to broadcast a message. However, it has some problems like control packet overhead, a collision in the requisition phase, and delay due to the requisition phase. Also, the broadcasting message causes energy loss. A traffic haul-based intra-cluster

communication technique was suggested in [46] to solve the energy loss problem, where CSMA used for slot reservations and TDMA used for data transmission. In the proposed scheme, the frame consists of four phases, where the first phase is used for the TDMA mechanism, and the remaining three phases are used for the CSMA mechanism. The slot requisition phase is also used in this protocol, so it needs message broadcasting. Nevertheless, broadcasting the message for slot reservation leads to energy consumption. Member nodes get the time slot those who have data to send, but in the middle of the round, if any data comes to the time slot non-owner node has to wait for the next round. An intra-cluster communication mechanism has been proposed considering these shortcomings, which will dynamically assign the TDMA time slots to the sensor nodes. So no time is wasted, and the energy consumption is reduced because no broadcasting is there, increasing the throughput of the network.

2.2 A brief overview of related works on time slot assignment

References	Method Name	Intra-cluster communication method	Drawback of the method
[43]	CBICCT	Use preamble for sensing channel	Gives better result for low traffic
[45]	Hybrid MAC protocol	CSMA is used for handling control message and TDMA is used for data transfer	Due to slot requisition, it causes energy loss.
[46]	TAICC	CSMA is used for slot requisition and TDMA is used for data transfer	Broadcasting message for slot requisition causes energy consumption

2.2 Summary

In this chapter we have reviewed several studies from which we got the information about WSN, clustering, different CH selection approaches and intra-cluster communication. In the table 2.1 and 2.2, different CH selection methods are mentioned. Limitations of the methods are also enlighten.

CHAPTER 3: PROPOSED METHODOLOGY

To improve the efficiency, throughput, as well as the lifetime of WSN, is the prime concern of the proposed methodology. In the proposed protocol, an efficient CH selection and also an intra-cluster communication strategy has been approached. Proper selection of cluster head can help to decrease the number of CH selection possibility in the cluster. Besides, successful intra-cluster communication will increase the network's throughput and lower the node's energy consumption. It also enhances the longevity of the network since the broadcasting of reservation message is not required. The network's performance is improved using a dynamic TDMA time slot assignment technique. This section focuses on the fundamental assumptions and the radio model used in the protocol and offers an overview of the protocol proposed.

3.1 Network Model and Assumptions

In the network model, nodes are deployed randomly. The network designed for 75 sensor nodes. Nodes are homogeneous in nature. Each node has unique ID and this sensor nodes collect data then send it to the BS. Proposed protocol is centralized protocol so cluster formation, CH selection and time slot assignment done by BS. For designing the wireless sensor network some assumptions are considered in this protocol:

- The network is homogeneous.
- All nodes are organized haphazardly over the network.
- All nodes consist of equal abilities for communication and equal initial energy.
- With adequate resources, the BS is located outside the network.
- The proposed protocol is centralized.

3.2 Radio Model

To evaluate energy consumption, a radio model is used [40]. This model is the model of the first order that is seen in Fig. 3.1.

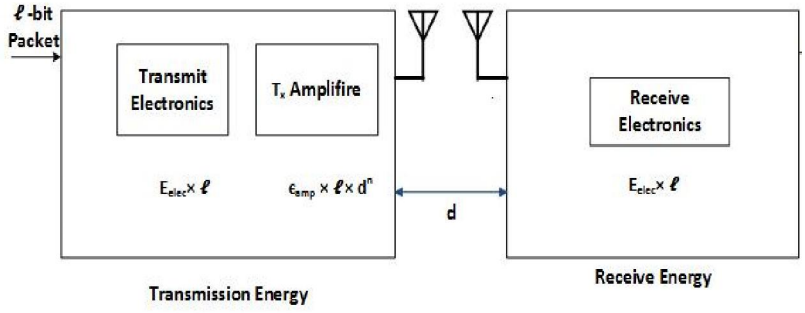


Figure 3.1: Network energy model

Let distance is d, l is the number of bits then required energy :

$$E_{TX(l,d)} = \begin{cases} E_{elec} \times l + \epsilon f_s \times l \times d^2 , for\ d < d_0 \\ E_{elec} \times l + \epsilon_{mp} \times l \times d^4 , for\ d \geq d_0 \end{cases} \quad (3.1)$$

Here, E_{elec} is the energy required for one bit processing, ϵf_s and ϵ_{mp} is the energy required in free space and multipath model for transmitting one bit to d_0 with an allowable bit error rate. The threshold value is d_0, where:

$$d_0 = \sqrt{\frac{\epsilon f_s}{\epsilon_{mp}}} \quad (3.2)$$

$$E_{RX} = E_{elec} \times l \quad (3.3)$$

Where E_{RX} is the energy dissipated per bit at receiver.

3.3 Protocol Description

Proposed protocol designed into two parts: efficient CH selection and an intra-cluster communication mechanism. The clustering protocol operates in several rounds, and each round is divided into two phases: i) the setup phase and ii) the steady-state phase.

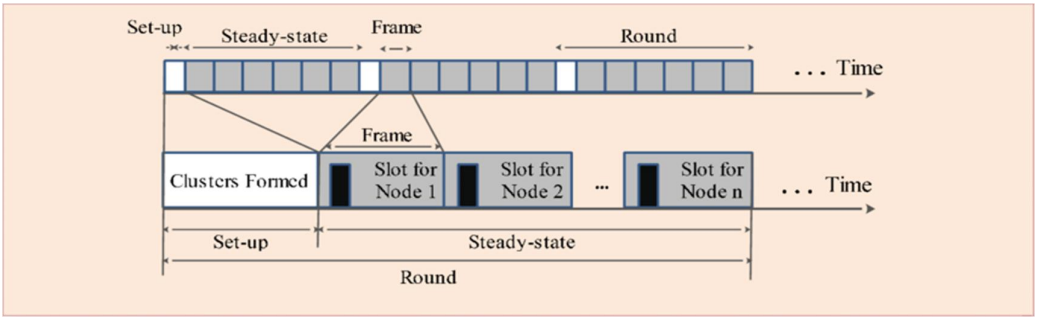


Figure 3.2: Two phases operation in a round of LEACH [47]

Selection of CH, cluster formation, and TDMA time slot distribution are performed in the setup phase. During the steady-state phase, the sensed data from the member node to CH and CH to BS are transmitted using the TDMA schedule.

3.3.1 Setup Phase

The network is split into several areas, called clusters, in the setup phase. Cluster creation can generally be categorized into centralized and distributed techniques. In centralized clustering, the control message for clustering computing is received from the central base station (BS) on the basis of information accumulated from all the sensors in the network [28]. Decisions relating to the creation of clusters are taken in the distributed approach by the sensor nodes without using a central entity (or BS) [48]. A centralized approach considered in the research work, where BS has ample resources for any computation.

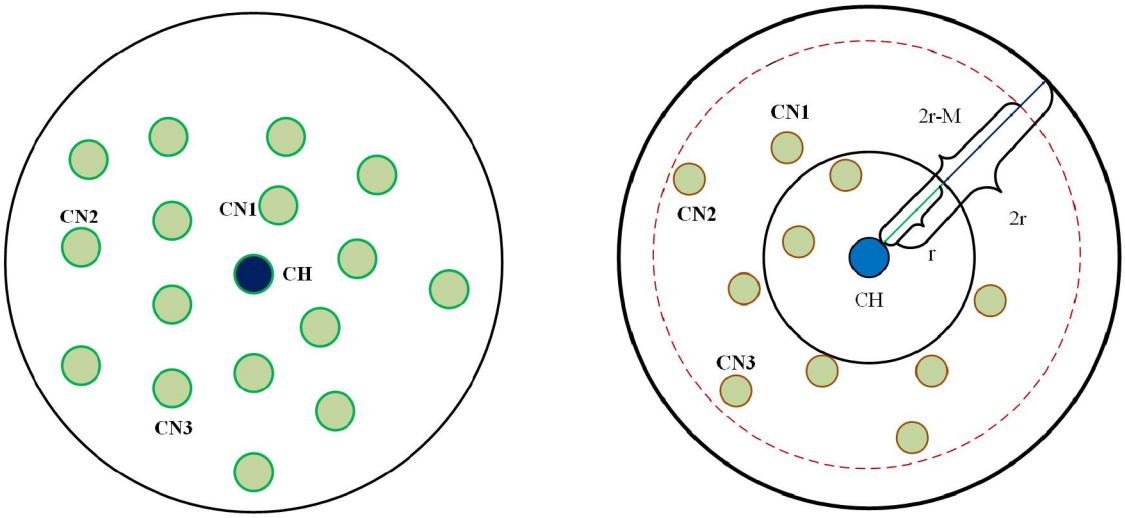
3.3.1.1 Cluster Head Selection Procedure

A centralized protocol is proposed protocol, where BS is responsible for selecting a CH according to the algorithm. A network consists of randomly deployed N sensor nodes, lead BS to determine first the radius for each node's neighbors and the optimum number of network clusters according to equations 3.4 and 3.5, [40], respectively. BS can also measure the neighbor set for each node in radius R.

$$R = \sqrt{\frac{M \times M}{\pi \times k}} \tag{3.4}$$

$$k = \frac{\sqrt{N}}{\sqrt{2 \times \pi}} \sqrt{\frac{\epsilon f_s}{\epsilon_{mp}}} \frac{M}{d^2{}_{toBS}} \tag{3.5}$$

Here, M×M is the network area, k is the optimum number of clusters, N is total nodes, d^2_{toBS} is the average distance between a nodes to BS.



(a) (b)

Figure 3.3: (a) Unnecessary CH selection and (b) Avoiding unnecessary CH selection

Studying different works it was found that different parameters were considered for efficient energy consumption. Some parameters such as number of neighbour nodes, residual energy, distance between CH and BS, one-hop neighbour information were considered in different studies. Some studies consider only distance, some consider only residual energy, some consider distance and remaining energy etc. From literature review different studies considering the mentioned parameters have some drawbacks. To solve this drawback five parameters were considered in the proposed algorithm that include the residual energy, the adjacent node number, the distance between CHs and BS, the one-hop neighbor info, and the CH to CH distance for CH selection. Again, since one-hop neighbor information has been considered, so the closest node of the one-hop neighbor of a CH has a chance to be CH. As a result, the number of CH will be increased, which causes more energy consumption. To solve this problem, 2r-M distance considered from a CH. If no neighbors in the 2r-M distance are CH, then add this node to the CH list. Each node will send its position to the BS with the residual energy. The BS will then determine the length of all the nodes on its own, the neighbor set, and the number of neighbors (NN) of each node within the cluster. BS will calculate the weight value (WV) for all nodes confer to the equation (3.6).

$$WV = RE + \alpha \times NN + \frac{\beta}{distance} \tag{3.6}$$

Here, alpha and beta represent a customized variable that fine-tunes the number of neighbors with remaining energy within R and all nodes' length. The BS will compute each node's WV using the equation (3.6) and prepare an ordinary node list (ON_List) considering all nodes as a normal node. If a node positioned very close to a one-hop neighbor of a CH is chosen, it will be treated as a common node, and the node will be

placed in the ON-List. Here the distance between CH to CH considered as 2r-M. Where r is a sensor's sensing area. M is a variable and the value of M can be adjusted according to the needs. Then BS will generate a CH list (CH_List) using distance 2r-M. If any node within this distance is not selected as CH, then it will add to the CH_List. In the proposed protocol, it is suggested that if any member nodes within the 2r-M are not CH, it will be a member of CH_List and consider as a CH candidate. Considering the distance of 2r-M, the number of CH has reduced. Consider figure 4.3 (b), here CN1 is not in one hop distance from the CH but closer to CH. According to our proposed algorithm, CN1 will not select as a CH. CN2 and CN3 may be select as CH.

3.3.1.2 Proposed Algorithm

The algorithm for CH selection is stated in Algorithm-[1], while the steps are graphically presented in figure 3.4.

