Energy efficient wireless sensor network for monitoring temperature and relative humidity in forest

Sinde, Ramadhani S.

NM-AIST

ENERGY EFFICIENT WIRELESS SENSOR NETWORK FOR MONITORING TEMPERATURE AND RELATIVE HUMIDITY IN FORESTS

Ramadhani Sinde

A Dissertation Submitted in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy in Information and Communication Science and Engineering of the Nelson Mandela African Institution of Science and Technology

Arusha, Tanzania

April, 2020

ABSTRACT

Monitoring the forest's weather has been essential to living things over the years. Currently, there is a shortage of information on real-time temporal and spatial environmental conditions of the forest that drive forest health condition. This work focuses on the sensing of humidity and temperature as weather data from the forest. Unlike the traditional systems used to collect weather information, the use of wireless sensor network (WSN) gives real-time data capture from every point of the forest. However, the WSN faces the number of challenges including low bandwidth, low power, and short battery lifespan. Reducing the network delay and improving the network lifetime are always big issues in the domain of WSN. To resolve these downsides, this work proposes an Energy Efficient Scheduling using Deep Reinforcement Learning (DRL) (E^2S-DRL) algorithm in WSN. E^2S-DRL contributes three phases to prolong network lifetime and to reduce network delay that is: Clustering phase, Duty-Cycling phase and Routing phase. E^2S-DRL starts with the clustering phase where it reduces the energy consumption incurred during data aggregation. It is achieved through Zone based Clustering (ZbC) scheme. In ZbC scheme, hybrid Particle Swarm Optimization (PSO) and Affinity Propagation (AP) algorithms are utilized. Duty cycling is adopted in the second phase by executing DRL algorithm. Here, the sensor node autonomously decides its sleep or wakeup time to transmit sensed data to the head node. From which, E^2S-DRL reduces the energy consumption of individual sensor nodes effectually. The transmission delay is mitigated in third (routing) phase using Ant Colony Optimization (ACO) and FireFly Algorithm (FFA). E^2S-DRL is modeled in the Network Simulator 3.26 (NS3) simulator. The results conquered are valuable in provisions of upcoming metrics including network lifetime, energy consumption, throughput and delay. From this evaluation, it is proved that E2S-DRL reduces energy consumption, delay up to 40% and enhances throughput and network lifetime up to 35% compared to the existing Time Division Multiple Access (cTDMA), Distributed Random Allocation (DRA) and improved Artificial bee colony (iABC) methods. The weather data will be stored in the database for further action. This study was conducted in the biodiversity-rich Usambara forest reserve in Tanzania. Timely collected environmental data will help other researchers to predict wildfire since the temperature rise can cause fire outbreak. Finally, to evaluate the performance of the proposed system using the following metrics namely network lifetime, energy consumption, throughput and delay.

DECLARATION

I, Ramadhani Sinde do hereby declare to the Senate of Nelson Mandela African Institution of Science and Technology that this dissertation is my own original work and that it has neither been submitted nor being concurrently submitted for degree award in anyother institution.

Ramadhani Sinde [illegible] 30.04.2020

Candidate Name and Signature **Date**

The above declaration is confirmed

Prof. Karoli N. Njau [illegible] 30.04.2020

Name and Signature of Supervisor **Date**

Dr. Shubi Kaijage [illegible] 30.04.2020

Name and Signature of Supervisor **Date**

COPYRIGHT

CERTIFICATION

The undersigned certify that they have read and hereby recommend for submission to the Nelson Mandela Institution of Science and Technology (NM-AIST) a dissertation titled Energy Efficient Wireless Sensor Network for Temperature and Relative Humidity Monitoring in the Forests, in fulfillment of the requirements for the degree of Doctor of Philosophy in Information and Communication Science and Engineering of the Nelson Mandela African Institution of Science and Technology.

Prof. Karoli N. Njau [illegible] 30.04.2020

Name and Signature of supervisor **Date**

Dr. Shubi Kaijage [illegible] 30.04.2020

Name and Signature of supervisor **Date**

ACKNOWLEDGEMENTS

First of all, I am thankful to the almighty God, who gave me guidance and wisdom to complete this thesis. It is a great pleasure to thank many people that supported me during my Ph.D studies. Words will never be enough to express my gratitude to my supervisor Dr. Shubi Kaijage for his constant and precious guidance throughout my doctoral program, for the time spent on discussing and accurately reviewing all the research work of this thesis. Besides his scientific competence, I enjoyed a lot their positive attitude to make a friendly work atmosphere. I learned much more than scientific concepts from him, and his enthusiasm has been fundamental to overcome all the difficulties which arose in these three years. My sincere gratitude is also due to Prof. Karoli Njau for his supervision and valuable suggestions throughout this study. To all of my colleagues, both staff and students, in the School of Computational and Communication Science and Engineering (CoCSE) at Nelson Mandela African Institution of Science and Technology (NM-AIST). You have made my time in the school that much more enjoyable. I feel honoured to have been a part of this school. Thanks also Canadian Queen Elizabeth II Diamond Jubilee Scholarships program (QES-AS) for giving opportunity to collaborate with Professors and students from Carleton university. Also special thanks go to all people facilitated my four months stay in Ottawa, Canada as follows: Prof. Michael Brklacich, Prof. Michel Rod, Prof. Onita Basu, Prof. Paul Mkandawile, Prof. Pius Adesanmi (R.I.P), Dr. Mwemezi Rwiza, Stephanie Pineau, Heloise Emdon, research mentors, research placement partners, research module leaders, research support staff, research funders, all international and Carleton QES steering committee. Thanks also for all the opportunities to meet important people in the research field, I am really grateful for my four months research stay in Ottawa. I would like to thank both the internal and external reviewers of this thesis, for their careful reading and for their valuable comments and suggestions. Unreserved thanks go to African Development Bank (AfDB) for giving me a scholarship of three years to pursue my studies at NM-AIST. Also a special thanks go to all AfDB staff at NM-AIST, Mr. Julius Lenguyana, Ms. Victoria Ndossi and Mr. Humprey Robert. Sincere thanks go to Prof. Dr. Andreas Winter from University of Oldenburg that showed credit in this work and my skills. He is a special person; I like his nice attitude to make me feel at ease every time we meet. I really appreciated the opportunity to collaborate with the Systems and Computer Engineering and CISCO research group headed by Prof. Mohamed Ibnkahla at Carleton University. I found very valuable colleagues with whom I shared work and entertainment. In particular, it was a great pleasure to work with Dr. Ayman

Sabbah, Dr. Zied Bouida, Anastazia, Karim, Abdullah and Ismael; I enjoyed our long discussions that always pointed out more questions than answers. I want to thank the University of Ghana, in particular the Computer Science department, I think that all of you are special persons. I want to especially thank Dr. Ferdinand Katsriku and Dr. Jamal with whom I had the chance to work with, and they gave me many useful and meaningful advices. Of course I cannot forget Dr. Benjamin Ofori for giving me opportunity to present my research at the institute of environmental sciences and receive massive of contribution from environmental expertise. Moreover, I want to thank the DEMIS project, and the people who facilitated my research visit at the Oldenburg University in Germany (August 2016, September 2017, October 2018), University of Ghana in Accra (February 2017) and Pedagogica´ University in Maputo (August 2018): Prof. Dr. Jorge Marx Gomez, Prof. Karoli Njau, Prof. Uranio Stefane Mahanjane, Dr. Dan Nukpezah, Barbara Rapp, Julia-Katharina Franke.

Last, but not least, I am very grateful to the DEMIS project funded by DAAD for their financial contributions to attend and present at the conferences and workshops. I could not have done this without all of you. Thank you!!

DEDICATION

I dedicate this project to God Almighty. I also dedicate this work to my mother, my family, colleagues and friends for their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS AND SYMBOLS

ACO	Anti-Colony Optimization
ADC	Analog Digital Converter
AFASR	Adaptive Forward Area Selection Routing Protocol
AP	Affinity Propagation
API	Application Programming Interface
BS	Base Station
CDS	Multi-Connected Dominated Set
CH	Cluster Head
CM	Cluster Member
CPU	Central Processing Unit
cTDMA	Cooperative Time Division Multiple Access
DDAR	Differentiated Data Aggregation Routing
DRA	Distributed Routing Algorithm
DRL	Deep Reinforcement Learning
EBRP	Energy Balance Routing Protocol
EEDC	Energy-Efficient Duty Cycling
EESSM	Energy-Efficient Sleep Scheduling Mechanism
ESQG	Multi-Energy-Aware scheduling with Quality Guarantee
FDS	Forward Area Division
FFA	Firefly Algorithm
FLS	Fuzzy Logic System
GA	Genetic Algorithm
GSA	Gravitational Search Algorithm
HB	Honey Bee Algorithm
iABC	improved Artificial Bee Colony
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
J-SIM	Java-based Simulation
LDC	Low Duty Cycle
LEACH	Low Energy Adaptive Clustering Hierarchical
LR-WPAN	Low-rate Wireless Personal Area Network
MANET	Mobile Ad Hoc Network

MDC	Mobile Data Collector
MDG	Mobile Data Gather
MH-GEER	Multi-Hop Graph based approach for Energy Efficient Routing Protocol
MOEA	Multi-Objective Evolutionary Algorithm
MPACO	Multiple Pheromone Ant Colony Optimization
NETSIM	Network Based Environment for Modeling and Simulation
NS-2	Network Simulator 2
NS-3	Network Simulator 3
OMNET++	Voice user interface
OPNET	Optimized Network Engineering Tools
PSO	Particle Swarm Optimization
QoS	Quality of Service
RAM	Random-access memory
REAL	REalistic And Large
REST	REpresentational State Transfer
SDMA	Space Division Multiple Access
E^2S-DRL	Efficient Scheduling using Deep Reinforcement Learning (DRL)
TDMA	Time Division Multiple Access
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
ZbC	Zone based clustering

CHAPTER ONE

INTRODUCTION

1.1 Background of the problem

There have been many changes in the field of wireless technology in the past decade, one of which is undoubtedly the introduction of Wireless Sensor Network (WSN). Over the past few years, WSN has become widely accepted in several applications including security surveillance of battle-fields, environmental monitoring, wildlife habit monitoring, fire detection, etc. (Verma *et al.*, 2018).

The WSN is useful productivity tool, with energy awareness and routing protocols becoming hotter research area for many researchers. The ability of WSN having self and auto-configuration of the nodes and operating without any human intervention for years provided the opportunity to increase productivity. Wireless sensor network is ease to deploy and more flexible. This wide acceptance of wireless sensor network originated from its ability to provide the real-time data collection from remote areas and low production cost regardless of underlying networking technologies, allowed development of Internet of Things (IoT) and designing of several protocols to enhance routing of packets from source node to destination at a minimum level of energy consumption.

Wireless Sensor Network consists of various sensor nodes that are capable of sensing and communicating the data in the monitoring area. The issues of reducing transmission delay and maximizing network lifetime network are important to research topics in the domain of WSN (Liu *et al.*, 2017). In order to reduce energy consumption and maximize network lifetime, two processes are included in WSN that are clustering and duty cycling. Cluster formation is one of the essential approaches in WSN to reduce energy consumption.

Clustering is a method of combining sensor node into clusters with cluster head. A bio-inspired algorithms such as hybrid PSO and Gravitational Search Algorithm (GSA) (Morsy *et al.*, 2018), Honey Bee (HB) algorithm (Ahmad *et al.*, 2018), Genetic Algorithm (GA) (Yuan *et al.*, 2017) and Multi-Objective evolutionary algorithm (MOEA) (Elhabyan, Shi, & St-Hilaire, 2018) are used to select optimum cluster head in WSN. In hybrid PSO and GSA, a cost function is computed using parameters such as closeness to the cluster head and residual energy. The cost function is used to allocate next hop for each cluster head to balance the load among cluster heads. In HB algorithm, a set of cluster head is selected from current nodes in the network and forms the clusters based on the cluster heads. In GA, the self-organizing network clustering method is proposed that affords framework for dynamically optimize wireless sensor node clusters. In MOEA, an efficient cluster head is selected using two evolutionary algorithms such as GA and Multi-Objective PSO. Fuzzy based cluster head selection is used by (Mehra *et al.*, 2020) to elect best cluster head among sensor nodes in WSN. Here, the eligibility index is calculated for each sensor node for the election of cluster head role.

Duty Cycling is one of the significant methods to reduce energy consumption in WSN. Energy-aware scheduling with Quality Guarantee method (ESQG) method is utilized (Xiao *et al.*, 2019) reduces sensor nodes energy consumption. Multi-Energy-Aware scheduling with Quality Guarantee (ESQG) method dynamically adapts each sensor node awakening frequency. Each node dynamically turns on through node awakening probability and node importance. A distributed delay efficient data aggregation scheduling is used (Kang *et al.*, 2017) for duty-cycled WSN. Here, the fast aggregation scheduling algorithm is used to schedule the sensor nodes. Delay aware tree construction and scheduling are proposed (Le *et al.*, 2018) for duty cycled WSN. Connected dominated set (CDS) tree is constructed for efficient scheduling. A distributed real Time Division Multiple Access (TDMA) scheduling is used (Ahmad & Hanzálek, 2017) to improve network lifetime in WSN. In this, the graph theory based algorithm is operated to schedule each sensor node.

Routing is one of the significant processes to reduce delay during data transmission. Energy efficient scheduling based cross layer based adaptive routing protocol is used (Singh & Verma, 2017) for WSN. The proposed routing protocol comprise of two models that includes the network model and radio energy model. Energy efficient clustering using multilevel routing is introduced (Muthukumaran *et al.*, 2018) for WSN. Three different routing mechanisms are hierarchical routing using cluster identification, hierarchical routing using multi-hop and multi-level. Multi-Objective hybrid routing algorithm is proposed (Kulkarni *et al.*, 2018) for WSN. In this, two approaches are used that are scalarization approach and lexico-graphical approach. Forwarding Area Division (FDS) and Adaptive Forwarding Area Selection Routing (AFASR) protocol (Hong *et al.*, 2018) is proposed for WSN. Forwarding area division ensures all nodes within the same forwarding sub area. From the aforesaid studies, it is seen that WSN still suffer from succeeding issues that are discussed as follows:

(i) Energy consumption is still a major problem in the WSN. Most of the works concentrating on clustering to reduce energy consumption. However, it lacks in electing optimal cluster head. Thus reduces the data aggregation efficiency.

(ii) Energy consumption of the sensor node is high due to the ineffective duty cycling process.

(iii) Lack of parameter consideration in scheduling leads to frequent dead of sensor nodes in the network.

(iv) The network delay is high due to distance and energy based path selection.

1.1.1 Structure of a sensor node

The wireless sensor network made of hundreds or thousands of sensor nodes, which are the fundamental unit of the system. As shown in (Fig. 1), a node constitutes a sensing unit, a processing unit, a wireless communication unit, and a power unit. Also, it has optional parts such as a position finding system, mobilizer, and a power generator, which can be energy harvest system.

The sensor unit is made up of two other subunits: sensors and analog to digital converters (ADCs) (Akyildiz *et al.*, 2002). The sensor unit is responsible for collecting environmental parameters such as pressure, relative humidity of the atmosphere, moisture and air temperature. These data then converted to digital input by using the analog-digital converter (ADC). The operation and control of the sensing activities, saving, coping with sensory data with its node and the binary information transferred from neighborhood nodes is carried out processing unit.

The wireless communication unit comprises of transmitter and receiver for sending and receiving information data respectively. Each component of the sensor node must be powered for it to operate. The power to run the nodes' units is generated by the power unit, which is one of the essential parts of the sensor network.

The power unit is a usual battery, and it can also be connected to the external power generator to increase the lifetime of the system. Wireless Sensor Networks (WSNs) face several challenges such as Quality of Service (QoS), energy conservation, fault tolerance and secure routing (Halawani & Khan, 2010). Other problems of WSNs are network dynamics, coverage, bandwidth allocation, connectivity, sensor network topology, environment, production costs, hardware constraints, transmission media, node deployment, the power consumption, and so forth (Halawani & Khan, 2010).

The deployment of the sensor node to the forest which is enormous and hostile area is an essential concern due to its influence on cost and the network capability of WSN. The proper amount of sensor nodes should be implemented to provide adequate coverage of a field. The focus of this study is the deployment issue and related concerns such as power consumption, coverage, connectivity, and energy efficiency, which have a high impact on the performance of WSNs.

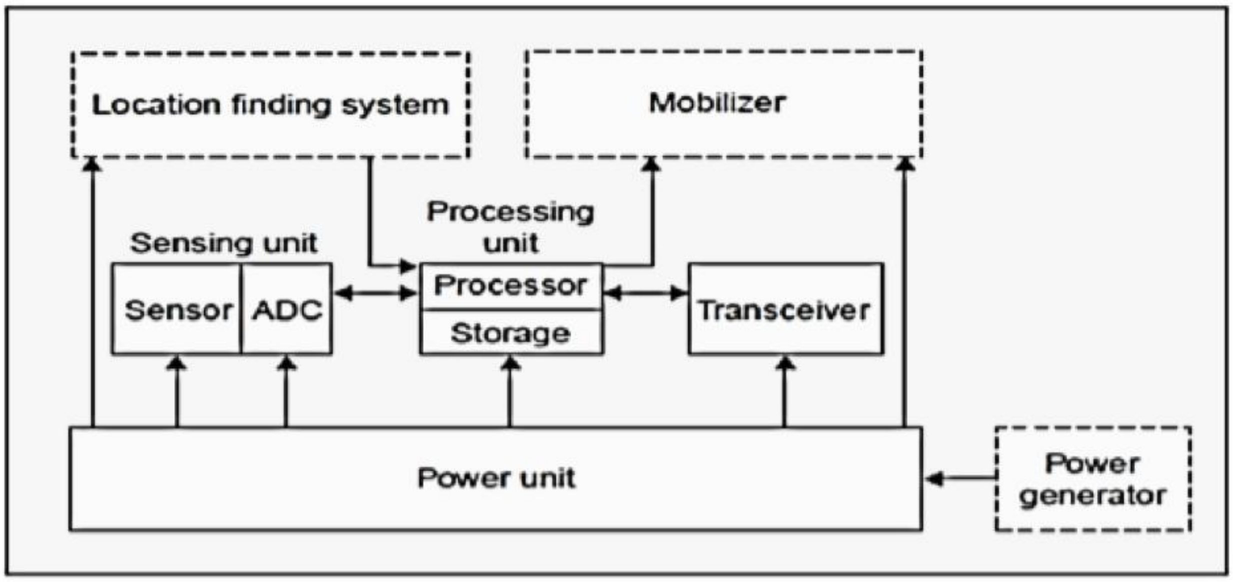

Figure 1: The structure of sensor node (Deng, 2014)

1.1.2 Comparison between WSN and traditional network

The WSN differ from traditional network in many ways including the following:

(i) There is no global addressing. The classical IP-based protocols cannot be applied in WSN

(ii) In WSN, the multiple nodes may generate same data within the vicinity of a phenomenon. Such redundancy needs to be exploited by routing protocols to improve bandwidth utilization and minimize energy consumption.

(iii) In WSN, energy management should be considered carefully, since the sensor nodes are constrained in terms of (a) Transmission power, (b) Processing power, (c) Storage and (d) On-board energy.

(iv) All applications of WSN require the flow of sensed data from multiple sources to particular destination (sink node). Therefore WSN is considered as a multiple-source single destination network.

Table 1: Comparison between WSN and traditional networks

Traditional Networks	**Wireless Sensor Networks**
General-purpose design; serving many applications	Single-purpose design; serving one specific application
Typical primary design concerns are network performance and latencies; energy is not a primary concern	Energy is the main constraint in the design of all nodes and network configuration
Networks are designed and engineered according to plans	Deployment, network structure, and resource use are often without planning(ad-hoc)
Devices and networks operate in controlled and mild environments	Sensor networks often operate in environments with harsh conditions
Maintenance and repair are common and networks are typically easy to access	Physical access to sensor nodes is often difficult or even impossible
Component failure is addressed through maintenance and repair	Component failure is expected and addressed in the design of the network
Obtaining global network knowledge is typically feasible and centralized management is possible	Most decisions are made localized without the support of central manager

1.1.3 Characteristics of wireless sensor nodes

Each sensor node has transmission area coverage and sensing area coverage based on its communication range and sensing range respectively. In WSN, a node has the limitation on both node's communication ability and sensing ability due to the communication and sensing range. A Wireless Sensor Nodes have following characteristics which make it a hot area of research:

(i) **Dense sensor node deployment**: The number of sensor nodes in WSN is usually higher compared to those in MANET (Almazeed & Mohamed, 2013; Chlamtac *et al*., 2003; Hoebeke *et al*., 2004), they are generally densely deployed in an area to be monitored (Pan *et al*., 2005).

(ii) **Limited energy resources**: Wireless sensor nodes are powered by small batteries. When theyare deployed in a harsh environment like the forest, where it would be very difficult orimpractical to replace or recharge the nodes batteries (Chow, 2009; Kosunalp & Cihan, 2017; Sheikhi *et al.*, 2019).

(iii) **Self and auto-configuration of nodes**: Sensor nodes could automatically configure to forma network when they are even randomly deployed without careful planning (Chetan & Potluri, 2009).

(iv) **Frequent Topology Change**: In WSN, there are some factors including node failure, channelfading or energy depletion which would lead to the topology changes (Alajlan, Dasari, Nossire, Elleithy & Pande, 2012).

(v) **Coverage Area and Data Redundancy**: In most applications, wireless sensor nodes are denselydeployed in remote areas. Therefore, there is a possibility more than one sensor nodeis monitoring a sensing region. Thus, the data sensed by multiple sensor nodes to havea certain amount of correlation or redundancy (Choudhuri & Das, 2019; Kanta & Prasad, 2015; Qin & Chen, 2018).

(vi) **Application Specific Nodes**: In WSNs, the sensor nodes are designed for a specific application. Therefore, the design requirements of a sensor network might change based on the condition of the application (Datta & Kundu, 2008; Tchakonté *et al.*, 2020).

1.1.4 Classification of sensor nodes

The WSNs may consist of many different types of sensors such as low sampling rate magnetic, seismic, infrared, thermal, visual, acoustic and radar, which can monitor a wide variety of ambient conditions that include the following (El Khediri *et al.*, 2011):

(i) Temperature

(ii) Humidity

(iii) Vehicular movement

(iv) Pressure

(v) Soil moisture

(vi) Level noise

(vii) Mechanical level of stress on attached objects, and

(viii) The current characteristics such as speed, direction, and size of object

Table 2: Classification and examples of sensors

Type	**Examples**
Temperature	Thermistors, thermocouples
Humidity	Capacitive and Resistive sensors, hygrometers, MEMS - based humidity sensors
Optical	Photodiodes, phototransistors, infrared sensors, CCD sensors, photovoltaic etc.
Pressure	Pressure gauges, barometers, ionization gauges
Flow	Anemometers, mass air flow sensors
Position	GPS, ultrasound-based sensors, infrared-based sensors, inclinometers
Radiation	Ionization detectors, Geiger-Mueller counters
Acoustic	Piezoelectric resonators, microphones
Motion, vibration	Accelerometers, gyroscopes, photo sensors
Mechanical	Strain gauges, tactile sensors, capacitive diaphragms, piezoresistive cells
Chemical	pH sensors, electrochemical sensors, infrared gas sensors
Electromagnetic	Hall-effect sensors, magnetometers

1.1.5 Types of Wireless Sensor Network Architecture

The application layer, transport layer, network layer, data link layer and physical layer of OSI model are the only layers applicable in the wireless sensor network (Abed *et al.*, 2012). The protocol stack consists of the application layer, transport layer, network layer, data link layer, physical layer, power management plane, mobility management plane, and task management plane (Akyildiz *et al*., 2002).

The application layer is responsible for the different application software which developed depending on the sensing tasks of the nodes. The transport layer responsible for the flowing of data is if the sensor network application requires it. The sensory data which are supplied by the transport need to be routed and the network layer responsible for it. Data link layer take care of MAC protocol which is responsible for collision avoidance of data from the neighbors' broadcast also considering the noise from the environment and the mobility of the sensor nodes. The datalink layer deals with Multiplexing of datastreams, data frame detection, medium access and error control. It ensures reliable point-to-point and point-to-multipoint connections in a communication network. The physical layer is responsible for the communication media, transmission techniques of the data including Modulation, etc. Task management, mobility plane, and the power management plane are used to monitor and lowering the overall power consumption, task distribution among the sensor nodes, and mobility of the nodes in the network. A transceiver may be switched off after receiving a signal from one of its neighbors, and this avoids a sensor node from receiving duplicated messages.

Sometimes, the powerlevel of the sensor node goes down, and therefore node should broadcast to the nearby nodes its status and that is not capable of routing or forwarding the messages. Only small power is reserved for sensing purpose (Akyildiz *et al*., 2002). All these are controlled by the power management plane. The task management plane is responsible for balancing and scheduling the sensing duties assigned to a particular area. Since, it is not necessary for all nodes in the field to perform the sensing task at the same time, which means depending on the power level some nodes perform the task more than others in the same network.

The importance of task management places has been proved in their ability to overcome selfishness behavior of the sensor nodes, make them work together in a power saving mode, route data in a mobile sensor network, and share resources between the nodes. Without task management planes, each sensor node will work individually and this is not an effective way of prolonging the network lifetime of the network.

The mobility management plane is responsible for identifying and recording the movement of sensor nodes in the network. Therefore, the sensor nodes can keep track of their neighbors and the route back to the user is always maintained. The importance of a node to know its neighbor nodes helps it balance their powers and control task usage (Akyildiz *et al*., 2002).

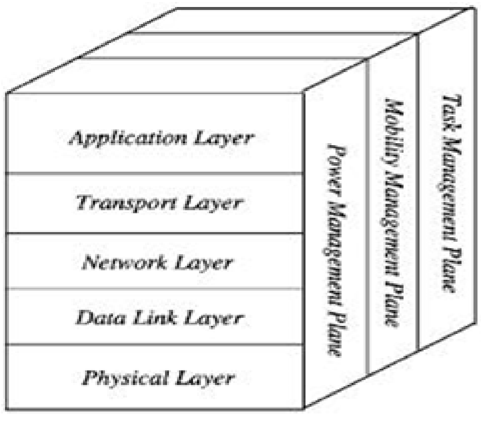

Figure 2: The architecture of wireless sensor network (Akyildiz *et al*., 2002)

Table 3: Difference of architectures OSI between WLAN and WSN (Kuorilehto *et al.*, 2005)

OSI Model	WLAN	Wireless sensor network
Application layer	Application programs	WSN Application
Presentation layer	Middleware	WSN Middleware
Session layer	Socket API	WSN Transport protocols
Transport layer	TCP/UDP	WSN Transport protocols
Network layer	IP	WSN routing protocols
Data link layer	WLAN Adapter & device driver	Error control
	WLAN MAC protocols	WSN MAC protocols
Physical layer	Transceiver	Transceiver

1.1.6 WSN as monitoring system

From the (Fig. 14) the following terminologies can be described as follows:

(i) **Sensor Field** the area in which the nodes are deployed for sensing and communication tasks. In this work, a sensor field is the forest.

(ii) **Sensor nodes** are the heart of the network. They perform sensing, collecting and routing of data back to a sink. In this case temperature and relative humidity sensors are used to collect sensor environmental data from the forest.

(iii) **Sink** are so known as BS is a special node for specific task of receiving, processing and storing data from the sensor field. It serves to reduce the total number of messages that need to be sent, hence reducing the overall energy consumption in the network. Sink can also be regarded as data aggregation point. In the network sink can be located inside or outside the sensor field, also in one WSN a single or multiple sinks can be used. Sink node can be stationary or mobile. In this work, one stationary sink is used, which is stationary located at the center of the sensor field.

(iv) **Task Manager** is a centralized point of control, usually outside the sensor field of WSN. It is a powerful data processing and storage center and an access point for human interface. The task manager is a laptop, a workstation or a database. Data is streamed to this task manager via wired or wireless communication.

1.1.7 Motivation

The importance of conserving forests has been a massive motivation for this research. Forests play an essential role in preventing global warming by absorbing greenhouses gases and building sustainable societies. Forests have a variety of functions, such as land conservation, securing of water sources, control of climate change, and creation of natural environs essential to human existence and regulating the temperature of the atmospheric environment.

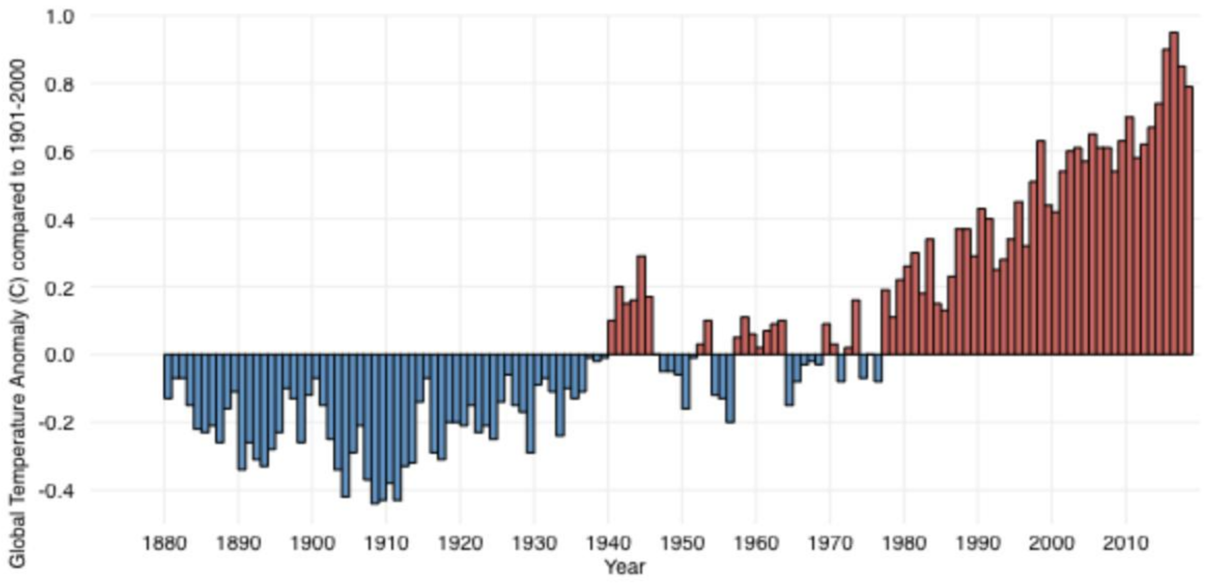

Figure 3: The average annual global temperatures as from 1880-2010

According to graph in (Fig. 3), it is clear that there is an increase in temperature above the average since 1940 and the situation becomes worst in the latest years. In this graph, the average temperature of the planet is represented by the zero line, whereas the blue colour shows the difference below the average and the red is above the average. There is a need to protect our forests for the basement of regulating temperature and reducing global warming. Several approaches including intensive researches to monitor forest environmental condition have been proposed.