Algorithm-1: CH selection algorithm

Input: Residual energy and position of all nodes
Output: CH_List
All nodes in the cluster considered as Vertex (V) and each cluster as connected Graph (G)

```
ON_List = ∀V            // Initially all nodes are in ON_List
CH_List= ∅
for ∀V ∈ G do
        Calculate R by using Equation 3.4
        Update ON_List
end for

CH_List= ∅                      // Initially set of CH_List is empty
for (each vertex v_i ∈ V) do
        calculate its weight WV_vi by using E.q. 3.6
end for
SORT (WV_vi)            //sort the weighted value in descending order
for i=1 to N do
        if (v_i ¬ ∈ (2r-M )) then
                CHs ← v_i ∪ CHs
        else
                v_i is a normal node
```

17: end if
18: end for

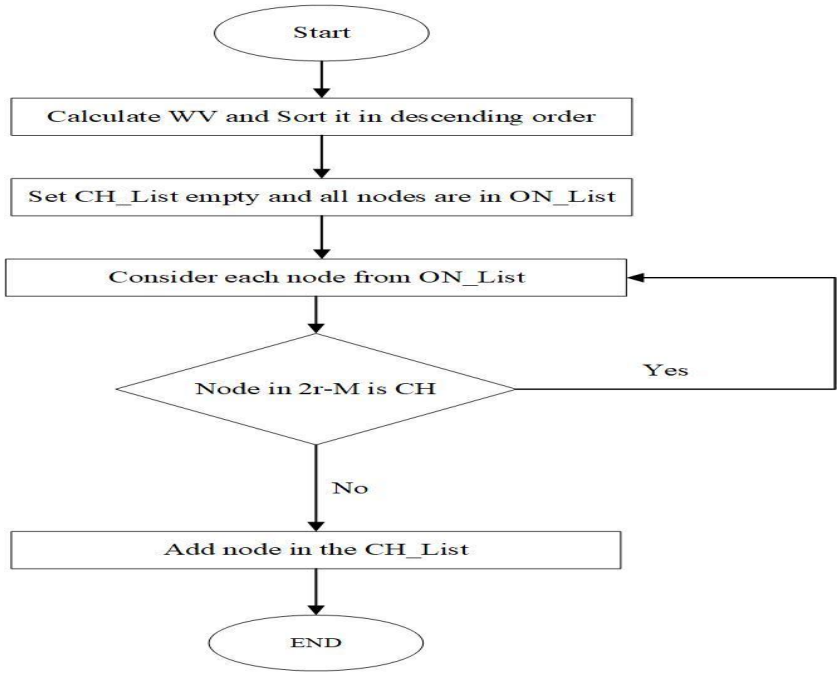


Figure 3.4: Proposed CH selection process

3.3.2 Steady-State Phase

Sensed data is transmitted from SN to CH and CH to BS using the allocated time slot in the steady-state phase. In the steady-state process, data transmission was carried out. In the literature review section, it is observed that lots of studies have been came out which are related to improvement of timeslot utilization technique and for changing the format of a TDMA frame. All of these studies have concentrated on improving the WSN network's throughput and lifespan. A time slot assignment strategy has been proposed where a beacon is used to minimize the waste of time slots. Different types of methods had been developed to minimize the wastage of time and wastage of energy. In [46], the slot reservations method is proposed, and broadcasting the slot reservations message leads to energy

consumption. Member nodes get the time slot those who have data to send, but in the middle of the round, if any data comes to the time slot non-owner node have to wait for the next round. An intra-cluster communication mechanism has been proposed where the concept of the beacon bit is used. In the proposed protocol, a hybrid intra-cluster communication is considered. At first, the network is divide into several clusters like LEACH-C [28]. In the clustering approach, lots of clustering protocols have been studied. Each protocol for clustering comprises of several rounds. A setup and a steady-state stage compose of each round. Clusters are created in the setup stage, and CHs are chosen, followed by the stage of data transmission. In the steady-state stage, at the beginning of the 1st round, member nodes that have data to send getting the time slot in the TDMA allocation schedule from the BS. Those member nodes have no data; they do not get the time slot. There is no broadcasting process; time slots are allocated by BS at the beginning of the round. In the middle of the round, if any data comes to the time slot non-owner node (a node that currently has no time slot), it sends the beacon with its ID to the nearest time slot owner node. Then time slot owner node combines the beacon and ID with its data and sends this packet to the BS. BS at first split the beacon and ID from the data. Then BS sends the time slot directly to that node using the ID of that node. The algorithm for Time slot assignment is stated in Algorithm-[2], while the steps are graphically presented in figure 3.5.

Algorithm-2: Time Slot Assignment

Input: Node transfer data using the allocated time slot by BS
Output: Time slot non-owner node get the time slot

```
All nodes know the TDMA of every node in a cluster
 if (node has data) then
        sends data to the CH
        if (node has no timeslot) then
                sends ID & beacon to the nearest time slot owner node
                if (timeslot owner node get ID & beacon) then
                        sends data to BS
                        BS get the ID and sends timeslot to that node directly.
```

```
9:                          end if
10:                 end if
11:     else
12:                 node remain silent
13:     end if
```

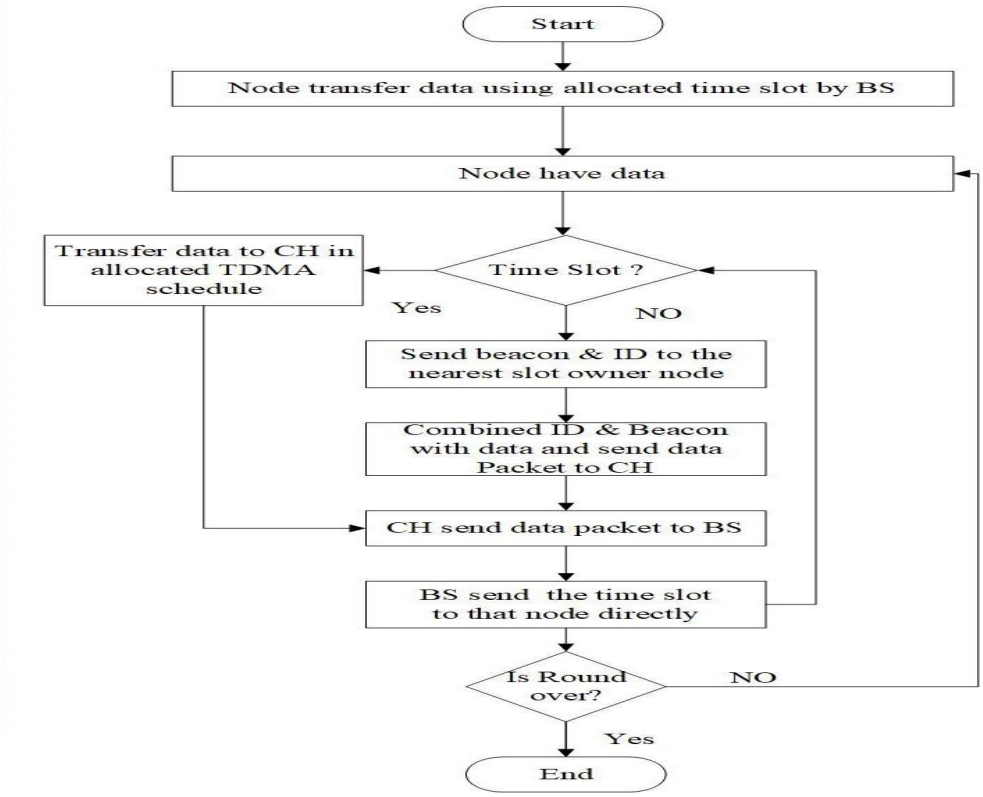


Figure 3.5: Proposed Time Slot Assignment Process

3.4 Summary

For wireless sensor networks in this chapter, a CH selection and an intra-cluster communication technique were suggested. A CH picked and an intra-cluster communication technique is implemented in this research. In this protocol, the BS decides to pick the cluster head based on residual energy, the number of adjacent nodes, one hop CH knowledge, the distance between BS and CHs, and the distance between CH and CH. One intra-cluster communication mechanism is suggested to minimize time slot wastage, reduce energy consumption, and increase throughput.

CHAPTER 4: RESULTS AND PERFORMANCE EVALUATION

4.1 Introduction

The suggested scheme simulated and evaluated for performance reasoning using the OMNeT++ simulation tool. The simulation examined considering different parameters (see Table 5.1) and deployed a total of 75 nodes. Each node was assigned preliminary energy of 0.5J, and the total energy was 35J. Simulation outcomes are compared with LEACH-C [32] and EEBCA [37] and concerning network lifespan. The simulation result illustrates that the proposed protocol has compelling improvement in a network lifetime.

Table 4.1: Network Parameters

Parameter	Value
Network Area	200 × 200 meter^2
Number of Nodes	75
Data Packet	512 bytes
Initial Energy	0.5 J
Network Primary Energy	35 J
Electronics energy(E_{elec})	50 nJ/bit
Free space loss (ϵf_s)	10 pJ/bit/m^2
Multipath loss (ϵ_{mp})	0.13 bit/m^2

4.2 Simulation Environment

The simulation came out using the OMNeT++ simulation tool. Few metrics have been considered to measure the effectiveness of the new scheme. Remaining Energy; consuming

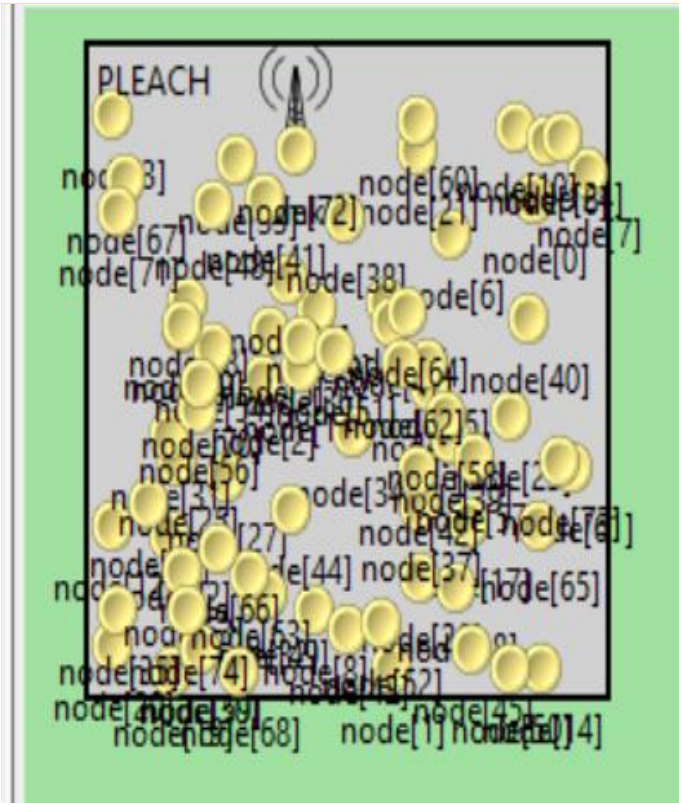


Figure 4.1: Simulation network implemented in OMNeT++ Environment

energy; dead nodes; no of CHs; First node dies (FND); Ten Node Die (TND) and Last Node Die (LND) concerning round; throughput; cumulative data packet send; Data frame send; data packet send and total data packet send – these were considered for evaluating the performance. It has been observed from performance evaluation that the proposed scheme provides better outcomes than the current process. The proposed network structure deploying 75 nodes is presented in figure 4.1. Nodes are randomly deployed here, and BS is positioned on the network's exterior/border. A unique ID is contained by every node.

4.3 Simulation of CH Selection with Various Metrics

For the proposed CH selection scheme, comparisons have come out with a pioneer clustering protocol LEACH-C and a latest energy efficient and balanced clustering work for improving throughput EEBCA. Proposed algorithm is centralized protocol like LEACH-C and EEBCA. Same network parameters have considered in this two works. After studying the literature review it was found that LEACH-C and EEBCA both works based on energy consumption considering different parameters. The results are showed and discussed elaborately to consider the different metrics as mentioned below:

4.3.1 Remaining Energy vs Round

The current energy of the node after transmitting or receiving packets is called remaining or residual energy. The assessment of the performance of the proposed scheme is an important measure. Here, the remaining energy vs round graph shows that the suggested scheme saves more energy than LEACH-C and EEBCA.

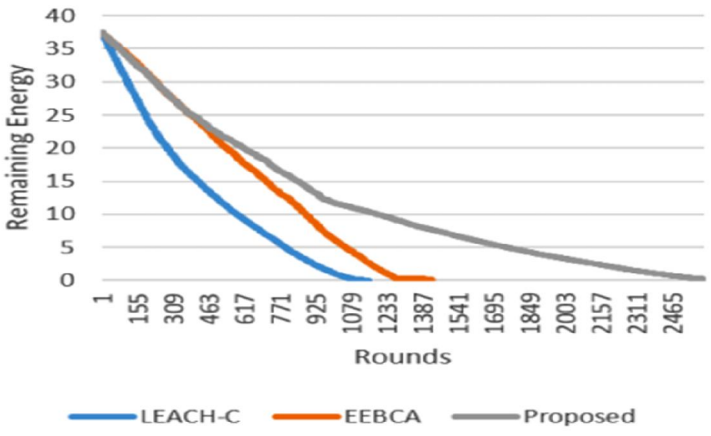


Figure 4.2: Remaining energy vs. Round

In LEACH-C, it executes up to round number 1155 but in the proposed on can go up to round number 2342. Because of less energy consumption proposed algorithm executes

more number of round which indicates more energy savings. It saves 114.95% and 39.44% power compare to LEACH-C and EEBCA, respectively.

4.3.2 Round vs Dead Node

In WSN, the network consists of many sensor nodes and gets energy from the battery source. If the charge of the battery is finished, then that node will die. A dead node will not collect or detect data from the area of detection or coverage and transfers the information to any other node. So if the sensor node can go more rounds, it increases both the lifetime and throughput of the network. Also, more dead nodes mean the degradation of network lifetime. Number of alive nodes indicate the network lifetime. If number of dead nodes are less, it reflects the network's long lifetime. Though proposed one's has a sharp increase in round number 925 but all nodes die in round number 2342. Considering LEACH-C and EEBCA it has found that all nodes die in round number 1155 and 1429 respectively.

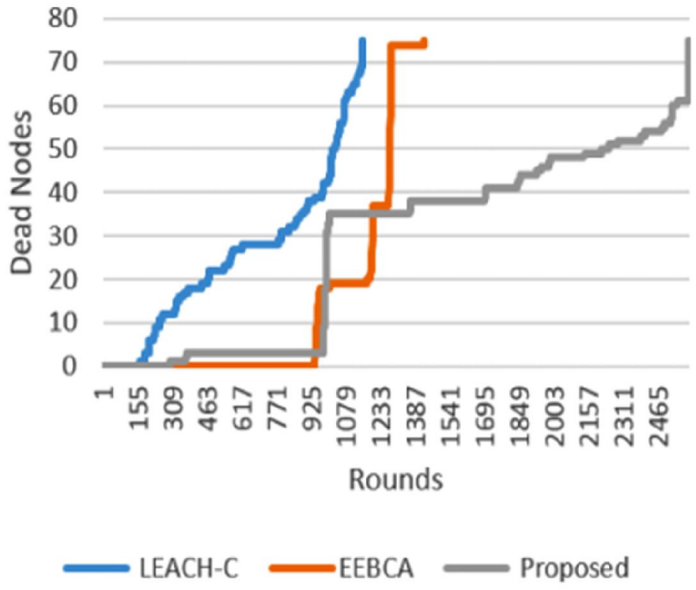


Figure 4.3: Rounds vs Dead Nodes

From the above figure, it shows that the proposed method maximizes the lifetime of 82.14% than EECBA. The number of dead nodes is less than the other two approaches. Less number of dead nodes indicates a longer lifetime of the network.

4.3.3 Rounds vs Throughput

The network output refers to how much information can be transmitted from source to destination during a given time.

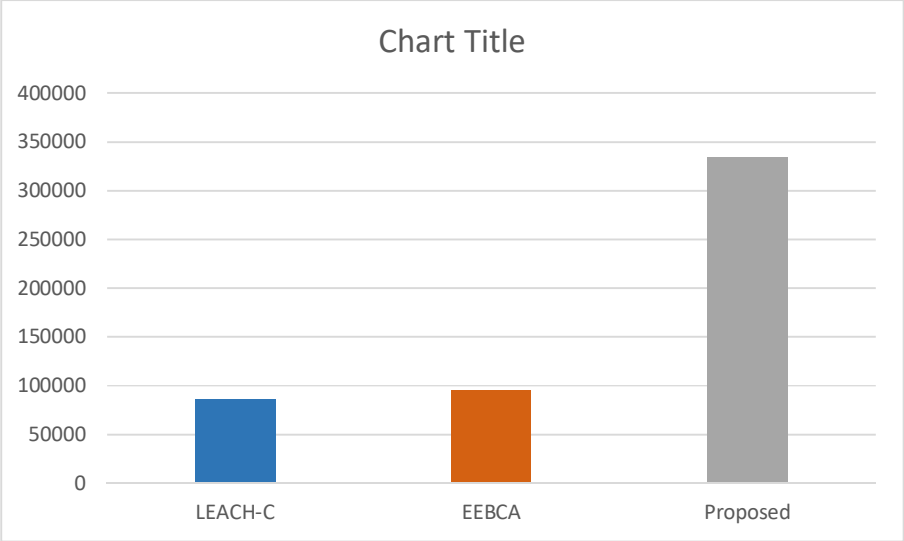


Figure 4.4: Round vs Throughput

The proposed scheme gave better throughput than LEACH-C and EEBCA. By the proposed scheme, five parameters considered which helps to save energy, and network lifetime has increased, accelerating the throughput of the network. From the above figure, it depicts that the proposed scheme got 99% more throughput than EEBCA and 135% more throughput than LEACH-C.

4.3.4 Round vs Energy Consumption

In WSN, one of the significant parameters is energy consumption. Energy consumption refers to absorb energy by the sensor nodes. To extend the lifetime of a WSN, it is a must to reduce the consumed energy. If energy consumption per round is decreased, then the network lifetime will be increased.

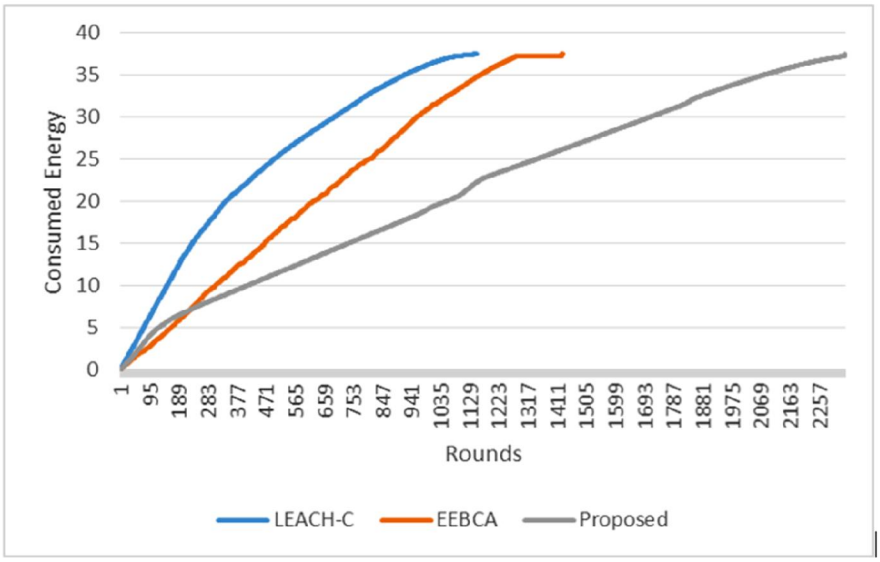


Figure 4.5: Rounds vs Energy Consumption

The above figure depicts that the proposed method consumed 50% less energy than LEACH-C and 29.3% less energy than EEBCA. Sensor nodes consumed less energy, so nodes can execute more rounds than the other two approaches.

4.3.5 Round vs number of CH

In WSN, to optimize the uses of energy by the sensor node clustering is the prime approach. When clustering, the network is split into a small region called a cluster, and one node is chosen as CH in each cluster to minimize energy consumption CH plays a very significant aspect in the cluster. So, a proper and energy-efficient CH selection scheme is essential for

improving the efficiency of the network. From various studies, it has been found that more number of CH in a cluster causes more energy consumption [49].From the analysis, it has been found that in the proposed scheme, number of CH selection is less than the existing methods, which cause longer life of nodes so that nodes can execute more rounds than the other two protocols.

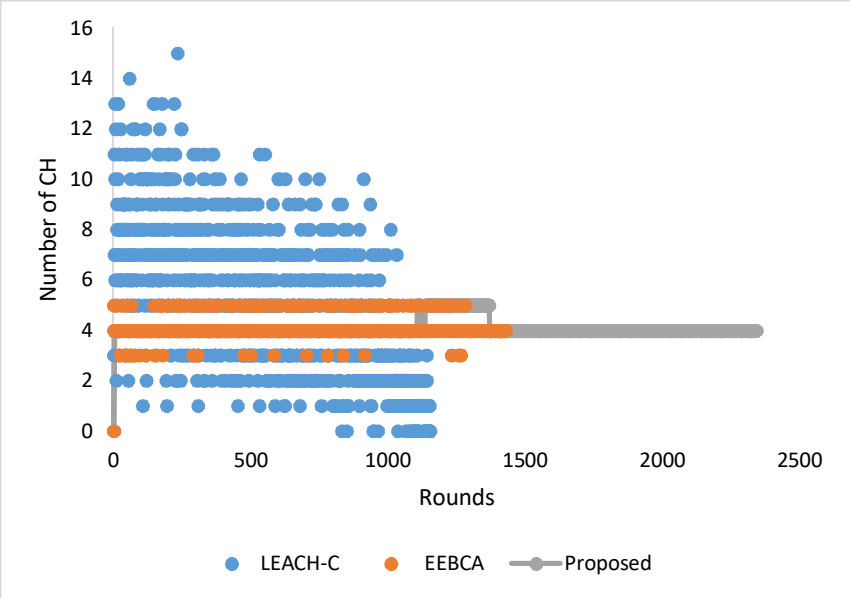


Figure 4.6: Rounds vs number of CH

4.3.6 FND, TND, and LND with Respect to Round

According to [50], WSN's lifespan is the cumulative amount of time for which the network stays operating. Currently, the node's lifetime depends on two variables: first, how much energy it uses over a round, and second, how much energy it absorbs to acquire, accumulate, and submit results. So, WSN's lifespan is the amount of time that the sensor nodes in the network can remain alive. Lifetime is often calculated as the network metrics of the First Node Dies (FND), Tenth Node Dies (TND), and the Last Node Dies (LND). FND means the time required until the first node dies from the beginning of the network

operation, TND means the time from the first operation to the 10th node dies, and LND means the time from the first operation to the last node dies. The lifetime of the network depends on the number of nodes that are still alive. If a sufficient number of nodes in the network remain intact, the network's life cycle increases. It increases throughput as well. Here, the first node, tenth node, and last node die has been considered with respect to round.

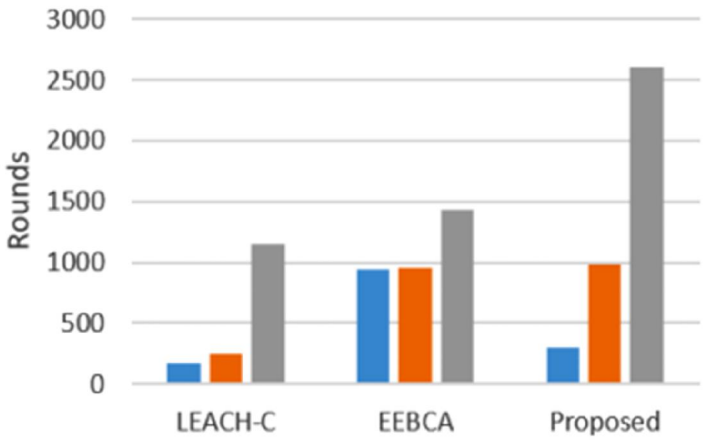


Figure 4.7: FND, TND and LND vs. Round

The proposed scheme increases the lifetime of the network where it shows that LND in the 2601 round. The FND, TND, and LND of LEACH-C, EEBCA, and the recommended scheme are illustrated in Figure 4.7. Observation shows that FND occurs at 168, 942, 297; TND at 254, 951, 984; LND at 1155, 1429, 2342 round in LEACH-C, EEBCA, and offered method, respectively. The proposed method's performance improved 125.58% and 82.14% of LND, higher than LEACH-C and EEBCA, respectively.

4.4 Simulation of Intra-Cluster Communication

For implementing the intra-cluster communication, the same network parameters have been considered. Different criteria for measuring the efficiency of the proposed technique have been considered. The output is also evaluated by contrasting it with LEACH-C works, and

current and recent works TAICC. In LEACH-C intra-cluster communication performed like LEACH. The proposed technique have compared in term of packets sending per round with traditional TDMA (LEACH-C) and a CSMA based slot requisition intra-cluster communication technique (TAICC).

4.4.1 Cumulative Data Packets Send to BS in Each Round

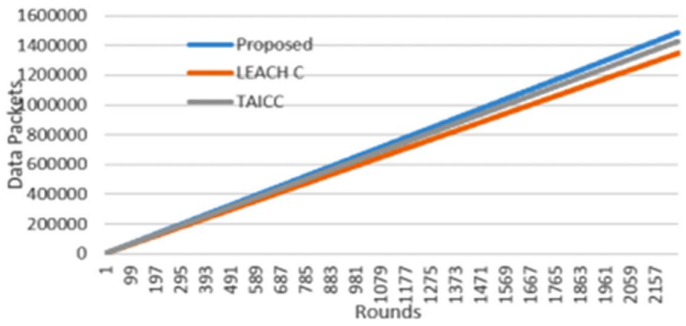


Figure 4.8: Data Packets vs Round

Figure 4.8 shows cumulative data packets send to BS of each round. We observed that the proposed protocol sends more data packets to compare to LEACH-C and TAICC in each round. So, the throughput of the suggested protocol is much better than LEACH-C and TAICC.

4.4.2 Data Frames Send to BS in Each Round

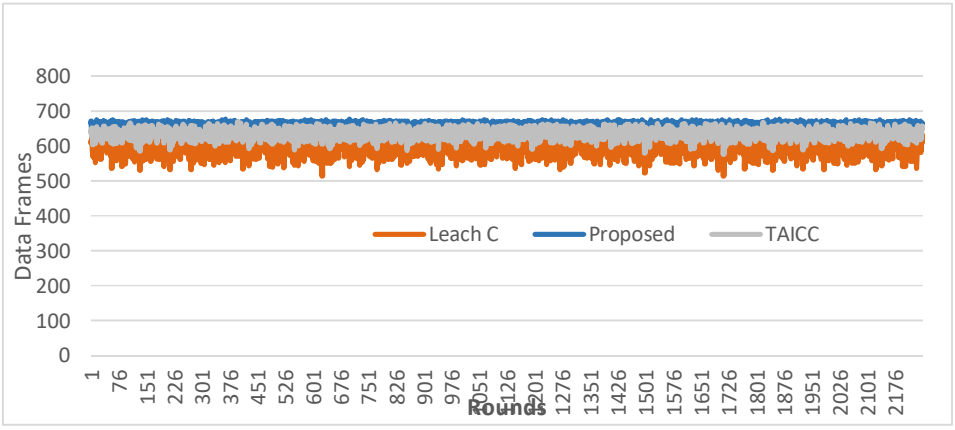


Figure 4.9: Number of Transmitted Frames to BS

Figure 4.9 shows that the offered scheme sends more data frames in each round in contrast to LEACH-C and TAICC. It reflects that it increases the throughput of the network. Proposed scheme can send more data frames than LEACH-C and TAICC. Network throughput of the proposed scheme is much greater than previous schemes.