The collection of forest data using WSN motivates more research in this field. WSNs consume a lot of energy during data transmission, although various algorithms have been developed to minimize energy consumption the problem is not adequately solved. Theaforesaid protocols are not concentrated on the optimal cluster head selection while routing the packets. Thus increases energy consumption and reduces the stability of the WSN. To resolve these shortcomings in existing communication protocols in WSN, E^2S-DRL approach constructs subsequent approaches:

(i) An efficient cluster head selection and clustering in WSN to reduce energy consumption during data aggregation.

(ii) To schedule the state of the individual sensor node to reduce the energy consumption.

(iii) To find an optimal path to deliver sensed data packet to sink without routing overhead and delay

1.2 Statement of the problem

There is an increase in temperature above the average since 1940 and the situation becomes worst in the latest years. Regardless of the negative impact of the global climate change, there is a shortage of weather information on real-time temporal and spatial environmental conditions of the forest that drive its health. Several measures have been proposed to monitor forest environmental conditions.

In recent years WSN technology has been used to monitor the forest environmental condition and this gives low-cost solution and real-time capturing for temperature, humidity, moisture, smoke and other parameters. The WSN faces the number of challenges including low bandwidth, low power and short battery lifespan. Since the sensor nodes are deployed in an inaccessible area and the sensors have limited batteries, which cannot be replaced in this situation. Careful management of power resources is needed to increase the lifetime of the WSN. Although most of the researches focus on lifetime enhancement of WSN, the problem is not adequately solved due to other factors like network delay, transmission range, hop count etc. which contribute to the energy consumption.

In order to reduce energy consumption and maximize network lifetime, two vital processes are included in WSN that are clustering and duty cycling. On the other hand, most of the researchers ignored the impact of temperature and relative humidity in a wildfire. They focus only on the detection of smoke which might not be acceptable since the existence of the smoke means the fire is about to start. Therefore it might be too late for the firefighter unit to extinguish the fire. In this research, the rise of temperature and humidity are considered very carefully as primary factors of the wildfire.

1.3 Rationale of the study

Wireless sensor network technology has been widely used in environmental monitoring, wildfire detection, structural health monitoring, pipeline monitoring, precision agriculture etc.

The WSN is useful productivity tool, with energy awareness and routing protocols becoming hotter research area for many researchers. The ability of WSN having self and auto-configuration of the nodes and operating without any human intervention for years provided the opportunity to increase productivity. Wireless sensor network is ease to deploy and more flexible. In a forests' environmental monitoring, WSN consists the following components: sensor field is the forest area in which the nodes are deployed for sensing and communication tasks. Sensor nodes are performing sensing, collecting and routing of data back to a sink. In this case temperature and relative humidity sensors are used to collect sensor environmental data from the forest. Sink are so known as Base Station (BS) is a special node for specific task of receiving, processing and storing data from the sensor field.Task Manager is a centralized point of control, usually outside the sensor field of WSN.

The WSN faces the number of challenges including low bandwidth, low power, and short battery lifespan. Since the sensor nodes are deployed in an inaccessible area and the sensors have limited batteries, which cannot be replaced in this situation. Careful management of power resources is needed to increase the lifetime of the WSN. Although most of the researches focus on lifetime enhancement of WSN, the problem is not adequately solved due to other factors like network delay, transmission range, hop count etc. which contribute to the energy consumption. In order to reduce energy consumption and maximize network lifetime, two vital processes are included in WSN that are clustering and duty cycling. On the other hand, most of the researchers ignored the impact of temperature and relative humidity in a wildfire. They focus only on the detection of smoke which might not be acceptable since the existence of the smoke means the fire is about to start. Therefore it might be too late for the firefighter unit to extinguish the fire. In this research, the rise of temperature and humidity are considered very carefully as primary factors of the wildfire.

Therefore, this study aimed at developing the energy efficient wireless sensor network temperature and relative humidity monitoring in the forests. This work proposes an Energy Efficient Scheduling using Deep Reinforcement Learning (DRL) (E^2S-DRL) algorithm in WSN. Efficient Scheduling using Deep Reinforcement Learning (E^2S-DRL) contributes three phases to prolong network lifetime and to reduce network delay that is: Clustering phase, Duty-cycling phase and routing phase. The E^2S-DRL starts with the clustering phase where it reduces the energy consumption incurred during data aggregation. It is achieved through Zone Based Clustering (ZbC) scheme. In ZbC scheme, hybrid particle swarm optimization (PSO) and Affinity Propagation (AP) algorithms are utilized. Duty cycling is adopted in the second phase by executing DRL algorithm. Here, the sensor node autonomously decides its sleep or wakeup time to transmit sensed data to the head node. From which, E^2S-DRL reduces the energy consumption of individual sensor nodes effectually. The transmission delay is mitigated in third (routing) phase using ant-colony optimization (ACO) and firefly algorithm (FFA).

1.4 Objectives

1.4.1 General objective

To develop energy efficient wireless sensor network for forest's temperature and relative humidity monitoring

1.4.2 Specific objectives

To answer research questions, the following research objectives were pursued:

(i) To identify requirement for energy efficient WSN

(ii) To develop a model of energy efficient WSN

(iii) To evaluate the developed model

1.5 Research questions

To better understand how WSN algorithms works to enhance network lifetime and reduce network delay, the research work was guided by the following research questions:

(i) What are the essential aspects or perspectives that are required to describe energy efficiency WSN?

(ii) How to describe a model for energy efficient WSN?

(iii) How to evaluate a proposed model for energy efficient WSN?

1.6 Significance of the study

There are several envisioned outputs and outcomes from this research, which are categorized in terms of expected effects on society and development, on research capacity building, and support to ongoing initiatives for global climate change and disaster management. Expected outputs are all focused on improving the livelihoods of semi-arid areas inhabitants and growth to the national economy. The best result for increasing the quality of forest environmental sensing with low energy consumption and low cost is what we are expecting from this research. The significance of this research includes:

(i) It will generally add up knowledge to the scientific body and can be used by other researchers in the field of environmental engineering, computer science and power opti-mization.

(ii) It will develop the general architecture that meets the strict efficiency and flexibility requirements of wireless sensor networks.

(iii) A sleep-wake up scheduler based Deep Reinforcement Learning will help to take decision about putting nodes in sleep and waking up on the basis of traffic pattern.

(iv) The research will assist to reduce the investment costs and man power; thus, enhance the growth of the industry, wildlife sector etc.

1.7 Delineation of this study

This research concerned with the use of energy efficient WSN to monitor temperature and relative humidity in the forests. This research uses WSN since it proved to be an effective requirement for the continuous monitoring of the hostile regions such as environment monitoring like sudden volcanic eruptions, forest fires, floods, etc. (Verma *et al*., 2018). The main aim of this research is to reduce energy consumption and network delay in the system during the operation. This work proposed the following approaches to reduce network delay and enhance the network lifetime:

(i) The sensor nodes are located in the corona field to aggregate data from all sensor nodes effectually and also we split coronas into zones to reduce energy consumption. To improve network lifetime, our work functioned based on three phases that are ZbC, duty cycling and routing.

(ii) The first phase reduces energy consumption during data aggregation through the ZbC scheme that comprises hybrid PSO and AP algorithm. Particle swarm optimization computes fitness function for following metrics energy factor and node degree. The energy factor is a combination of residual energy and distance between the source node and sink node.

(iii) The second phase enhances network lifetime through duty cycling. Duty cycling is performed using a Distributed Routing Algorithm that schedules each sensor node in distributed manner. Each scheduling slot considers three modes that are sleep, listen and transmit.

(iv) In the last phase, routing is performed to reduce data transmission delay by finding the best path between source and cluster head. Anti-colony optimization (ACO) algorithm is used to choose multiple paths between source and cluster head. Here, ACO algorithm considers the following metrics to compute fitness function such as residual energy, the distance between source and destination node, hop count and bandwidth. From the selected multiple paths, best path between source and cluster head is selected using firefly algorithm. Firefly algorithm considers succeeding metrics such as expected delay, packet delivery ratio and load.

CHAPTER TWO

LITERATURE REVIEW

2.1 Forest environmental monitoring systems

Forests play an essential role in preventing global warming by absorbing greenhouses gases and building sustainable societies. Forests have a variety of functions, such as land conservation, securing of water sources, control of climate change, and creation of natural environs essential to human existence.

Bayo *et al.* (2010) developed an early detection and monitoring of wildfire using low-rate wireless personal area network (LR-WPAN) system in the mesh topology. Their sys-tem designed to take measurements of different parameters from the different heights of tree depending on the relief of the forest. For the achievement of early detection of the wildfire (Bayo *et al*., 2010), proposed to collect some critical environmental data such as barometric pressure, temperature, light intensity, relative humidity, soil moisture, and smoke. In order to reduce power consumption, the author introduced the analog switches to disconnect the sensors nodes when the system is in sleep mode. This approach does not consider the randomness deployment of nodes with unknown locations and the size of the forest.

Lloret *et al.* (2009) designed an integrated fire detection system using a wire-less sensor network and wireless local area network (WLAN). The integrated system detects the presence of fire by using sensor nodes with IP-based camera. The sensor alarm will propagate to the server via the wireless network when the nodes discover a fire. The central server selects the closest wireless camera to the multisensor, and it sends a request for it to retrieve real-time figures from the region. Although this study integrates sensory data with the image the main advantage, still this system might not be compatible with the harsh environments like most the forests.

Aslan *et al.* (2012) developed a comprehensive framework for the use of wireless sensor networks for forest fire detection and monitoring. Their framework includes the wireless sensor network architecture, sensor deployment scheme, and clustering and communication protocols. The developed framework aims to detect a fire threat as early as possible. The researchers consider the factors that may affect the activity level of the network. They mentioned two factors as follows: the energy consumption of the sensor nodes and the environmental conditions.

Bouabdellaha *et al.* (2013) proposed a solution using WSN to detect fire on the forest of Oran city in Algeria. They used two concurrent detection systems: the Canadian system and the Korean system. They conducted real experiments using a sensor Test-bed based on MICA-Z plate-form. In their study claim that most of the existing research choose to simulate their proposed solutions instead of doing experiments in real test-bed environments.

Hefeeda and Bagheri (2009) presented the design and evaluation of wireless sensor network for forest fire detection. They presented forest fire modeling by analysing the Fire Weather Index (FWI) System. Their analysis helped to show how the different components can be used to design efficient fire detection systems. Also, they implemented a distributed algorithm to solve the coverage problem in WSN. They also claimed that the proposed algorithm achieves various coverage degrees at different subareas of the forest. Finally, they presented a simple data aggregation scheme based on the FWI System. The main contribution of this approach is to provide unequal monitoring quality of forest zones. The presented design also maximaze the network lifetime by using data aggregation scheme in which it delivers only data that is of interest to the application.

Gao and Huang (2015) constructed a multi-sensor based system with the 4G network to monitor fire in the forest. The system was designed based on Zigbee standard and 4G mobile communication. For the hardware, CC2530 was used to collect and transmit sensor data via the 4G network to the remote server. The use of multi-sensor with high data rate 4G network give early warning of the wildfire. The challenge of using the 4G network is that not all forest can have full coverage of high data rate mobile network.

Xu *et al.* (2018) demonstrated a use WSN framework to detect and monitor wildfire. This work uses a clustering routing protocol to minimize energy consumption of WSN and the activity-rate selection to enhance the detection speed of the forest fire.

Unfortunately, most of these studies consider the detection of wildfire Al-Habashneh *et al.* (2011), Antoine-Santoni *et al.* (2009), Aslan *et al.* (2012), Bernardo *et al.* (2007), Li *et al.* (2006), Pripužić *et al.* (2008), Sazak and Abdullah (2009) and Soliman *et al.* (2010) only a few types of research focus on the primary sources on the fire in the forest. This study proposes a collection of weather data from the forest such as temperature and humidity, since these two parameters will help to analyse and predict the occurrence of wildfire.

Díaz-Ramírez *et al.* (2012) proposed two WSN algorithms for forest fire detection. These are information fusion algorithms. The first one uses temperature, light, humidity sensors and a threshold method, whereas the Dempster-Shafer theory used in the second algorithm.

Dehwah *et al.* (2017), Dondi *et al.* (2008), Horng *et al.* (2014), Ibrahim *et al.* (2017), Noh and Kang (2011), Voigt *et al.* (2003) and Yue and Ying (2011), used solar power systems and a lithium-ion battery instead of using disposable batteries as power supplies. Some other researches by Jalali *et al.* (2012) evaluated the impact of solar power systems in WSNs.

Thayananthan and Alzranhi (2014) proposed the enhancement of energy conservation technolo-gies in WSN. In their method, they used a combination of routing protocols and solar power to enhance energy conservation. Their approach increases the lifetime of the sensor node and the battery life.

Ibrahim *et al*. (2017) and Frezzetti *et al.* (2015) presented the development of a solar energy harvesting mechanism for Wireless HART sensor node using photovoltaic (PV) cell array, controller, converter and UltraCapacitor (UC).

Baranov *et al.* (2016) presented a hybrid power supply to reduce power consumption of the node without degrading the sensing capabilities. In this study, they are integrated solar energy, wind and electrochemical carbon monoxide sensor.

Table 4: Comparison of WSNs and other wildfire systems (Alkhatib, 2014)

Comparison	**Human based observation**	**Satellite-based systems**	**Optical cameras**	**WSN**
Cost	Low	Very high	High	Medium
Efficiency and Practicality	Low	Low	Medium	High
Faulty Alarms repetition	Low	Low	Medium	Medium
Fire localizing accuracy	Low	Medium	Medium	High
Detection delay	Long	Very long	Long	Small
Fire behavior information	-	Yes	-	Yes
Can be used for other purposes	No	Yes	No	Yes

All the researches based on the solar systems can improve the lithium-ion battery life and ensure the business continuity of the system. But for the implementation of the solar power system still needs to redo the experimentation on the operation of the charge and discharge control to make sure the command and estimation of battery power accuracy fulfil the system requirements. Most of the researchers did not pay much attention to the temperature and relative humidity as sources of wildfire, rather their focuses based on the smoke. It might be too late and very risky to detect smoke in the forest as this is a premature level of uncontrolled fire in the forest. Therefore this work proposed to collect and analyze variations of environmental data. The collected environmental data will help to predict the wildfire by for example extensively rises in temperature.

This research intends to develop energy efficient and a real-time monitoring wireless sensor network for collecting temperature and humidity data in Forest. Most of them focus on wildfire monitoring only, do not consider the energy constraints of the sensor nodes, transmission and avoidance of data collision.

The importance of conserving forests has been a massive motivation for this research. It was observed that several monitoring systems in the forest do not focus on the energy constraints of the sensor nodes and the transmission delay. Three novel approaches are proposed in work to prolong the network lifetime and the transmission delay: Zone based clustering, deep reinforcement learning and optimum path selection for the data routing.

2.2 Energy awareness in WSN

Different approaches have been applied to reduce the energy consumption and prolong the network lifetime in WSN technology. These methods include energy serving (Rault *et al.*, 2014), renewable energy from other sources like wind, water etc, and energy harvesting such as solar power, mechanical, thermal, commercial energy harvesters etc. Energy harvesting is a promising technology for many WSNs applications which require less human intervention as it excludes the need to replace batteries. Seah *et al.* (2009) proposed the WSNs powered by ambient energy harvesting in which energy from environment is converted into electricity to power the nodes. All of these methods can increase the network lifetime. However, the current state of technology in energy harvesting is still unable to provide sustained energy supply to enable WSNs continuously. Moreover, the ability to harvest energy from the environment is highly dependent on many environmental factors which need further research to understand and exploit.

Due to the fluctuation with time of the energy during harvesting process causes the complexity to estimate the sleep-wakeup time of the node. In this case, it is very difficult to ensure that the next-hop node is awake to receive a packet. The uncertainty in how long it takes a node to harvest enough energy before it can function again makes existing sleep-wake scheduling schemes for WSNs unusable since a node may not have harvested sufficient energy at the scheduled wakeup time (Seah *et al.*, 2009).

Table 5: Comparison of WSNs battery operated and with energy harvesters

	Battery operated WSNs	WSNs with Energy Harvesters
Goal	Throughput and latency are usually traded off for longer network lifetime	Longer network lifetime achieved by supplementing battery power with harvest energy
Protocol design	Sleep-and-wakeup schedules can be determined precisely	Sleep-and-wakeup schedules can be determined if future energy availability is correctly predicted
Energy model	Energy model is well understood	Energy model can be predicted to high accuracy

2.2.1 Energy conservation scheme in WSN

Energy conservation scheme Anastasi *et al.* (2009) and Rault *et al*. (2014) can be divided into three methods duty cycling, data-driven and mobility based. Figure 4 summarizes the several approaches used to conserve the energy of the sensor nodes.

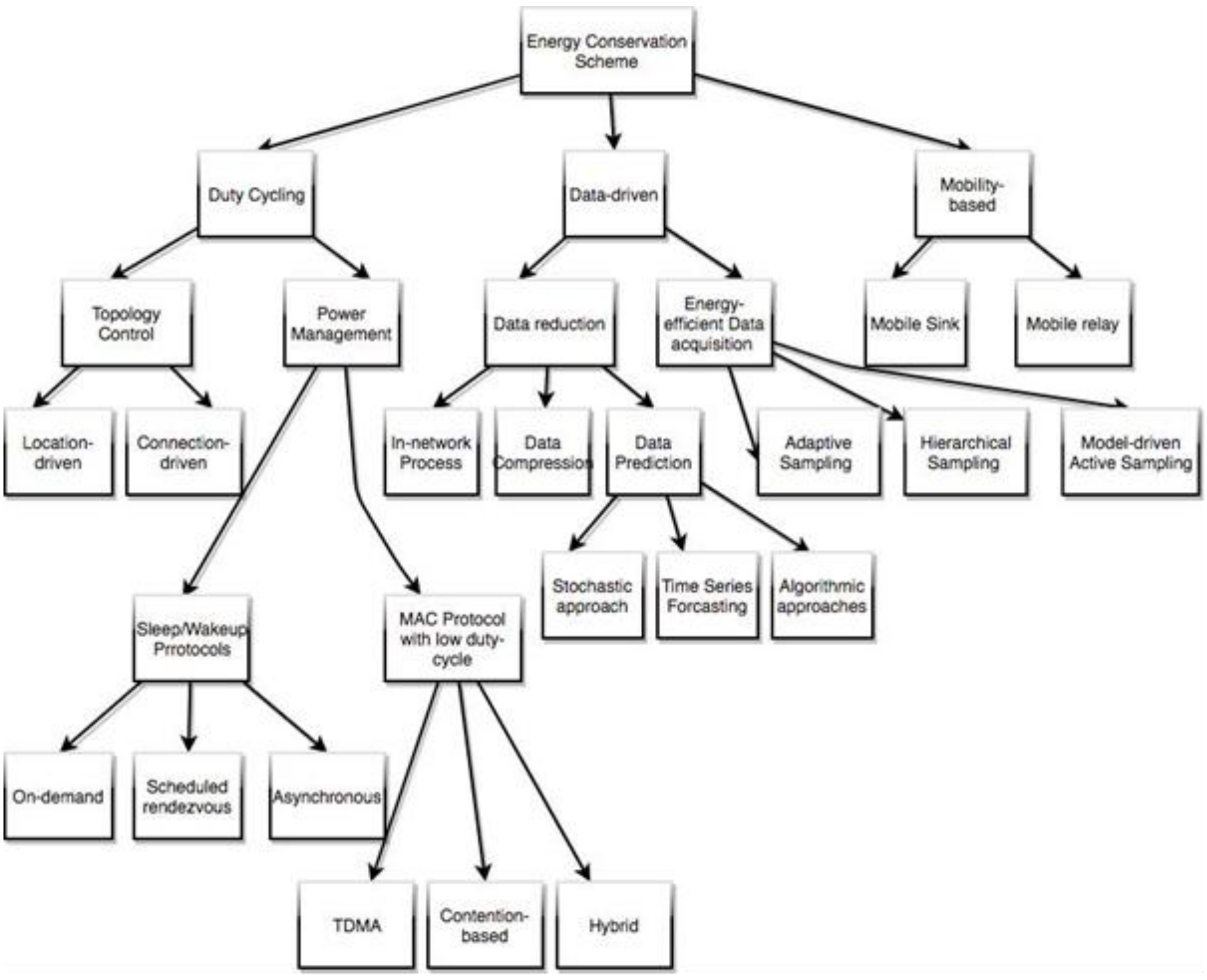

Figure 4: Energy conservation scheme in WSN (Anastasi *et al*., 2009)

2.2.2 Routing protocols in wireless sensor network

Routing is a way or mechanism for finding a communication path between the source and the destination node. Due to the nature of WSN compared to the traditional network, the routing protocols face several challenges. The following are challenges of routing protocol and design that affect routing process in WSNs (Al-Karaki & Kamal, 2004; Singh & Singh, 2016).

(i) **Nature of the node** - In WSN, sensor nodes are divided into two groups, homogeneous in which all nodes in the network have the same transmission range, sensing range, battery life, processing power, and all other capabilities (Al-Karaki & Kamal, 2004). The heterogeneous nodes have different capabilities (Al-Karaki & Kamal, 2004). Several architectures in WSN assume that nodes are stationary in the network, although in several applications mobility of BS and other nodes is necessary (Ye *et al.*, 2002).

(ii) **Scalability** - In WSN, sensors nodes are deployed on the orders of hundreds or thousands to monitor a given field. In some applications, the numbers of nodes may reach an extreme value of millions. A routing protocol must be able to work with a large number of the nodes and also utilize the high density of the sensor nodes. In a given region of less than 10m in diameter, the density can range from few nodes to the few hundreds. According to Ke *et al.* (2016), the network density μ(R) in terms of number of nodes per nominal coverage area can be calculated as follows:

$$\mu(\mathrm{R}) = \frac{N\pi R^2}{A} \qquad (1)$$

Where by:

N - The number of scattered sensor nodes in a region of area A

R - Radio transmission range (nominal range)

(iii) **Node deployment -** The application of WSN influences the methods of sensor node deployment. In WSN the performance of routing protocol is highly affected by the deployment of nodes. The deployment is either deterministic (manual) or self-organizing (random) (Ke *et al.,* 2016). In deterministic situations, the sensors are manually placed, and data is routed through pre-determined paths. Whereas in self-organizing systems, the sensor nodes are scattered randomly creating an infrastructure in an ad hoc manner.

(iv) **Coverage -** The coverage area is essential during the routing protocol design since each node prevails a particular view of the environment. Both accuracy and the transmission range limit the environmental view of the sensor node (Hoblos *et al.,* 2000).

(v) **Energy consumption -** Energy considerations highly influenced by the creation of an infrastructure, and the process of setting up the routes (Halawani & Khan, 2010). In WSN, the transmission power of a wireless radio is proportional to the squared distance between a transmitter and the receiver, or even higher order in the presence of obstacles, multi-hop routing will consume less energy than direct communication. The sensor node has three main units: communication, sensing and processing unit. Among these three units, communication which is made of transmitter and receiver causes more battery energy depletion compared to other units (Shen *et al.*, 2001). This is due to the fact that a node spends most of its battery power on transmitting and receiving data (Raghunathan *et al.*, 2002).

(vi) **Quality of service** - The suitable routing protocol should be able to give a QoS need for a particular application of WSN. In WSN, the parameters of QoS include bandwidth, jitter, delivery delay, throughput, etc. For example, some applications such as object tracking and detection require low transmission delay for the time-sensitive data. While other applications require high throughput, e.g., multimedia networks (Lin *et al.*, 2011).

(vii) **Number of nodes** - The routing protocol must stable to work with a massive number of nodes in the network (Hoblos *et al.*, 2000). Usually, in WSN, sensor nodes are deployed in thousands or even millions to cover a sensor field. A suitable routing protocol makes each node maintain a global knowledge of network topology when the number of nodes is extensive (Al-Karaki & Kamal, 2004).

(viii) **Application** - The sensor networks are application specific, and the nodes are designed depending on the particular application of WSN (More & Raisinghani, 2015). Therefore, the design requirements of a sensor network might change based on the need application.

(ix) **Fault tolerance** - Fault tolerance is the ability to maintain WSN functionalities without any interruption due to sensor node failures (Hoblos *et al.*, 2000). If in the WSN sensor node fails due to lack of power, environmental interference or has any type of physical damage, MAC and routing protocols must accommodate the formation of alternative links in such a way that a node failure should be accommodated and not affect the overall performance of the WSN (Sohrabi *et al.*, 2000; Woo & Culler, 2001). A study Hoblos *et al.* (2000) is modeled on the fault tolerance or reliability $R_k(t)$ using Poisson distribution as follows:

$$R_k(\mathrm{t}) = e^{-\lambda}k^{t}\beta \qquad (2)$$

Where by:

λ - Failure rate of sensor node k

t - Time period

This model shows the probability of avoiding a network failure in the time interval of $(0, t)$.

(x) **Transmission media**- In WSN, communicating sensor nodes are linked by the wireless media. Radio, infrared or optical media can form these links. The choice of transmission media should be made worldwide to enable global operation of WSNs. The industrial, scientific and medical (ISM) band also known as unregulated frequencies is a good option for the communication links between sensor nodes. The ISM offer license-free communication in most countries. The International Table of Frequency Allocation contained in Article S5 of the Radio Regulations (Volume 1), species some frequency bands that may be made available for ISM applications.

Table 6 list frequency bands, in which some of them are being used in most of the wireless devices such as cordless phones and wireless local area networks (WLANs). For sensor networks, a small-sized, low-cost, ultralow power transceiver is required. According to Porret *et al.* (2000), certain hardware constraints and the trade-off between antenna efficiency and power consumption limit the choice of a carrier frequency for such transceivers to the ultrahigh frequency range.

The use of 433 MHz industrial, scientific and medical band was proposed for Europe while the 915 MHz ISM band for North America. The design of transceiver for these bands was discussed by Lin *et al.* (2016). ISM bands have several advantages including large spectrum allocation, free radio and their availability globally. These bands are free to use and not bound to particular standard, therefore can be implemented strategically to save energy in WSN. Shih *et al*. (2001) use 2.4 GHz transceiver for Bluetooth standard whereas in Woo and Culler (2001) used transceiver of a single RF channel operating at 916 MHz. Shen *et al*. (2001) proposed WINS architecture which also uses radio links for communication.

Another possible mode of internode communication in sensor networks is by infrared. Infrared communication is license-free and robust to interference from electrical devices. In today's technology many electronic devices such as Personal Digital Assistance (PDA), laptops, smartphone, tablets use data association via infrared since transceivers for infrared are cheaper and easier to build. The main drawback though is the requirement of a line of sight between sender and receiver. Therefore infrared a reluctant choice for transmission medium in the sensor network scenario.

Table 6: Frequency bands available for ISM applications

Frequency band	Center frequency
6765.00–6795.00 kHz	6780.00 kHz
13,553.00–13,567.00 kHz	13,560.00 kHz
26,957.00–27,283.00 kH	27,120.00 kHz
40.66–40.70 MHz	40.68 MHz
433.05–434.79 MHz	433.92 MHz
902.00–928.00 MHz	915.00 MHz
2400.00–2500.00 MHz	2450.00 MHz
5725.00–5875.00 MHz	5800.00 MHz
24.00–24.25 GHz	24.125 GHz
61.00–61.50 GHz	61.25 GHz
122.00–123.00 GHz	122.50 GHz

2.2.3 Classification of routing protocols in WSN

In WSN, routing protocols can be classified into five groups, based on the way of establishing routing paths, according to the initiator of communications, according to the network structure, based on protocol operation, and according to how a protocol selects the next hop on the route to forward the packets (Singh & Sharma, 2015).

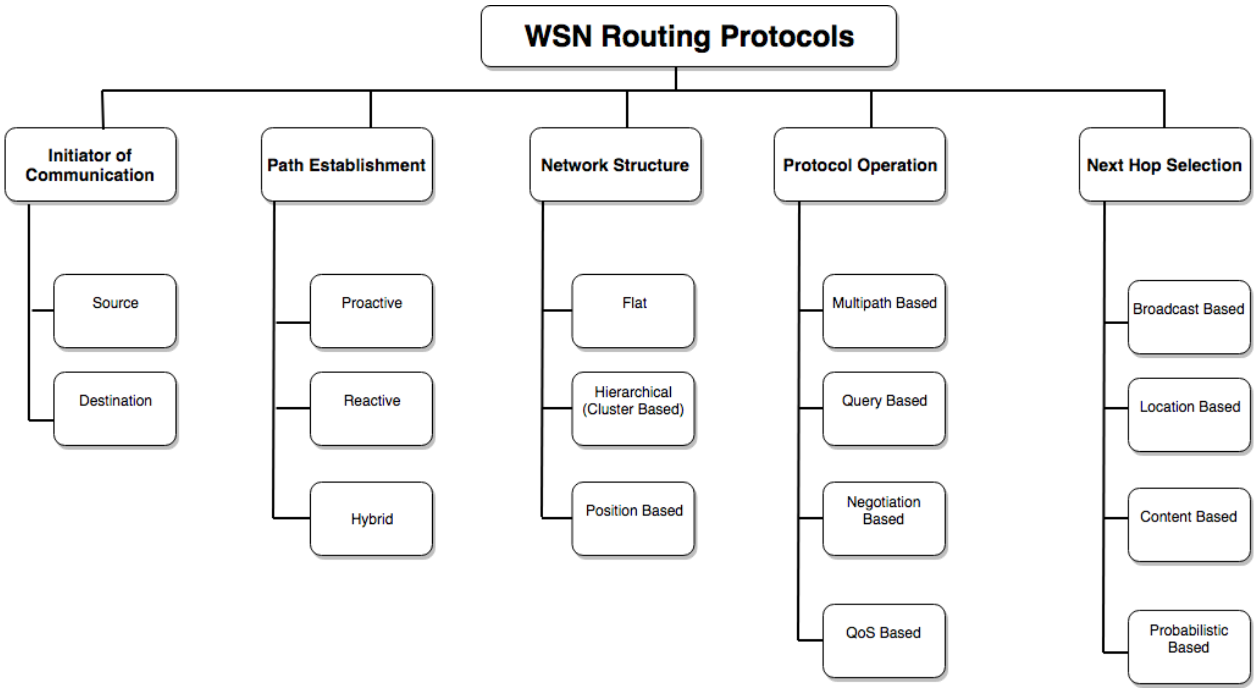

Figure 5: Classification of routing protocols in WSN (Singh & Sharma, 2015)

This study adopted the cluster based routing protocol (Boyinbode *et al*., 2011; Liu, 2012 & Naeimi *et al*., 2012) since it is an energy efficient method. In cluster based those nodes having high residue energies are selected for aggregation and sending data, while nodes those having low energies are used for sensing and transmitting data to the cluster heads (CHs). This property of clustering reduces energy consumption, increase scalability and prolong the network lifetime. The cluster based routing protocol playsapivotal role in achieving application specific goals such as large areas monitoring (Gupta & Younis, 2003; Dehghani *et al*., 2018).

2.2.4 Low energy adaptive clustering hierarch

Low Energy Adaptive Clustering Hierarchy is the one first and most popular routing protocol in WSNs (Tyagi & Kumar, 2013). The LEACH as the first cluster-based protocol was adopted and modified by number of researchers. The LEACH protocol like all others has its pros and cons. The idea behind LEACH is to save energy of sensor nodes as possible, and improve the network lifetime (Heinzelman *et al*., 2000). The cluster head threshold is given by equation 3.

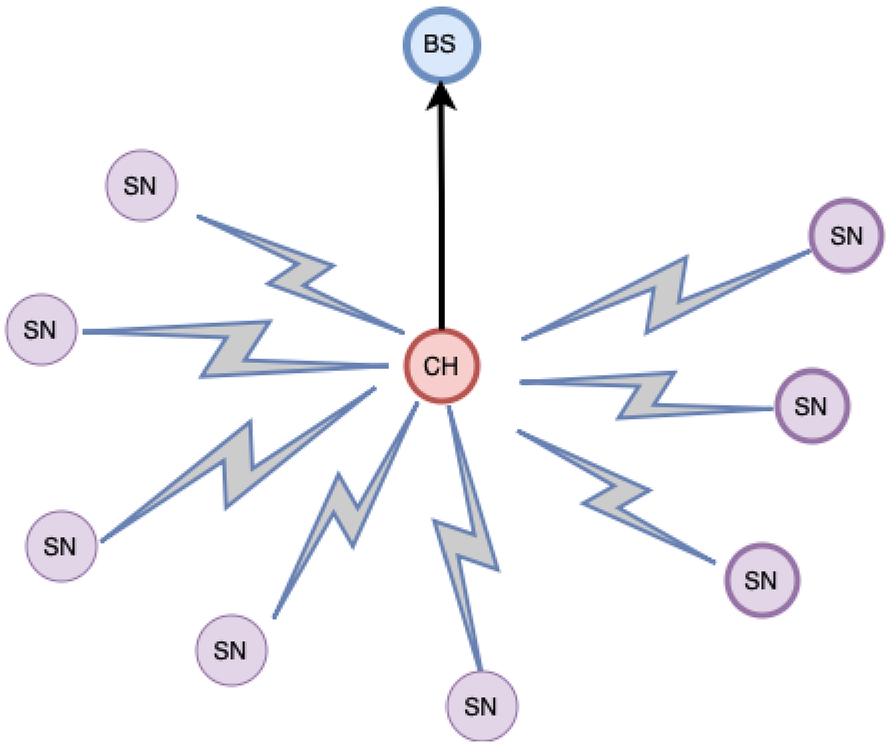

Figure 6: LEACH protocol, single hop in WSN (Heinzelman *et al*., 2000)

$$T(n) = \begin{cases} \dfrac{p}{1 - p * \left[r \bmod \left(\frac{1}{p}\right) \right]}, & if\ n \in G \\ 0, & elsewhere \end{cases} \qquad (3)$$

Where:

(i) p - percentage of choosing cluster heads

(ii) r - the current round

(iii) G - set of sensors that have not been cluster heads in $1/p$ round

2.2.5 Advantages of LEACH protocol

The LEACH protocol as one of the cluster based routing protocol has the following advantages:

(i) It enables data aggregation at the cluster head to discard the redundant and uncorrelated data; thereby, it saves energy of the sensor nodes.

(ii) It is easier to manage packets routing because only CHs need to maintain the local route set up of other CHs and thus require small routing information, this improves the scalability of the network significantly.

(iii) The communication bandwidth is well conserved since non-CH members in the cluster communicate only with their CH and thus avoid redundant messages to exchange among themselves.

(iv) The communication in LEACH follows the clustering concept, and therefore increase the network lifetime by reducing the number of the packet transmitted to the sink node.

(v) In LEACH there is intra-cluster collision avoidance by allocating schedule for nodes to sleep using TDMA. This help to prolong the network lifetime.

(vi) All sensor nodes in LEACH have an equal chance to become a CH at least once in every round. The random selection of the CH reduces the energy consumption of the nodes.

(vii) LEACH routing protocol does not require any global information to operate due to its distributed nature.

2.2.6 Limitations of LEACH

(i) Although the random selection of the CH enhance the network lifetime, but this randomized choice the nodes does not consider their residual energy.

(ii) LEACH is designed to work well only on small area. Single hop as shown in (Fig. 6).

(iii) In LEACH there is a limitation of TDMA as each node will send data when its time is reached without considering the freshness of the data.

(iv) LEACH protocol fails to differentiate small and large clusters. As it is clear that distribution of nodes in the clusters might be different such that small size clusters consume more energy and die faster than the big ones. This is due to the fact small clusters will send data to the sink node with higher frequency.

(v) LEACH assumes that all sensors have enough energy to communicate with the BS. So, more energy consumed if sensors are far from the BS.

(vi) In LEACH, all nodes in the network are assumed to be homogeneous.

(vii) Sensornodesuserandomnumbersbetween0and1. If a number is less than the threshold, it will become the CH. There are no other criteria for the selection. The energy efficiency of the node is affected by the number of using as using as a CH.

2.2.7 Modification of cluster heads selection

Based on the limitation of LEACH, several researches proposed the modification to select the CHs in the network. Most of the studies consider the following factors to modify the CHs selections:

(i) Residual energy of each sensor node

(ii) The distance between a CH and the sink node

(iii) The number of the neighbor nodes

Handy *et al.* (2002) proposed the sensor nodes with more neighbors than the other has a chance to be elected as a CH. They considered the radius of a neighborhood of the node to determine the node as a neighbor. For the nodes to be neighbors they must be in the neighborhood radius. The neighborhood radius is given in equation 4:

$$R_{neighborhood} = \sqrt{\frac{M^2}{\pi * C}} \qquad (4)$$

Equation 5 defines the distance between a sensor node and the CH.

$$d_{CNtoCH} = \frac{M}{\sqrt{2\pi C}} \qquad (5)$$

Finally, the distance between CHs and the sink node is given by equation 6.

$$d_{CNtoCH} = \frac{0.755 * M}{2} \qquad (6)$$

The $T(n)$ proposed by Heinzelman *et al.* (2002) in equation 3 is multiplied by a factor representing the current energy level of the sensor as shown in equation 7. The remaining energy level in each sensor node will increase the network lifetime (Handy *et al*., 2002).

$$T(n)_1 = \begin{cases} T(n) * \frac{E_{remaining}}{E_{initial}}, & if\ n \in G \\ 0, & elsewhere \end{cases} \qquad (7)$$

Where by:

(i) $E_{remaining}$ - Residual sensor node energy in the current round.

(ii) $E_{initial}$ - Initial sensor node energy

Since the CH threshold is too low, there is a high possibility the network to stuck after some rounds. This phenomenon will affect even those nodes with enough energy to transmit data to the sink node. The solution was proposed to extend $T(n)_1$ by a factor E_{avg} that increases the cluster head threshold. The improved threshold is improved by equation 8. This improvement helps the nodes with high remaining energy to have the chance to become CHs.

$$T(n)_2 = \begin{cases} T(n)_1 * (1 - \frac{1}{E_{avg}}), & if\ n \in G \\ 0, & elsewhere \end{cases} \qquad (8)$$

Where by:

E_{avg} – Average energy of all sensor nodes in the current round.

Since the distance between CHs and sink node contributes to the energy consumption of the network. Therefore, it is cost effective to select a node to which is near to the sink to become a CH. Equation 9 demonstrates the modification of CH threshold using the distance parameters.

$$T(n)_3 = \begin{cases} T(n)_2 * \dfrac{d_{toBS_{av}}}{toBS_n}, & if\ n\ \in G \\ 0, & elsewhere \end{cases} \qquad (9)$$

Where by:

(i) $d_{toBS_{av}}$ - The average distance of nodes to the sink node

(ii) $toBS_n$ - The distance between sensor node to the sink node

Also, the number of times a node selected as a CH and the distance from sending to receiving node can affect the choice of cluster heads as demonstrated in equation 10.

$$T(n)_4 = \begin{cases} T(n)_3 * (1 - \log_{10} d) * \dfrac{1}{CH_s} * Nb_n, & if\ n\ \in G \\ 0, & elsewhere \end{cases} \qquad (10)$$

Where by:

(i) CH_s - The time that node selected as a cluster head.

(ii) Nb_n - The number of neighbors of n nodes.

From equation 10 each node generate a random number between 0 and 1, the generated number will be compared with threshold value, and a node a less random number will be selected as a CH for the current round. Wan *et al.* (2018), calculated the energy consumption of all active nodes with a range of CH's competition radius to transmit their collected data to the CH by using equation 11:

$$E_{toCH} = \sigma \int_{d_{CH}}^{0} 2\pi\rho\big(nE_{elec} + n\varepsilon_{fs}x^2\big)dx = \sigma n\pi\rho(E_{elec}d_{CH}^2 + \frac{1}{2}\varepsilon_{fs}d_{toBS}^4) \qquad (11)$$

Where by:

(i) n - The length of the packets in bits.

(ii) $\rho = \frac{N}{M^2}$ - The node density, M - length of monitoring area and N - number of nodes

(iii) d_{toBS} - The transmission distance from CH to sink node

(iv) $\sigma = 1 - (\frac{V}{k-1})$ - The percentage of active member nodes in the cluster, V - number of redundant nodes in the cluster

(v) $2\pi x\rho dx$ - The number of nodes being located in the ring annular area with the length$x(0 \leq x \leq ri)$

(vi) E_{re} - The energy consumed by CH for receiving such packet

(vii) E_{ag} - The energy consumed of CH for data aggregation

(viii) E_{toBS} - The energy consumed by CH for data forwarding to the sink node

(ix) R_i - The distance from the CH to the BS

Finally, the research by Wan *et al.* (2018) concluded that the total energy consumed by a cluster is given by equation 12:

$$\begin{aligned} E_{cluster} = E_{toCH} + E_{toBS} + E_{re} + E_{ag} \\ = n\pi\rho r_i^2 \left(\sigma + \frac{1}{2}\sigma\varepsilon_{fs}r_i^2 + E_{elec} + E_{DA}\right) \\ + n\varepsilon_m R_i^4 \qquad (12) \end{aligned}$$

Elshrkawey *et al.* (2018) proposed the following steps to calculate the total energy dissipated in the clusters: A network with N sensor nodes has been considered, and there are C clusters divided the network, which means there are $\frac{N}{C}$ average numbers of nodes per cluster. For a single frame, the energy consumed by the CH is given by:

$$E_{CH} = nE_{elec}\, {}^{N}/_{C} + nE_{D}A\, {}^{N}/_{C} + n\varepsilon_m d_{toBS}^4 \qquad (13)$$

Where:

(i) E_{DA} - The energy consumed in aggregation of data

(ii) d_{toBS} - The distance from the CH to BS

The energy consumption in non-cluster head nodes for transmitting the n-bit packet to the CH is derived from equation (20) and (21) as follows:

$$E_{non\_CH} = nE_{elec} + n\varepsilon_f d_{toCH}^2 \qquad (14)$$

Where:

$$d_{toCH}^2 = \frac{M^2}{2\pi C} \qquad (15)$$

(i) The network radius - R

(ii) Area of each cluster - $\frac{M^2}{C}$

(iii)The cluster radius - $\frac{M}{\sqrt{C}}$

Therefore, the total energy dissipated by a cluster in a single frame is given by:

$$E_{cluster} = E_{CH} + E_{non\_CH} {}^{N}/_{C} \qquad (16)$$

The total energy dissipated by all clusters in a network for a frame is derived from equation (24) as follows:

$$E_{total} = cE_{cluster} \qquad (17)$$

Where:

(i) E_{total} - Total energy dissipated by all clusters

(ii) $E_{cluster}$ - Total energy dissipated by a cluster in a single frame

Morsy *et al*. (2018) proposed energy efficient algorithm for clustering and routing in WSN. In this work, efficient energy consumption is achieved using Gravitational Search Algorithm (GSA) clustering in WSN. The proposed protocol begins with a setup phase which contains two phases. They are neighbor discovery phase and flooding to BS phase. In neighbour discovering phase, each node sends hello packet which includes its ID and updates neighbour table with ID. In flooding to BS phase, broadcast its ID, energy and its neighbour. In the steady state phase, data transmission is performed in which each node sends sensed data in its TDMA slot and goes to sleep mode to save energy. Gravitational search algorithm is used for cluster head selection and routing which has a weak local searching mechanism. It affects the convergence speed and accuracy of the cluster formation.

Mann and Singh (2018) proposed optimal node clustering and scheduling in WSN. Herein, improved Artificial Bee colony (iABC) algorithm is used to select the optimal cluster head accompanied by optimal cluster head scheduling in WSN. The optimal cluster head selection is accomplished using four phases of iABC algorithm. Four phases are precisely initialization phase, employed bee phase, onlooker bee phase and scout bee phase. Here, the fitness function is computed using subsequent metrics specifically residual energy, minimum transmitted power to transmit aggregated data and distance between cluster head and sink node. The proposed iABC algorithm has the number of objective function evaluations, and it lacks in using secondary information.

Rodriguez-Zurrunero *et al.* (2018) proposed an adaptive scheduler for real-time operating systems to extend WSN nodes lifetime. In this work, an adaptive scheduler is intended to extend the WSN nodes lifetime which dynamically delays the execution of low priority tasks while maintaining real-time capabilities on high priority tasks. In this duty cycle, a decision algorithm is proposed to improve the network lifetime. The adaptive scheduler is used to schedule task which has the highest priority, not like round robin scheduler which schedule all available tasks. Therefore this proposed scheduler useful in the nodes with the rechargeable batteries by delaying the least critical tasks.

Purkar and Deshpande (2018) proposed energy efficient clustering protocol to enhance the performance of WSN. The network is divided into four zones based on the population. Herein, nodes are recognized as three types namely normal, advanced and supernodes based on their energy level. The cluster head is selected using a non-probabilistic manner using clustering protocol. Clustering protocol considers node quality index which is devised parameter based on the initial energy, remaining energy and hops count required by a particular node with respect to the sink node. Each selection of the cluster head takes more time hence the node quality computation is complex.

Mosavvar and Ghaffari (2019) proposed cluster based data aggregation in WSN. Here, sensor nodes are divided into several sensing areas using clustering operation. Operation of distributed sensor nodes within each cluster is performed using the firefly algorithm. Cluster head selection operation is carried out using a combination of firefly algorithm and Low

Energy Adaptive Clustering Hierarchy (LEACH) model. Cluster based firefly algorithm computes fitness function using distance and residual energy. Cluster head selection is not optimum since it considers less metrics for cluster head election.

Kaur and Mahajan (2018) proposed hybrid Meta heuristic based routing for WSN. A hybrid ACO and PSO algorithm based energy efficient protocol are used to form a cluster. ACO based path selection technique is utilized that form spanning tree between the cluster head and sink node. Here, next hop is selected based on the distance between nodes. Herein, energy aware threshold function proposed to elect CHs. The node with higher remaining energy has a chance to become a CH. The random numbers generated by each node and the remaining energy are proposed as criteria for CH selection. A node can become a CH if and only if it is a random number is less than the evaluated Threshold ($T(i)$). The $T(i)$ is evaluated using the following equation:

$$T(i) = \frac{P_{opt}}{1 - P_{opt}(r.mod(\frac{1}{P_{opt}}))} * \frac{E_i(r)}{E_{avg}(r)} \qquad (18)$$

For all nodes if $E_i(r) > 0$

Here, r represents the current round in WSNs network lifetime, $E_i(r)$ is the current energy of given node i.

E_{avg} - Average remaining energy which is evaluated as follows:

$$E_{avg} = \frac{\sum_{i=1}^{N} E_i(r)}{N} \qquad (19)$$

(i) For every node i

(ii) N - A total number of nodes.

Kang *et al*. (2017) proposed a distributed delay-efficient data aggregation scheduling for duty-cycled WSNs. Distributed delay efficient data aggregation scheduling scheme is introduced which generates a collision-free aggregation delay. The proposed work contains two novel ideas in the tree construction phase and scheduling phase. A distributed algorithm is proposed to construct degree constrained data aggregation which reduces the effect of high degree nodes aggregation delay. In this duty cycle is a fraction of one period in which the signal is active.

Arora *et al.* (2019) proposed ACO optimized self- organized energy balanced algorithm for WSN. The proposed method consists of three phases including cluster formation, multi-path creation and data transmission. In cluster formation, desired numbers of sensor nodes selected as a cluster head and remaining nodes join the nearest CH to form the cluster. Multiple paths between the cluster head and cluster member nodes are explored using ACO algorithm. Anti-colony optimization selects energy efficient, optimized route for data transmission between cluster member and cluster head nodes. The above discussed routing algorithm has a slow convergence rate; hence next hop selection is not effective.

Rhim *et al*. (2018) proposed Multi-Hop Graph based approach for Energy Efficient Routing (MH-GEER) protocol for WSN. The proposed protocol aim is to distribute energy consumption between the clusters with a balanced rate, and this improves network lifespan. Multi-Hop Graph based approach for Energy Efficient Routing (MH-GEER) deals with node clustering and inter cluster multi-hop routing selection phase. In the clustering phase, K-Means algorithm is utilized to form centralized clusters based on the randomly set ’K’ clusters. In the routing phase, new inter cluster routing protocol is used to find a path between the cluster head and sink node. Here, cluster head launches agent to carry data and find the path to sink node to deliver carried data. In this work, the K-Means clustering algorithm is used for cluster formation in which K value selection is critical.

Huang *et al.* (2018) proposed an energy efficient routing protocol for WSN. Arithmetic progression method is used to divide the network areainto the different cluster with various sizes. To route, the data to sink node interlevel multi hop routing algorithm is presented. Each node maintains its neighbour table to store information of neighbour, and it also knows its location and residual energy. The next hop node is selected based on remaining energy to route the data between the cluster head and sink node.

Liu *et al*. (2017) proposed distributed routing algorithm for WSN. Distributed routing strategy is utilized to achieve better data aggregation for WSN. The proposed algorithm performs better tradeoff between latency and energy conservation. This method finds the globally optimal path to achieve minimum average end to end delay with less energy consumption. The discussed routing methods have more data transmission delay since this process does not consider the distance between the sink node and cluster head that leads to more energy consumption.

Le *et al*. (2018) proposed a delay-aware tree construction and scheduling for data aggregation in duty-cycled wireless sensor networks. In this work, novel data aggregation scheme is proposed to minimize the data aggregation delay in duty cycle WSN. The proposed technique takes sleeping delay between sensor nodes into account to construct a connected dominating set tree in the first phase. The proposed CDS tree is useful for efficient data aggregation scheduling in the second phase as a virtual backbone. The proposed scheme does not allow any collision in a schedule to conserve the energy for transmission.

Bahbahani and Alsusa (2017) proposed a cooperative clustering protocol for sensor networks. The cooperative TDMA scheduling was used to schedule the sensor nodes. Here, each sensor node follows duty cycle to determine how often it will become CH in each round that was decided in the beginning of each round. Cooperative TDMA (cTDMA) slot was divided into direct transmission sub slot and cooperative transmission sub slot. During direct transmission sub slot, active nodes transmit their packets to the sink and cooperative nodes. The relay nodes were transmitting their packet in cooperative transmission sub slot. The proposed cTDMA cannot support dynamic changing of slots in scheduling.

Nguyen *et al.* (2018) proposed an efficient minimum latency scheduling algorithm for WSN. Here, the independent set based scheduling algorithm is used. Independent set based scheduling algorithm aggregates a fixed number of data into one packet to reduce data transmission delay. To minimize the required time slot for sensor nodes, this method forwards as many packets from the source node to the destination node. The proposed method has less sleep time that leads to high energy consumption.

Elshrkawey *et al*. (2018) proposed an enhancement approach for reducing energy consumption in WSN. The cluster head is selected based on the number of neighbors count and residual energy of the node. Herein, highest neighbor count node acquires more chance to be selected as a cluster head. Modified TDMA algorithm is used for scheduling that contains two phases setup and steady. Steady phase has more time slot than the setup phase. The largest cluster has minimum sleep time than the smallest cluster. Largest cluster has more active time that leads to a decrease in network lifetime.

Wan *et al*. (2018) proposed an energy efficient sleep scheduling mechanism with a similarity measure to reduce the energy consumption in wireless sensor networks. The proposed method schedules the sensors into active and sleep mode to minimize energy consumption effectively. At first, optimal competition radius is computed to organize sensor into several clusters to balance the energy consumption. Secondly, data collected by the member nodes fuzzy matrix are introduced to find the similarity in the collected data. Redundant nodes are selected to put into sleep mode. The selection of redundant nodes using fuzzy matrix requires more data for computation and high computational time which increases the delay in the sensor network and energy consumption.

Jiang *et al.* (2017) proposed low latency and energy efficient data preservation mechanism for Low Duty Cycle (LDC) WSN. A probability broadcast mechanism is used to disseminate data in a network. Each node decides whether to become a relay node based on computed probability. Here, each node wakeup at the time when its neighbor wakeup. After sending a packet to the neighbor node, it will wait for an acknowledgement message from the receiver. Nodes are wakeup based on neighbors wakeup time which leads to an increase in energy dissipation. Relay nodes send the acknowledgement to the sender; it may lead to an increase in energy consumption of sensor nodes.

Vijayalakshmi and Manickam (2019) proposed cluster based mobile data gathering using Space Division Multiple Access (SDMA) and PSO techniques for WSN. In this, sensors are grouped into clusters, and Mobile Data Collector (MDC) is deployed with multiple antennas which operate on SDMA technique. The cluster is formed using PSO algorithm which selects a cluster head based on the residual energy and degree of each sensor node. When the MDC meets the cluster point, the cluster members within the relevant clusters scheduled to communicate with Mobile Data Gather (MDG). Particle swarm optimization is used to select a cluster head based on the residual energy and degree of each sensor node. In this study, cluster head selection leads to high energy consumption, since it does not consider the distance between the cluster head and sink node which increases transmission delay.

Edla *et al.* (2019) proposed a PSO based routing with novel fitness function for improving the lifetime of WSNs. The proposed algorithm contains particle initialization, evaluation of fitness function and updating velocity and position phases. In particle initialization phase each gateway is initialized with the randomly generated number from uniform distribution range between 0 and 1. In this fitness function is computed based on three objectives; they are the distance between gateways and base station, the number of relay node between gateways to base station and relay load factor of the network. Based on this computed fitness function routes are selected between gateway and base station. Particle swarm optimization is used to select an optimum path between a source and a sink node. Since, PSO computes fitness function for succeeding metrics such as a distance between a source and the sink node, relay load factor and relay node count. Based on the computed fitness function, PSO chooses the optimum path to transmit data from source to sink node. The PSO does not consider node residual energy while selecting a path between source and destination that leads to an increase in energy consumption of the sensor network.

Sert *et al.* (2018) constructed a Wireless Sensor Network with Sensor nodes and Base Station, and it performs processing in two tiers. In tier one; Distributed Fuzzy Logic clustering is performed by estimating the parameters as distance, remaining energy and relative node connectivity. Using fuzzy, predict Competition Radius with which the cluster head adds cluster members. Then cluster head performs data aggregation by collecting sensed data from cluster members. A set of 27 fuzzy rules are deployed in clustering phase. In tier two, Fuzzy Routing is performed by computing the parameters relative distance and average link residual energy for determining a path. A possible path that satisfies fuzzy is selected and data transmission is begun on the selected path. The drawback of this concept is that the clustering head selection is based on the connectivity of the particular node, which ignored the major constraint of energy in sensor network. Then each node has to check each rule to predict its competition radius. Also, the fuzzy rules can be altered, since change in relative node connectivity also impacts in the competition radius. Whereby this competition radius could be predicted based on the distance and position of a node.

Li *et al*. (2018) proposed a differentiated data aggregation routing (DDAR) scheme that ensure providence of route based on the satisfaction of Quality of Service (QoS). This network is constructed with the entities as: sink node, sensor nodes and aggregator nodes. The aggregator performs two phases as receiving data from sensor and transmitting the aggregated data to sink. The sensors are responsible for sensing and transmitting the data to aggregator. As per this work, the sensor nodes are in sleep state, while the aggregators are transmitting data to sink. A service tag is generated for each aggregator that is formulated using data aggregation ratio, value of service requirement, proportion of aggregator and probability of data packet of sensor. Using this service tag of aggregator the sensors chose nearest aggregator for data transmission. Based on this approach the network is deployed with 57 aggregators and 70 sensors, here the aggregators covers nearly 85% as the number of sensor nodes which is unnecessary. Since the aggregator node is equipped to gather data from multiple nodes. Here the sensor node selects aggregators based on the service level, if no aggregator is available to satisfy the service level, it choose the lesser satisfied service level from the list.

Wan *et al*. (2018) proposed an energy efficient sleep scheduling with similarity measure for the purpose of minimizing energy consumption. Clusters are created and the data is collected by fuzzy matrix based on the estimation of similarity degree. The cluster head selection probability function is formulated by considering the parameters as largest distance, nearest distance and residential energy. The optimal competition radius is defined for cluster heads to balance the energy consumption. Fuzzy matrix is used for clustering and then based on the similarity of the sensed data; they are categorized further into clusters. Energy-Efficient Sleep Scheduling Mechanism (EESSM) is applied over the unequal clusters created in this WSN environment. The competition radius is estimated from node to base station, however, the sensors far from base station require head selection; else it may be isolated from the network. In this work, scheduling of sensor nodes into sleep state is based on the number of redundant nodes chosen in accordance collected data within a cluster. Here the evaluation of similarity is complex. Increased number of sensors in sleep state may lead to cause poor network management (sensing information).

The Fast Adaptive and Energy-Efficient Data collection Protocol in Multi-Channel-Multi-Path which is operated on two phases was proposed (Liew *et al.,* 2018). Node-channel assignment is performed in first phase and scheduling, packet forwarding in second phase. In the second phase, the time is divided into cycles; each cycle is composed of scheduling sub-cycle and packet forwarding sub-cycle. Here, coordination between two neighbors is determined at particular time slot. This detection also includes the traffic load, where higher traffic load has many pairs and at lower traffic, the nodes can be put in sleep mode. The parent and child nodes exchange grant and accept message for getting paired. The proposed data collection by scheduling and packet forwarding method is better in higher traffic load; however it consumes higher energy consumption. Due to high energy consumption, the network lifetime is poor. Energy is consumed due to the following reasons: (a) selection of parent for each time slot and (b) a parent transmitting grant message for each time slot.

Li and Li (2018) proposed Energy Balanced Routing Protocol (EBRP). This EBRP performs clustering using K-means++ algorithm and the cluster head is selected using fuzzy logic system (FLS). Initially K-means++ algorithm is applied and the nodes are divided into clusters based on their positions by the sink node. Then cluster head is selected by FLS, the fuzzy rules for cluster head selection is generated using genetic algorithm that performs selection, crossover and mutation processes. The three parameters involved in cluster head selection are distance between sensors and sink, distance between sensor and cluster center and residual energy. A set of 45 rules are generated for selecting a cluster head. Further data transmission is performed based on TDMA scheduling. The rules in fuzzy logic need to be defined individually based on the varying number of node, however the parameters are same then it is not required to define rules repeatedly. Also the genetic algorithm consumes higher computational time and slower processing which takes time to define fuzzy rules for cluster head selection. This caused by inappropriate selection of k-value in K-means++ algorithm.

Table 7: Contribution, objectives and limitations of related studies

Author Name	Method	Contribution	Objective	Limitations
Mann and Singh (2018)	iABC	It selects the optimal cluster head for data transmission in WSN.	To select the cluster head fast using the optimization algorithm.	The iABC-based cluster head selection lacks in analyzing the secondary information of the sensor nodes. This results in frequent cluster head selection.
Nguyen *et al*. (2019)	BAT	It selects a cluster head and forms clusters in the sensor network.	To select the optimal cluster head to reduce energy consumption.	It consumes more time to select the cluster head. This reduces the energy consumption.
Mosavvar and Ghaffari (2019)	LEACH	It forms clusters and selects a cluster head using an optimization algorithm.	To perform efficient data aggregation in WSN.	The cluster head selection is not optimal that affects the data aggregation efficiency.
Kang *et al.* (2018)	E-LEACH	It forms clustering and utilizes the distributed algorithm to pass the message.	To form clusters effectually to reduce energy consumption.	More parameters are required to select the best cluster head. Hence, it has frequent clustering in the network.
Kozłowski and Sosnowski (2019)	EEDC	It provides a duty-cycle based on transmitting and receiving energy.	To provide an energy-efficient duty-cycle.	It increases the delay during data transmission that tends to packet loss.
Bahbahani and Alsusa (2017)	cTDMA	It allocates slots to transmit data packets based on the cluster head occurrence count.	To increase the lifetime of the WSN network.	It cannot change the transmission slots dynamically that leads to more energy drainage.
Nguyen *et al*. (2018)	ISS	It provides a slot to each node based on the number of data packet count.	To minimize the data transmission delay in WSN.	It has high energy consumption due to a lack of consideration of energy-oriented metrics.
Elshrkawey *et al*. (2018)	Modified TDMA	It allocates slots using set and steady phases.	To reduce the energy consumption of the cluster head.	It follows a complex computation process that decreases the performance of the system.
Kaur and Mahajan (2018)	ACO-PSO	The cluster-based routing is performed using the hybrid ACO-PSO algorithm.	To provide better data aggregation in data transmission.	The distance parameter only considered for next-hop selection thus increases the energy consumption.
Arora *et al.* (2019)	ACO	It routes the packet to the optimal path.	To balance the energy among the cluster head node in WSN.	The distance and energy are not sufficient to attain less delay in WSN routing.
Arora *et al.* (2020)	MPACO	It routes the packet by selecting the optimal path.	To design energy-efficient routing in WSN.	The parameters considered for routing are not sufficient to reduce packet loss in WSN.
Rhim *et al*. (2018)	MH-GEER	It provides cluster-based routing to route the sensed data.	Reducing the sensor node energy consumption to increase performance.	The next-hop is selected based on the energy, which thus leads to an increase in the load of each sensor node.
Liu *et al*. (2017)	DRA	It selects the optimal path to transmit the data packet.	To prolong the lifetime of the WSN.	The distance between the source and destination node is not considered thus increases latency.
Jiang *et al.* (2017)	LDC	It provides Low-Latency and Energy-EfficientData Preservation Mechanism	To reduce the latency and energy consumption in WSN	The nodes are woken up based on the neighbor's wakeup time which leads to the increase in energy dissipation of nodes
Vijayalakshmi and Manickam (2019)	SDMA	It provides data gathering in WSN using SDMA and PSO	To prolong the lifetime of the WSN	CH selection was not effective due to the absence of significant parameter consideration such as distance and energy.

Energy Conservation has become a significant factor in the design of any system nowadays. Wireless sensor networks consume a lot of energy, and various algorithms have been developed to minimize energy consumption. Therefore, this dissertation proposes the following:

(i) ACO-Firefly algorithm based routing which reduces the delay and increase the lifetime through minimization of energy consumption.

(ii) Scheduling sensor node using Deep Reinforcement Learning algorithm which reduces energy consumption and delay.

(iii) The proposed hybrid PSO-AP algorithm for cluster head selection which selects the optimum cluster head which reduce the energy consumption and delay

CHAPTER THREE

MATERIALS AND METHODS

Researches by Akkaya and Younis (2005), Al-Karaki and Kamal (2004), Gherbi *et al.* (2017), Singh *et al.* (2010) and Singh and Sharma (2015) have been published a comprehensive survey on clusters based routing protocols in WSN. These researches have outlines the pros and cons of routing protocols in WSN as well as proposing taxonomy of cluster-based routing scheme. As clearly explained in chapter 2, in this research the clustering routing protocol was selected among the others due to the fact that it is more energy efficient and suitable for large areas.