4.4.3 Data Packets Send Comparison in Each Round

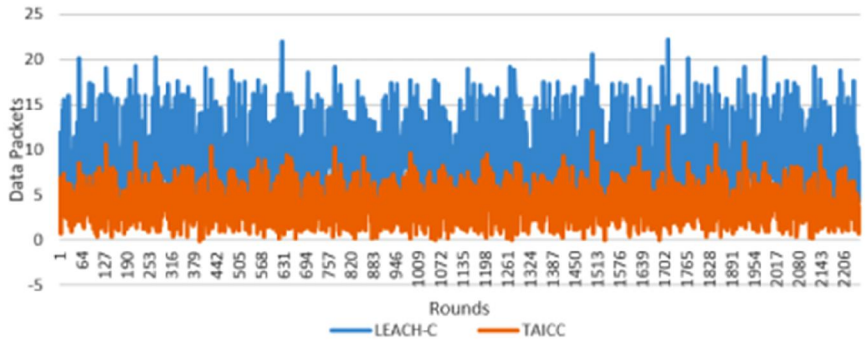


Figure 4.10: Number of Transmitted Packets to BS

From the above figure it shows that more data in percentage send in each round of the proposed protocol in contrast to LEACH-C and TIACC. This analysis observed that the

proposed protocol sends on average 8.46% and 3.56% more data packets compared to LEACH-C and TIACC, respectively.

4.4.4 Total Data Packets Send

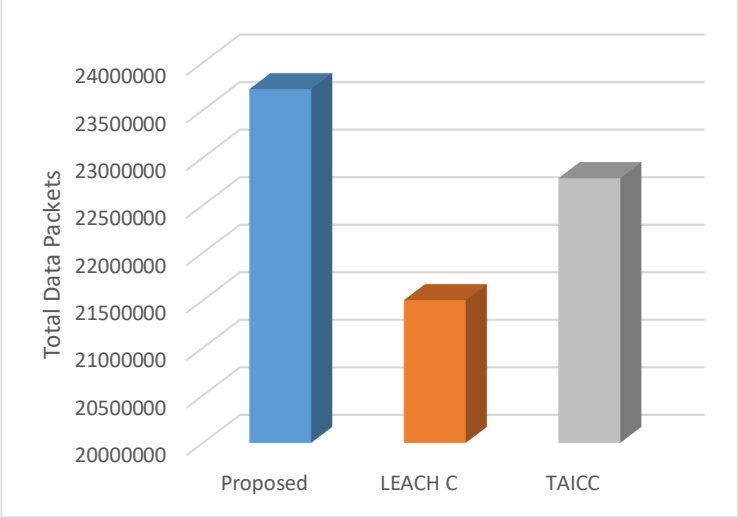


Figure 4.11: Total Data Sending Analysis

The figure shows the total data sending analysis of three protocols. This figure depicts that the proposed protocol, LEACH-C, and TIACC protocol sends a total of 23720614, 21501522, and 22784348 data packets, respectively. It also observed that the proposed protocol sends 2219092 and 936266 more packets compare to LEACH-C and TIACC, respectively.

Table 4.2: Performance comparison of competitive protocols for CH selection

Parameters	Clustering Protocols			Discussion
	LEACH-C	EEBCA	Proposed	
Throughput	86169	95545	333945	99% more throughput than EEBCA and 135% throughput than LEACH-C

Table 4.2: Performance comparison of competitive protocols for CH selection (cont.)

Parameters	Clustering Protocols			Discussion
	LEACH-C	EEBCA	Proposed	
Throughput	86169	95545	333945	99% more throughput than EEBCA and 135% throughput than LEACH-C
Energy Consumption	28629.27	20215.21	14273.32	Consumed 50% less energy than LEACH-C and 29.3% than EEBCA.
Number of Rounds execute	1155	1429	2342	Proposed scheme goes 103% and 64% more rounds than LEACH-C and EECBA respectively.
Remaining Energy	17383.13	26788.05	37364.81	Remaining energy 114.95% and 39.4% more compare to LEACH-C and EEBCA, respectively.
Number of Dead Nodes	All nodes died in 1155 rounds	All nodes died in 1429 rounds	All nodes died in 2342 rounds	Executes more number of rounds than LEACH-C and EEBCA.

Table 4.2: Performance comparison of competitive protocols for CH selection (cont.)

Parameters	Clustering Protocols			Discussion
	LEACH-C	EEBCA	Proposed	
1st , 10th & Last node die w.r.t round	1st : 168 10th : 254 Last Node: 1155	1st : 942 10th : 951 Last Node: 1429	1st : 297 10th : 984 Last Node: 2342	Network lifetime has increased, so proposed scheme's last node die in 2342 round

Table 4.3: Performance comparison of competitive protocols for timeslot assignment

Parameters	Clustering Protocols			Discussion
	LEACH-C	TAICC	Proposed	
Data frames sends in each round	13,48,404	14,28,820	14,87,487	Sends more data frames in each round compare to LEACH-C and TAICC.
Data packet send comparison in each round	21041.95	8877.906	22988.23	Sends on average 8.46% and more data packets compare to LEACH-C and TAICC respectively.
Total data packet sends	2,15,01,522	2,27,84,348	2,37,20,614	Sends 2219092 and 936266 more packets compare to LEACH-C and TAICC respectively.

4.5 Summary

The proposed protocol simulated and contrasted with several current protocols in different parameters in this chapter. It also measured and shown a significant increase in longevity and development. From the results and graphical representation, it is clear that the proposed approach provide better result than the other existing methods. The results are listed in a tabular form below, where different metrics or parameters are shown.

CHAPTER 5: CONCLUSION

5.1 Thesis Outcomes

A CH selection protocol and a time slot assignment methodology were suggested in this research work. The literature review noted that much research has already been conducted on cluster development, balancing cluster, CH selection, and intra-cluster communication strategies in a central and dispersed approach. Different studies consider different parameters to find an appropriate clustering protocol that can easily overcome the energy consumption problem of WSN. In some cases, residual energy or distance from nodes or CHs to BS, positioning, or both considered for cluster head selection. The number of member nodes and one-hop neighbor data is considered in certain instances. Two works were carried out in this study: (a) choice of CH using residual energy, number of neighbors, one-hop neighbor data, distance from BS to CHs and distance from CH to CH-these five parameters and (b) a new technique for TDMA time slot assignment in intra-communications technique. It also showed a better result in comparison to the other two existing protocols- LEACH-C and EEBCA.

a) Five parameters (Residual Energy, Number of Neighbors, one-hop neighbor data, distance from BS to CHs, and CH to CH distance) were taken into account in this research study for CH selection, which consumed less energy, increased network life, and throughput. It also showed a better result in comparison to the other two existing protocols- LEACH-C and EEBCA.
b) In WSN, for communication purposes, most of its energy is spent. One of them is the Intra-Cluster Communication Technique, where most of the energy is spent. In our study, a new Intra-Cluster Communication Technique has been proposed, where the beacon concept has been used to reduce energy consumption. The slot

reservation and broadcasting are not required. So that consumption of energy is less. In this case, the number of packets sent to CH and BS from the node is higher, so the throughput is higher and higher than in previous studies.

To simulate the proposed protocol, we used the OMNeT++ simulator and compared it to LEACH-C, EECBA, and TAICC with different terms and showed a significant improvement in lifetime and throughput. All of the findings show that the proposed scheme's network performance is better than the clustering methods previously proposed. The appropriate choice of CH and an intra-cluster communication technique achieves this improvement.

5.2 Thesis Limitations

In the propose protocol, prime concern is minimize energy consumption. WSN comprises many sensor nodes and these sensor nodes perform its task using energy. Lifetime of sensor node depends on battery power. If the energy or battery power of the node run out fast then that node will die fast. So minimize the usage of energy plays a vital role to increase the longevity of the sensor node as well as the network. It also increases the throughput of the network. Limitations of the proposed approach is that, others parameters such as-delay, stability, security, robustness, coverage are not considered. Outcomes of the proposed approach show the better result than other existing protocols for energy consumption but it may shows different results for other parameters. Get the result of the proposed protocol from simulation implementations where some assumptions have considered.

5.3 Future Work

In this protocol, we only considered homogeneous and static nodes. Nodes are deployed randomly and after depletion nodes are considered static. We only considered energy, but

we did not consider the other parameters, such as delay, coverage, security, and data aggregation. Thus, addressing the mobile node in the environment and promoting the other parameters may be a future extension of this protocol. Implement the proposed approach considering other metrics or parameters in the future work.

REFERENCES

[1] S. L. Ullo and G. R. Sinha, "Advances in Smart Environment Monitoring Systems Using IoT and Sensors," *Sensors*, vol. 20, no. 11, p. 3113, May 2020.

[2] M. Elhoseny and A. E. Hassanien, "Optimizing Cluster Head Selection in WSN to Prolong Its Existence," in *Dynamic Wireless Sensor Networks*, vol. 165, Cham: Springer International Publishing, 2019, pp. 93–111.

[3] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[4] M. Dong, K. Ota, and A. Liu, "RMER: Reliable and Energy-Efficient Data Collection for Large-Scale Wireless Sensor Networks," *IEEE Internet Things J.*, vol. 3, no. 4, pp. 511–519, Aug. 2016.

[5] J. Ben-Othman and B. Yahya, "Energy efficient and QoS based routing protocol for wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 70, no. 8, pp. 849–857, Aug. 2010.

[6] J. Wang, Z. Zhang, F. Xia, W. Yuan, and S. Lee, "An Energy Efficient Stable Election-Based Routing Algorithm for Wireless Sensor Networks," *Sensors*, vol. 13, no. 11, pp. 14301–14320, Oct. 2013.

[7] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[8] C. Del-Valle-Soto, C. Mex-Perera, J. A. Nolazco-Flores, R. Velázquez, and A. Rossa-Sierra, "Wireless Sensor Network Energy Model and Its Use in the Optimization of Routing Protocols," *Energies*, vol. 13, no. 3, p. 728, Feb. 2020.

[9] M. Shafiq, "Systematic Literature Review on Energy Efficient Routing Schemes in WSN – A Survey," *Mob. Netw Appl*, p. 14.

[10] R. R. Rout and S. K. Ghosh, "Enhancement of Lifetime using Duty Cycle and Network Coding in Wireless Sensor Networks," *IEEE Trans. Wirel. Commun.*, vol. 12, no. 2, pp. 656–667, Feb. 2013.

[11] M. R. Senouci, A. Mellouk, M. A. Senouci, and L. Oukhellou, "Belief functions in telecommunications and network technologies: an overview," *Ann. Telecommun. - Ann. Télécommunications*, vol. 69, no. 3–4, pp. 135–145, Apr. 2014.

[12] M. Abdullah and A. Ehsan, "Routing Protocols for Wireless Sensor Networks: Classifications and Challenges," p. 12.

[13] S. Karthik, "Challenges of Wireless Sensor Networks and Issues associated with Time Synchronization," p. 5, 2015.

[14] S. Sharma, R. K. Bansal, and S. Bansal, “Issues and Challenges in Wireless Sensor Networks,” in *2013 International Conference on Machine Intelligence and Research Advancement*, Katra, India, pp. 58–62., Dec. 2013.

[15] S. Asheer and S. Kumar, “A comprehensive review of cooperative MIMO WSN: its challenges and the emerging technologies,” *Wirel. Netw.*, Nov. 2020.

[16] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” *Ad Hoc Netw.*, vol. 3, no. 3, pp. 325–349, May 2005.

[17] H. Landaluce, L. Arjona, A. Perallos, F. Falcone, I. Angulo, and F. Muralter, “A Review of IoT Sensing Applications and Challenges Using RFID and Wireless Sensor Networks,” *Sensors*, vol. 20, no. 9, p. 2495, Apr. 2020.

[18] N. Shabbir and S. R. Hassan, “Routing Protocols for Wireless Sensor Networks (WSNs),” in *Wireless Sensor Networks - Insights and Innovations*, P. Sallis, Ed. InTech, 2017.

[19] N. Gupta, N. Kumar, and D. S. Jain, “COVERAGE PROBLEM IN WIRELESS SENSOR NETWORKS: A SURVEY,” p. 8.

[20] I. S. Akila, S. V. Manisekaran, and R. Venkatesan, “Modern Clustering Techniques in Wireless Sensor Networks,” in *Wireless Sensor Networks - Insights and Innovations*, P. Sallis, Ed. InTech, 2017.

[21] M. M. Zanjireh and H. Larijani, “A Survey on Centralised and Distributed Clustering Routing Algorithms for WSNs,” in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, Glasgow, United Kingdom, pp. 1–6, May 2015.

[22] A. More and V. Raisinghani, “A survey on energy efficient coverage protocols in wireless sensor networks,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 29, no. 4, pp. 428–448, Oct. 2017.

[23] R. Sruthi, “Medium Access Control Protocols for Wireless Body Area Networks: A Survey,” *Procedia Technol.*, vol. 25, pp. 621–628, 2016.

[24] M. Ahmed, M. Salleh, and M. I. Channa, “Routing protocols based on protocol operations for underwater wireless sensor network: A survey,” *Egypt. Inform. J.*, vol. 19, no. 1, pp. 57–62, Mar. 2018.

[25] S. A. Jesudurai and A. Senthilkumar, “An improved energy efficient cluster head selection protocol using the double cluster heads and data fusion methods for IoT applications,” *Cogn. Syst. Res.*, vol. 57, pp. 101–106, Oct. 2019.

[26] A. B. M. Alim Al Islam, C. Sayeed Hyder, H. Kabir, and M. Naznin, “Finding the Optimal Percentage of Cluster Heads from a New and Complete Mathematical Model on LEACH,” *Wirel. Sens. Netw.*, vol. 02, no. 02, pp. 129–140, 2010.

[27] O. Younis, “HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks,” *IEEE Trans. Mob. Comput.*, vol. 3, no. 4, p. 14, 2004.

[28] S. K. Singh, P. Kumar, and J. P. Singh, “A Survey on Successors of LEACH Protocol,” *IEEE Access*, vol. 5, pp. 4298–4328, 2017.

[29] P. K. Batra and K. Kant, “LEACH-MAC: a new cluster head selection algorithm for Wireless Sensor Networks,” *Wirel. Netw.*, vol. 22, no. 1, pp. 49–60, Jan. 2016.