3.1 Study area

The Usambara Mountains found in the Lushoto District of Tanga region in the north-eastern part of Tanzania. The mountains are about 90 km long and about half that wide. They composed of Precambrian metamorphic rocks. The range of the mountains is split into two sub-ranges known as; the Eastern Usambara which is nearer the coast and receives more rainfall, and the West Usambara Mountains which are being higher at 90 km long, 31 km wide. These mountains are rich in biodiversity and endemic species of fauna and flora (Meliyo *et al.*, 2018). In this study, we have selected the west Usambara Mountain because it has unique features such as; weather patterns, massive movement of different species of plants and animals, population dynamics or activities. Although there have been extensive studies on plant and animals species, little effort has been made on collecting and assessing the impact of weather data which may influence the wildfire. The monitoring of the weather data can help to predict the wildfire, rainfall etc. Lushoto district is one of the fastest growing areas in Tanzania. This growth of population and industrialization affect pollution, deforestation and global climate changes (Meliyo *et al.*, 2018). This research focuses on the conservation of national forests by monitoring their weather conditions using WSN. Forests have several advantages including habitats for animals, watershed protection and mitigation of climate by absorbing harmful greenhouse gasses (Xia *et al.*, 2001). Usambara forests are not much affected by wildfires like others in Europe, America and Northern Africa, but our proposed system will be in place in case of gradual changes of temperature and humidity. Therefore this research aims to reduce the negative impact of weather condition to the environment.

Lushoto is considered a warm and temperature district. It receives significant rainfall from January to December even in the driest months. The average annual temperature and average precipitation in Lushoto are 17.3 ºC and 1074 mm respectively. Its climate classified according to the KOppen-Geiger system as Cfb.

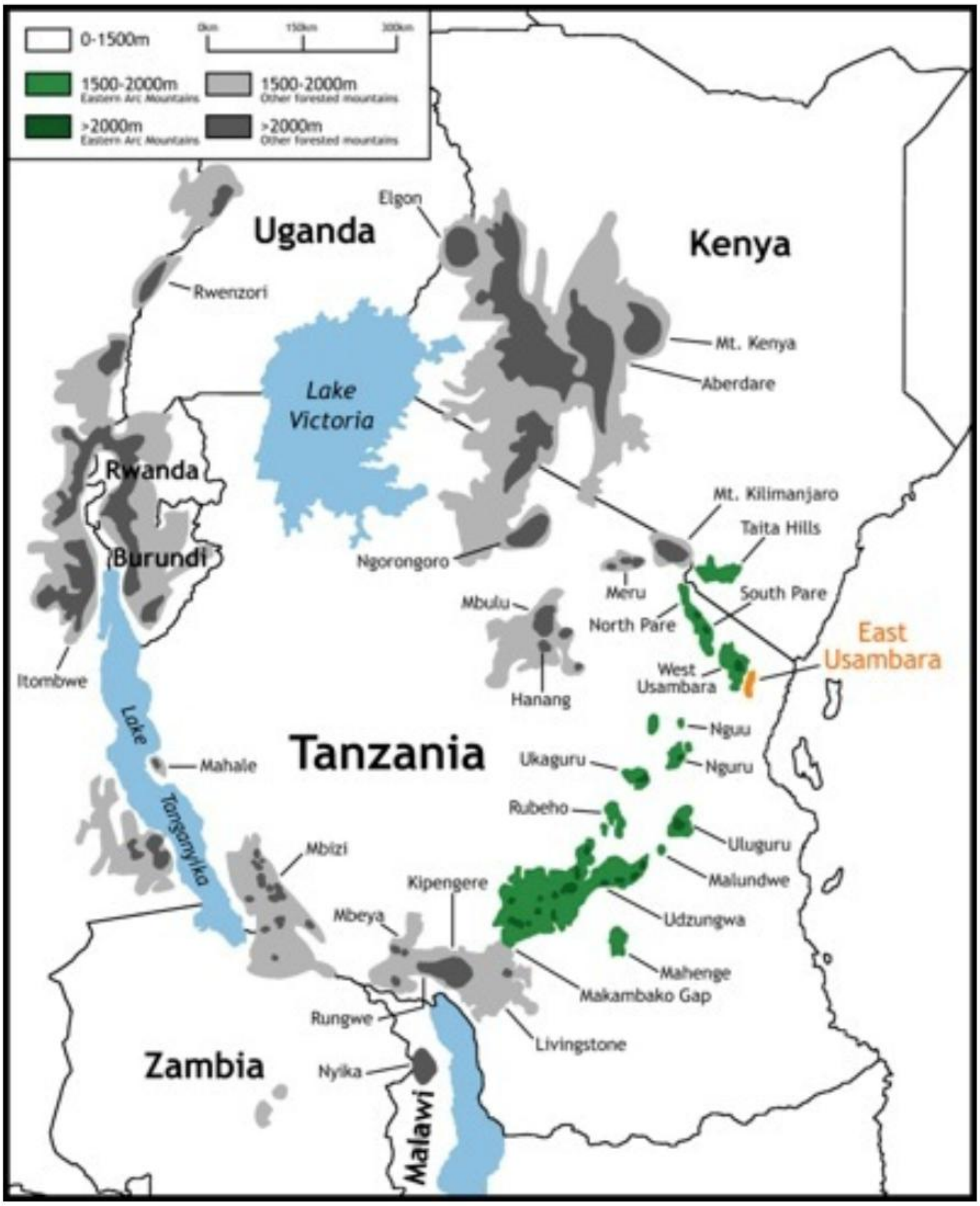

Figure 7: The study area in Usambara Mountains, Tanzania. Map credit: Rebecca Banksias and William H. Stanley (The Field Museum)

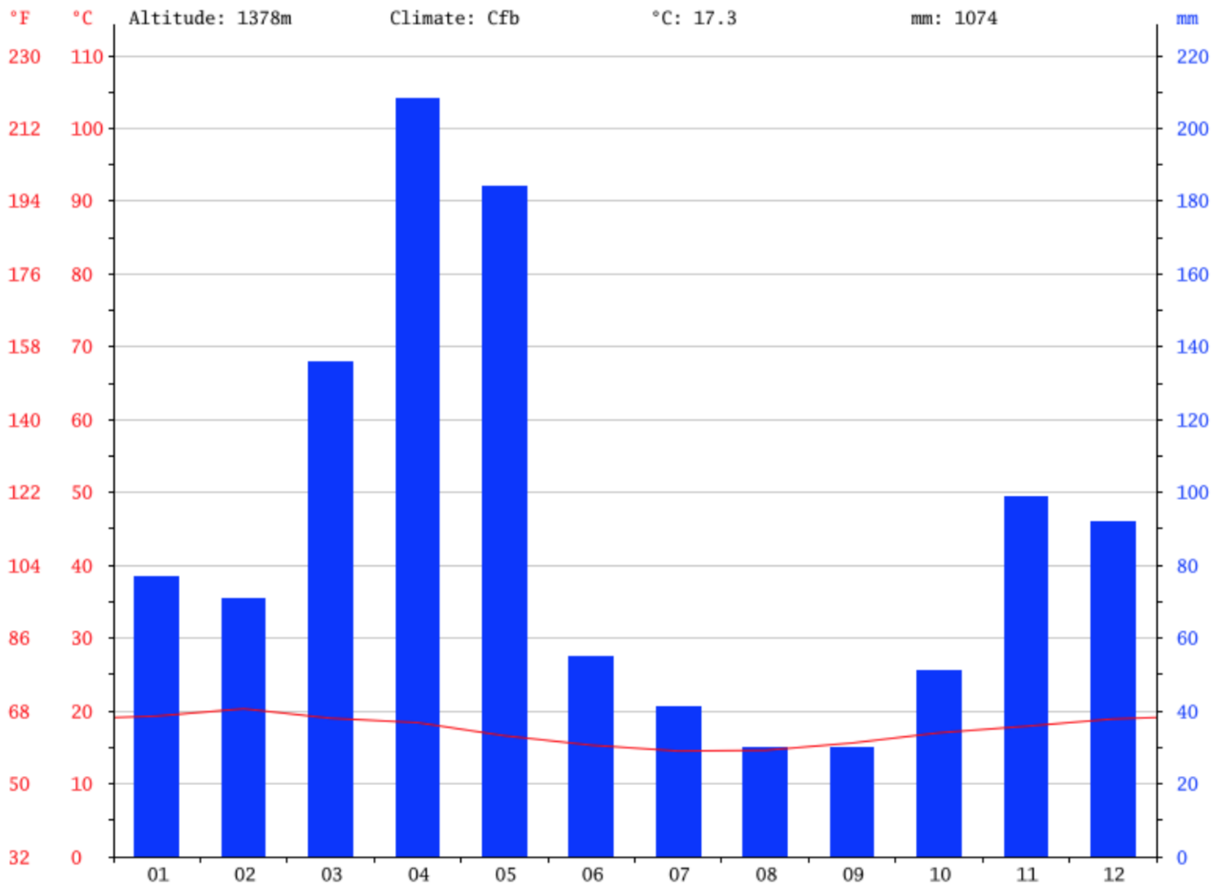

Figure 8: The average amount of rainfall yearly https://en.climate-data.org/

Figure 8 clearly shows the average amount of precipitation received in August which is 30 mm, whereas the most rainfall of 208 mm occurs in April.

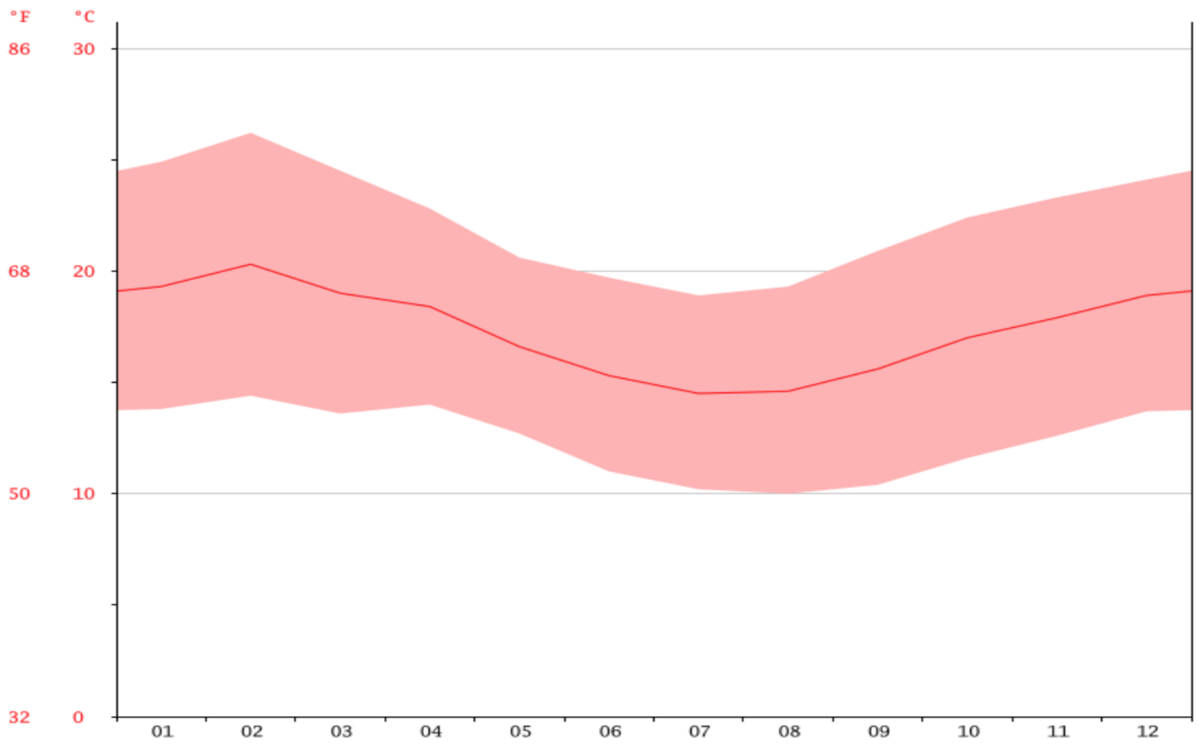

Figure 9: The average temperature yearly (https://en.climate-data.org/)

The temperature reaches its highest value in February, at around 20.3 ºC, and the average lowest temperature of 14.5 ºC seen in July.

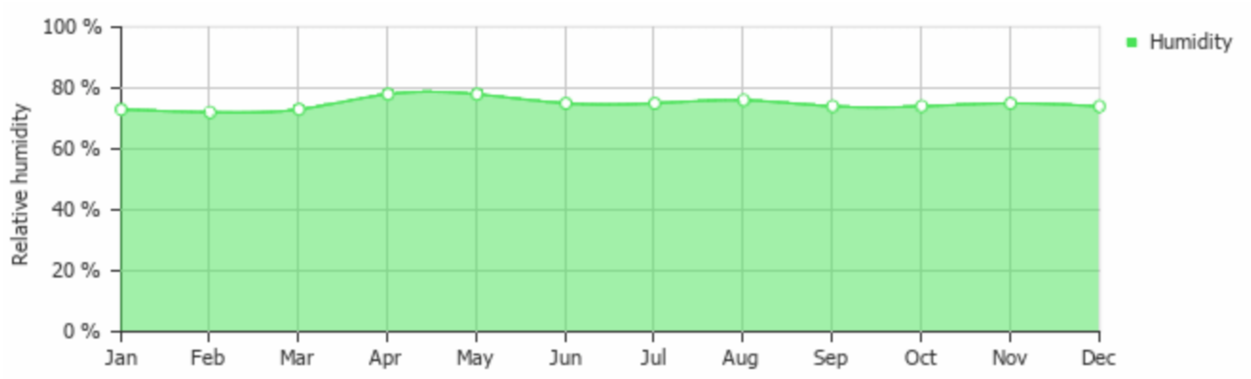

Figure 10: The average relative humidity in Lushoto (2019) (www.weather-and-climate.com/)

Figure 10 shows the mean monthly relative humidity over the year in Lushoto, Tanzania. The most humidor this May while the least humid month is February. The average annual percentage of humidity is: 75.0%. Table 7 shows the variation of temperatures and rainfall between wettest and driest months. The average variation of temperature yearly is by 5.8 ºC while that of precipitation is 178 mm.

Table 8: Lushoto weather by month https://en.climate-data.org/

Temperature and Precipitation	January	February	March	April	May	June	July	August	September	October	November	December
Avg. Temperature(◦C)	19.3	20.3	19	18.4	16.6	15.3	14.5	14.6	15.6	17	17.9	18.9
Min. Temperature (◦C)	13.8	14.4	13.6	14	12.7	11	10.2	10	10.4	11.6	12.6	13.7
Max. Temperature (◦C)	24.9	26.2	24.5	22.8	20.6	19.7	18.9	19.3	20.9	22.4	23.3	24.1
Avg. Temperature(◦F)	66.7	68.5	66.2	65.1	61.9	59.5	58.1	58.3	60.1	62.6	64.2	66.0
Min. Temperature (◦F)	56.8	57.9	56.5	57.2	54.9	51.8	50.4	50.0	50.7	52.9	54.7	56.7
Max. Temperature (◦F	76.8	79.2	76.1	73.0	69.1	67.5	66.0	66.7	69.6	72.3	73.9	75.4
Precipitation / Rainfall (mm)	77	71	136	208	184	55	41	30	30	51	99	92

3.2 Communication media/technology

There are numerous types of wireless technologies in the market nowadays including Wi-Fi, Bluetooth, Zig Bee, 3G and 4G. The (Table 9) shows comparison of different technologies used in wireless devices. These technologies offer many different wireless applications, but ZigBee is the best for limited bandwidth and low-power devices. The most comparable wireless technology to ZigBee is Bluetooth (Choudhury *et al.*, 2015). The Bluetooth is designed explicitly for file transferring in the Ad-hoc network increasing the complexity of its protocols. Wireless characteristics of Bluetooth differ from those of Zigbee in capabilities, for instance, Bluetooth as a maximum operating range of 10 m, a maximum of seven slave devices, and a maximum of battery lifetime of up seven days (Franceschinis *et al.*, 2013). ZigBee is a personal wireless network with low-cost, low data-rate, low-cost consumption, two-way wireless networking based on the IEEE 802.15.4 standard. This standard allows for the creation of lost cost and low power networks that run for years. These networks are created from sensors and actuators and can wirelessly control many electrical products such as remote controls, medical, industrial and security sensors. More than 200 companies including Chipcon, Ember, Freescale, Honeywell, Mitsubishi Electric, Motorola, Philips, Samsung and Texas Instruments have adopted this technology (Vancin *et al.*, 2015). The following are features which make ZigBee more suitable for this work:

(i) ZigBee devices operate at low radio frequency (RF) power of 1 mW

(ii) ZigBee devices can sleep when not involved in communication

(iii) The devices can operate without power cable

(iv) The devices can last for a long time in a single power battery

Table 9: ZigBee compared to other wireless technologies Vanc¸in and Erdem (2015)

Market Name Standard	**ZigBee 802.15.4**	**GSM/GPRS CDMA/1xRTT**	**Wi-Fi 802.11b**	**Bluetooth 802.15.1**
Application focus	Monitoring & control	Wide Area Voice & Data	Web, Email, Video	Cable replacement
System resources	4 KB – 32 KB	16 MB+	1 MB+	250 KB+
Battery life (days)	100 – 1,000+	1 - 7	.5 - 5	1 - 7
Network size	Unlimited (2^{6+})	1	`32	7
Maximum data rate (KB/s)	20 - 250	64 – 128+	11,000+	720
Transmission range (meters)	1 – 100+	1,000+	1 - 100	1 – 10+
Success metrics	Reliability, power, costs	Reach, quality	Speed, flexibility	Cost, convenience

3.3 Network simulator

Several network simulators with different features exist (ns-2, OPNET, NetSim, ns-3, OMNET++, REAL, QualNet and J-Sim). Some of them are open access simulators while others are commercial tools. Table 10 shows the comparison of different open source simulators.

Table 10: Comparison of different open source network simulators

Name of Simulator	Operating System	Programming Language	Licenses	Network Supoort	User Interface	API
NS-2	Windows, Linux	C++,Otcl	Free, Open Source	Wired and Wireless Network, Ad-Hoc mode, Wireless Managed mode, Wired cum Wireless, Cannot simulate problems of bandwidth or power consumption in WSN	Command Line Interface	Pure Event Based
OMNET++	Windows, Unix-based, Mac OS	C++	Free, non-commercial license, Commercial license	Wired Network, Wireless Managed mode	Graphical User Interface	Event based
JSIM	Truly platform independent	Java, Tcl	Free, Open Source	Wired and Wireless Network, WSN, radio channels and power consumption	Command Line Interface on Linux and GUI	Completely Process Driven including Thread Synchronization
NS-3	Network size	C++, Python	Free, GNU General Public License	Wired and Wireless Network, WSN, IP and non-IP based networks	`Command Line Interface	Low-level users can mix and match between the simpler API
GloMoSim	Windows, Unix, Linux-based	Parsec (C-based simulation language)	Free Open Source, Commercial GloMoSim based product is QualNet	Wired and Wireless Network, Mobile Ad-hoc network	GUI	Parallel discrete event simulation

NS-3 is primarily used on Linux Systems, and designed as a set of libraries that can be combined together with other external software libraries. While some simulators platforms provide use of a single integrated graphical environment in which all tasks are carried out. The ns-3 is more modular in this regards. Several external animators, data analysis and visualization tools can be used with ns-3. The simulation conducted by Khan *et al.* (2012) suggested that GloMoSim, OMNET++ and ns-3 can be used in large scale simulations for the wireless network. In additional, ns-3 has shown the best simulation performance in terms of computational time compared to other open source network simulators. Also, ns-3 has proved to reduce CPU simulation when applications run in parallel and therefore can fully utilize the computer processor.

The key features of ns-3 which make it useful in this research are as follows:

(i) Different software core: The core of ns-3 is written in C++ and with Python scripting interface. Several advanced C++ design patterns are also used.

(ii) Attention to realism: protocol entities are designed to be closer to real computers.

(iii) Software integration: support the incorporation of more open-source networking software and reduce the need to write models for simulation.

(iv) Support for visualization: lightweight virtual machines are used.

(v) Tracing architecture: ns-3 is developing in tracing and statistics gathering framework trying to enable customization of the output without rebuilding the simulation core.

The following are ns-3 resources:

(i) The Web
(ii) Mercurial
(iii) Waf
(iv) Develop Socket Programming

3.4 Radio model

This work adopts a light model for radio hardware as introduced by Heinzelman *et al.* (2000). The energy dissipations in transmit and receiving model influenced by different radio characteristics. This work uses characteristics of radio that dissipate transmit electronics E_{elec}at a rate of $50nJ/bit$. This is a minimum requirement for the transmitter and receiver circuitry to run at a given transmit amplifier of $\in_{amp}$ equals to $120pJ/bit/m^2$ in order to achieve an acceptable of SNR.

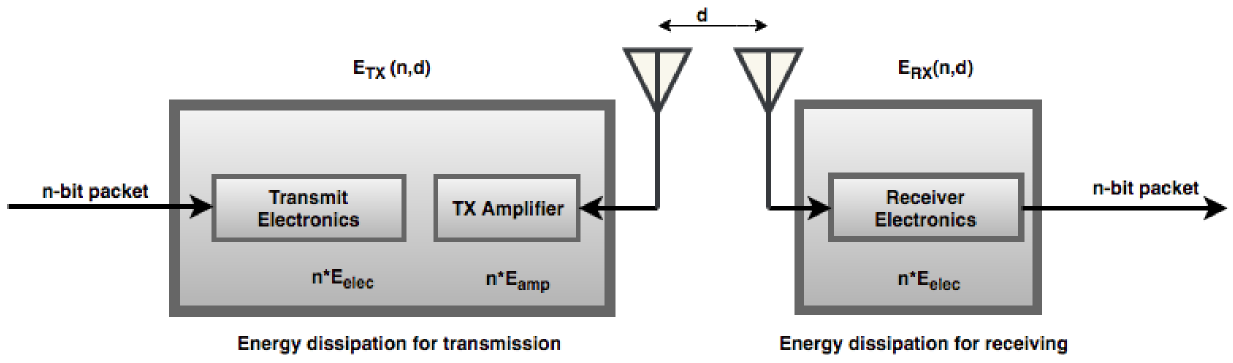

Figure 11: Radio model (Heinzelman *et al.*, 2000)

Figure 11 shows the distance d in meters between transmitter and receiver to determine how much energy can be conversed in the transmission process. Transmission power, P in a free space mode is directly proportion to the squared distance, d^2 may be higher in the presence of obstacles. For multi path propagation model this energy is proportional to the distance d^4 due to different paths that take the transmitted message to reach the receiver.

$$P \propto d^2$$

Where:

d - Distance in meters

This work adopted multi-hop communication nevertheless it brings more data delay, but consumes less energy than single hop (direct transmission). Two models have been used for energy consumption:

(i) Free space model $\varepsilon_f d^2$

(ii) Multipath fading model$\varepsilon_f d^4$

Both models are relying on the distance between transmitter and receiver. To transmit a n-packet at a distance d, we use the radio models as follows:

$$E_{Tx}(n,d) = E_{Tx-elec}(n) + E_{Tx-amp}(n,d) \qquad (20)$$

Whereby are from equation (20) $E_{Tx-elec}$ is a function of n, in which the higher the number of packet to transmitted, the higher the $E_{Tx-elec}$ is needed. Also E_{Tx-amp} is a function of both n and d, which means the high amount of transmit amplifier will be needed to transmit a large number of packet in a large distance.

From equation (20) the energy expended for free space propagationE_{Tx-fs} and multi path propagationE_{Tx-mp} can be derived as follows:

$$E_{Tx-fs}(n,d) = n * E_{elec} + n * \varepsilon_f d^2 \qquad (21)$$

$$E_{Tx-mp}(n,d) = n * E_{elec} + n * \varepsilon_{mp} d^4 \qquad (22)$$

If it is assumed that energy expended for free space propagation and multi path propagation are equal, then by equating equation (21) and (22) the crossover distance d_0 can be obtained as follows:

$$E_{Tx-fs}(n,d) = E_{Tx-mp}(n,d) \qquad (23)$$

$$n * \varepsilon_{fs} d^2 = n * \varepsilon_{mp} d^4 \qquad (24)$$

The equation (25) produced a relationship of d_0, ε_{fs} and ε_{mp}

$$d_0 = \sqrt{\frac{\varepsilon_{fs}}{\varepsilon_{mp}}} \qquad (25)$$

Whereby d_0 defines the propagation transition from direct path to multipath model

$$E_{Tx}(n,d) = \begin{cases} n * E_{elec} + n * \varepsilon_f d^2, & if\ d < d_0 \\ n * E_{elec} + n * \varepsilon_m d^4, & if\ d < d_0 \end{cases} \qquad (26)$$

$$E_{Rx}(n) = E_{Rx-elec}(n) + E_{Rx-amp}(n,d) \qquad (27)$$

From equation (20) to (27):

(i) E_{Tx} - Energy required to transmit packet

(ii) E_{Rx} - Energy required to receive packet

(iii) E_{elec} - Electronic energy for filtering, modulation the digital coding and spreading of the signal (Minimum electronics energy required to run Tx or Rx circuitry)

(iv) d_0 - Equals to the square root of the dividing EDA free space model by multipath fading model

(v) n - Number of message in bits

(vi) $\in_{amp}$ - transmit amplifier energy consumption

3.5 Proposed work

This research overwhelms the problems that are discussed in existing energy consumption in WSN. The main objective is to reduce energy consumption, improve the network lifetime and reducedelay. Therefore this work proposed to enhance network lifetime through energy efficient scheduling using DRL algorithm in WSN. Considering a network consists of static nodes with a sink node as depicted in Fig. 12. The sensor nodes are deployed in a sensing field as three coronas in which a sink node in the center point of them. Each corona is split into four partitions based on a sink node position. Three sequential phases such as Zone based clustering (ZbC), duty cycling and routing are used. The sensor network contains three components specifically sensor node, cluster head and sink node. Initially, the sensor nodes are deployed in the coronas (circular field). The proposed work considers three coronas namely C1, C2 and C3 for this work. The sink node is placed at the center of the coronas. Based on the center point of the corona, we split each corona into four partitions. Each partition consists of three zones namely Z1, Z2 and Z3. The first phase diminishes energy consumption through data aggregation using ZbC scheme. In ZbC scheme, a hybrid PSO and AP algorithm is used to establish clusters in each zone. Herein, PSO is applied to choose the best exemplar for AP using fitness function computation which contemplates succeeding metrics such as node degree, residual energy and distance. Based on the elected exemplar, AP forms clusters in each zone efficiently. The second phase enhances network lifetime through duty scheduling that is accomplished using DRL algorithm. The proposed scheduling algorithm adaptively changes each sensor node scheduling modes. In the last phase network delay is minimized via ACO and FFA algorithm. Here, FFA is used to select the best path from multiple paths selected by ACO. Anti-colony optimization selects multiple paths by considering subsequent parameters that are residual energy, hop count, bandwidth and distance. Firefly Algorithm (FFA) selects the best path by taking into account the following metrics for instance expected delay, packet delivery ratio and load.

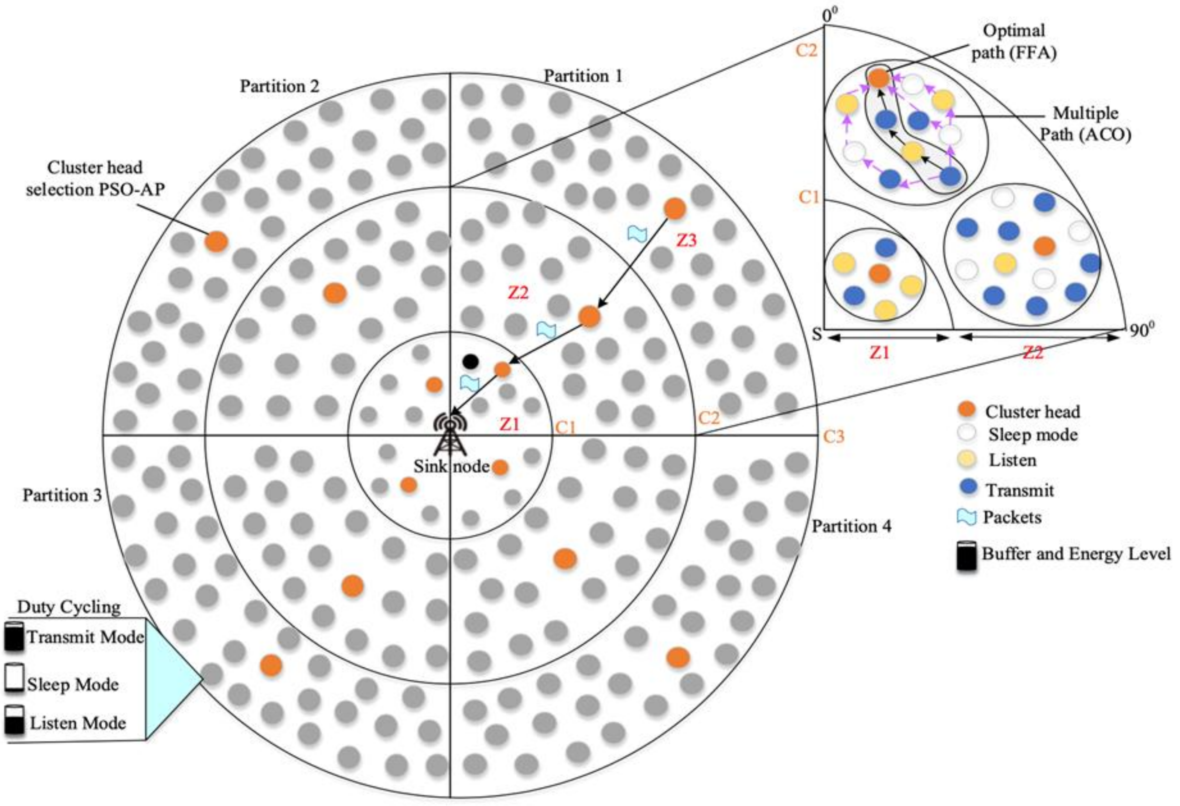

Figure 12: Architecture for proposed work

3.5.1 Zonal based clustering phase

In WSN, clustering plays a vital role in terms of energy consumption, since it minimizes energy consumption during data aggregation. To overthrow this problem, we pursue ZbC scheme that forms an efficient cluster in each zone using hybrid PSO and AP algorithm. The shortcoming of AP algorithm is the number of exemplar selection is not effective. To overthrow this problem, the combined PSO with AP is used.