[30] Lan Tien Nguyen, X. Defago, R. Beuran, and Yoichi Shinoda, “An energy efficient routing scheme for mobile wireless sensor networks,” in *2008 IEEE International Symposium on Wireless Communication Systems*, Reykjavik, Iceland, pp. 568–572, 2008.

[31] W. Abushiba, P. Johnson, S. Alharthi, and C. Wright, “An energy efficient and adaptive clustering for wireless sensor network (CH-leach) using leach protocol,” in *2017 13th International Computer Engineering Conference (ICENCO)*, Cairo, pp. 50–54, Dec. 2017.

[32] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Trans. Wirel. Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[33] T. G. Nguyen, C. So-In, and N. G. Nguyen, “Two energy-efficient cluster head selection techniques based on distance for wireless sensor networks,” in *2014 International Computer Science and Engineering Conference (ICSEC)*, Khon Kaen, Thailand, pp. 33–38, Jul. 2014.

[34] M. C. M. Thein and T. Thein, “An Energy Efficient Cluster-Head Selection for Wireless Sensor Networks,” in *2010 International Conference on Intelligent Systems, Modelling and Simulation*, Liverpool, United Kingdom, pp. 287–291, Jan. 2010.

[35] Mao Ye, Chenfa Li, Guihai Chen, and Jie Wu, “EECS: an energy efficient clustering scheme in wireless sensor networks,” in *PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, 2005.*, Phoenix, AZ, USA, pp. 535–540, 2005.

[36] Yuvaraj P. and K. V. L. Narayana, “EESCA: Energy efficient structured clustering algorithm for wireless sensor networks,” in *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, Pune, India, pp. 523–527, Dec. 2016.

[37] A. Karmaker, M. S. Alam, Md. M. Hasan, and A. Craig, “An energy-efficient and balanced clustering approach for improving throughput of wireless sensor networks,” *Int. J. Commun. Syst.*, vol. 33, no. 3, p. e4195, Feb. 2020.

[38] Z. Beiranvand, A. Patooghy, and M. Fazeli, “I-LEACH: An efficient routing algorithm to improve performance & to reduce energy consumption in Wireless Sensor Networks,” in *The 5th Conference on Information and Knowledge Technology*, shiraz, Iran, pp. 13–18, May 2013.

[39] A. Karmaker, Md. M. Hasan, S. S. Moni, and M. S. Alam, "An efficient cluster head selection strategy for provisioning fairness in wireless sensor networks," in *2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, Pune, India, pp. 217–220, Dec. 2016.

[40] S. Islam, N. I. Khan, S. M. J. Islam, and J. Akhtar, "Cluster Head Selection Technique Using Four Parameters of Wireless Sensor Networks," in *2019 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, Tamil Nadu, India, pp. 1–4, Jan. 2019.

[41] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-Efficient Routing Protocols in Wireless Sensor Networks: A Survey," *IEEE Commun. Surv. Tutor.*, vol. 15, no. 2, pp. 551–591, 2013, doi: 10.1109/SURV.2012.062612.00084.

[42] S. Kassan, P. Lorenz, and J. Gaber, "Low Energy and Location Based Clustering Protocol for Wireless Sensor Network," p. 6.

[43] A. Karmaker and M. M. Hasan, "A CSMA based intra cluster communication technique for saving cluster head energy," in *16th Int'l Conf. Computer and Information Technology*, Khulna, pp. 267–270, Mar. 2014.

[44] T. Ahmad, M. Haque, and A. M. Khan, "An Energy-Efficient Cluster Head Selection Using Artificial Bees Colony Optimization for Wireless Sensor Networks," in *Advances in Nature-Inspired Computing and Applications*, S. K. Shandilya, S. Shandilya, and A. K. Nagar, Eds. Cham: Springer International Publishing, pp. 189–203, 2019.

[45] B. Srikanth, M. Harish, and R. Bhattacharjee, "An energy efficient hybrid MAC protocol for WSN containing mobile nodes," in *2011 8th International Conference on Information, Communications & Signal Processing*, Singapore, pp. 1–5, Dec. 2011.

[46] A. Karmaker, M. Hasan, and M. S. Alam, "A Traffic Aware Intra-Cluster Communication Mechanism for Wireless Sensor Networks," in *2018 10th International Conference on Electrical and Computer Engineering (ICECE)*, Dhaka, Bangladesh, pp. 389–392, Dec. 2018.

[47] N. Tan, "An Improved LEACH Routing Protocol for Energy-Efficiency of Wireless Sensor Networks," *Smart Comput. Rev.*, vol. 2, no. 5, 2012.

[48] A. Chamam and S. Pierre, "A distributed energy-efficient clustering protocol for wireless sensor networks," *Comput. Electr. Eng.*, vol. 36, no. 2, pp. 303–312, Mar. 2010.

[49] M. M. Zanjireh, A. Shahrabi, and H. Larijani, "ANCH: A New Clustering Algorithm for Wireless Sensor Networks," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, Barcelona, pp. 450–455, Mar. 2013.

[50] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. Hanzo, "A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 2, pp. 828–854, 2017.