(i) PSO based exemplar selection

Particle Swarm Optimization (PSO) algorithm operates based on the swarming nature of flocks of birds. In this work, the PSO is pursued to choose optimum exemplar for AP. The PSO computes fitness function forsubsequent metrics including node degree, residual energy and distance. Particle Swarm Optimization selects exemplar in each zone to form effective clusters. Particle Swarm Optimization comprises of three phases that are initialization, fitness function evaluation, Global best G_b and Local best L_b computation.

Initialization

In PSO, each sensor node is considered as a particle that is associated with initialize position, node degree, residual energy and distance.

Fitness function computation

In this phase, PSO computes fitness function for each node by means of succeeding parameters such as node degree, residual energy and distance that are explained as follows:

- **Node degree**

  Node degree is described as the number of neighbor nodes connected to the particular node. This metric elects node having the highest node degree that can communicate with more number of neighbour node that tends to increase in network performance. Node degree is represented as 'N_D' that can be conveyed as follows:

  $$N_D = N_c(N_i) \qquad (28)$$

  Where, $N_c(N_i)$ represents the neighbor count of the node N_i.

- **Residue energy**

  Residual energy is described as the difference between the total primary energy and consumed energy. Herein, residual energy metric is used to elect highest energy node to avoid frequent death of sensor node in a network. It is represented as 'E_r' that is expressed as follows:

  $$E_r = E_p - E_c \qquad (29)$$

  Where, E_p represents total primary energy and E_c represents consumed energy.

- **Distance**

  Distance parameter is referred to as the distance between sensor and a sink node. This metric also involved in network delay, since distance is directly proportional to the delay. It is represented as '$D_{i,s}$' follows:

  $$D_{i,s} = \sqrt{((N_{s,x} - N_{i,x})^2 + (N_{s,y} - N_{i,y})^2)} \qquad (30)$$

  Where,$N_{s,x}, N_{s,y}$ represents the position of the sink node and $(N_{i,x}, N_{i,y})$ represents the position of the sensor node. By means of the above metrics, PSO computes the fitness function for the exemplar using the following expression (Liu *et al*., 2017).

$$f_e = \sigma_1(E_r) + \sigma_2\left(D_{i,s}\right) + \sigma_3(N_D) \qquad (31)$$

Where,σ_1,σ_2 and σ_3 are weightage parameters.

(ii) Global Best and Local Best Computation

After computing fitness function for each node, this phase compares each node local best 'L_b' with other nodes to find the global best 'G_b' node. This process is functioned until stopping criterion reached. Finally, the optimum node with the highest fitness value is elected as an exemplar for the AP process.

(iii) AP cluster formation

Based on selected exemplar, AP forms clusters; here examplar nodes are elected by computing fitness function for each node. Using selected 'k' exemplar, AP forms clusters by means of similarity between sensor node N_i and exemplar N_e that is computed using the following equation:

$$Sim\,(N_e,N_i) = |N_e - N_i|^2 \qquad (32)$$

Where $|N_e - N_i|^2$represents similarity between Ni and Ne in terms of Euclidean distance. AP has two matrices that are responsibility matrix and available matrix. Responsibility matrix and available matrix are updated using computed similarity. Responsibility matrix 'R_m' is updated as follows:

$$R_m(N_e, N_i) = Sim\,(N_e,N_i) - max_{i' \neq i}\,(A(e,i') + s(e,i')) \qquad (33)$$

Where $A(e,i')$ represents the availability matrix that is expressed as follows:

$$A(e,i') = \min\left(0, R_m(N_e,N_i)\right) + \sum_{i' \neq e,i} \max\left(0, R_m(N_e,N_i)\right) \qquad (34)$$

Using two equations 33 and 34 responsibility and availability matrix are computed for each sensor node and exemplar node. This process is repeated until possible clusters are formed. The proposed clustering hybrid algorithm forms effective clusters in each zone that reduces energy consumption through data aggregation.

3.5.2 Duty cycling phase

Duty cycling is one of the significant processes to reduce the energy consumption of WSN. This work proposes DRL algorithm for duty cycling that adaptively decides each node scheduling modes.

(i) DRL based clustering

The DRL algorithm is utilized to schedule each sensor node scheduling modes adaptively. Here, the time period is divided into a different slot that is allocated to each sensor node. Hence each sensor node contains a unique slot that tends to avoid data collision between sensor nodes during transmission. In each slot, nodes adaptively change their mode using the DRL algorithm. Three modes are considered in scheduling that are sleep, listen and transmit conveyed as follows:

Sleep mode

During sleep mode, each sensor node turns off their radio that reduces their energy consumption of each node which in turn increases the lifetime of the network. A sensor node cannot receive or transmit any data when it is sleeping.

Listen mode

In listen mode, each sensor node senses its surrounding field and also it receives data from neighbor nodes. During a listen mode sensor nodes turn off their transmitter circuitry, hence transceiver can able receive data only.

Transmit mode

In transmit mode, each sensor node transmits its sensed data to the sink node and also it transmits data from neighbor sensor node. Figure 13 depicts the DRL scheduling of the sensor nodes with three modes such as Transmit (T), Sleep (S) and Receive (R). Each node changes these three modes in each period using DRL algorithm. In DRL, the deep Q-Learning is adopted that comprises of following operations such as actions, states, Q-Value, payoff, and reward. Deep Q-leaning algorithm learns the environment state information and obtains the best action. The steps involved in the DRL scheduling are discussed as follows:

- **Actions** ($\mathcal{A}$): This work proposes three actions that are Sleep (1), Listen (2) and Transmit (3). Actions are undertaken by each node based on the computed Q-value.

- **States**: Four states are proposed for each node that are $\mathbb{S}_0, \mathbb{S}_1, \mathbb{S}_2$and $\mathbb{S}_3$. $\mathbb{S}_0$ indicates node does not have any storage in its buffer$BS = 0$. $\mathbb{S}_1$ indicates node has storage in its buffer as $\frac{BS}{BS}$. $\mathbb{S}_2$ indicates node has storage in its buffer as $\frac{BS}{4}$. $\mathbb{S}_3$ indicates node has storage in its buffer as $\frac{BS}{2}$.
- **Q-value**: Q value is computed to select actions for each node. Here, Q-value is computed using the payoff value of the particular node.
- **Payoff**: Payoff value is computed using the probability of node to select the action. Payoff value is computed for each node to select particular action is each period.
- **Reward**: Reward is provided to each action on the basis of successful transmission of packets to the neighbors.

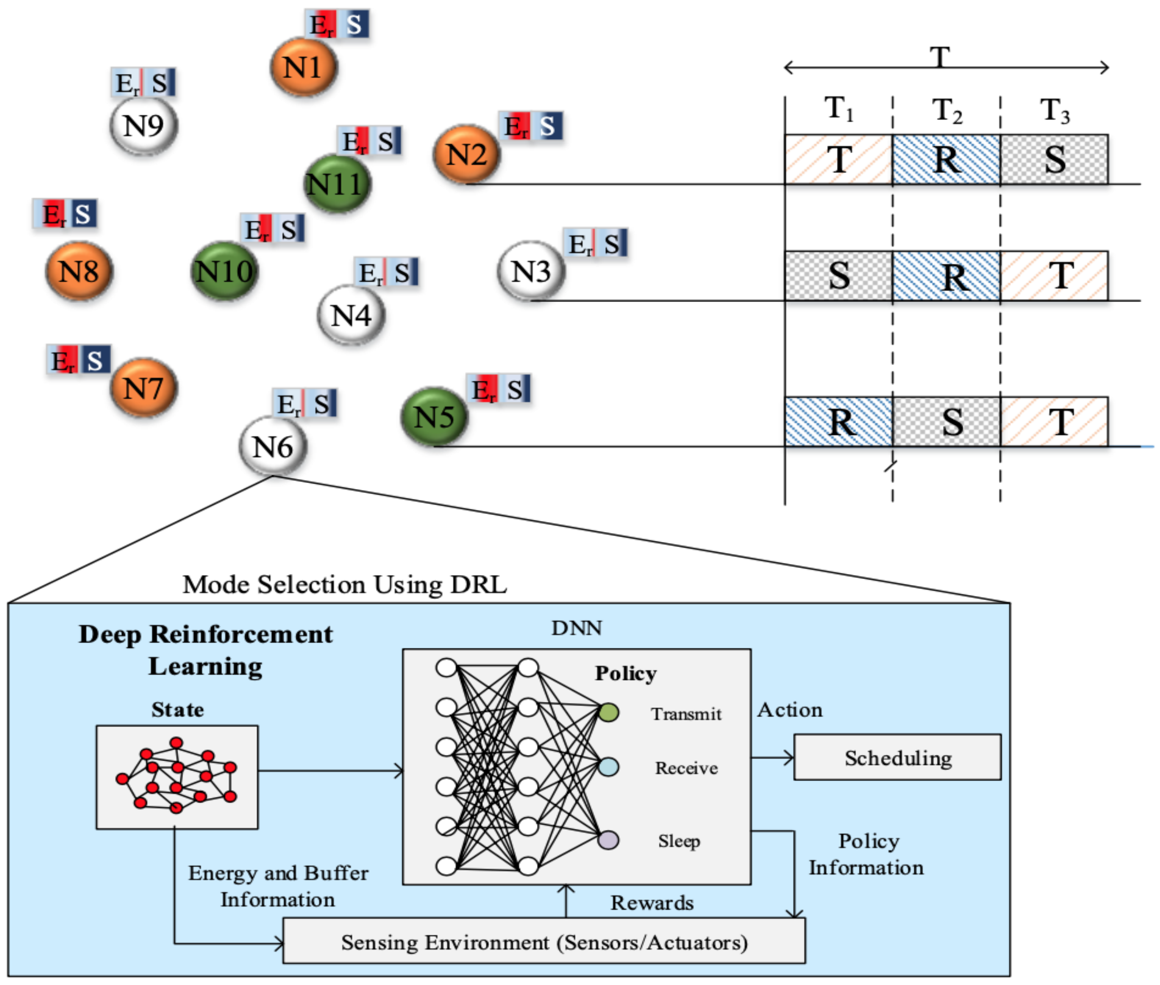

Figure 13: Deep reinforcement learning scheduling

Considering two nodes N_i and N_j need to select actions during the allocated period then it determines to pay off value for each node. This payoff value is computed using the probability distribution function over actions. If the node N_i and N_j selects action l and m respectively, then node N_i receives payoff $N_{i_{lm}}$and node N_j obtains payoff $N_{j_{lm}}$. Let $\alpha_1 - \alpha_3$, denote probability for node 'N_i' to select actions 1 to 3 where $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Let β_1 - β_3; denote probability for node 'N_j' to select actions 1 to 3 where $\beta_1 + \beta_2 + \beta_3 = 1$. The node '$N_i$' expected payoff is represented as:

$$\mathcal{P}_{N_i} = \sum_{1 \leq l \leq 3} \left(\sum_{1 \leq m \leq 3} N_{i_{lm}} \alpha_l \beta_l \right) \qquad (35)$$

The node N_j expected payoff is represented as,

$$\mathcal{P}_{N_j} = \sum_{1 \leq l \leq 3} \left(\sum_{1 \leq m \leq 3} N_{j_{lm}} \alpha_m \beta_m \right) \qquad (36)$$

Where payoff values $N_{i_{lm}}$ and $N_{j_{lm}}$ for nodes N_i and N_j are defined as the energy utilized by each node. This energy represents remaining energy of node which is calculated using equation 29. The reward is given to each node if a packet is successfully transmitted that is represented as '$\mathfrak{R}$'. This factor is included to the energy consumption only if a packet is successfully transmitted. For example, if node 'N_i' has packets to transmit, selects transmit action, then node 'N_j' selects listen to action that induces packets successfully transmitted.Here, payoffs for both nodes are positive which can be calculated using energy consumed to transmit/receive packet plus reward constant $\mathfrak{R}$. In transmit state node 'N_i' intend to transmit a packet that consumes energy σ_{ei} and N_j consumes energy σ_{ej} to receive packets. Then, the payoff for node N_i is$-\sigma_{ei} + \mathfrak{R} = \mathcal{P}_{N_i}$and a payoff for node $N_j$is$-\sigma_{ej} + \mathfrak{R} = \mathcal{P}_{N_j}$. This way of packet transmission reduces collision effectually.

The Q value is computed using the following equation,

$$Q(\mathbb{S}, \mathcal{A}; \theta) = \mathfrak{E}(\mathfrak{R}_i, + \delta \mathfrak{R}_2 + \delta^{\,2} \mathfrak{R}_3 + \cdots) \qquad (37)$$

Where δ represent discount rate for action and θ is weight parameter. Discount rate determines the importance of future reward. Value of discount rate relies on the range of $[0 \; to \; 1]$. Thus, $\delta = 0$ represents node is biased in current rewards whereas $\delta = 1$ represents node achieves the high reward. The deep Q learning learns the parameter θ of action function $Q(S, A; \theta)$ by minimizing sequence of a loss function, where i^{th} loss function $L_i(\theta_i)$is given by,

$$L_i(\theta_i) = \mathbb{E}[\Re_n + \underset{\mathcal{A}_{n+1}}{max} Q(\mathbb{S}_{n+1}, \mathcal{A}_{n+1}; \theta_{i-1}) - Q(\mathbb{S}_n, \mathcal{A}_n; \theta_i)] \qquad (38)$$

Where θ_i is neutral network parameter at i^{th} update and parameters from the previous updateθ_{i-1} are held to fixed during optimization loss function $L_i(\theta_i)$. The term $\Re_n + \underset{\mathcal{A}_{n+1}}{max} Q(\mathbb{S}_{n+1}, \mathcal{A}_{n+1}; \theta_{i-1}) - Q(\mathbb{S}_n, \mathcal{A}_n; \theta_i)$ is the target for iteration i that depends on the neutral network parameters from the last update. Differentiating loss function with respect to the neutral network parameter at iteration i, θ_i gives the upcoming gradient,

$$\nabla\theta_i L_i(\theta_i) = \mathbb{E}\left[\Re_n + \underset{\mathcal{A}_{n+1}}{max} Q(\mathbb{S}_{n+1}, \mathcal{A}_{n+1}; \theta_{i-1}) - Q(\mathbb{S}_n, \mathcal{A}_n; \theta_i)\nabla\theta_i Q(\mathbb{S}_n, \mathcal{A}_n; \theta_i)\right] (39)$$

Where$\nabla\theta_i$represents a gradient of θ_i.

Algorithm 1. Node Scheduling

Require: Energy and Buffer of Node
Ensure: Selected Mode of Operation for Node

```
    Initialize → replay memory $\mathbb{D}$ to the capacity 'C'
    Initialize → Q network with random weights $\theta$.
    Initialize → target Q network with random weights$\theta^- = \theta$
    for($\mathbb{S}_i < S$ &&$\mathcal{A}_i < A$)do
    {
    payoff $\mathcal{P}$ select → action $\mathcal{A}$;
    payoff 1- $\mathcal{P}$ select → $\mathcal{A}_n$ maximizes $\mathbb{S}, \mathcal{A}; \theta$;
    }
    Execute → action $\mathcal{A}_n$ to obtain reward $\Re_n$ and $\mathbb{S}_{n+1}$
    Store ($\mathcal{A}_n, \mathbb{S}_n, \mathbb{S}_{n+1}, \Re_n$) → $\mathbb{D}$
    If (n==n+1)
    Set target → $\Re_n$;
    Else
    Set target → $\Re_n + \underset{\mathcal{A}_{n+1}}{max} Q(\mathbb{S}_{n+1}, \mathcal{A}_{n+1}; \theta^-)$;
    Update$\theta^-$ → $\theta$ ;
    End for
```

Algorithm one describes the processes in each time slot in the period. Initially, replay memory $\mathbb{D}$, weight θ and random weight parameters θ^- are initialized. At first, the node selects an action depends on the probability distribution over three actions sleep, listen and transmit respectively in the current state. Node carried out selected action and observes the reward and new state. Node adjusts probability distribution over three actions in state S based on the payoff. In each iteration, θ^- target parameter is updated effectually. In this way, scheduling each sensor node in the network helps to avoid collision and reduces the energy consumption of the sensor node. In order to avoid divergence of DRL two techniques are introduced that are experience replay and the fixed target network. Using these techniques DRL minimizes the loss function,

$$L_i(\theta_i) = \mathbb{E}_{\mathbb{S}_n,\mathcal{A}_n,\Re_n,\mathbb{S}_{n+1}} \sim \mathbb{D}[\Re_n + \max_{\mathcal{A}_{n+1}} Q(\mathbb{S}_{n+1}, \mathcal{A}_{n+1}; \theta^-) - Q(\mathbb{S}_n, \mathcal{A}_n; \theta_i)] \quad (40)$$

Where, $\mathbb{D}$ denotes experience replay memory and θ^- is parameters of target Q network. Using these equations, deep Q learning algorithm adaptively changes each node action effectually.

3.5.3 Routing phase

Transmission delay is more in WSN due to inefficient path selection between the source and sink node. To overthrow network delay problem, ACO and FFA algorithm based routing are proposed. Multipath selection reduces delay in data transmission between source Cluster Member (CM) and exemplar. Hence, using ACO the multiple paths between CM and exemplar are selected. From selected multipath, optimum path is selected using FFA.

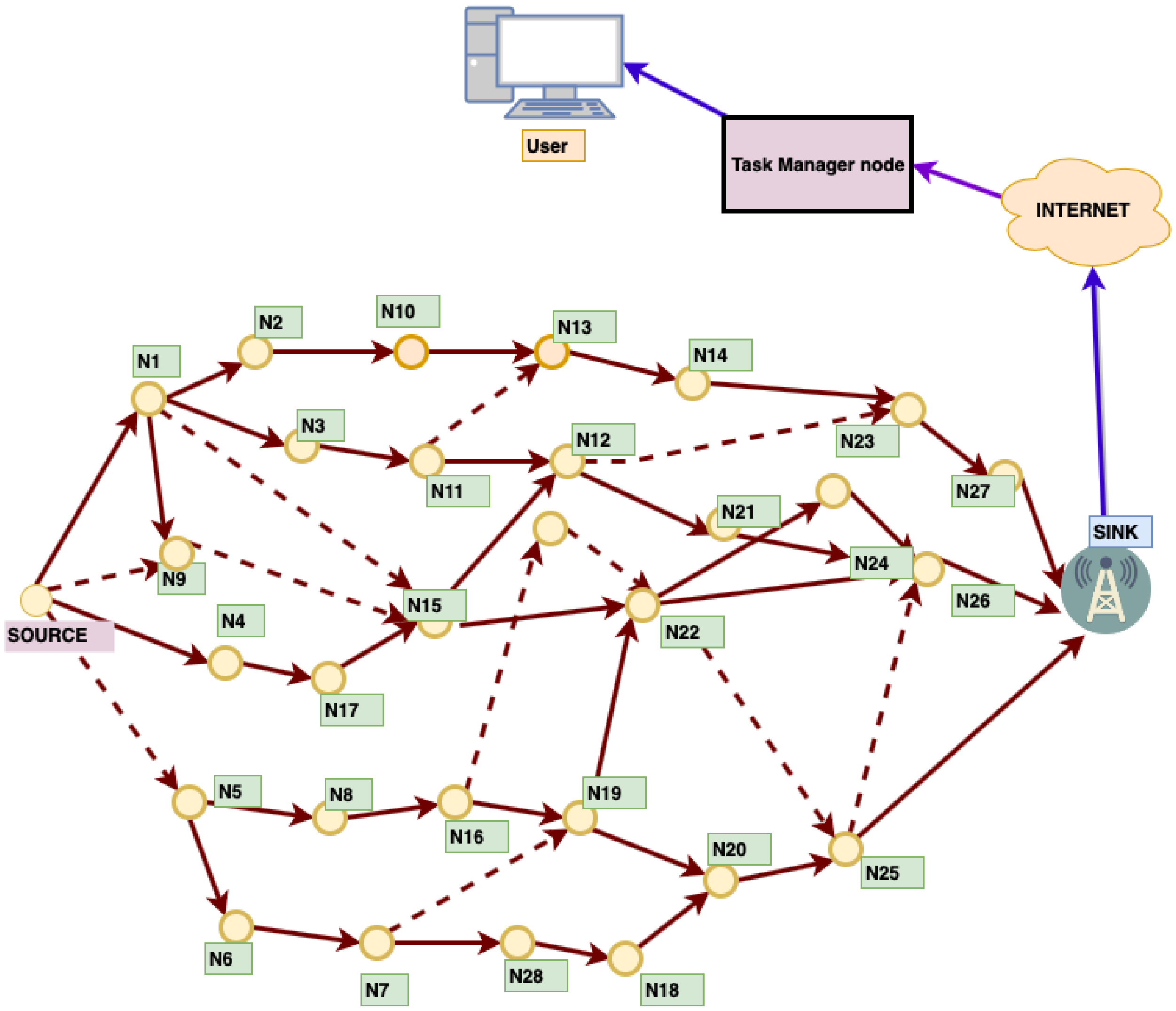

Figure 14: Selection of optimum path using ACO and FFA

(i) ACO algorithm

Particle swarm optimization algorithm works on the basis of real ants behaviour. The ACO algorithm computes fitness function using subsequent parameters such as hop count, bandwidth, residual energy and distance. In ACO multipath are selected based on the pheromone value of each node. Pheromone value is updated in each iteration to discover the best path between source CM and exemplar. Pheromone value is computed using the following parameters that are explained as follows:

Hop count

Hop count is described as the number of nodes between source CM and exemplar. This metric is used to reduce delay in data transmission since delay tends to increase the energy consumption of the sensor node. It is represented as 'H_c' that is explained as follows:

$H_c = N_c(N_{CM_i}, N_e)$ (41)

Where, $N_c(N_{CM_i}, N_s)$ represent hop count between source cluster member node N_{CM_i} and exemplar N_e.

Distance

Distance parameter is referred to as the distance between source cluster member and exemplar node. This metric also involved in network delay, since distance is directly proportional to the delay. It is represented as '$D_{CM_{i,e}}$' that can be computed as same as equation 30.

Bandwidth

Bandwidth parameter is considered to know the capacity of the link. Typically, bandwidth is measured in terms of bits per second. Bandwidth metric is used to elect the best path to route the sensed data. It is represented as 'B_p' that is expressed as follows:

$$B_p = \frac{b_n}{s} \quad (42)$$

Where b_n indicates a number of bits and s indicates seconds. In proposed ACO algorithm ants are working based on two rules. At first, ants select paths in the initial rule and then it finds an optimum path in revised rule.

- **Initial rule**

  In this rule probability of selecting a path is computed using pheromone. Pheromone function is computed using fitness value and pheromone intensity. At first, ACO compute fitness values for each node to elect multiple paths between source CM and exemplar node. Fitness value 'V_f' is computed using below expression,

$$V_f = \sum (E_r + H_c + B_p + \frac{1}{D_{i,s}}) \qquad (43)$$

  Equation 43 composed of all computed parameters such as residual energy, hop count, distance and bandwidth. By means of computed fitness value, ACO further computes pheromone for each path for data transmission. The pheromone function 'P_f' is computed using the succeeding expression:

$$V_f = P_f = \frac{{\tau_{i,j}}^{\alpha} * {V_f}^{\beta}}{\sum_{i=0}^{n} {\tau_{i,j}}^{\alpha} * {V_f}^{\beta}} \qquad (44)$$

Where, $\tau_{i,j}$indicates pheromone intensity, α and β are control parameters. The probability of selecting a path between source CM and exemplar node is expressed as follows:

$$\rho_k = \frac{\alpha + [P_f] * \beta}{\sum_{i=0}^{n} \alpha + [P_f] * \beta} \qquad (45)$$

Equation 45 indicates the probability of selecting path as a transmission path between source CM and exemplar node.

- **Revising rule**

In this rule, pheromone is updated in each iteration using following expression,

$$\tau_{i,j}(t+1) = (1-\varepsilon)\tau_{i,j}(t) + \varepsilon\Delta\tau_{i,j}(t) \qquad (46)$$

Where ε represents local pheromone corrosion and $\Delta\tau_{i,j}(t)$represents pheromone enhancement. By means of updating pheromone values in each iteration, ACO discovers multiple paths between source CM and exemplar node.

(ii) Fire fly algorithm

From the selected multiple paths, FFA elects best the path between source CM and exemplar. FFA work on the principle of flashing lights of fireflies. Here, FFA computes fitness function for succeeding metrics such as packet delivery ratio, expected delay and load.

Packet delivery ratio

Packet delivery ratio is described asthe ratio betweena number of packets successfully transmitted by the source node and number packets successfully received by the exemplar node. It is expressed '$\mathbb{d}_r$' as follows:

$$\mathbb{d}_r = \frac{p_s}{p_r} \qquad (47)$$

Where p_s represents packets successfully transmitted and p_r indicates packets successfully received.

Expected delay

Expected delay is described as the time required to transmit data from a source to the destination node. This parameter is taken to measure delay in the selected path since it affects energy consumption of the network. It is represented as '$\mathbb{E}_d$' that is expressed as follows:

$$\mathbb{E}_d = T(N_{CM_i}, N_e) \qquad (48)$$

Where, $T(N_{CM_i}, N_e)$represents the time required to transmit data between source CM node N_{CM_i} and exemplar node N_e.

Load

Load parameter is used to measure the load of each sensor node in the network. Load parameter is directly proportional to the energy consumption; hence the load of the sensor node increase automatically energy consumption also increases. This parameter is expressed as 'N_L' that can be conveyed as follows:

$$N_L = L(N_i) \qquad (49)$$

Where N_L represents the load of the node N_i. FFA computes fitness function for each node to find an effective path between source CM and exemplar node. Fitness function $f(x)$ is expressed as follows:

$$f(x) = \sum \frac{1}{E_d + N_L} + D_r \qquad (50)$$

The light intensity of each node is computed using the above explained fitness function $f(x)$. Light intensity 'I_l' is expressed as follows:

$$I_l = I_o e^{-\gamma r} \qquad (51)$$

Where I_o indicates original light intensity. The attractiveness of FFA is computed using below expression,

$$\omega = \omega_0 e^{-\gamma r^2} \qquad (52)$$

Where ω_0 indicates attractiveness at $r = 0$. Here, firefly moves to the best position from i to j by using equation 53,

$$N_i = N_i + \omega_0 e^{-\gamma r^2}(N_j - N_i) + \delta\,(rand - \frac{1}{2}) \qquad (53)$$

By using the above function firefly discover optimum path between source CM and exemplar node. From the above process, this work effectually reduces network delay that in turn enhances network lifetime.

3.6 Data storage

3.6.1 DATA LOG from sensor

Data from Sink Node to server will be communicated via Rest API. This API request end point takes the following parameters:

(i) Temperature

(ii) Humidity

(iii) Time

The request sent to server shows different statuses based on which it was a success or failure

```
Status:
1. Success
{
     "status": 200,
     "message": "Data log success"
}
2. Failure:
{
     "status": 202,
     "message": "Invalid Data"
}
```

Then send REST API request to the server. This request canbe performed via unix curl command as follows:

```
`curl  -H  "Content-Type:  application/json"  -X  POST  -d  '{"time":"150","temperature":"37",
"humidity":"33"}' https://wsn.duckdns.org/api/v1/log_data.json
{"status":200,"message":"Data log success"}%
Result:
   "status": 200,
    "message": "Data log success"
```

For the missing value of temperature, the codes will be displayed as follows:

```
`curl  -H  "Content-Type:  application/json"  -X  POST  -d  '{"time":"150","temperature":"",
"humidity":"33"}' https://wsn.duckdns.org/api/v1/log_data.json
{"status":200,"message":"Data log success"}%
Result:
    "status": 202,
    "message": "Invalid Data"
```

All the fresh weather data are sent wirelessly in ad-hoc fashion to a sink node, which in turn transmits data to the control center via a transport network such as 3G, 4G, Satellite, TCP/IP networks. This research uses real-time based networked system to forward the sensor data.

3.7 Experiment setup

3.7.1 Requirement for prototype

The following are equipmentused in this experiment:

(i) Raspberry Pi - Base station

(ii) Arduino UNO - Sensor nodes

(iii) DHT22 Modules - Temperature and Humidity sensor

(iv) XBee S2 - ZigBee radio communication module

(v) 2PCS New 9V Battery Holder Box Case Wire for Arduino UNO

(vi) RMS LCD Digital Auto Range Multimeter AC/DC Tester Meter

(vii) Male to Male + Male to Female + Female to Female Jumper Cable

(viii) ZigBee XBee Modules Shield V03 Modules

(ix) XBee Explorer USB Dongle

3.7.2 Configuration of ZigBee module

All ZigBee modules in this experiment are configured by connecting them to the Windows computerusing their serial adapter. The X-Ctu software has been used to configure the ZigBee module. The serial port of computer set to 9600 baudrate, 8 data bits, 1 stop bit and no parity.

3.7.3 Interfacing ZigBee with arduino

The following code helps to establish a communication between ZigBee modules and Arduino boards.

3.7.4 Lab setup design for weather monitoring

The lab setup comprises of the following components: Sensor nodes are equipped with the arduino UNO board, DHT22, ZigBee radio communication module, ZigBeeXBee Modules Shield V03 Modules, and the 9 V batteries. Base station is equipped with the Raspberry Pi, XBee Explorer USB Dongle, ZigBee radio communication module and interface to the task manager.

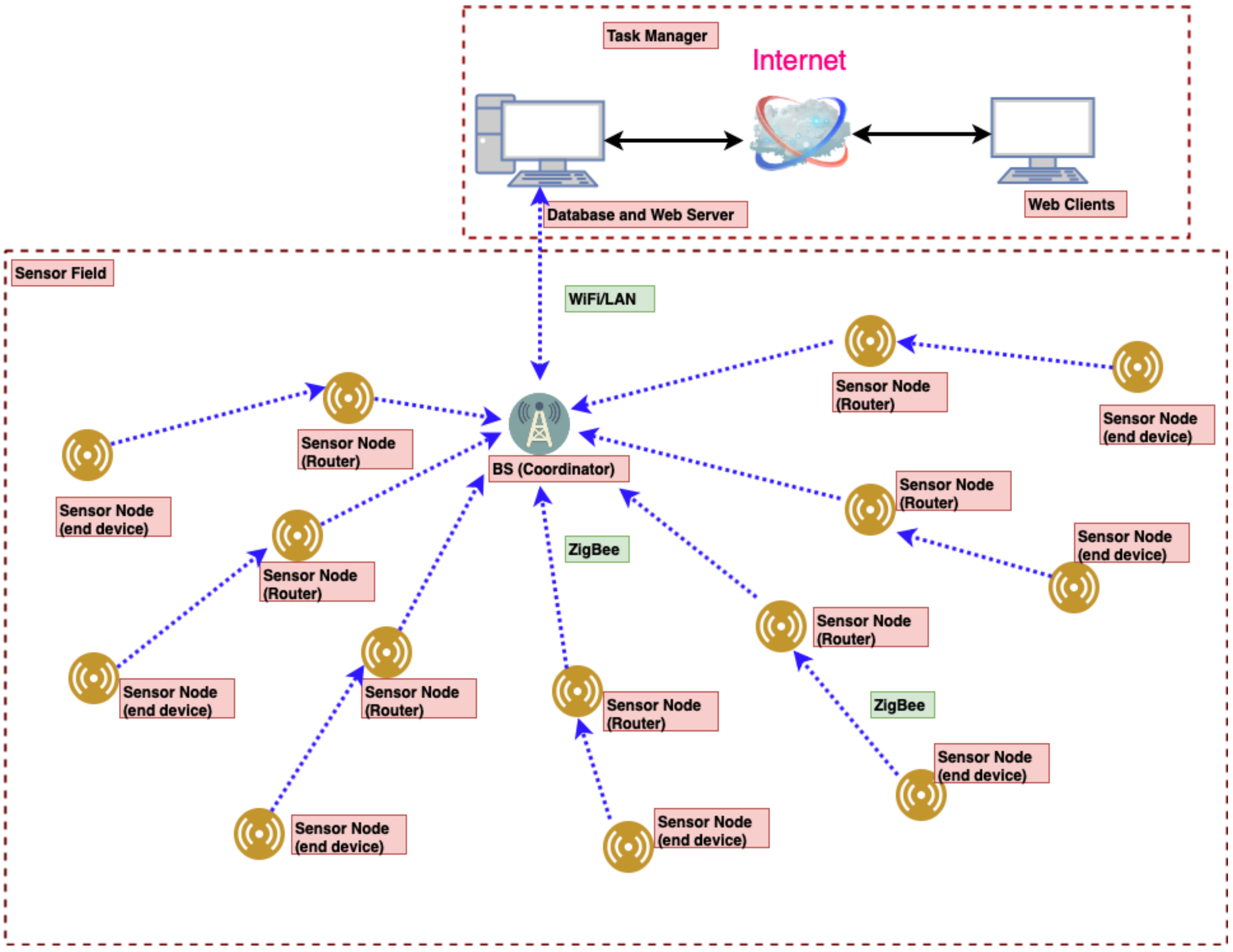

Figure 15: Lab setup design for weather monitoring

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Performance evaluation

This section describes the evaluation performance of the E^2S-DRL with existing methods using the network lifetime, energy consumption, throughput and the transmission delay. This section is further divided into three sections that are simulation environment, performance metrics and comparative analysis. The performance of the proposed work has been compared with iABC, cTDMA, LDC, DRA and MPACO methods.

4.1.1 Simulation environment

The proposed work is implemented in Network Simulator 3 (ns-3) tool that is installed in the Ubuntu operating system. NS-3 is a discrete event simulator that provides simulations of different types of network; hence we preferred this simulator for proposed energy efficient sleep scheduling in WSN environment.

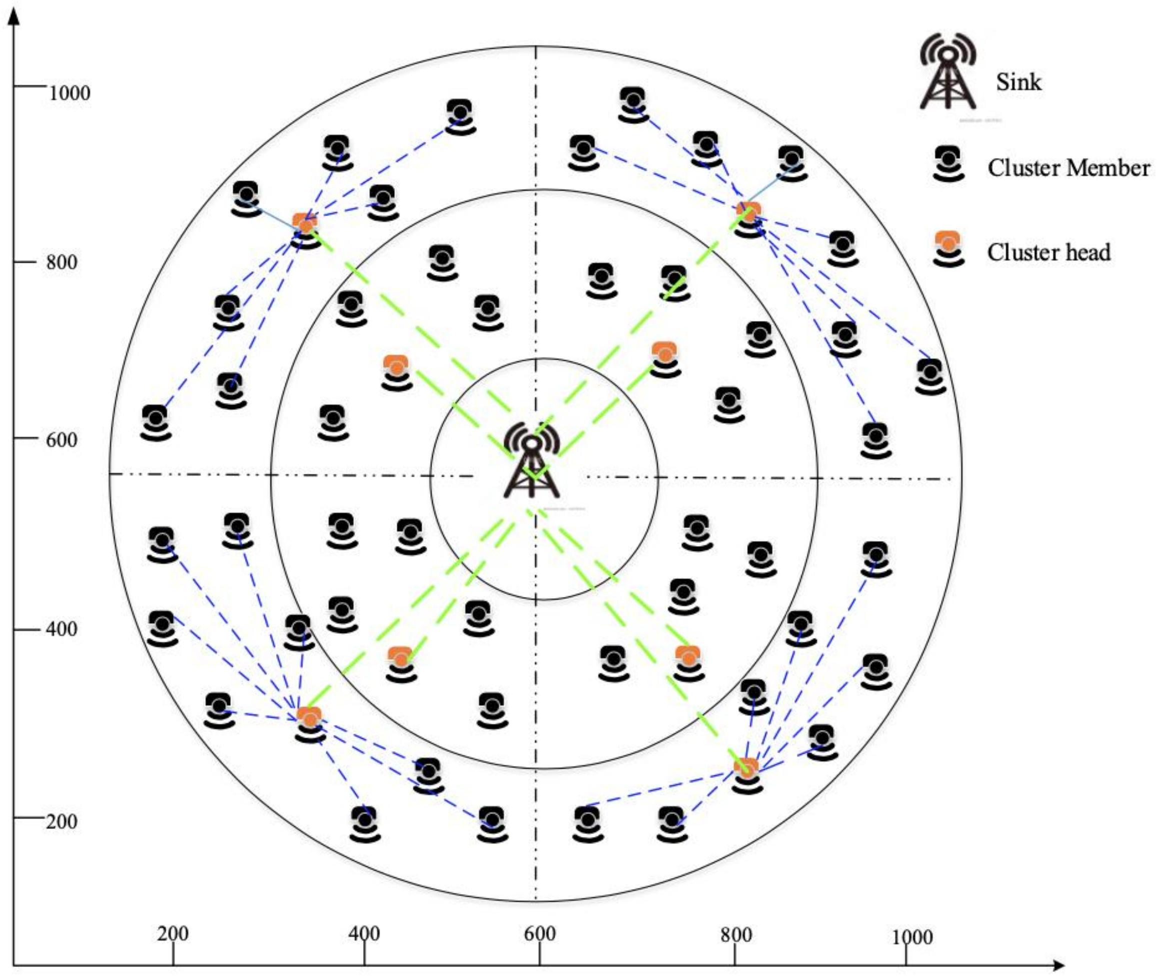

Figure 16: Simulation environment

Figure 16 illustrates the simulation environment of the proposed work. This simulation environment composed of 100 sensor nodes that are deployed in the three coronas. The sink node is positioned at the center of coronas; based on the sink node position each corona is further split into zones. Zones comprise of clusters in which each cluster contain one cluster head respectively.

Table 11: Simulation parameters

Parameters		Value
Network Parameters	Simulation Area	1000*1000m
	Number of sensor nodes	100
	Number of Sink node	1
	Initial Energy of the Node	1200J
Packet Parameters	Number of Packets	≈ 1000
	Number of retransmission	Max 7
	Packet size	512KB
	Packet Interval	0.1s
	Traffic Type of Packet	Constant Bit Rate (CBR)
Communication Parameters	Sensor Communication Range	100m
	Data Rate	20Mbps
Transmission Slot Parameters	Number of Slots	16
	Data Packet length	840bits
	Slot length	1050bits
	Slot Duration	10 s
PSO Parameters	Number of Particles	[20–80]
	Maximum Inertia Weight	0.9
	Minimum Inertia Weight	0.1
ACO Parameters	Number of Ants	100
	α	0.6
	β	0.6
FFA parameters	Firefly population	100
	ω	0.9
	γ	1
Number of Run		**1000**
Simulation time		**100s**

Table 11 illustrates the simulation parameter of the proposed work that composed of 1000*1000 m simulation area with 100 sensor node and one sink node. The positioned sensor nodes have 100 m communication range to transmit sensed information to neighbour nodes.

4.1.2 Performance metrics

The evaluation the performance of this work has been done using performance metrics such as network lifetime, energy consumption, throughput and delay that are summarized as follows:

(i) Network lifetime

Network lifetime metric is used to measure the lifetime of the sensing network. Typically, network lifetime is referred to as the time at which the first node dies in the network. It is also defined as the operational time of the node during which it can able to perform the allocated task. Network lifetime is represented as '$\mathcal{n}_{\ell}$' that can be conveyed as follows:

$$\mathcal{n}_{\ell} = \frac{\mathcal{I}_{\mathcal{E}} - \mathcal{w}_{e}}{\mathcal{C}_{p} + \mathfrak{a}_{r}.\mathcal{R}_{e}} \quad (54)$$

Where $\mathcal{I}_{\mathcal{E}}$ denotes initial energy of the network, $\mathcal{w}_{e}$ denotes wasted energy, $\mathcal{C}_{p}$. denotes continuous power consumption of the network, $\mathfrak{a}_{r}$ represents average sensor reporting rate, and $\mathcal{R}_{e}$ represents estimated reporting energy.

(ii) Energy consumption

Energy consumption metric is described as the amount of energy consumed over sensing, data transmitting and data receiving in the network. It is represented as $\mathfrak{E}_{\mathfrak{C}}$ that is represented as follows:

$$\mathfrak{E}_{\mathfrak{C}} = \sum (\mathcal{D}_{t} + \mathcal{D}_{r} + \mathfrak{s}_{f}) \quad (55)$$

Where $\mathcal{D}_{t}$ represents the energy consumed during data transmission,$\mathcal{D}_{r}$ represents the energy consumed during data receiving, and $\mathfrak{s}_{f}$ represents the energy consumed during sensing the field.

(iii) Throughput

Throughput metric is described as the number of packets successfully received by the receiver in the given amount of time. It is represented as '$\mathcal{T}$' that is conveyed as follows:

$$\mathcal{T} = \frac{\sum_{i=1}^{n} \mathfrak{p}_{i} * \mathfrak{p}_{\ell}}{s(t)} \quad (56)$$

Where, $\mathfrak{p}_{i}$ represents a number of packets in node 'i', $\mathfrak{p}_{\ell}$ denotes data packet length and $s(t)$represents simulation time.

(iv) Delay

Delay is defined as the time required to deliver sensed data from the source to the destination node. It is represented as '$\mathfrak{D}$' that is conveyed as follows:

$$D = \frac{T_s}{T_d} (57)$$

Where T_s represents the time required to send data and T_d represents data received by the destination node.

4.1.3 Comparative analysis

This section demonstrates a comparison on the proposed work performances with the existing methods iABC (Mann & Singh., 2018), LDC (Jiang *et al.,* 2017), cTDMA (Bahbahani & Alsusa, 2017) and DRA (Liu *et al*., 2017) methods. The contributions of these methods are similar to the contribution of the E^2S-DRLwork in WSN. Hence, these methods are selected to compare with the proposed work. From this comparison, it is clear that the proposed work is more efficient than existing methods that are discussed on the previous session.

(i) Analysis of network lifetime

Network lifetime is a significant metric to analyze the performance of the proposed work. It demonstrates the efficiency of a proposed work concerning the lifetime of the network. In this work, the network lifetime metric is measured with the number of nodes or simulation time. Figure 17 shows the proposed work achieves high network lifetime compared to the existing methods DRA, LDC, iABC and cTDMA. The proposed duty cycling method achieves better network lifetime through reduce energy consumption, it provides sleep schedule to the sensor nodes adaptively using DRL algorithm. Furthermore, ZbC method improves energy consumption via reduced energy consumption during data aggregation.

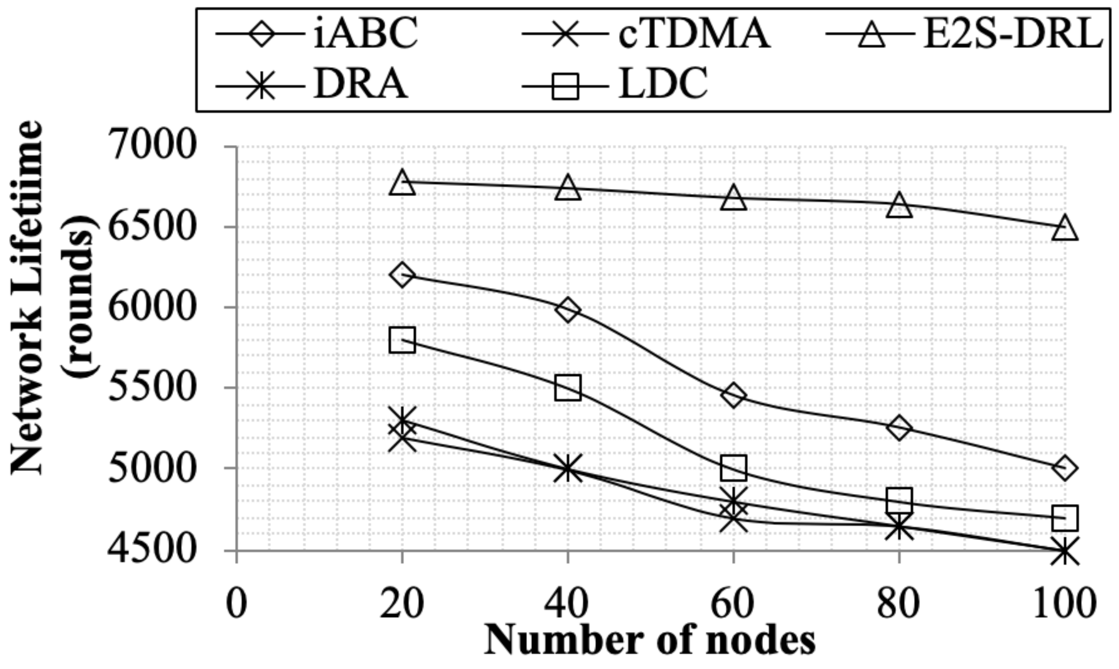

Figure 17: Network lifetime vs. Number of nodes

The proposed method achieves a maximum of 6780 rounds network lifetime in the presence of 100 nodes. Meanwhile, existing method iABC and cTDMA achieves minimum network lifetime for 100 nodes. Here, cTDMA and DRA methods have a minimum lifetime of 4600 rounds for 100 nodes. Figure 18 illustrates network lifetime with respect to the simulation time that demonstratesthe proposed work achieves better network lifetime compared to the existing methods DRA, LDC, iABC and cTDMA. The iABC method achieves minimum network lifetime compared to the E^2S-DRL, because iABC method takes more objective function to select optimum cluster head and it also lacks in using secondary information. cTDMA method has minimum network lifetime compared to the E^2S-DRL method, since it cannot change the scheduling slot dynamically which reduce energy of each sensor node drastically. Likewise, LDC and DRA method has low network lifetime compared to the E^2S-DRL method due to the lack of concentration on energy metric of sensornodes. These drawbacks pull down network lifetime minimum for DRA, LDC, iABC and cTDMA methods. Network lifetime is decrease dramatically, when simulation time increases. The proposed method achieves maximum average network lifetime of 6720 rounds for 100 sec of simulation time whereas existing method DRA, LDC, iABC and cTDMA methods minimum average network lifetime 5430 rounds, 5630 rounds, 5650 rounds and 4870 rounds for 100 sec of simulation time respectively.

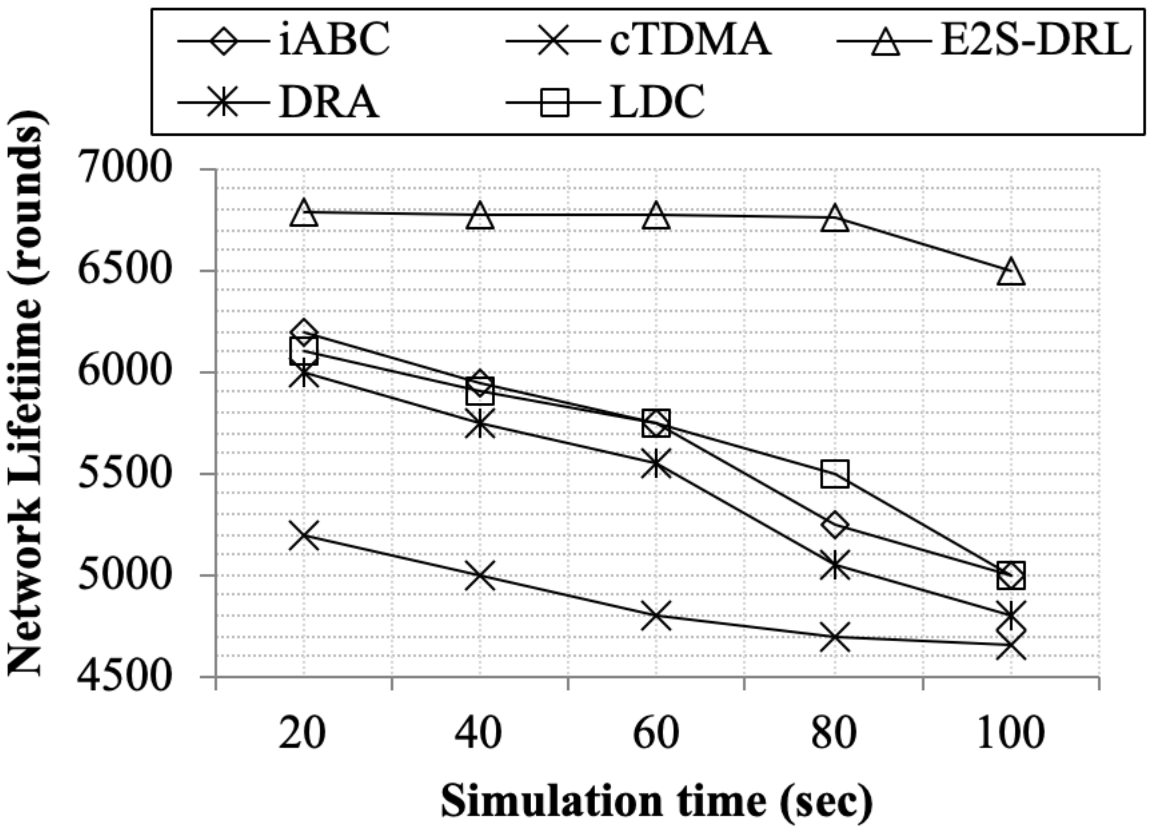

Figure 18: Comparison on Network Lifetime vs. Simulation time

(ii) Analysis of Energy Consumption

Energy consumption is an important metric to improve the lifetime of the network. Since, the energy consumption is inversely proportional to the network lifetime, if the energy consumption of the network increases, then automatically lifetime of the network decreases drastically. Thus, energy consumption metric is reduced as possible in the network. In this research, energy consumption of network is measured with the number of nodes or simulation time.

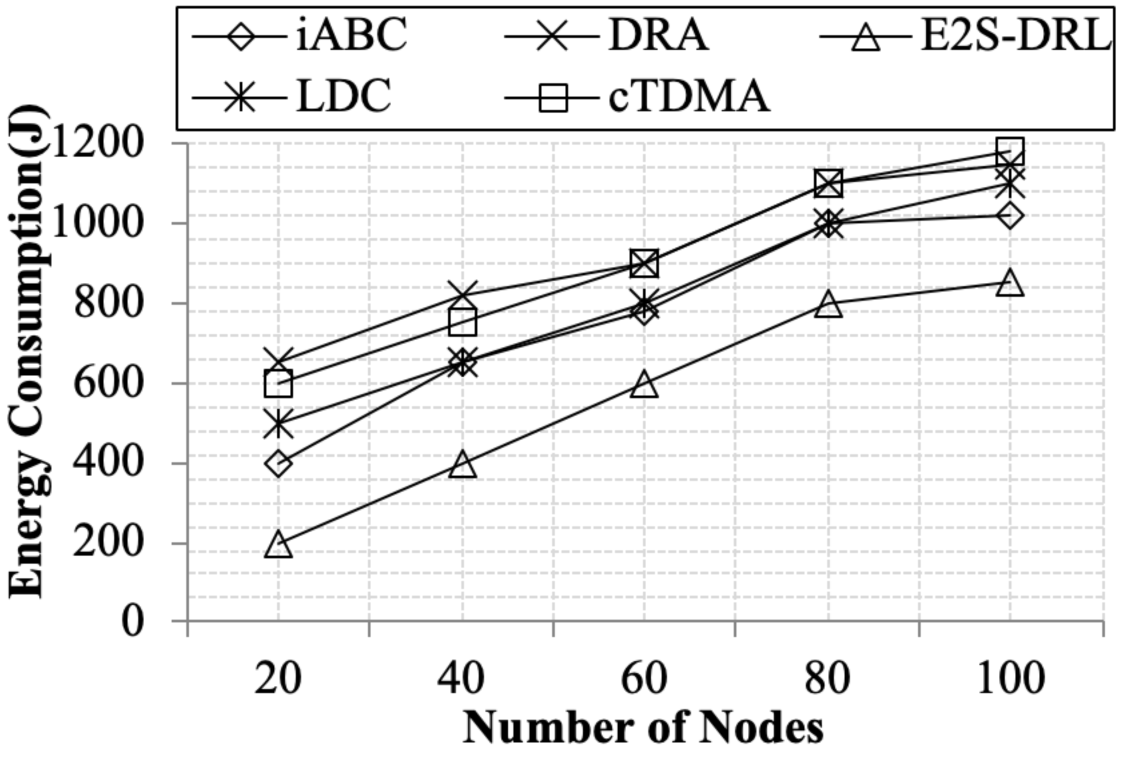

Figure 19: Comparison on Energy Consumption vs. Number of Nodes

Figure 19 illustrates the energy consumption with respect to the number of nodes, thus concludes proposed method achieves better energy consumption compared to the existing methods like LDC, cTDMA, iABC and DRA. The sensor nodes are deployed in coronas field to support data aggregation efficiently that reduces energy consumption. In addition to it, this research furthermore reduces energy consumption through duty cycling and ZbC schemes. In duty cycling, DRL algorithm is proposed which provides scheduling slot for each sensor node based on the energy and states. Zone based Clustering scheme reduce energy consumption through effective cluster head selection, since optimum cluster head can reduce energy consumption in data aggregation. These processes reduce energy consumption of network effectually. The proposed method achieves minimum of 200J and maximum of 850J for 100 nodes. Distributed Routing Algorithm (DRA) method consumes more energy, it consume maximum of 1800J for 100 nodes compared to the other existing methods.

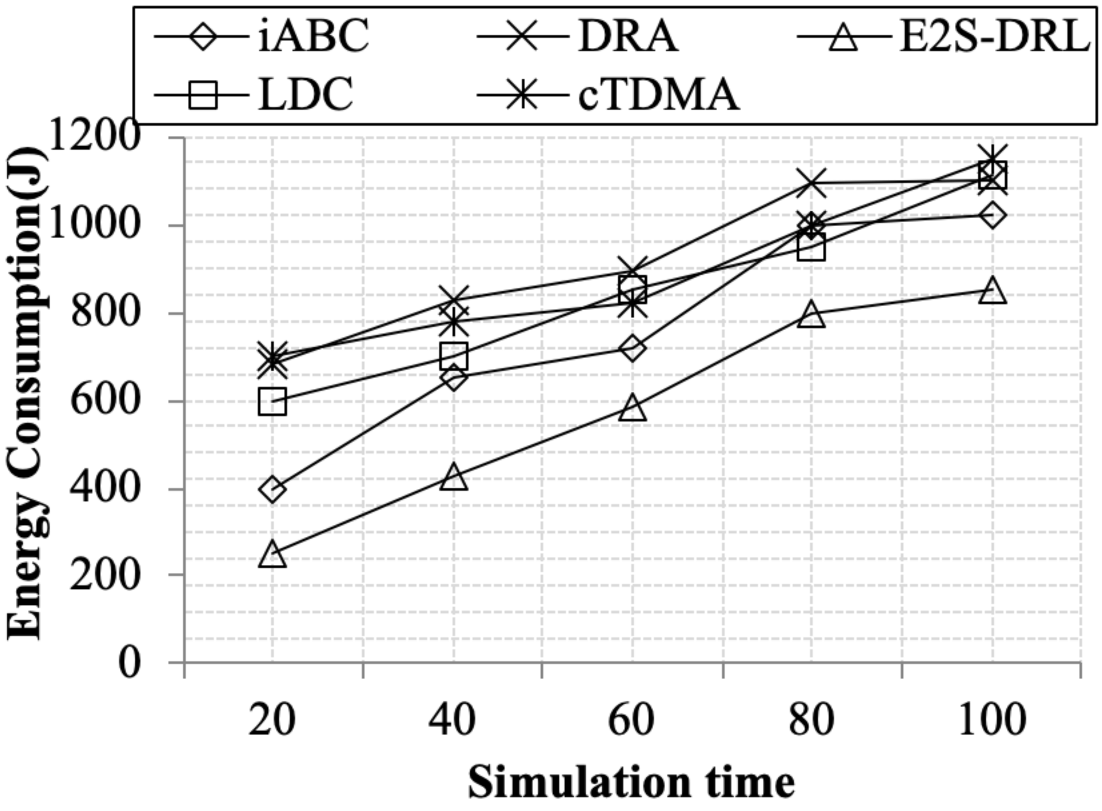

Figure 20: Comparison on Energy Consumption vs. Simulation time

Figure 20 demonstrates energy consumption with respect to the simulation time, thus concludes E^2S-DRL achieves reduced energy consumption compared to the simulation time. DRA method has more energy consumption, since it does not elect optimum cluster head to transmit sensed data. Meanwhile iABC method also consumes more energy due to its objective function evaluation. Likewise, LDC and cTDMA methods are achieved high energy consumption. The reason for this is that these methods do not concentrate on proper slot allocation for sleep and wakeup period of sensor nodes. These drawbacks lead to more energy consumption of DRA and iABC methods. The proposed method achieves average energy consumption of 583J for 100 sec of simulation time. Whereas, LDC, cTDMA, DRA and iABC methods consumes average energy of 842J, 890J, 920J and 757J for 100 sec of simulation time respectively. From the comparative results, it is clear that the proposed work achieves better energy consumption compared to the existing method cTDMA, LDC iABC and DRA.

(iii) Analysis of Throughput

Throughput is a metric related to the performance of the proposed network. Hence, throughput has increased as much as possible in this work. In performance evaluation throughput metric is measured using the number of nodes or number of rounds.

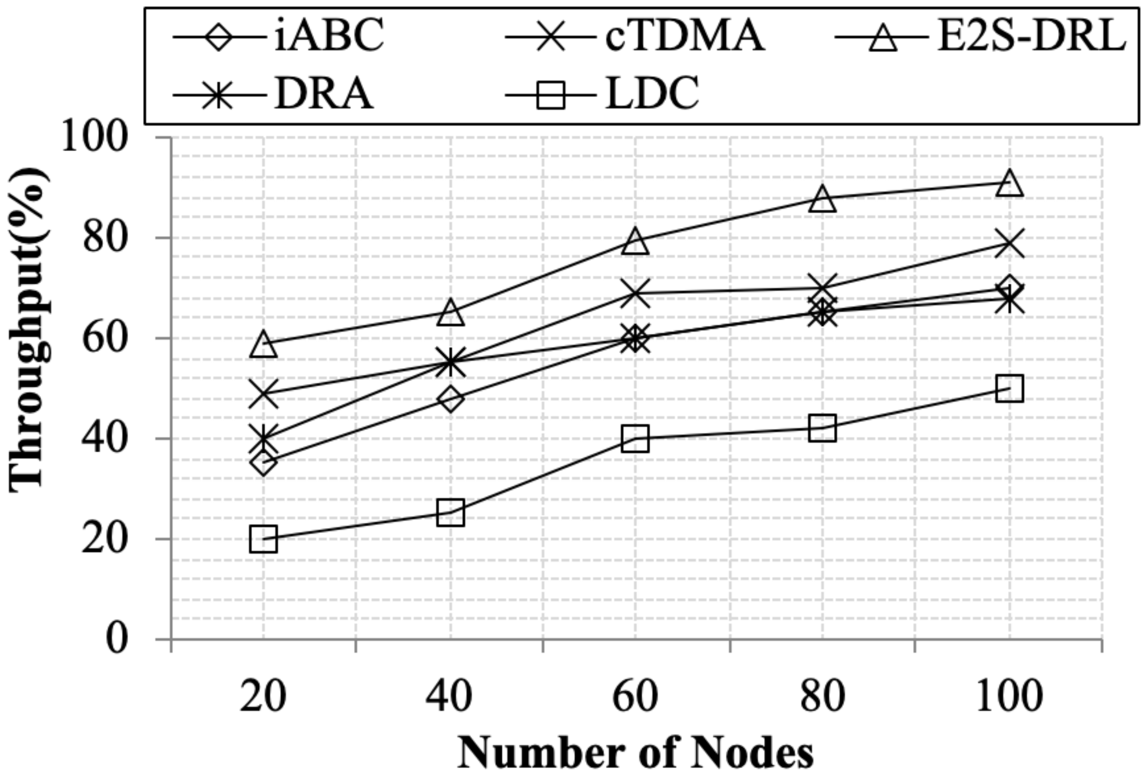

Figure 21: Comparison of throughput

In existing methods like iABC and cTDMA losses data packets in transmission due to poor path selection between the source and sink node. Hence, this research proposes an effective routing method that efficiently reduces data packet loss through minimizing delay in the network. Delay of the network reduced through effective selection of path between source and destination. Here, ACO and FFA are pursued to select an optimum path between source and destination. The ACO selects multiple paths between source and destination, from which FFA elects the best path between source and destination. This way of selecting path diminishes delay effectually that in turn improves the throughput of the proposed work. The proposed work achieves a maximum of 91% throughput for 100 nodes. This improved Artificial Bee Colony (iABC) achieves a maximum of 70%, and cTDMA method achieves a maximum of 79% for 100 nodes. Thus, the results show the proposed work achieves better throughput that in turn improves the performance of the network. Figure 21 describes that the

E^2S-DRL produces high throughput compared to the existing methods such as DRA, LDC, iABC and cTDMA. In existing methods like iABC and cTDMA losses data packets in transmission due to poor path selection between source and sink node. Likewise, DRA and LDC methods do not concentrate on the buffer related and link related metrics to route the data packet. Thus reduce the throughput of the WSN effectually. Hence, this study proposes an effective routing method that efficiently reduces data packet loss through minimizing delay in the network. Delay of the network reduced through effective selection of paths between source and destination. Here, the proposed work pursues ACO and FFA to select an optimum path between source and destination. Anti-Colony Optimization (ACO) selects multiple paths between source and destination, from which FFA elects best path between source and destination. This way of selecting path diminishes delay effectually that in turn improves throughput of the proposed work. The efficient scheduling using Deep Reinforcement Learning Achieves Maximum of 91% throughput for 100 nodes. Meanwhile, the existing methods DRA achieves maximum of 68%, LDC method achieves maximum of 50%, iABC achieves maximum of 70% and cTDMA method achieves maximum of 79% for 100 nodes. Thus, the results show E^2S-DRL achieves better throughput that in turn improves the performance of the network.

(iv) Analysis of Delay

The delay metric is significant for improving the lifetime of the network since it is directly proportional to the energy consumption and network lifetime metric. Thus, this work reduces delay as much as possible. Usually, delay metric is measured using the number of nodes or number of rounds.