APPENDIX: SAMPLE CODES

Node.cc

```
#include <Node.h>
#include <Sink.h>
#include<RANDOM.H>
#include<custMsg_m.h>
#include<algorithm>
#include<string>
#include<stdlib.h>
#include<cUtility.h>
#include<cmath>

Define_Module(Node);
int setInf=0;

int numberOfCluster=0;
int clusterHeadListShortest[5000];
int l=4000;
int sortedNodesListFromBS[5000];
int spetialZoneNodes[1000];
int proCHIndex;
Node::Node() {
    // TODO Auto-generated constructor stub
    this->X = 0;
    this->Y = 0;
    batteryPower = 0.5;
    this->G = 0.0;
    this->type = 'N'; //Normal Node

    //Sleep time
    sleepTime = 0.10;

    Efs = 10 * 0.000000000001;
    Emp = 0.0013 * 0.000000000001;
    Do = sqrt(Efs / Emp);

    CHETx = 0.0;
    CHERx = 0.0;
    NETX = 0.0; //Node ETX
    NERX = 0.0; //Node ERX

    ETX = 50 * 0.000000001;
    ERX = 50 * 0.000000001;

    EDA = 5 * 0.000000001;

    CHIndex = 0;
    roundInterval = 1.0;

    optimalClusterFactor = 100; //Long value as inifinity

    alpha = 0.1;
```

```
    efactor =  0.0;

     dToBS = 109.06;

    thresholdEnergy = 0.23;

    noDataSentToCH = 0;
    noDataSentToSink = 0;
}

Node::~Node() {
    // TODO Auto-generated destructor stub

//    dataQueue->clear();
//    delete dataQueue;

}

void Node::initialize() {

    //EV<<"PI: " <<PI<<endl;
    //endSimulation();

    dataQueue = new cQueue;  //Queue for incoming message
    chDataQueue = new cQueue;

    wakeup = CreateCustMsg("Wakeup");
    noOfWirelessNode = getParentModule()->par("noOfWirelessNode");
    clusterHeadPercentage = getParentModule()->par("clusterHeadPercentage");

    this->netSizeX = getParentModule()->par("netSizeX");
    this->netSizeY = getParentModule()->par("netSizeY");


    SetCoordinate();

    this->K = (sqrt(noOfWirelessNode) / sqrt(2 * PI)) * sqrt(Efs / Emp)
            * ((double) netSizeX / (dToBS * dToBS));


    this->R = sqrt((double) (netSizeX * netSizeX) / (PI * K)); //==55.3215

ev<<"   K="<<this->K ;
    //Set Initial Cluster head and cluster
    if (simTime().dbl() == 0.0 && getIndex() == (noOfWirelessNode - 1)) {
        CalculateAvgDistanceToBS();

        //endSimulation();

        CalculateNeighborNode();
        SetEnergyMarker();
        CalculateWGTV();
        //endSimulation();
        ClusterHeadSelection(0);
        ClusterFormation(0);
```

```
    }

    /*************************************************************************
     * Initial schedule for each node
     * ***********************************************************************/
    scheduleAt(
            simTime().dbl()
                    + ((double) noOfWirelessNode - getIndex())
                            / (double) noOfWirelessNode, wakeup);
}

void Node::handleMessage(cMessage *msg) {

    int networkStatus = getParentModule()->par("networkStatus");
    double lastRoundTime = getParentModule()->par("lastRoundTime");
    roundNumber = getParentModule()->par("roundNumber");
    int noOfNodeDied = getParentModule()->par("noOfNodeDied");

    /*Casting incomming msg to custMsg(Customize message)*/
    custMsg *inMsg = check_and_cast<custMsg *>(msg);

    //Handle incoming data message
    if (strcmp("DataMsg", inMsg->getFullName()) == 0) {

        //Power consumtion for incomming message
        if (this->type == 'C' && this->batteryPower >= 0) {
            //For Cluster head
            this->batteryPower = this->batteryPower - this->CHERx;

            if (this->batteryPower <= 0) {
                noOfNodeDied++;
                getParentModule()->par("noOfNodeDied") = noOfNodeDied;

                if (noOfNodeDied == 1) {
                    getParentModule()->par("fstNodeDieRound") = roundNumber;
                }

                if (noOfNodeDied == 10) {
                    getParentModule()->par("tenNodeDieRound") = roundNumber;
                }

                if (noOfNodeDied == noOfWirelessNode) {
                    getParentModule()->par("allNodeDieRound") = roundNumber;
                }
            }
        }

        inMsg->setPacketReachTime(simTime().dbl());
        chDataQueue->insert(inMsg);
    }

    //Initial setup
    if (roundNumber <= 0 && getIndex() == noOfWirelessNode - 1) {
        EV << "Initial Setting round time and round number" << endl;
        roundNumber = roundNumber + 1;
```

```
    lastRoundTime = simTime().dbl();
    getParentModule()->par("roundNumber") = roundNumber;
    getParentModule()->par("lastRoundTime") = lastRoundTime;
    getParentModule()->par("networkStatus") = 2; //Means now nodes will send data to sink.
}

//Send data to CH
if (networkStatus == 2) {
    if (this->type == 'N') {
        SendDataToCH();

    }
    if (this->type == 'C')
    {
        int queueLen = dataQueue->length();

          if (queueLen > 0)
          {
        noDataSentToCH = getParentModule()->par("noDataSentToCH");
    getParentModule()->par("noDataSentToCH") = noDataSentToCH + 1;
    queueLen--;
          }
                              //sendSelfData();
    }
    if (getIndex() == noOfWirelessNode - 1) {
        //Means all data sent to CH and CH will send data to Sink
        getParentModule()->par("networkStatus") = 3;
    }
}

//Send data to Sink
if (networkStatus == 3) {
    if (this->type == 'C') {
        //sendSelfData();
        SendDataToSink();
    }

    if (getIndex() == noOfWirelessNode - 1) {
        //Means all data sent to Sink, ready for setup round
        getParentModule()->par("networkStatus") = 1;
        //EV<<"All data sent to CH in round : "<<roundNumber<<endl;
        //endSimulation();
    }
}

//Setup state
if (roundNumber > 0
        && simTime().dbl() >= lastRoundTime + this->roundInterval
        && getIndex() == noOfWirelessNode - 1 && networkStatus == 1) {

    roundNumber = roundNumber + 1;
    lastRoundTime = simTime().dbl();
    getParentModule()->par("roundNumber") = roundNumber;
    getParentModule()->par("lastRoundTime") = lastRoundTime;
    //ClusterHeadSelectionOld(roundNumber);
```

```
        //OptimalClusterFormation();

        SetEnergyMarker();
        CalculateWGTV();
        ClusterHeadSelection(roundNumber);
        ClusterFormation(roundNumber);
        getParentModule()->par("networkStatus") = 2; //Means now nodes will send data to sink.
    }

    if (msg->isSelfMessage()) {
        custMsg *dataMessage = CreateCustMsg("DataMsg");
        dataQueue->insert(dataMessage);
        scheduleAt(simTime().dbl() + sleepTime, msg->dup()); //Schedule after 0.5 second
    }
}
void Node::sendSelfData()
{
    int noOfNodeDied = getParentModule()->par("noOfNodeDied");
      int destNodeIndex = this->CHIndex;
      tempDestModule = getParentModule()->getSubmodule("node", destNodeIndex);

      int queueLen = dataQueue->length();

      if (queueLen > 0) {
          //Added to calculate throughput
          noDataSentToCH = getParentModule()->par("noDataSentToCH");
          getParentModule()->par("noDataSentToCH") = noDataSentToCH + 1;

      }

}
void Node::finish() {

    WriteOneTenAllNodeDeadHistory();


}

void Node::SetCoordinate() {

    int corX = 0;
    int corY = 0;
    int startAfterX = 5;
    int endBeforeX = netSizeX - 5;
    int startAfterY = 20;
    int endBeforeY = netSizeX - 5;

    //Take Rand value >10 and <=750
    do {
        corX = intrand(netSizeX); //old = 800
    } while (corX <= startAfterX || corX >= endBeforeX); //Old 10,750

    do {
        corY = intrand(netSizeY);
    } while (corY <= startAfterY || corY >= endBeforeY); //old 70, 550
```

```
    this->X = corY;
    this->Y = corY;

    getDisplayString().setTagArg("p", 0, corX);
    getDisplayString().setTagArg("p", 1, corY);
}

void Node::OptimalClusterFormation() {

    //endSimulation();

    int sortedNode[noOfWirelessNode];
    //double sorted

    for (int i = 0; i < noOfWirelessNode; i++) {
        sortedNode[i] = i;
    }



    cModule *tempCurModule;
    Node *tempCurNode;

    cModule *tempNextModule;
    Node *tempNextNode;

    Node *NodeList[noOfWirelessNode];

    //Bubble sort algorithm : Working
    for (int i = 0; i < noOfWirelessNode; i++) {
        //int iValue = sortedNode[i]; //Node index

        for (int j = 0; j < noOfWirelessNode - i - 1; j++) {
            int jValue = sortedNode[j];
            int jNextValue = sortedNode[j + 1];

            tempCurModule = getParentModule()->getSubmodule("node", jValue);
            tempCurNode = check_and_cast<Node *>(tempCurModule);

            tempNextModule = getParentModule()->getSubmodule("node",
                    jNextValue);
            tempNextNode = check_and_cast<Node *>(tempNextModule);

            if (tempCurNode->optimalClusterFactor
                    > tempNextNode->optimalClusterFactor) {
                int temp = sortedNode[j]; //Node index
                sortedNode[j] = sortedNode[j + 1];
                sortedNode[j + 1] = temp;
            }
        }

        //Set Cluster head for first 5 optimum cluster others are consider as normal node
    }
```

```
//    EV<<"clusterHeadPercentage: "<<clusterHeadPercentage <<endl;
//    endSimulation();

    int optimumCluster = (int) noOfWirelessNode * clusterHeadPercentage;
    int optimumClusterCounter = 0;
    EV << "optimumCluster: " << optimumCluster << endl;
    //endSimulation();

    for (int i = 0; i < noOfWirelessNode; i++) {
        int sortedValue = sortedNode[i];
        EV << "sortedValue: " << sortedValue << endl;
        tempCurModule = getParentModule()->getSubmodule("node", sortedValue);
        tempCurNode = check_and_cast<Node *>(tempCurModule);
        if (optimumClusterCounter >= optimumCluster) {
            tempCurNode->type = 'N';
            continue;
        }
        if (tempCurNode->type == 'C') {
            optimumClusterCounter++;

            int noOfCH = getParentModule()->par("noOfCH");
            noOfCH = noOfCH + 1;
            getParentModule()->par("noOfCH") = noOfCH;

            char buffer[33];
            itoa(sortedValue, buffer, 10);
            std::string strCH(buffer);
            //String conversion end

            //std::string clusterIndex = std::to_string(i);

            if (noOfCH <= 1) {
                getParentModule()->par("lstCH") = strCH;
            } else {

                std::string prevList = getParentModule()->par("lstCH");
                getParentModule()->par("lstCH") = prevList + "," + strCH;

            }

        }
    }
}

void Node::ClusterHeadSelection(int roundNo) {

    int noOfCH = 0;
    noOfCH = getParentModule()->par("noOfCH");

    //Calculate noOf dead node before creating new node cluster
    int noOfNodeDied = getParentModule()->par("noOfNodeDied");
    ThresholdCheck();

    WriteDeadNodeHistory(noOfNodeDied, roundNo - 1);
    WriteNetworkEnergyHistory(roundNo - 1);
```

```
    CountSinkPacket(roundNo - 1);
    CountCH(roundNo - 1, noOfCH);
    CountThroughput(roundNo - 1);

    //Reset parameter for each round
    ResetParam();

    getParentModule()->par("noOfCH") = 0; //Reset cluster head
    getParentModule()->par("noOfCluster") = 0; //reset no of cluster

/*    double temp_rand;
    double p = clusterHeadPercentage;
    int roundNumber = roundNo;*/
    double distance = 0.0;
   /* int noOfCH = 0;*/
    int noOfCluster = 0;
/*    int initialCH = 0;
    double optimalRandValue = dblrand();*/

    //Initially reset all node as normal node
    for (int i = 0; i < noOfWirelessNode; i++) {
        tempModule = getParentModule()->getSubmodule("node", i);
        tempNode = check_and_cast<Node *>(tempModule);
        tempNode->type = 'N';
        tempNode->CHIndex = 0;
    }

    //endSimulation();

    cModule *tempCurModule;
    Node *tempCurNode;

    cModule *tempNextModule;
    Node *tempNextNode;

    int sortedNode[noOfWirelessNode];


    for (int i = 0; i < noOfWirelessNode; i++) {
        sortedNode[i] = i;
    }

    //Bubble sort algorithm (according to WGTV)
    //output value are in ascending order like 5, 7, 14, 50

    for (int i = 0; i < noOfWirelessNode; i++) {
        for (int j = 0; j < noOfWirelessNode - i - 1; j++) {
            int jValue = sortedNode[j];
            int jNextValue = sortedNode[j + 1];

            tempCurModule = getParentModule()->getSubmodule("node", jValue);
            tempCurNode = check_and_cast<Node *>(tempCurModule);

            tempNextModule = getParentModule()->getSubmodule("node",
                    jNextValue);
```

```
            tempNextNode = check_and_cast<Node *>(tempNextModule);

            if (tempCurNode->WGTV > tempNextNode->WGTV) {
                int temp = sortedNode[j]; //Node index
                sortedNode[j] = sortedNode[j + 1];
                sortedNode[j + 1] = temp;
            }
        }
    }

    //Since it is ascending order
    cModule *tempNeigborModule;
    Node *tempNeigborNode;

    for (int i = noOfWirelessNode - 1; i >= 0; i--) {
        bool isCH = true;
        int sortedValue = sortedNode[i];

        EV<<"sorted value: " <<sortedValue<< endl;
        //endSimulation();

        tempCurModule = getParentModule()->getSubmodule("node", sortedValue);
        tempCurNode = check_and_cast<Node *>(tempCurModule);

        //tempCurNode->energyMarker = i + 1;
//        if(i == noOfWirelessNode-1)
//        {
//            isCH = true;
//        }
//        else
//        {
        for (int j = 0; j < tempCurNode->neighborNode.size(); j++) {
            int neighborNodeIndex = tempCurNode->neighborNode[j];
            tempNeigborModule = getParentModule()->getSubmodule("node",
                    neighborNodeIndex);
            tempNeigborNode = check_and_cast<Node *>(tempNeigborModule);
            if (tempNeigborNode->type == 'C') {
                isCH = false;
                break;
            }
        }


        if (isCH == 1) {

            distance = CalculateDistanceToBS(i);
            tempCurNode->distanceToBS = distance;




            for (int s = 0; s < noOfWirelessNode; s++) {
```

```
            cModule *curModule1;
            Node *curNode1;
           float neighborDistance1 = CalculateDistance(i, s);
            curModule1 = getParentModule()->getSubmodule("node", s);
            curNode1 = check_and_cast<Node *>(curModule1);

            int noOfNodeDied1 = getParentModule()->par("noOfNodeDied");
   // if (noOfNodeDied1 > 30)
       {
              if(noOfWirelessNode==75)
              {
                    if (neighborDistance1 <= curNode1->R*1.9 && neighborDistance1 >
curNode1->R )
                       curNode1->alpha=curNode1->alpha*1.180;

              }
              if(noOfWirelessNode==50)

                    if (neighborDistance1 <= curNode1->R*1.9 && neighborDistance1 >
curNode1->R )
                       curNode1->alpha=curNode1->alpha*1.180;

          }

          }
          CalculateWGTV();




       //test purpose
       //if(tempCurNode->distanceToBS >= tempCurNode->Do)
      if(tempCurNode->distanceToBS >= setInf)
              {
         continue;
       }

       tempCurNode->type = 'C';
       noOfCluster = getParentModule()->par("noOfCluster");
       getParentModule()->par("noOfCluster") = noOfCluster + 1;
       //EV << "Cluster No: " << noOfCluster + 1 << endl;


       int noOfCH = getParentModule()->par("noOfCH");
       noOfCH = noOfCH + 1;
       getParentModule()->par("noOfCH") = noOfCH;

       char buffer[33];
       itoa(sortedValue, buffer, 10);
       std::string strCH(buffer);
       //String conversion end

       //std::string clusterIndex = std::to_string(i);
```

```
        if (noOfCH <= 1) {
            getParentModule()->par("lstCH") = strCH;
        } else {

            std::string prevList = getParentModule()->par("lstCH");
            getParentModule()->par("lstCH") = prevList + "," + strCH;

        }



        EV << "DO: " << tempCurNode->Do << endl;

        //endSimulation();
        EV << "distanceToBS" << tempCurNode->distanceToBS << endl;

        if (tempCurNode->distanceToBS > tempCurNode->Do) {


            EV << "Multipath " << endl;
            EV << "Index: " << sortedValue << "distance to bs"
                    << tempCurNode->distanceToBS << endl;
            //endSimulation();


            tempCurNode->CHETx = (tempCurNode->ETX + tempCurNode->EDA)
                    * (4000)
                    + tempCurNode->Emp * 4000
                            * (distance * distance * distance * distance);
            }
            if (tempCurNode->distanceToBS <= tempCurNode->Do) {

                EV<<"Free space " <<endl;
                        EV<<"Index: " <<sortedValue << "distance to bs" <<tempCurNode-
>distanceToBS <<endl;
                        //endSimulation();


                tempCurNode->CHETx = (tempCurNode->ETX + tempCurNode->EDA)
            * (4000)
            + tempCurNode->Efs * 4000 * (distance * distance);
            }

            //EV<<"tempCurNode->CHETx: " <<tempCurNode->CHETx<<endl;
            //endSimulation();
            //Energy loss of CH to receive data
            tempCurNode->CHERx = (ERX + EDA) * 4000;

            //Added factor - Testing
            tempCurNode->CHETx = tempCurNode->CHETx - efactor;
            tempCurNode->CHERx = tempCurNode->CHERx - efactor;
```

```
        }
    }









}

//Calculate node distance from Sink
int Node::CalculateDistanceToBS(int senderIndex) {

    int senderX, senderY, receiverX, receiverY, baseDistance;


    senderNode = getParentModule()->getSubmodule("node", senderIndex);
    receiverNode = getParentModule()->getSubmodule("sink"); //Base station node
    //getParentModule()->getSubmodule("node", receiverIndex);

    tempTargetModule = check_and_cast<Node *>(senderNode);
    senderX = tempTargetModule->X;
    senderY = tempTargetModule->Y;

    tempBaseModule = check_and_cast<Sink *>(receiverNode);
    receiverX = tempBaseModule->X;
    receiverY = tempBaseModule->Y;

    baseDistance = (int) sqrt(
            (receiverX - senderX) * (receiverX - senderX)
                    + (receiverY - senderY) * (receiverY - senderY));

    return baseDistance;
}

void Node::clusterHeadListOfCurrentRound() {


    int  roundNumber = getParentModule()->par("roundNumber");
   int noOfCH= getParentModule()->par("noOfCH");
     int noOfNodeDied = getParentModule()->par("noOfNodeDied");
   //noOfWirelessNode=50;
   cModule *tempCHModule;
   Node *tempCHNode;
   int tempDistance = 0;

   for(int j=0;j<500;j++)
        clusterHeadListShortest[j]=0;

numberOfCluster=0;
     for(int i=0;i<noOfWirelessNode;i++)
     {
        tempCHModule = getParentModule()->getSubmodule("node", i);
              tempCHNode = check_and_cast<Node *>(tempCHModule);
```

```
            if(tempCHNode->type == 'C')
               clusterHeadListShortest[numberOfCluster++]=i;
    }



}

double   Node::getMinDitanceOfaNode(int nodeID)
{
   int minDistanceClusterIndex=0;
     double minDistance=9999;
     double TampDist=0;


    // ev<<endl<<" Distance From:"<<nodeID<<endl;
     for(int clstrNo=0;clstrNo<numberOfCluster;clstrNo++)
         {
       //  TampDist=distanceFromCH[nodeID][clstrNo];
         TampDist= CalculateDistance(nodeID,clusterHeadListShortest[clstrNo]);
       //  ev<<TampDist<<" ";
          if(minDistance>TampDist)
          {
             minDistanceClusterIndex=clstrNo;
             minDistance=TampDist;
          }

        }



     ev<<" Min Dist:"<<minDistance;
    return minDistance;
}
int  Node::isABorderNode(int nodeID,int minDistClusterNo)
{
   int clstrNo;
   int borderNode=1;
   double b=0,c=0;

//ev<<endl<<"Own Dis:"<<clusterHeadListShortest[minDistClusterNo]<<endl;
   for( clstrNo=0;clstrNo<numberOfCluster;clstrNo++)
   {
      {
       b=CalculateDistance(nodeID,clusterHeadListShortest[clstrNo]);

       c=CalculateDistance(minDistClusterNo,clusterHeadListShortest[clstrNo]);

      ev<<endl<<nodeID<<" to "<<clusterHeadListShortest[clstrNo]<<"="<<b<<"
"<<minDistClusterNo<<"to"<<clusterHeadListShortest[clstrNo]<<"="<<c;

        if(b<c)
          {
                        borderNode=0;
                       break;
```

```
            }
        }

    }
    return borderNode;
}

int  Node::getMinDitanceClusterId(int nodeID)
{
    int minDistanceClusterIndex=0;
    double minDistance=9999;
    double TampDist=0;


   ev<<endl<<endl<<" Distance From:"<<nodeID<<endl;
    for(int clstrNo=0;clstrNo<numberOfCluster;clstrNo++)
        {
       // TampDist=distanceFromCH[nodeID][clstrNo];
        TampDist= CalculateDistance(nodeID,clusterHeadListShortest[clstrNo]);
       ev<<TampDist<<" ";
          if(minDistance>TampDist)
          {
             minDistanceClusterIndex=clstrNo;
             minDistance=TampDist;
          }

        }



ev<<" Min Dist Cluster Index:"<<minDistanceClusterIndex;
  return minDistanceClusterIndex;

}

void Node::ClusterFormation(int roundNo) {

    clusterHeadListOfCurrentRound();
   // int minDistance = 32000;    //infinity
          cModule *tempCHModule;
          Node *tempCHNode;