I

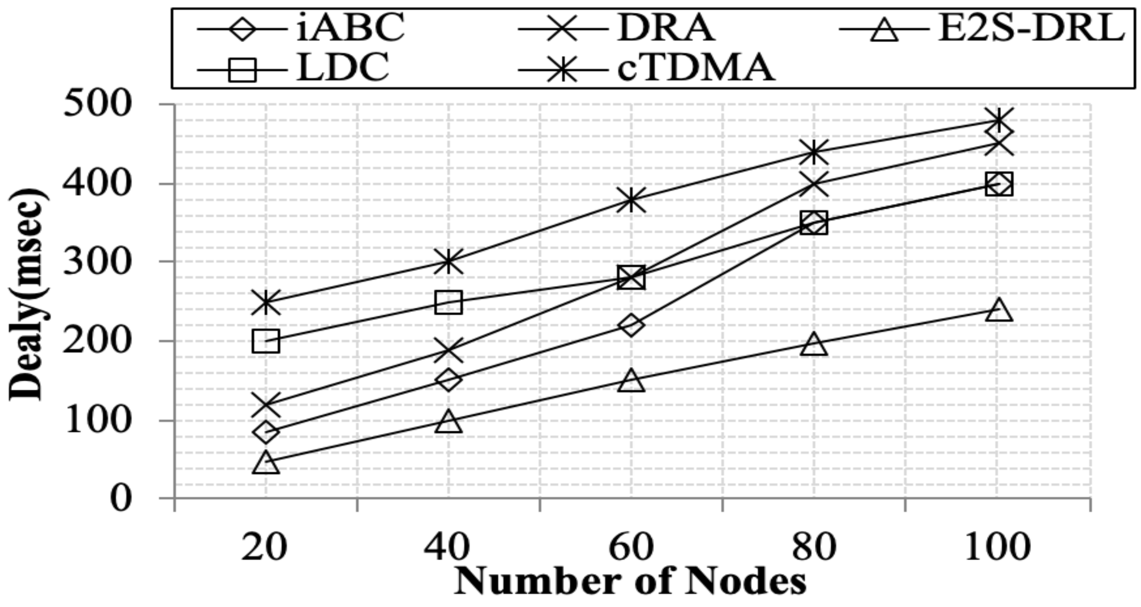

Figure 22: Comparison of Delay

Figure 22 demonstrates that the delay of the proposed work is minimum compared to the existing methods LDC, cTDMA, iABC and DRA. If the number of node increases, automatically delay also increase. The above comparison clearly shows E^2S-DRL transmits packets with minimum delay. Since, this study proposes efficient routing and scheduling schemes. The proposed scheduling scheme reduces delay through allocating slots based on the packets in its buffer. Furthermore, the delay is reduced through effective path selection between source and destination. Here, ACO selects multiple paths between source and destination and FFA selects best path among multiple paths that in turn reduces delay effectively. The E^2S-DRL achieves minimum of 240 msec for 100 nodes. The cTDMA and DRA method have poor data transmission; since they have maximum delay of 450 and 480 msec for 100 nodes respectively due to the lack of consideration of distance and buffer related metrics for comparison. Likewise, iABC and LDC methods also acquire high delay as 400 msec for 100 nodes. It is due to slow processing performance of the utilized algorithms. From the above comparison, it can be concluded that the proposed method achieves better data transmission with minimum delay in the network.

(v) Memory consumption analysis

This analyzes the performance of the proposed work with the memory consumption. Also, it compares the memory consumption of the proposed work with the existing methods including iABC, LDC, DRA and cTDMA. Memory is measured with the aid of varying the number of nodes in the network. As represented in Fig. 23, E^{2}S-DRL achieves better results in the memory or space consumption compared to the other existing methods. This is achieved because of the proposed algorithms for clustering, duty cycling and routing. The proposed algorithms including PSO-AP, DRL and ACO-FFA are consuming less memory to select optimal cluster head, mode of operation and optimal path. The routing selects the path with high energy and delay to transmit data to the destination. Thus reduces the memory consumption of the sensor node effectually. Meanwhile, the existing methods acquire high memory consumption due to its poor processing performance of the selected algorithms. It leads to high memory consumption in each sensor node in the network. Efficient Scheduling Using Deep Reinforcement Learning (E^{2}S-DRL) method reduces 40% in memory consumption compared to the existing methods including iABC, DRA, cTDMA and LDC.

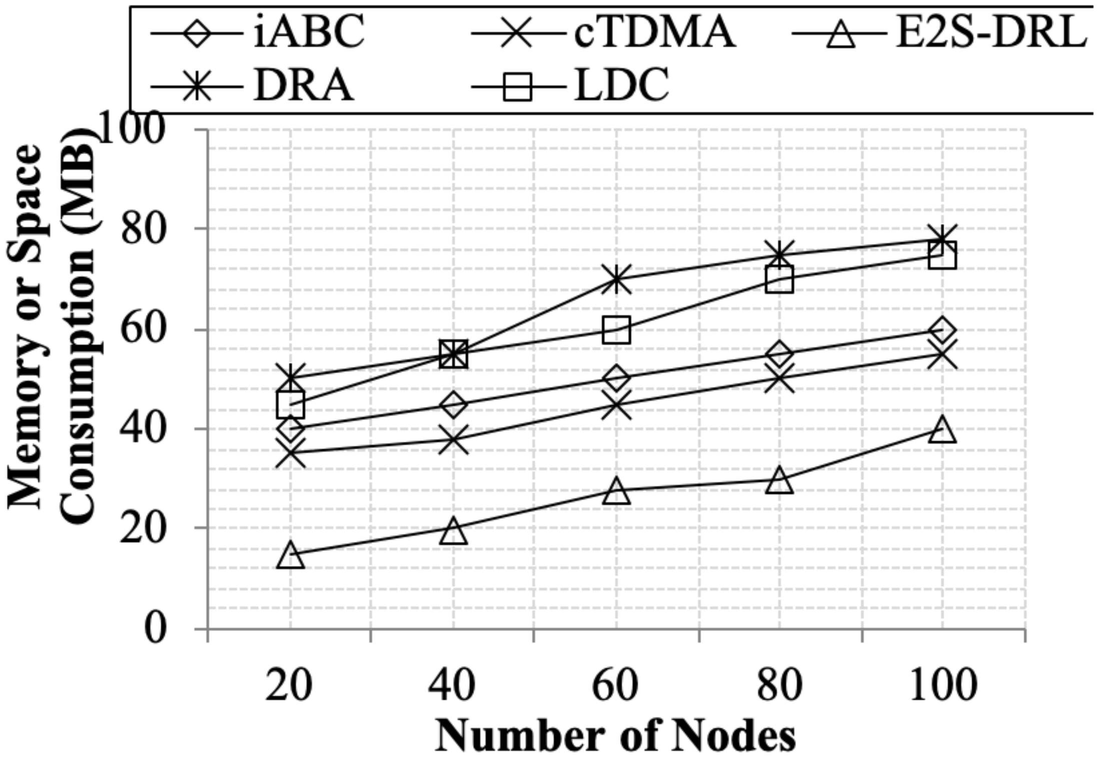

Figure 23: Comparison of memory

(vi) Computational complexity analysis

This portion discusses the performance of the proposed work in terms of the complexity. Here, the time complexity of proposed optimization algorithm and reinforcement algorithm compared with the existing iABC and cTDMA algorithms.

Table 12: Time complexity analysis

Process	Method	Time Complexity
Clustering	iABC	$O(2 * {}^{P}/_{2} * D * I^2)$
	PSO-AP	$\mathbf{O(P * I + n^2)}$
Duty Cycling	cTDMA	$3O(n)$
	DRL	$\mathbf{O(n)}$
Routing	LEACH	$nO(L)$
	ACO-FFA	$\mathbf{O(n^2 + nlogn)}$

From the Table 12, it has been proved that the computation complexity of the E^{2}S-DRL is less than the other existing methods present in the clustering, duty cycling and routing. In clustering, the existing iABC has $O(2 * {}^{P}/_{2} * D * I^2)$where P represents the number of population, D represents the dimension and I represents the number of iteration. Meanwhile, the E^{2}S-DRL method has less complexity as $0(P * I + n^2)$. For duty cycling, cTDMA has time complexity of three time of the number of operations to be performed i.e. $3O(n)$. Here, the n represents the number of operations. The E^{2}S-DRL has only $O(n)$ operations to perform duty cycling. Likewise, LEACH routing has the $nO(L)$ operations to complete the routing in WSN where E^{2}S-DRL method only consumes $O(n^2 + nlogn)$ time to complete the routing. From the above analysis, it is clear that E^{2}S-DRL achieves less computational complexity compared to the existing algorithms. The overall time complexity of the proposed work is $O(n^2 + nlogn)$.

4.1.4 Benefits of the proposed energy efficient approach

The algorithms incorporated in the aforesaid phases are discussed with its benefits in terms of the performance metrics in Table 13.

Table 13: Benefits of proposed algorithms

Proposed Algorithm	Benefits Related to Performance Metrics
PSO-AP	The PSO selects the optimal exemplar for the AP algorithm based on the energy-related metrics. Hence, the selected cluster head has the ability to sustain for a long time. This reduces the energy drain of the cluster head effectively. Therefore, the E^2S-DRL method enhances energy consumption in WSN. Besides, the proposed AP algorithm forms clusters quickly compared to the traditional clustering algorithms such as K-means, etc. Thus, this reduces the time required during the setup phase.
DRL	The proposed DRL algorithm provides the proper mode of operation for each sensor node. For this purpose, it considers the buffer size parameter of each sensor node. Based on the buffer size, it allocates the modes to each node. This results in the reduction of energy consumption which tends to increase in the network lifetime drastically. Besides, it also reduces the network delay by considering the buffer size-based mode of operation allocation.
ACO-FFA	The throughput of the proposed work is increased by selecting an optimal path from the multiple paths. Here, ACO selects the best multiple paths for transmission with less amount of time. From the selected multiple paths, FFA selects the optimal path to transmit the message between the source and sink node. Here, expected delay, PDR and load parameters are considered to achieve less delay during packet transmission. These metrics also increase the throughput of the proposed system.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In this research, a cluster-based network to collect forest's climatological data using WSN has been proposed. To improve network lifetime and reduce the network delay, this study proposes energy efficient sleep scheduling using the DRL algorithm (E^2S-DRL). The proposed method comprises three major phases such as Clustering, Duty cycling and Routing. At first, clustering operation is functioned via the ZbC scheme that is executed through hybrid PSO and AP algorithm that reduce energy consumption during data aggregation. Here, PSO is used to select the optimal exemplar to form the AP clusters. In duty cycling phase, the DRL algorithm is proposed that effectively schedules each node based on the energy and state adaptively that in turn improves network lifetime via reduced energy consumption and also it reduces data collision among sensor nodes. The routing is adopted to reduce the network delay that is executed by employing ACO and FFA. Here, the ACO selects the multiple paths between source and destination node. From the multiple paths, the FFA selects the optimal path to transmit packet to the sink node. At last, the performance of the proposed solution was evaluated by implementing it in a simulation environment. The simulations are conducted using ns-3.26 and to get precise plots, confidence interval is taken. The performance of E^2S-DRL has been compared with existing methods like LDC, iABC, cTDMA and DRA using network lifetime, throughput, energy consumption and delay metrics. It is concluded that, E^2S-DRL method reduces 40% in delay and energy consumption and increases 35% in network lifetime and throughput compared to the existing methods. The climatological data will be stored in a database for further analysis.

5.2 Recommendations

This research could be done differently by extending it to monitor and detect wildfire with enough money and time. Different types of sensor nodes for soil moisture, pressure, smoke would be needed to predict fire. The proposed solution can be integrated with a framework used for fire detection. The detection of wildfire might be done by setting threshold values of the weather parameters etc. Implementation of the data science techniques and machine learning concept would be better for the analysis of the collected weather data and fire prediction. Also, this research can be integrated with other systems for precision agriculture in huge farms, since network lifetime and delay have been carefully considered. The number of nodes deployed in the present study was relatively small. A more significant amount of sensor nodes would probably give a clearer picture of the situation. Therefore, future studies may attempt to use a larger number of sensor nodes. Also, the deployment issues, network security and data quality still open for further research. As it is known that, the forest is a large and harsh area which is not easily accessible for nodes deployment. The random deployment of sensors by dropping them from the helicopter or drones would fit the purpose than trying to use a deterministic approach.

REFERENCES

Abed, A., Alkhatib, A., & Baicher, G. S. (2012). Wireless Sensor Network Architecture. *International Conference on Computer Networks and Communication Systems*, *35*, 11–15.

Ahmad, A., & Hanzálek, Z. (2017). Distributed Real Time TDMA Scheduling Algorithm for Tree Topology WSNs. *IFAC-PapersOnLine*, *50*(1), 5926–5933. https://doi.org/10.1016 /j.ifacol.2017.08.1484

Ahmad, M., Ikram, A. A., Wahid, I., Inam, M., Ayub, N., & Ali, S. (2018). A bio-inspired clustering scheme in wireless sensor networks: BeeWSN. *Procedia Computer Science*, *130*, 206–213. https://doi.org/10.1016/j.procs.2018.04.031

Akkaya, K., & Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, *3*(3), 325–349. https://doi.org/10.1016/J.ADHOC.2003. 09.010

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, *38*(4), 393–422. https://doi.org/10.1016/S1389-1286(01)00302-4

Al-Habashneh, A. A. Y., Ahmed, M. H., & Husain, T. (2011). Reliability analysis of Wireless Sensor Networks for forest fire detection. *2011 7th International Wireless Communications and Mobile Computing Conference*, 1630–1635. https://doi.org/10. 1109/IWCMC.2011.5982779

Al-Karaki, J. N., & Kamal, E. (2004). Wireless Sensor Networks Routing Techniques in Wireless Sensor Networks : A Survey. *IEEE Wireless Communications*, *11*, 6–28. https://doi.org/10.1109/MWC.2004.1368893

Alajlan, A., Dasari, B., Nossire, Z., Elleithy, K., & Pande, V. (2012). Topology Management in Wireless Sensor Network: Multi-state Algorithms. *International Journal of Wireless & Mobile Networks*, *4*(6). https://doi.org/10.5121/ijwmn.2012.4602

Alkhatib, A. A. A. (2014). A review on forest fire detection techniques. *International Journal of Distributed Sensor Networks*, 2014. https://doi.org/10.1155/2014/597368

Almazeed, A., Evaluation, A. A. P., & 2013, U. (2016). The Science and Information Organization (SAI) ®. *Citeseer*, 1–2. Retrieved from www.ijacsa.thesai.org

Anastasi, G., Conti, M., Di Francesco, M., & Passarella, A. (2009). Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, *7*(3), 537–568. https://doi.org/10.1016/j.adhoc.2008.06.003

Antoine-Santoni, T., Santucci, J. F., De Gentili, E., Silvani, X., & Morandini, F. (2009). Performance of a Protected Wireless Sensor Network in a Fire. Analysis of Fire Spread and Data Transmission. *Sensors*, *9*, 5878–5893. https://doi.org/10.3390/s90805878

Arora, V. K., Sharma, V., & Sachdeva, M. (2019). ACO optimized self-organized tree-based energy balance algorithm for wireless sensor network: AOSTEB. *Journal of Ambient Intelligence and Humanized Computing*, *10*(12), 4963–4975. https://doi.org/10.1007/s12652-019-01186-5

Arora, V. K., Sharma, V., & Sachdeva, M. (2020). A multiple pheromone ant colony optimization scheme for energy-efficient wireless sensor networks. *Soft Computing*, *24*(1), 543–553. https://doi.org/10.1007/s00500-019-03933-4

Aslan, Y. E. Y., Korpeoglu, I., & Ulusoy, Ö. (2012). A framework for use of wireless sensor networks in forest fire detection and monitoring. *Computers, Environment and Urban Systems*, *36*(6), 614–625. https://doi.org/10.1016/j.compenvurbsys.2012.03.002

Awang, A., & Suhaimi, M. (2007). RIMBAMON©: A forest monitoring system using wireless sensor networks. *Intelligent and Advanced Systems*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4658555

Bahbahani, M. S., & Alsusa, E. (2017). A Cooperative Clustering Protocol With Duty Cycling for Energy Harvesting Enabled Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, *1*. https://doi.org/10.1109/TWC.2017.2762674

Bandyopadhyay, S., & Coyle, E. J. (2003). An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks. In *ieeexplore.ieee.org*. Retrieved from https://ieeexplore.ieee.org/abstract/document/1209194/

Baranov, A., Spirjakin, D., Akbari, S., Somov, A., & Passerone, R. (2016). “Perpetual” operation of CO wireless sensor node with hybrid power supply. *Sensors and Actuators A*, *238*, 112–121. https://doi.org/10.1016/j.sna.2015.12.004

Bayo, A., Antolín, D., Medrano, N., Calvo, B., & Celma, S. (2010). Early detection and monitoring of forest fire with a wireless sensor network system. *Procedia Engineering*, *5*, 248–251. https://doi.org/10.1016/j.proeng.2010.09.094

Bernardo, L., Oliveira, R., Tiago, R., & Pnto, P. (2007). A fire monitoring application for scattered wireless sensor networks : A peer-to-peer cross-layering approach. *WINSYS 2007 - International Conference on Wireless Information Networks and Systems, Proceedings*, 189–196. https://doi.org/10.5220/0002149701730180

Bouabdellaha, K., Noureddine, H., & Larbi, S. (2013). Using wireless sensor networks for reliable forest fires detection. *Procedia Computer Science*, *19*, 794–801. https://doi.org/10.1016/j.procs.2013.06.104

Boyinbode, O., Le, H., & Takizawa, M. (2011). A survey on clustering algorithms for wireless sensor networks. In *International Journal of Space-Based and Situated Computing*, 1. Retrieved from https://ieeexplore.ieee.org/abstract/document/5636152/

Chetan, S., & Potluri, A. (2009). LBAA: Level based address auto-configuration for ad hoc networks. *5th International Conference on Wireless Communication and Sensor Networks, WCSN-2009*, 94–99. https://doi.org/10.1109/WCSN.2009.5434797

Chlamtac, I., Conti, M., & Liu, J. J. N. (2003). Mobile ad hoc networking: Imperatives and challenges. *Ad Hoc Networks*, *1*(1), 13–64. https://doi.org/10.1016/S1570-8705(03) 00013-1

Choudhuri, R., & Das, R. K. (2019). Efficient Area Coverage in Wireless Sensor Networks Using Optimal Scheduling. *Wireless Personal Communications*, *107*(2), 1187–1198. https://doi.org/10.1007/s11277-019-06331-z

Choudhury, S., Kuchhal, P., Singh, R., & Anita, G. (2015). ZigBee and bluetooth network based sensory data acquisition system. *Procedia Computer Science*, *48*(C), 367–372. https://doi.org/10.1016/j.procs.2015.04.195

Chow, B. S. (2009). A limited resources-based approach to coding for wireless video sensor networks. *IEEE Sensors Journal*, *9*(9), 1118–1124. https://doi.org/10.1109/JSEN.2009.2026518

Datta, D., & Kundu, S. (2008). An application-specific reliable data transfer protocol in wireless sensor networks. *Journal of Networks*, *3*(6), 69–81. https://doi.org/10.4304/jnw.3.6.69-81

Dehghani, S., Barekatain, B., & Pourzaferani, M. (2018). An Enhanced Energy-Aware Cluster-Based Routing Algorithm in Wireless Sensor Networks. *Wireless Personal Communications*, *98*(1), 1605–1635. https://doi.org/10.1007/s11277-017-4937-1

Dehwah, A. H., Elmetennani, S., & Claudel, C. (2017). An energy estimation and forecast scheme for solar powered wireless sensor networks. *Journal of Network and Computer Applications*, *90*, 17–25. https://doi.org/10.1016/j.jnca.2017.04.003

Díaz-Ramírez, A., Tafoya, L. A., Atempa, J. A., & Mejía-Alvarez, P. (2012). Wireless Sensor Networks and Fusion Information Methods for Forest Fire Detection. *Procedia Technology*, *3*(4322), 69–79. https://doi.org/10.1016/j.protcy.2012.03.008

Dondi, D., Bertacchini, A., Brunelli, D., Larcher, L., & Benini, L. (2008). Modeling and optimization of a solar energy harvester system for self-powered wireless sensor networks. *IEEE Transactions on Industrial Electronics*, *55*(7), 2759–2766. https://doi.org/10.1109/TIE.2008.924449

Edla, D. R., Kongara, M. C., & Cheruku, R. (2019). A PSO Based Routing with Novel Fitness Function for Improving Lifetime of WSNs. *Wireless Personal Communications*, *104*(1), 73–89. https://doi.org/10.1007/s11277-018-6009-6

el Khediri, S., Kachouri, A., & Nasri, N. (2011). Diverse synchronization issues in wireless sensor networks. *ICM 2011 Proceeding*, 1–6. https://doi.org/10.1109/ICM.2011.6177374

Elhabyan, R., Shi, W., & St-Hilaire, M. (2018). A Pareto optimization-based approach to clustering and routing in Wireless Sensor Networks. *Journal of Network and Computer Applications*, *114*, 57–69. https://doi.org/10.1016/j.jnca.2018.04.005

Elshrkawey, M., Elsherif, S. M., & Elsayed, W. M. (2018). An Enhancement Approach for Reducing the Energy Consumption in Wireless Sensor Networks. *Journal of King Saud University - Computer and Information Sciences*, *30*(2), 259–267. https://doi.org/10.1016/j.jksuci.2017.04.002

Franceschinis, M., Pastrone, C., Spirito, M. A., & Borean, C. (2013). On the performance of ZigBee Pro and ZigBee IP in IEEE 802.15.4 networks. *International Conference on Wireless and Mobile Computing, Networking and Communications*, 83–88. https://doi.org/10.1109/WiMOB.2013.6673344

Frezzetti, A., Manfredi, S., & Pagano, M. (2015). A design approach of the solar harvesting control system for wireless sensor node. *Control Engineering Practice*, *44*, 45–54. https://doi.org/10.1016/j.conengprac.2015.07.004

Gao, Z., & Huang, L. (2015). A Forest Fire Monitoring and Early Warning System Based on the Technology of Multi-sensor and Multilevel Data Fusion. *Atlantis-Press.Com*. https://doi.org/10.2991/icecee-15.2015.126

Gherbi, C., Aliouat, Z., & Benmohammed, M. (2017). A survey on clustering routing protocols in wireless sensor networks. *Sensor Review*, 37, 12–25. https://doi.org/10.1108/SR-06-2016-0104

Gupta, G., & Younis, M. (2003). Performance evaluation of load-balanced clustering of wireless sensor networks. *10th International Conference on Telecommunications, 2*, 1577–1583. https://doi.org/10.1109/ICTEL.2003.1191669

Halawani, S., & Khan, A. W. (2010). Sensors Lifetime Enhancement Techniques in Wireless Sensor Networks - A Survey. *Journal of Computing*, *2*(5), 34–47.

Handy, M. J., Haase, M., & Timmermann, D. (2002). Low energy adaptive clustering hierarchy with deterministic cluster-head selection. *2002 4th International Workshop on Mobile and Wireless Communications Network, 2002*, 368–372. https://doi.org/10.1109/MWCN.2002.1045790

Hefeeda, M., & Bagheri, M. (2008). Forest fire modeling and early detection using wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, *7*, 169–224. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Forest+fire+modeling +and+early+detection+using+wireless+sensor+networks#0

Hefeeda, M., & Bagheri, M. (2009). Forest fire modeling and early detection usingwireless sensor networks. *Ad-Hoc and Sensor Wireless Networks*, *7*(3–4), 169–224. Retrieved from https://www.researchgate.net/publication/220419284

Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, *1*(c), 10. https://doi.org/ 10.1109/HICSS.2000.926982

Heinzelman, W. B., Chandrakasan, A. P., & Balakrishnan, H. (2002). An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Transactions On Wireless Communications*, *1*(4). https://doi.org/10.1109/TWC.2002.804190

Heinzelman, W. B. (2000). Application-Specific Protocol Architectures for Wireless Networks. *Cambridge*, (PhD Dissertation). Retrieved from http://citeseerx.ist.psu.edu/ viewdoc/download?doi=10.1.1.27.5727&amp;rep=rep1&amp;type=pdf

Hoblos, G., Staroswiecki, M., & Aitouche, A. (2000). Optimal design of fault tolerant sensor networks. *IEEE Conference on Control Applications - Proceedings*, *1*, 467–472. https://doi.org/10.1109/cca.2000.897468

Hoebeke J., Moerman, I., Dhoedt, B., & Demeester, P. (2004). An overview of mobile ad hoc networks: Applications and challenges. *Journal of the Communications Network*, *3*(3), 60–66. https://doi.org/10.1109/MPRV.2009.2

Hong, C., Zhang, Y., Xiong, Z., Xu, A., Chen, H., & Ding, W. (2018). Circular/Spherical Sector based Forwarding Area Division and Adaptive Forwarding Area Selection routing protocol in WSNs. *Ad Hoc Networks*, *70*, 121–134. https://doi.org/10. 1016/j.adhoc.2017.11.013

Horng, G. J., Chang, T. Y., & Cheng, S. T. (2014). An effective node-selection scheme for the energy efficiency of solar-powered WSNs in a stream environment. *Expert Systems with Applications*, *41*(7), 3143–3156. https://doi.org/10.1016/j.eswa.2013.11.016

Huang, W., Ling, Y., & Zhou, W. (2018). An Improved LEACH Routing Algorithm for Wireless Sensor Network. *International Journal of Wireless Information Networks*, *25*(3), 323–331. https://doi.org/10.1007/s10776-018-0405-4

Ibrahim, R., Chung, T. D., Hassan, S. M., Bingi, K., & Salahuddin, S. K. B. (2017). Solar Energy Harvester for Industrial Wireless Sensor Nodes. *Procedia Computer Science*, *105*, 111–118. https://doi.org/10.1016/j.procs.2017.01.184

Jalali, F., Khodadoustan, S., & Ejlali, A. (2012). Cooperative hybrid ARQ in solar powered wireless sensor networks. *Microelectronics Reliability*, *52*(12), 3043–3052. https://doi.org/10.1016/j.microrel.2012.07.009

Jiang, C., Li, T. S., Liang, J. B., & Wu, H. (2017). Low-latency and energy-efficient data preservation mechanism in low-duty-cycle sensor networks. *Sensors (Switzerland)*, *17*(5). https://doi.org/10.3390/s17051051

Kang, B., Nguyen, P. K. H., Zalyubovskiy, V., & Choo, H. (2017). A Distributed Delay-Efficient Data Aggregation Scheduling for Duty-Cycled WSNs. *IEEE Sensors Journal*, *17*(11), 3422–3437. https://doi.org/10.1109/JSEN.2017.2692246

Kang, J., Sohn, I., & Lee, S. H. (2018). Enhanced Message-Passing Based LEACH Protocol for Wireless Sensor Networks. *Sensors (Basel, Switzerland)*, *19*(1), 1–17. https://doi.org/10.3390/s19010075

Kanta, P., & Prasad, A. M. (2015). Area Coverage Redundancy and Node Positioning in Wireless Sensor Networks. In *International Journal of Computer Applications*, 111. Retrieved from www.ijcaonline.org

Kaur, S., & Mahajan, R. (2018). Hybrid meta-heuristic optimization based energy efficient protocol for wireless sensor networks. *Egyptian Informatics Journal*, *19*(3), 145–150. https://doi.org/10.1016/j.eij.2018.01.002

Ke, W., Yangrui, O., Hong, J., Heli, Z., & Xi, L. (2016). Energy aware hierarchical cluster-based routing protocol for WSNs. *Journal of China Universities of Posts and Telecommunications*, *23*(4), 46–52. https://doi.org/10.1016/S1005-8885(16)60044-4

Kosunalp, S., & Cihan, A. (2017). Harvesting solar energy for limited-energy problem in wireless sensor networks. *2017 25th Signal Processing and Communications Applications Conference, SIU 2017*. https://doi.org/10.1109/SIU.2017.7960535

Kozłowski, A., & Sosnowski, J. (2019). Energy Efficiency Trade-Off Between Duty-Cycling and Wake-Up Radio Techniques in IoT Networks. *Wireless Personal Communications, 107*(4), 1951–1971. https://doi.org/10.1007/s11277-019-06368-0

Kulkarni, N., Prasad, N. R., & Prasad, R. (2018). Q-MOHRA: QoS Assured Multi-objective Hybrid Routing Algorithm for Heterogeneous WSN. *Wireless Personal Communications, 100*(2), 255–266. https://doi.org/10.1007/s11277-017-5064-8

Kuorilehto, M., Hännikäinen, M., & Hämäläinen, T. D. (2005). A survey of application distribution in wireless sensor networks. *Eurasip Journal on Wireless Communications and Networking*, 2005, 774–788. https://doi.org/10.1155/WCN.2005.774

Le, D. T., Lee, T., & Choo, H. (2018). Delay-aware tree construction and scheduling for data aggregation in duty-cycled wireless sensor networks. *Eurasip Journal on Wireless Communications and Networking*, *2018*(1). https://doi.org/10.1186/s13638-018-1108-3

Li, L., & Li, D. (2018). An Energy-Balanced Routing Protocol for a Wireless Sensor Network. *Journal of Sensors*, *2018*. https://doi.org/10.1155/2018/8505616

Li, X., Liu, W., Xie, M., Liu, A., Zhao, M., Xiong, N. N., … Dai, W. (2018). Differentiated data aggregation routing scheme for energy conserving and delay sensitive wireless sensor networks. *Sensors (Switzerland), 18*(7). https://doi.org/10.3390/s18072349

Li, Y., Wang, Z., & Song, Y. (2006). Wireless sensor network design for wildfire monitoring. *Proceedings of the World Congress on Intelligent Control and Automation*, *1*, 109–113. https://doi.org/10.1109/WCICA.2006.1712372

Liew, S., Tan, C., Gan, M., & Goh, H. G. (2018). *Soung-Yue Liew, Cheng-Kiat Tan, Ming-Lee Gan, and Hock Guan Goh*. (February), 30–40.