          int tempDistance = 0;
          double minDistance=999;
          int baseX,baseY,midPointX,midPointY,midPointDistance1=0,midPointDistance2=0;
          int noOfNodeDied = getParentModule()->par("noOfNodeDied");
          receiverNode = getParentModule()->getSubmodule("sink");
          tempBaseModule = check_and_cast<Sink *>(receiverNode);
          baseX = tempBaseModule->X;
          baseY = tempBaseModule->Y;

    //int noOfCluster = getParentModule()->par("noOfCluster");
ev<<endl<<endl<<" CH LIst: ";

for(int i=0;i<numberOfCluster;i++)
```

```
    ev<<" "<<clusterHeadListShortest[i];

    for (int i = 0; i < noOfWirelessNode; i++) {
      tempModule = getParentModule()->getSubmodule("node", i);
      tempNode = check_and_cast<Node *>(tempModule);
      minDistance = 32000;
      tempNode->spType=0;
//BORDER NODES TUNNING

      if (tempNode->batteryPower > 0)
        {
                int     minDistClusterNo=clusterHeadListShortest[getMinDitanceClusterId(i)];
                        int   TmpminDistance=getMinDitanceOfaNode(i);
                       int yn=isABorderNode(i,minDistClusterNo);

        if(tempNode->type=='D')
         {
          tempNode->CHIndex= spetialZoneNodes[proCHIndex];
          minDistance=CalculateDistance(i,spetialZoneNodes[proCHIndex]);
          tempNode->type='N';
        }



        else
          if(yn)  //ONLY FOR CORENER NODE
            {
                                        tempNode->CHIndex=minDistClusterNo;
                                        ev<<endl<<" Border Node: "<<i<<" CH:
"<<minDistClusterNo;
                                        minDistance=TmpminDistance;

            }

        else

          if(tempNode->type == 'N')
        {
           //   getParentModule()->par("noOfCluster");
             // tempNode->getDisplayString().setTagArg("b",3,"red");
               int  firstClosestCH=clusterHeadListShortest[getMinDitanceClusterId(i)];

                tempNode->CHIndex=firstClosestCH;
               minDistance=CalculateDistance(i,firstClosestCH);
        }

      //FOR ENERGY CALCULATION
      if (minDistance > this->Do) {


        tempNode->NETX = ETX * (l)
              + Emp * l
                    * (minDistance * minDistance * minDistance
                          * minDistance);
      }
```

```
        if (minDistance <= Do) {
            tempNode->NETX = ETX * (l)
                    + Efs * l * (minDistance * minDistance);
        }
        tempNode->NETX = tempNode->NETX - efactor;
    }
    }
}

custMsg* Node::CreateCustMsg(const char *name) {

    custMsg *createMsg = new custMsg(name);
    createMsg->setPacketGenerateTime(simTime().dbl());
    createMsg->setPacketReachTime(simTime().dbl());

    createMsg->setSourceId(getIndex());

    return createMsg;
}

int Node::CalculateDistance(int senderIndex, int receiverIndex) {

    int senderX, senderY, receiverX, receiverY, baseDistance;

    cModule *senderNode;
    cModule *receiverNode;
    Node *tempTargetModule;

    senderNode = getParentModule()->getSubmodule("node", senderIndex);
    receiverNode = getParentModule()->getSubmodule("node", receiverIndex);

    tempTargetModule = check_and_cast<Node *>(senderNode);
    senderX = tempTargetModule->X;
    senderY = tempTargetModule->Y;

    tempTargetModule = check_and_cast<Node *>(receiverNode);
    receiverX = tempTargetModule->X;
    receiverY = tempTargetModule->Y;

    baseDistance = (int) sqrt(
            (receiverX - senderX) * (receiverX - senderX)
                    + (receiverY - senderY) * (receiverY - senderY));

    return baseDistance;
}

//Not used
void Node::TempDataSendToCH() {
    int noOfCH = getParentModule()->par("noOfCH");
    EV << "DataSendToCH, No of CH: " << noOfCH << endl;

    std::string lstCH = getParentModule()->par("lstCH");

    EV << "CH Lists: " << lstCH << endl;
```

```
    cUtility *objcUtility = new cUtility;
    std::vector<std::string> result = objcUtility->split(lstCH, ',');
    std::vector<int> vlstCH = objcUtility->convertToInt(result);

    for (int i = 0; i < vlstCH.size(); i++) {
        EV << "Converted CH: " << vlstCH[i] << endl;
    }

    //endSimulation();
    for (int ch = 0; ch < vlstCH.size(); ch++) {
        int destNodeIndex = vlstCH[ch];
        tempDestModule = getParentModule()->getSubmodule("node", destNodeIndex);
        tempDestNode = check_and_cast<Node *>(tempDestModule);

        for (int i = 0; i < noOfWirelessNode; i++) {
            tempSrcModule = getParentModule()->getSubmodule("node", i);
            tempSrcNode = check_and_cast<Node *>(tempSrcModule);
            if (tempSrcNode->CHIndex == destNodeIndex
                    && tempSrcNode->type == 'N') {
                EV << "Node index: " << i << "CH index: " << destNodeIndex
                        << endl;

                int queueLen = tempSrcNode->dataQueue->length();
                if (queueLen > 0) {
                    cObject *cObj = tempSrcNode->dataQueue->pop();
                    custMsg *qMsg = check_and_cast<custMsg *>(cObj);
                    //sendDirect(qMsg, baseModule, "radioIn");
                }




            }
        }

    }

    //endSimulation();

}

void Node::TempDataSendToSink() {
}

void Node::SendDataToCH() {

    int noOfNodeDied = getParentModule()->par("noOfNodeDied");
    int destNodeIndex = this->CHIndex;
    tempDestModule = getParentModule()->getSubmodule("node", destNodeIndex);

    int queueLen = dataQueue->length();

    if (queueLen > 0) {
        cObject *cObj = dataQueue->pop();
        custMsg *qMsg = check_and_cast<custMsg *>(cObj);
```

```
        if (this->batteryPower > 0) {
            this->batteryPower = this->batteryPower - this->NETX;

            if (this->batteryPower <= 0) {
                noOfNodeDied++;
                getParentModule()->par("noOfNodeDied") = noOfNodeDied;

                if (noOfNodeDied == 1) {
                    getParentModule()->par("fstNodeDieRound") = roundNumber;
                }

                if (noOfNodeDied == 10) {
                    getParentModule()->par("tenNodeDieRound") = roundNumber;
                }

                if (noOfNodeDied == noOfWirelessNode) {
                    getParentModule()->par("allNodeDieRound") = roundNumber;
                }

                deadTime = simTime().dbl();
            }
        }


        noDataSentToCH = getParentModule()->par("noDataSentToCH");
        getParentModule()->par("noDataSentToCH") = noDataSentToCH + 1;

        sendDirect(qMsg, tempDestModule, "radioIn");
    }

}

void Node::SendDataToSink() {

    int noOfNodeDied = getParentModule()->par("noOfNodeDied");
    sinkModule = getParentModule()->getSubmodule("sink");

    int queueLen = chDataQueue->length();

    int noPacketSentToSink = getParentModule()->par("noPacketSentToSink");
    getParentModule()->par("noPacketSentToSink") = noPacketSentToSink
            + queueLen;

    while (queueLen > 0) {
        cObject *cObj = chDataQueue->pop();
        custMsg *qMsg = check_and_cast<custMsg *>(cObj);
        sendDirect(qMsg, sinkModule, "radioIn");
        queueLen--;
    }


    noDataSentToSink = getParentModule()->par("noDataSentToCH");
    getParentModule()->par("noDataSentToSink") = noDataSentToSink;
```

```
    if (this->batteryPower > 0) {
        this->batteryPower = this->batteryPower - this->CHETx;

        if (this->batteryPower <= 0) {
            noOfNodeDied++;
            getParentModule()->par("noOfNodeDied") = noOfNodeDied;
            deadTime = simTime().dbl();

            if (noOfNodeDied == 1) {
                getParentModule()->par("fstNodeDieRound") = roundNumber;
            }

            if (noOfNodeDied == 10) {
                getParentModule()->par("tenNodeDieRound") = roundNumber;
            }

            if (noOfNodeDied == noOfWirelessNode) {
                getParentModule()->par("allNodeDieRound") = roundNumber;
            }
        }
    }

}

void Node::WriteDeadNodeHistory(int noOfDeadNode, int roundNumber) {

    const char* filename = "02-RoundVsDeadNode-BP2.csv";
    const char* filenameAliveNode = "03-RoundVsAliveNode-BP2.csv";

    FILE *file;
    FILE *fileAliveNode;

    if (roundNumber < 1) {
        file = fopen(filename, "w+");
        fileAliveNode = fopen(filenameAliveNode, "w+");
        fprintf(file, "Round Number, Dead Node\n");
        fprintf(fileAliveNode, "Round Number, Alive Node\n");
    } else {
        //Append or update mode
        file = fopen(filename, "a+");
        fileAliveNode = fopen(filenameAliveNode, "a+");
    }

    //FILE *file = fopen(filename, "w+");

    fprintf(file, "%d,%d\n", roundNumber, noOfDeadNode);
    fclose(file);

    fprintf(fileAliveNode, "%d,%d\n", roundNumber,
            noOfWirelessNode - noOfDeadNode);
    fclose(fileAliveNode);

}
```

```
void Node::WriteOneTenAllNodeDeadHistory() {
    const char* filename = "01-1st10thAllDie-BP2.csv";
    //const char* filenameAliveNode = "03-RoundVsAliveNode-BP2.csv";

    FILE *file;
    FILE *fileAliveNode;

    file = fopen(filename, "w+");
    fprintf(file, "Dead Node, Round Number\n");

    int fstNodeDieRound = getParentModule()->par("fstNodeDieRound");
    fprintf(file, "%d,%d\n", 1, fstNodeDieRound);

    int tenNodeDieRound = getParentModule()->par("tenNodeDieRound");
    fprintf(file, "%d,%d\n", 10, tenNodeDieRound);

    int allNodeDieRound = getParentModule()->par("allNodeDieRound");
    ;
    fprintf(file, "%d,%d\n", noOfWirelessNode, allNodeDieRound);

    fclose(file);
}

void Node::WriteNetworkEnergyHistory(int roundNumber) {

    //if(roundNumber <1 ) return;
    double initialBatteryPower = 0.5;
    double netInitialEnergy = initialBatteryPower * (double) noOfWirelessNode;
    double netRemainingEnergy = 0.0;
    double netConsumptionEnergy = 0.0;

    for (int i = 0; i < noOfWirelessNode; i++) {
        calModule = getParentModule()->getSubmodule("node", i);
        calNode = check_and_cast<Node *>(calModule);
        netRemainingEnergy = netRemainingEnergy + calNode->batteryPower;
    }

    if (netRemainingEnergy <= thresholdEnergy) {
        netRemainingEnergy = 0;
    }

    getParentModule()->par("totalRemainingEnergy") = netRemainingEnergy;
    getParentModule()->par("avgRemainingEnergy") = netRemainingEnergy /
(double)noOfWirelessNode;


    netConsumptionEnergy = netInitialEnergy - netRemainingEnergy;

    if (netConsumptionEnergy >= netInitialEnergy) {
        netConsumptionEnergy = netInitialEnergy;
    }

    const char* filenameRemainingEnergy = "04-RoundVsRemainingEnergy-BP2.csv";
    const char* filenameEnergyConsumption = "05-RoundVsEnergyConsumption-BP2.csv";
```

```
    FILE *fileRemainingEnergy;
    FILE *fileEnergyConsumption;

    if (roundNumber < 1) {
        fileRemainingEnergy = fopen(filenameRemainingEnergy, "w+");
        fileEnergyConsumption = fopen(filenameEnergyConsumption, "w+");
        fprintf(fileRemainingEnergy,
                "Round Number, Network Remaining Energy\n");
        fprintf(fileEnergyConsumption,
                "Round Number, Total Energy Consumption\n");
    } else {
        //Append or update mode
        fileRemainingEnergy = fopen(filenameRemainingEnergy, "a+");
        fileEnergyConsumption = fopen(filenameEnergyConsumption, "a+");
    }

    fprintf(fileRemainingEnergy, "%d,%lf\n", roundNumber, netRemainingEnergy);
    fclose(fileRemainingEnergy);

    fprintf(fileEnergyConsumption, "%d,%lf\n", roundNumber,
            netConsumptionEnergy);
    fclose(fileEnergyConsumption);
}

void Node::CountSinkPacket(int roundNumber) {
    if (roundNumber <= 0) {
        return;
    }

    int noPacketSentToSink = getParentModule()->par("noPacketSentToSink");
    int sinkPackets = 0;

    calSinkModule = getParentModule()->getSubmodule("sink");
    calSinkNode = check_and_cast<Sink *>(calSinkModule);
    sinkPackets = calSinkNode->noDataInSink;

    const char* filenameSinkPackets = "06-RoundVsSinkPackets-BP2.csv";
    FILE *fileSinkPackets;

    if (roundNumber < 1) {
        fileSinkPackets = fopen(filenameSinkPackets, "w+");
        fprintf(fileSinkPackets, "Round Number, Number of packets in Sink\n");
    } else {
        //Append or update mode
        fileSinkPackets = fopen(filenameSinkPackets, "a+");
    }

    fprintf(fileSinkPackets, "%d,%d\n", roundNumber, noPacketSentToSink);
    fclose(fileSinkPackets);

}

void Node::CalculateNeighborNode() {

    int neighborDistance = 0;
```

```
    for (int i = 0; i < noOfWirelessNode; i++) {
        cModule *curModule;
        Node *curNode;

        curModule = getParentModule()->getSubmodule("node", i);
        curNode = check_and_cast<Node *>(curModule);

        for (int j = 0; j < noOfWirelessNode; j++) {
            if (i == j) {
                continue;
            }

            neighborDistance = CalculateDistance(i, j);
            int noOfNodeDied = getParentModule()->par("noOfNodeDied");




            if (neighborDistance <= curNode->R) {
                curNode->neighborNode.push_back(j);







            }
        }


    }

}

void Node::SetEnergyMarker() {

    //int roundNumber =  getParentModule()->par("roundNumber");
    int sortedNode[noOfWirelessNode];

    for (int i = 0; i < noOfWirelessNode; i++) {
        sortedNode[i] = i;
    }

    cModule *tempCurModule;
    Node *tempCurNode;

    cModule *tempNextModule;
    Node *tempNextNode;

    //Node *NodeList[noOfWirelessNode];

    //Bubble sort algorithm (according to energy) : Working
    for (int i = 0; i < noOfWirelessNode; i++) {
```

```
        for (int j = 0; j < noOfWirelessNode - i - 1; j++) {
            int jValue = sortedNode[j];
            int jNextValue = sortedNode[j + 1];

            tempCurModule = getParentModule()->getSubmodule("node", jValue);
            tempCurNode = check_and_cast<Node *>(tempCurModule);

            tempNextModule = getParentModule()->getSubmodule("node",
                    jNextValue);
            tempNextNode = check_and_cast<Node *>(tempNextModule);

            if (tempCurNode->batteryPower > tempNextNode->batteryPower) {
                int temp = sortedNode[j]; //Node index
                sortedNode[j] = sortedNode[j + 1];
                sortedNode[j + 1] = temp;
            }
        }
    }



    for (int i = 0; i < noOfWirelessNode; i++) {
        int sortedValue = sortedNode[i];
        tempCurModule = getParentModule()->getSubmodule("node", sortedValue);
        tempCurNode = check_and_cast<Node *>(tempCurModule);
        tempCurNode->energyMarker = i + 1;

        EV << "Node: " << sortedValue << " Energy Marker: "
                << tempCurNode->energyMarker << endl;
    }

    //endSimulation();
}

void Node::CalculateAvgDistanceToBS() {

    //double dToBS = getParentModule()->par("dToBS");
    double totalDistance = 0.0;
    double avgDistance = 0.0;
    int distanceToBS = 0.0;
    for(int i=0; i<noOfWirelessNode; i++)
    {
        distanceToBS = CalculateDistanceToBS(i);
        totalDistance = totalDistance + distanceToBS;
    }

    EV<<"totalDistance: " << totalDistance <<endl;
    avgDistance = totalDistance / (double)noOfWirelessNode;
    getParentModule()->par("dToBS") = avgDistance;
    //endSimulation();
}
int Node::getBeta()
{

int x=netSizeX;
```

```
int y=netSizeY;
int bFact;
if(x==200&&y==200)
{setInf=150;
   if(noOfWirelessNode>=50 &&noOfWirelessNode<=59)
      bFact=70;
   else
      if(noOfWirelessNode>=60 &&noOfWirelessNode<=69)
         bFact=70;
      else
            if(noOfWirelessNode>=70 &&noOfWirelessNode<=79)
               bFact=70;
               else
                     if(noOfWirelessNode>=80 &&noOfWirelessNode<=89)
                        bFact=70;
                     else
                           if(noOfWirelessNode>=90 &&noOfWirelessNode<=99)
                              bFact=80;
                           else
                                 if(noOfWirelessNode>=100 &&noOfWirelessNode<=110)
                                    bFact=100;
                                 else
                                       if(noOfWirelessNode>=111 &&noOfWirelessNode<=120)
                                          bFact=130;
}

if(x==300&&y==200)
{
   setInf=120;
   if(noOfWirelessNode>=50 &&noOfWirelessNode<=59)
      bFact=50;
   else
      if(noOfWirelessNode>=60 &&noOfWirelessNode<=69)
         bFact=70;
      else
            if(noOfWirelessNode>=70 &&noOfWirelessNode<=79)
               bFact=70;
               else
                     if(noOfWirelessNode>=80 &&noOfWirelessNode<=89)
                        bFact=80;
                     else
                           if(noOfWirelessNode>=90 &&noOfWirelessNode<=99)
                              bFact=80;
                           else
                                 if(noOfWirelessNode>=100 &&noOfWirelessNode<=110)
                                    bFact=70;
                                 else
                                       if(noOfWirelessNode>=111 &&noOfWirelessNode<=120)
                                          bFact=70;
}
ev<<endl<<"X="<<x<<" y="<<y;
return bFact;
}
void Node::CalculateWGTV() {
   //int roundNumber =  getParentModule()->par("roundNumber");
```

```
    int mindist=0;
    int maxdist=0;
    int beta=getBeta();
    for (int i = 0; i < noOfWirelessNode; i++)
     {
         int tempdist=CalculateDistanceToBS(i);
         if(maxdist>tempdist)
            maxdist=tempdist;
     }
    int noOfNodeDied = getParentModule()->par("noOfNodeDied");
    double dFactor=0;
    for (int i = 0; i < noOfWirelessNode; i++) {
        cModule *curModule;
        Node *curNode;
        double d=CalculateDistanceToBS(i);
        if(d>0)
             dFactor=1/d;
            else
               dFactor=1;

        curModule = getParentModule()->getSubmodule("node", i);
        curNode = check_and_cast<Node *>(curModule);


       curNode->WGTV = curNode->energyMarker + curNode->alpha * curNode-
>neighborNode.size()+beta*dFactor;
    }

}

void Node::ThresholdCheck() {


    double netRemainingEnergy = (double) (getParentModule()->par(
          "totalRemainingEnergy"));

    int roundNo = getParentModule()->par("roundNumber");

    if (netRemainingEnergy <= thresholdEnergy && roundNo > 0) {

        getParentModule()->par("allNodeDieRound") = roundNo - 1;
        getParentModule()->par("noOfNodeDied") = noOfWirelessNode;

        int noOfNodeDied = getParentModule()->par("noOfNodeDied");
        WriteDeadNodeHistory(noOfNodeDied, roundNumber -1 );
        WriteNetworkEnergyHistory(roundNumber - 1);
        int noOfCH = getParentModule()->par("noOfCH");
        CountCH(roundNumber - 1, noOfCH);
        CountSinkPacket(roundNumber -1);
        EV<<"netRemainingEnergy: " << netRemainingEnergy <<"roundNo: " << roundNo <<endl;
        endSimulation();
    }
}

void Node::CountCH(int roundNumber, int noOfCH) {
```

```
    const char* filenameCHVsRound= "07-RoundVsCH-BP2.csv";
    //const char* filenameEnergyConsumption = "05-RoundVsEnergyConsumption-BP2.csv";

    FILE *fileCHVsRound;
    //FILE *fileEnergyConsumption;

    if (roundNumber < 1) {
        fileCHVsRound = fopen(filenameCHVsRound, "w+");
        fprintf(fileCHVsRound,
                "Round Number, No of CH\n");
    } else {
        //Append or update mode
        fileCHVsRound = fopen(filenameCHVsRound, "a+");
    }

    fprintf(fileCHVsRound, "%d,%d\n", roundNumber, noOfCH);
    fclose(fileCHVsRound);
}

void Node::ResetParam() {
    noDataSentToCH = 0;
    noDataSentToSink = 0;

    getParentModule()->par("noDataSentToCH") = 0;
    getParentModule()->par("noDataSentToSink") = 0;
}

void Node::CountThroughput(int roundNumber) {

    //Must remove
    int factorValue = 0;   //PRIVIOUS VALUE 12

    noDataSentToSink = getParentModule()->par("noDataSentToSink");
    noDataInSink = getParentModule()->par("noDataInSink");
    getParentModule()->par("noDataInSink") = noDataInSink + noDataSentToSink +
factorValue;
    int noDataInSinkAtPresent = getParentModule()->par("noDataInSink");

    const char* filenameThroughput = "08-RoundVsThroughput-BP2.csv";
    const char* filenameNoDataInSink = "09-RoundVsNoDataInSink-BP2.csv";

    FILE *fileThroughput;
    FILE *fileNoDataInSink;

    if (roundNumber < 1) {
        fileThroughput = fopen(filenameThroughput, "w+");
        fileNoDataInSink = fopen(filenameNoDataInSink, "w+");
        fprintf(fileThroughput,
                "Round Number, Throughput\n");

        fprintf(fileNoDataInSink,
                    "Round Number, No data in Sink\n");

    } else {
```

```
        //Append or update mode
        fileThroughput = fopen(filenameThroughput, "a+");
        fileNoDataInSink = fopen(filenameNoDataInSink, "a+");
    }

    fprintf(fileThroughput, "%d,%d\n", roundNumber, noDataSentToSink);
    fprintf(fileNoDataInSink, "%d,%d\n", roundNumber, noDataInSinkAtPresent);

    fclose(fileThroughput);
    fclose(fileNoDataInSink);
}
```

Node.h

```
#ifndef NODE_H_
#define NODE_H_

#include <omnetpp.h>
#include <Sink.h>
#include <custMsg_m.h>
#include<vector>

class Node: public cSimpleModule {

public:
    int X;
    int Y;
    int netSizeX;
    int netSizeY;
    int cornerNodeEffect;

    int noOfWirelessNode;
    double batteryPower;
    double G;
    double clusterHeadPercentage;
    char type;
    double distanceToBS;
    double sleepTime;

    double Efs; //10*0.000000000001;
    double Emp; //0.0013*0.000000000001;
    double Do;
    double CHETx;
    double CHERx;
    double NETX; //Node ETX
    double NERX; //Node ERX
```

```
double ETX;
double ERX;

double EDA;
int CHIndex;
double roundInterval;
double deadTime;
int roundNumber;
double optimalClusterFactor;

int M; //Network size
double R; //Radius
//double PI; //const value 3.14
double K; //optimum cluster
double dToBS;
int energyMarker;
double alpha;
double efactor;
int spType;

std::vector<int> neighborNode;

double WGTV;

double thresholdEnergy;

int noDataSentToCH;
int noDataSentToSink;
int noDataInSink;

custMsg *wakeup;
custMsg *data;
cQueue *dataQueue;
cQueue *chDataQueue;

cModule *senderNode;
cModule *receiverNode;
Node *tempTargetModule;
Sink *tempBaseModule;

cModule *tempModule;
Node *tempNode;

cModule *tempSrcModule;
Node *tempSrcNode;

cModule *tempDestModule;
Node *tempDestNode;

cModule *calModule;
Node *calNode;

cModule *calSinkModule;
Sink *calSinkNode;
```

```
    cModule *sinkModule;

    Node();
    virtual ~Node();

private:
    void SetCoordinate();
    void CalculateAvgDistanceToBS();
    void CalculateNeighborNode();
    void SetEnergyMarker();
    void CalculateWGTV();
    void ClusterHeadSelection(int roundNo);

    //void ClusterHeadSelectionOld(int roundNo);
    void ClusterFormation(int roundNo);

    int CalculateDistanceToBS(int nodeindex);
    custMsg* CreateCustMsg(const char *name);
    int CalculateDistance(int senderIndex, int receiverIndex);

    void SendDataToCH();
    void SendDataToSink();

    void WriteDeadNodeHistory(int noOfDeadNode, int roundNumber);
    void WriteOneTenAllNodeDeadHistory();
    void WriteNetworkEnergyHistory(int roundNumber);
    void CountSinkPacket(int roundNumber);
    void CountThroughput(int roundNumber);


    void TempDataSendToCH();
    void TempDataSendToSink();
    void OptimalClusterFormation();
    void ThresholdCheck();
    void CountCH(int roundNumber, int noOfCH); //Cound CH vs Round

    void ResetParam();
int getMinDitanceClusterId(int nodeID);
double getMinDitanceOfaNode(int nodeID);

void centralCornerNodesTunningNew();

    void clusterEngeryCalculation() ;
    void clusterHeadListOfCurrentRound();
    void showAllClusterInfo();
    void calculateAllClusterEnergy();
void sortClusterListAccordingToEnergyCurrentRound();
void getIndividualClusterInfo(int clusterNo) ;
void generateAllNodesDistanceTable();
void clusterBalancing();
void centralBorderTunning();
void shortCHFromZeroZero();
void showNecessaryInfo();
void clusterBalance();
```

```
void centralBorderNodesTunningNew1();
void centralBorderNodesTunningNew();
void balanceCentralBorderNodesTunning();
void newProposedInfo();
int isABorderNode(int nodeID,int minDistClusterNo);
void otherNodesTunningNew();
void chkValue();
void engergyBalanceingNew();
int getNearestCH(int CH);
void sendEnergyToCluster(int FromclstrNo,int toClstrNo,double avgEnergy);
int getSecondMinDitanceClusterId(int nodeID);
void centralBorderNodesTunning1();
void showSortingNodesFromBS();
void sortedNodeFromBS();
void sendSelfData();
//void clusterEngeryCalculation();
int getBeta();
protected:
   virtual void initialize();
   virtual void handleMessage(cMessage *msg);
   void finish();
   void adjustNext2Engr(int ClsNo,int Rno);
   void adjustNext1Engr(int ClsNo);
};

#endif /* NODE_H_ */
```