Lin, K., Rodrigues, J. J. P. C., Ge, H., Xiong, N., & Liang, X. (2011). Energy efficiency QoS assurance routing in wireless multimedia sensor networks. *IEEE Systems Journal*, *5*(4), 495–505. https://doi.org/10.1109/JSYST.2011.2165599

Lin, Z., Mak, P. I., & Martins, R. P. (2016). *A Sub-GHz Multi-ISM-Band ZigBee Receiver Using Function-Reuse and Gain-Boosted N-Path Techniques for IoT Applications*. https://doi.org/10.1007/978-3-319-21524-2_5

Liu, F., Wang, Y., Lin, M., Liu, K., & Wu, D. (2017). A Distributed Routing Algorithm for Data Collection in Low-Duty-Cycle Wireless Sensor Networks. *IEEE Internet of Things Journal*, *4*(5), 1420–1433. https://doi.org/10.1109/JIOT.2017.2734280

Liu, X. (2012). A Survey on Clustering Routing Protocols in Wireless Sensor Networks. *Sensors*, *12*, 11113–11153. https://doi.org/10.3390/s120811113

Lloret, J., Garcia, M., Bri, D., & Sendra, S. (2009). A Wireless Sensor Network Deployment for Rural and Forest Fire Detection and Verification. *Sensors*, *9*, 8722–8747. https://doi.org/10.3390/s91108722

Mann, P. S., & Singh, S. (2018). Optimal Node Clustering and Scheduling in Wireless Sensor Networks. *Wireless Personal Communications*, *100*(3), 683–708. https://doi.org/10.1007/s11277-018-5341-1

Mehra, P. S., Doja, M. N., & Alam, B. (2020). Fuzzy based enhanced cluster head selection (FBECS) for WSN. *Journal of King Saud University - Science*, *32*(1), 390–401. https://doi.org/10.1016/j.jksus.2018.04.031

Meliyo, J. L., Masuki, K. F. G., Msanya, B. M., Kimaro, D. N., & Mulungu, L. S. (2018). Ecological Biogeography of West Usambara Mountains: A Study on the Influence of Abiotic Factors to Spatial Distribution of Plant and Animal Species. In L. Hufnagel (ed.) *Pure and Applied Biogeography*, pp. 143-166. https://doi.org/10.5772/intechopen.72068

More, A., & Raisinghani, V. (2015). Discharge Curve Backoff Sleep Protocol for Energy Efficient Coverage in Wireless Sensor Networks. *Procedia Computer Science*, *57*, 1131–1139. https://doi.org/10.1016/j.procs.2015.07.402

Morsy, N. A., AbdelHay, E. H., & Kishk, S. S. (2018). Proposed Energy Efficient Algorithm for Clustering and Routing in WSN. *Wireless Personal Communications*, *103*(3), 2575–2598. https://doi.org/10.1007/s11277-018-5948-2

Mosavvar, I., & Ghaffari, A. (2019). Data Aggregation in Wireless Sensor Networks Using Firefly Algorithm. *Wireless Personal Communications*, *104*(1), 307–324. https://doi.org/10.1007/s11277-018-6021-x

Muthukumaran, K., Chitra, K., & Selvakumar, C. (2018). An energy efficient clustering scheme using multilevel routing for wireless sensor network. *Computers and Electrical Engineering*, *69*, 642–652. https://doi.org/10.1016/j.compeleceng.2017.10.007

Naeimi, S., Ghafghazi, H., Chow, C. O., & Ishii, H. (2012). A Survey on the Taxonomy of Cluster-Based Routing Protocols for Homogeneous Wireless Sensor Networks. *Sensors*, *12*, 7350–7409. https://doi.org/10.3390/s120607350

Nguyen, N. T., Liu, B. H., Pham, V. T., & Liou, T. Y. (2018). An Efficient minimum-latency collision-free scheduling algorithm for data aggregation in wireless sensor networks. *IEEE Systems Journal*, *12*(3), 2214–2225. https://doi.org/10.1109/JSYST.2017.2751645

Nguyen, T. T., Pan, J. S., & Dao, T. K. (2019). A compact bat algorithm for unequal clustering in wireless sensor networks. *Applied Sciences, (Switzerland)*, *9*(10). https://doi.org/10.3390/app9101973

Noh, D. K., & Kang, K. (2011). Balanced energy allocation scheme for a solar-powered sensor system and its effects on network-wide performance. *Journal of Computer and System Sciences*, *77*(5), 917–932. https://doi.org/10.1016/j.jcss.2010.08.008

Pan, J., Cai, L., Hou, Y. T., Shi, Y., & Shen, S. X. (2005). Optimal Base-Station Locations in Two-Tiered Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, *4*(5), 458–473. https://doi.org/10.1109/TMC.2005.68

Porret, A. S., Melly, T., Enz, C. C., & Vittoz, E. A. (2000). A low-power low-voltage transceiver architecture suitable for wireless distributed sensors network. *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, *1*(1), 56–59. https://doi.org/10.1109/ISCAS.2000.857025

Pripužić, K., Belani, H., & Vuković, M. (2008). Early forest fire detection with sensor networks: Sliding window skylines approach. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 1,* 725–732. https://doi.org/10.1007/978-3-540-85563-7-91

Purkar, S. V., & Deshpande, R. S. (2018). Energy Efficient Clustering Protocol to Enhance Performance of Heterogeneous Wireless Sensor Network: EECPEP-HWSN. *Journal of Computer Networks and Communications, 2018*. https://doi.org/10.1155/2018/2078627

Qin, N. N., & Chen, J. L. (2018). An area coverage algorithm for wireless sensor networks based on differential evolution. *Research Article International Journal of Distributed Sensor Networks, 14*(8). https://doi.org/10.1177/1550147718796734

Raghunathan, V., Schurgers, C., Park, S., & Srivastava, M. B. (2002). Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, *19*(2), 40–50. https://doi.org/10.1109/79.985679

Rault, T., Bouabdallah, A., & Challal, Y. (2014). Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks*, *67*, 104–122. https://doi.org/10.1016/j.comnet.2014.03.027

Rhim, H., Tamine, K., Abassi, R., Sauveron, D., & Guemara, S. (2018). A multi-hop graph-based approach for an energy-efficient routing protocol in wireless sensor networks. *Human-Centric Computing and Information Sciences*, *8*(1). https://doi.org/10.1186/s13673-018-0153-6

Rodriguez-Zurrunero, R., Utrilla, R., Romero, E., & Araujo, A. (2018). An Adaptive Scheduler for Real-Time Operating Systems to Extend WSN Nodes Lifetime. *Wireless Communications and Mobile Computing, 2018*. https://doi.org/10.1155/2018/4185650

Sazak, N., & Abdullah, H. (2009). The importance of using wireless sensor networks for forest fire sensing and detection in Turkey. *Proceedings of the 5th International Advanced Technologies Symposium (IATS'09),* 12–15. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.623.8596&rep=rep1&type=pdf

Seah, W. K. G., Zhi, A. E., & Tan, H. P. (2009). Wireless Sensor Networks Powered by Ambient Energy Harvesting (WSN-HEAP) - Survey and challenges. *Proceedings of the 2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology, Wireless VITAE 2009*, 1–5. https://doi.org/10.1109/WIRELESSVITAE.2009.5172411

Sert, S. A., Alchihabi, A., & Yazici, A. (2018). A Two-Tier Distributed Fuzzy Logic Based Protocol for Efficient Data Aggregation in Multihop Wireless Sensor Networks. *IEEE Transactions on Fuzzy Systems*, *26*(6), 3615–3629. https://doi.org/10.1109/TFUZZ. 2018.2841369

Sheikhi, M., Sedighian Kashi, S., & Samaee, Z. (2019). Energy provisioning in wireless rechargeable sensor networks with limited knowledge. *Wireless Networks*, *25*(6), 3531–3544. https://doi.org/10.1007/s11276-019-01948-1

Shen, C. C., Srisathapornphat, C., & Jaikaeo, C. (2001). Article in IEEE Personal Communications. *Ieeexplore.Ieee.Org*. https://doi.org/10.1109/98.944004

Shih, E., Cho, S. H., Ickes, N., Min, R., Sinha, A., Wang, A., & Chandrakasan, A. (2001). Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking - MobiCom '01*, 272–287. https://doi.org/10.1145/ 381677. 381703

Singh, S. K., Singh, M. P., & Singh, D. K. (2010). Routing Protocols in Wireless Sensor Networks – A Survey. *International Journal of Computer Science & Engineering Survey*, *1*(2), 63–83. https://doi.org/10.3390/s91108399

Singh, H., & Singh, D. (2016). Taxonomy of Routing Protocols in Wireless Sensor Networks: A Survey. *International Conference on Contemporary Computing and Informatics*, 822–830. https://doi.org/10.1109/IC3I.2016.7918796

Singh, R., & Verma, A. K. (2017). Energy efficient cross layer based adaptive threshold routing protocol for WSN. *AEU - International Journal of Electronics and Communications*, *72*, 166–173. https://doi.org/10.1016/j.aeue.2016.12.001

Singh, S. P., & Sharma, S. C. (2015). A Survey on Cluster Based Routing Protocols in Wireless Sensor Networks. *Procedia Computer Science*, *45*, 687–695. https://doi.org/10. 1016/J. PROCS.2015.03.133

Sohrabi, K., Gao, J., Ailawadhi, V., & Pottie, G. J. (2000). Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, *7*(5), 16–27. https://doi.org/ 10.1109/98.878532

Soliman, H., Sudan, K., & Mishra, A. (2010). A smart forest-fire early detection sensory system: Another approach of utilizing wireless sensor and neural networks. *Sensors, 2010 IEEE*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber =5690033

Tchuani, T. D., Simeu, E., & Tchuente, M. (2020). Lifetime optimization of wireless sensor networks with sleep mode energy consumption of sensor nodes. *Wireless Networks*, *26*(1), 91–100. https://doi.org/10.1007/s11276-018-1783-3

Thayananthan, V., & Alzranhi, A. (2014). Enhancement of energy conservation technologies in wireless sensor network. *Procedia Computer Science*, *34*, 79–86. https://doi.org/10. 1016/j.procs.2014.07.052

Tyagi, S., & Kumar, N. (2013). A systematic review on clustering and routing techniques based upon LEACH protocol for wireless sensor networks. *Journal of Network and Computer Applications*, *36*(2), 623–645. https://doi.org/10.1016/j.jnca.2012.12.001

ur Rehman Khan, A., Bilal, S. M., & Othman, M. (2012). A Performance Comparison of Network Simulators for Wireless Networks. *2012 IEEE International Conference on Control System, Computing and Engineering*, (November), 34–38. https://doi.org/10. 1109/ICCSCE.2012.6487111

Vancin, S., Erdem, E., & Vançİn, S. (2015). Design and Simulation of Wireless Sensor Network Topologies Using the ZigBee Standard. *International Journal of Computer Networks and Applications*, *2*(3), 135–143. https://doi.org/10.1109/IC4.2015. 7375648

Verma, S., Sood, N., & Sharma, A. K. (2018). Design of a novel routing architecture for harsh environment monitoring in heterogeneous WSN. *IET Wireless Sensor Systems*, *8*(6), 284–294. https://doi.org/10.1049/iet-wss.2018.5025

Vijayalakshmi, K., & Manickam, J. M. L. (2019). A cluster based mobile data gathering using SDMA and PSO techniques in WSN. *Cluster Computing*, *22*, 12727–12736. https://doi.org/10.1007/s10586-018-1748-4

Voigt, T., Ritter, H., & Schiller, J. (2003). Utilizing solar power in wireless sensor networks. *Proceedings - Conference on Local Computer Networks*, 416–422. https://doi.org/10.1109/LCN.2003.1243167

Wan, R., Xiong, N., & Loc, N. T. (2018). An energy-efficient sleep scheduling mechanism with similarity measure for wireless sensor networks. *Human-Centric Computing and Information Sciences*, *8*(1). https://doi.org/10.1186/s13673-018-0141-x

Woo, A., & Culler, D. E. (2001). A transmission control scheme for media access in sensor networks. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, 221–235. https://doi.org/10.1145/381677.381699

Xia, Y., Fabian, P., Winterhalter, M., & Zhao, M. (2001). Forest climatology: Estimation and use of daily climatological data for Bavaria, Germany. *Agricultural and Forest Meteorology*, *106*(2), 87–103. https://doi.org/10.1016/S0168-1923(00)00210-0

Xiao, K., Wang, R., Deng, H., Zhang, L., & Yang, C. (2019). Energy-aware scheduling for information fusion in wireless sensor network surveillance. *Information Fusion*, *48*, 95–106. https://doi.org/10.1016/j.inffus.2018.08.005

Xu, Y. H., Sun, Q. Y., & Xiao, Y. T. (2018). An environmentally aware scheme of wireless sensor networks for forest fire monitoring and detection. *Future Internet*, *10*(10). https://doi.org/10.3390/fi10100102

Ye, F., Luo, H., Cheng, J., Lu, S., & Zhang, L. (2002). A two-tier data dissemination model for large-scale wireless sensor networks. *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, 148–159. https://doi.org/10.1145/570645.570664

Yuan, X., Elhoseny, M., El-Minir, H. K., & Riad, A. M. (2017). A Genetic Algorithm-Based, Dynamic Clustering Method Towards Improved WSN Longevity. *Journal of Network and Systems Management*, *25*(1), 21–46. https://doi.org/10.1007/s10922-016-9379-7

Yue, R., & Ying, T. (2011). A water quality monitoring system based on wireless sensor network & solar power supply. *2011 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, 2011*, 126–129. https://doi.org/10.1109/CYBER.2011.6011777

APPENDICES

Appendix 1: Particle Swarm Optimization source codes

```
#include <time.h> // for time()
#include <math.h> // for cos(), pow(), sqrt() etc.
#include <float.h> // for DBL_MAX
#include <string.h> // for mem*
///////////////////////////#include <gsl/gsl_rng.h>
#include "pso.h"
namespace ns3 {
// calulate swarm size based on dimensionality
int pso::pso_calc_swarm_size(int dim) {
int size = 10. + 2. * sqrt(dim);
return (size > PSO_MAX_SIZE ? PSO_MAX_SIZE : size);
}
//==============================================================
//          INERTIA WEIGHT UPDATE STRATEGIES
//==============================================================
// calculate linearly decreasing inertia weight
double pso::calc_inertia_lin_dec(int step, pso_settings_t *settings) {
   int dec_stage = 3 * settings->steps / 4;
   if (step <= dec_stage)
 return settings->w_min + (settings->w_max - settings->w_min) *        \
     (dec_stage - step) / dec_stage;
   else
 return settings->w_min;
}
//==============================================================
//          NEIGHBORHOOD (Angle) MATRIX STRATEGIES
//==============================================================
// global neighborhood
void pso::inform_global(int *Angle, double *pos_nb,
          double *pos_b, double *Distance,
          double *Belief, int Energy,
          pso_settings_t *settings)
{
   int i;
   // all particles have the same attractor (Belief)
   // copy the contents of Belief to pos_nb
   for (i=0; i<settings->size; i++)
 memmove((void *)&pos_nb[i*settings->dim], (void *)Belief,
       sizeof(double) * settings->dim);
}
// ==============================================================
// general inform function :: according to the connectivity
// matrix Angle, it copies the best position (from pos_b) of the
// informers of each particle to the pos_nb matrix
void pso::inform(int *Angle, double *pos_nb, double *pos_b, double *Distance,
 int Energy, pso_settings_t * settings)
```

```
{
   int i, j;
   int b_n; // best neighbor in terms of fitness
   // for each particle
   for (j=0; j<settings->size; j++) {
 b_n = j; // self is best
 // who is the best informer??
 for (i=0; i<settings->size; i++)
    // the i^th particle informs the j^th particle
    if (Angle[i*settings->size + j] && Distance[i] < Distance[b_n])
       // found a better informer for j^th particle
       b_n = i;
 // copy pos_b of b_n^th particle to pos_nb[j]
 memmove((void *)&pos_nb[j*settings->dim],
       (void *)&pos_b[b_n*settings->dim],
       sizeof(double) * settings->dim);
   }
}
// =============
// ring topology
// =============
// topology initialization :: this is a static (i.e. fixed) topology
void pso::init_comm_ring(int *Angle, pso_settings_t * settings) {
   int i;
   // reset array
   memset((void *)Angle, 0, sizeof(int)*settings->size*settings->size);

   // choose informers
   for (i=0; i<settings->size; i++) {
 // set diagonal to 1
 Angle[i*settings->size+i] = 1;
 if (i==0) {
    // look right
    Angle[i*settings->size+i+1] = 1;
    // look left
    Angle[(i+1)*settings->size-1] = 1;
 } else if (i==settings->size-1) {
    // look right
    Angle[i*settings->size] = 1;
    // look left
    Angle[i*settings->size+i-1] = 1;
 } else {
    // look right
    Angle[i*settings->size+i+1] = 1;
    // look left
    Angle[i*settings->size+i-1] = 1;
 }

   }
}
```

```
void pso::inform_ring(int *Angle, double *pos_nb,
        double *pos_b, double *Distance,
        double *Belief, int Energy,
         pso_settings_t * settings)
{
    // update pos_nb matrix
    inform(Angle, pos_nb, pos_b, Distance, Energy, settings);
}
// ============================
// random neighborhood topology
// ============================
void pso::init_comm_random(int *Angle, pso_settings_t * settings) {

    int i, j, k;
    // reset array
    memset((void *)Angle, 0, sizeof(int)*settings->size*settings->size);
    // choose informers
    for (i=0; i<settings->size; i++) {
 // each particle informs itself
 Angle[i*settings->size + i] = 1;
 // choose kappa (on average) informers for each particle
 for (k=0; k<settings->nhood_size; k++) {
     // generate a random index
    ////////////////////////// j = gsl_rng_uniform_int(settings->rng, settings->size);
j=5;
     // particle i informs particle j
 Angle[i*settings->size + j] = 1;
 }
    }
}
void pso::pso_algo(int *Angle, double *pos_nb,
        double *pos_b, double *Distance,
        double *Belief, int Energy)
{
pso_settings_t * settings = new pso_settings_t;
inform_random(Angle, pos_nb, pos_b, Distance, Belief, Energy, settings);
}
void pso::inform_random(int *Angle, double *pos_nb,
          double *pos_b, double *Distance,
          double *Belief, int Energy,
          pso_settings_t * settings)
{
    // regenerate connectivity??
    if (!Energy)
 init_comm_random(Angle, settings);
    inform(Angle, pos_nb, pos_b, Distance, Energy, settings);
}
//==============================================================
// return default pso settings
void pso::pso_set_default_settings(pso_settings_t *settings) {
    // set some default values
    settings->dim = 30;
    settings->x_lo = -20;
    settings->x_hi = 20;
    settings->goal = 1e-5;
    settings->size = pso_calc_swarm_size(settings->dim);
    settings->print_every = 1000;
    settings->steps = 100000;
    settings->c1 = 1.496;
    settings->c2 = 1.496;
    settings->w_max = PSO_INERTIA;
    settings->w_min = 0.3;
    settings->clamp_pos = 1;
    settings->nhood_strategy = PSO_NHOOD_RING;
    settings->nhood_size = 5;
    settings->w_strategy = PSO_W_LIN_DEC;
    settings->rng = NULL;
    settings->seed = time(0);
}
//=============================================================
//                  PSO ALGORITHM
//=============================================================
void pso::pso_solve(pso_obj_fun_t obj_fun, void *obj_fun_params,
        pso_result_t *solution, pso_settings_t *settings)
{

//////////////////   int free_rng = 0; // whether to free settings->rng when finished
    // Particles
    double pos[settings->size][settings->dim]; // position matrix
    double vel[settings->size][settings->dim]; // velocity matrix
    double pos_b[settings->size][settings->dim]; // best position matrix
    double fit[settings->size]; // particle fitness vector
    double Distance[settings->size]; // best fitness vector
    // Swarm
//////////////    double pos_nb[settings->size][settings->dim]; // what is the best informed
                                          // position for each particle
    int Angle[settings->size][settings->size]; // Angleunications:who informs who
                                         // rows : those who inform
                                         // cols : those who are informed
///////////    int Energy; // whether solution->error was Energy during
                 // the last iteration

    int i, d, step;
 //////////////  double a, b; // for matrix initialization
//////////////double rho1, rho2; // random numbers (coefficients)
///////    double w; // current omega
////////////////////    void (*inform_fun)(); // neighborhood update function
////////////////////    double (*calc_inertia_fun)(); // inertia weight update function
    // CHECK RANDOM NUMBER GENERATOR
    if (! settings->rng) {
```

```
/*
// initialize random number generator
 gsl_rng_env_setup();
 // allocate the RNG
 settings->rng = gsl_rng_alloc(gsl_rng_default);
 // seed the generator
 gsl_rng_set(settings->rng, settings->seed);

*/
 // remember to free the RNG
////////////   free_rng = 1;
    }

    // SELECT APPROPRIATE NHOOD UPDATE FUNCTION
    switch (settings->nhood_strategy)
 {
 case PSO_NHOOD_GLOBAL :
     // Angle matrix not used
////////////       inform_fun = inform_global;
     break;
 case PSO_NHOOD_RING :
     init_comm_ring((int *)Angle, settings);
  //////////////   inform_fun = inform_ring;
     break;
 case PSO_NHOOD_RANDOM :
     init_comm_random((int *)Angle, settings);
//////////////////         inform_fun = inform_random;
     break;
 }

    // SELECT APPROPRIATE INERTIA WEIGHT UPDATE FUNCTION
    switch (settings->w_strategy)
 {
 /* case PSO_W_CONST : */
 /*     calc_inertia_fun = calc_inertia_const; */
 /*     break; */
 case PSO_W_LIN_DEC :
///////////////            calc_inertia_fun = calc_inertia_lin_dec;
     break;
 }
    // INITIALIZE SOLUTION
    solution->error = DBL_MAX;
    // SWARM INITIALIZATION
    // for each particle
    for (i=0; i<settings->size; i++) {
 // for each dimension
 for (d=0; d<settings->dim; d++) {
     // generate two numbers within the specified range
/*
     a = settings->x_lo + (settings->x_hi - settings->x_lo) * \
        gsl_rng_uniform(settings->rng);
     b = settings->x_lo + (settings->x_hi - settings->x_lo) *   \
        gsl_rng_uniform(settings->rng);
*/
     // initialize position
////////////////          pos[i][d] = a;
     // best position is the same
/////////          pos_b[i][d] = a;
     // initialize velocity
//////////          vel[i][d] = (a-b) / 2.;
 }
 // update particle fitness
 fit[i] = obj_fun(pos[i], settings->dim, obj_fun_params);
 Distance[i] = fit[i]; // this is also the personal best
 // update Belief??
 if (fit[i] < solution->error) {
     // update best fitness
     solution->error = fit[i];
     // copy particle pos to Belief vector
     memmove((void *)solution->gbest, (void *)&pos[i],
            sizeof(double) * settings->dim);
 }
   }
   // initialize omega using standard value
  ///////////// w = PSO_INERTIA;
   // RUN ALGORITHM
   for (step=0; step<settings->steps; step++) {
 // update current step
 settings->step = step;
 // update inertia weight
 // do not bother with calling a calc_w_const function
 if (settings->w_strategy)
/////      w = calc_inertia_fun(step, settings);
 // check optimization goal
 if (solution->error <= settings->goal) {
     // SOLVED!!
     if (settings->print_every)
 //     printf("Goal achieved @ step %d (error=%.3e) :-)\n", step, solution->error);
     break;
 }
 // update pos_nb matrix (find best of neighborhood for all particles)
 ////////////////////////////////////////  inform_fun(Angle,  pos_nb,  pos_b,  Distance,  solution-
 >Belief,Energy, settings);
 // the value of Energy was just used; reset it
/////   Energy = 0;
 // update all particles
 for (i=0; i<settings->size; i++) {
     // for each dimension
     for (d=0; d<settings->dim; d++) {
        // calculate stochastic coefficients
```

```
/*
       rho1 = settings->c1 * gsl_rng_uniform(settings->rng);
       rho2 = settings->c2 * gsl_rng_uniform(settings->rng);
*/
       // update velocity
///////////////       vel[i][d] = w * vel[i][d] +rho1 * (pos_b[i][d] - pos[i][d]) +rho2 *
 (pos_nb[i][d] - pos[i][d]);
       // update position
       pos[i][d] += vel[i][d];
       // clamp position within bounds?
       if (settings->clamp_pos) {
          if (pos[i][d] < settings->x_lo) {
             pos[i][d] = settings->x_lo;
             vel[i][d] = 0;
          } else if (pos[i][d] > settings->x_hi) {
             pos[i][d] = settings->x_hi;
             vel[i][d] = 0;
          }
       } else {
          // enforce periodic boundary conditions
          if (pos[i][d] < settings->x_lo) {
             pos[i][d] = settings->x_hi - fmod(settings->x_lo - pos[i][d],
                                     settings->x_hi - settings->x_lo);
             vel[i][d] = 0;
             /* printf("%f < x_lo=%.0f (v=%f) (mod=%f)\n", */
             /*       pos[i][d], settings->x_lo, vel[i][d], */
             /*       settings->x_hi - fmod(settings->x_lo - pos[i][d], */
             /*                   settings->x_hi - settings->x_lo)); */
             /* assert(pos[i][d] > settings->x_lo && pos[i][d] < settings->x_hi); */

             //vel[i][d] = 0;
          } else if (pos[i][d] > settings->x_hi) {

             pos[i][d] = settings->x_lo + fmod(pos[i][d] - settings->x_hi,
                                     settings->x_hi - settings->x_lo);
             vel[i][d] = 0;

             /* printf("%f > x_hi=%.0f (v=%f) (mod=%f)\n", */
             /*       pos[i][d], settings->x_hi, vel[i][d], */
             /*       settings->x_lo + fmod(pos[i][d] - settings->x_hi, */
             /*                   settings->x_hi - settings->x_lo)); */
             /* assert(pos[i][d] > settings->x_lo && pos[i][d] < settings->x_hi); */
          }
       }
    }
    // update particle fitness
    fit[i] = obj_fun(pos[i], settings->dim, obj_fun_params);
    // update personal best position?
    if (fit[i] < Distance[i]) {
       Distance[i] = fit[i];
```

```
        // copy contents of pos[i] to pos_b[i]
        memmove((void *)&pos_b[i], (void *)&pos[i],
              sizeof(double) * settings->dim);
     }
     // update Belief??
     if (fit[i] < solution->error) {
////////////          Energy = 1;
        // update best fitness
        solution->error = fit[i];
        // copy particle pos to Belief vector
        memmove((void *)solution->gbest, (void *)&pos[i],
              sizeof(double) * settings->dim);
     }
  }

    /////////////// if (settings->print_every && (step % settings->print_every == 0))
      ///////////////   printf("Step %d (w=%.2f) :: min err=%.5e\n", step, w, solution->error);

    }

    // free RNG??
   /////////////// if (free_rng)
 ////////////////////gsl_rng_free(settings->rng);
}
}
```

Appendix 2: Affinity Propagation source codes

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
#include <algorithm>
#include "affinity_propagation.h"
#include <iostream>
#include <ctime>
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/csma-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"

using namespace std;
namespace ns3 {

//N is the number of two-dimension data points
//S is the similarity matrix
//R is the responsibility matrix
//A is the availabiltiy matrix
//iter is the maximum number of iterations
//lambda is the damping factor
const int N = 250;
double S[N][N] = {0};
double R[N][N] = {0};
double A[N][N] = {0};
int iter = 230;
double lambda = 0.9;

void affinity_propagation::Position(NodeContainer nod)
{

AnimationInterface::SetConstantPosition (nod.Get (0), 525, 525);
AnimationInterface::SetConstantPosition (nod.Get (1), 675, 675);
AnimationInterface::SetConstantPosition (nod.Get (2), 825, 825);

AnimationInterface::SetConstantPosition (nod.Get (3), 375, 525);
AnimationInterface::SetConstantPosition (nod.Get (4), 225, 675);
AnimationInterface::SetConstantPosition (nod.Get (5), 75, 825);

AnimationInterface::SetConstantPosition (nod.Get (6), 375, 375);
AnimationInterface::SetConstantPosition (nod.Get (7), 225, 225);
AnimationInterface::SetConstantPosition (nod.Get (8), 75, 75);
```

```
AnimationInterface::SetConstantPosition (nod.Get (9), 525, 375);
AnimationInterface::SetConstantPosition (nod.Get (10), 675, 225);
AnimationInterface::SetConstantPosition (nod.Get (11), 825, 75);

}
void affinity_propagation::algo(int exemplar)
{
 ////readS(S, dataFileName);


for(int x=0; x<100; x++) {

for(int y=0; y<100; y++) {
S[x][y]=(x*y);
}
}

 for(int m=0; m<iter; m++) {
 //update responsibility
        for(int i=0; i<N; i++) {
                for(int k=0; k<N; k++) {
                        double max = -1e100;
                        for(int kk=0; kk<k; kk++) {
                                if(S[i][kk]+A[i][kk]>max)
                                        max = S[i][kk]+A[i][kk];
                        }
                        for(int kk=k+1; kk<N; kk++) {
                                if(S[i][kk]+A[i][kk]>max)
                                        max = S[i][kk]+A[i][kk];
                        }
                        R[i][k] = (1-lambda)*(S[i][k] - max) + lambda*R[i][k];
                }
        }
 //update availability
        for(int i=0; i<N; i++) {
                for(int k=0; k<N; k++) {
                        if(i==k) {
                                double sum = 0.0;
                                for(int ii=0; ii<i; ii++) {
                                        sum += max(0.0, R[ii][k]);
                                }
                                for(int ii=i+1; ii<N; ii++) {
                                        sum += max(0.0, R[ii][k]);
                                }
                                A[i][k] = (1-lambda)*sum + lambda*A[i][k];
                        } else {
                                double sum = 0.0;
                                int maxik = max(i, k);
                                int minik = min(i, k);
```

```
                                        for(int ii=0; ii<minik; ii++) {
                                                sum += max(0.0, R[ii][k]);
                                        }
                                        for(int ii=minik+1; ii<maxik; ii++) {
                                                sum += max(0.0, R[ii][k]);
                                        }
                                        for(int ii=maxik+1; ii<N; ii++) {
                                                sum += max(0.0, R[ii][k]);
                                        }
                                        A[i][k] = (1-lambda)*min(0.0, R[k][k]+sum) + lambda*A[i][k];
                                }
                        }
                }
        }

        //find the exemplar
        double E[N][N] = {0};
        vector<int> center;
        for(int i=0; i<N; i++) {
                E[i][i] = R[i][i] + A[i][i];
                if(E[i][i]>0) {
                        center.push_back(i);
                }
        }
        //data point assignment, idx[i] is the exemplar for data point i
        int idx[N] = {0};
        for(int i=0; i<N; i++) {
                int idxForI = 0;
                double maxSim = -1e100;
                for(int j=0; j<250; j++) {
                        int c = center[j];
                        if (S[i][c]>maxSim) {
                                maxSim = S[i][c];
                                idxForI = c;
                        }
                }
                idx[i] = idxForI;
        }
        //output the assignment
        for(int i=0; i<N; i++) {
                //since the index of data points starts from zero, I add 1 to let it start from 1
                cout << idx[i]+1 << endl;
        }
}

}
```

Appendix 3: Deep Reinforcement Learning source codes

```
#include "Deep_Reinforcement_scheduler.h"

#include "ns3/simulator.h"
#include "ns3/packet-burst.h"
#include "ns3/log.h"
#include<iostream>
using namespace std;
namespace ns3 {
void DeepReinforcementscheduler::findWaitingTime(int processes[], int n, int bt[], int
 wt[], int quantum)
{
 int rem_bt[n];
 for (int i = 0 ; i < n ; i++)
        rem_bt[i] = bt[i];
 int t = 0;
 while (1)
 {
        bool done = true;
        for (int i = 0 ; i < n; i++)
        {
                if (rem_bt[i] > 0)
                {
                        done = false;
                        if (rem_bt[i] > quantum)
                        {
                                t += quantum;
                                rem_bt[i] -= quantum;
                        }
                        else
                        {
                                t = t + rem_bt[i];
                                wt[i] = t - bt[i];
                                rem_bt[i] = 0;
                        }
                }
        }
        if (done == true)
        break;
 }
}

void DeepReinforcementscheduler::findTurnAroundTime(int processes[], int n,
                                        int bt[], int wt[], int tat[])
{
 for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}
void DeepReinforcementscheduler::findavgTime(int processes[], int n, int bt[],
                                                      int quantum)
{
 int wt[n], tat[n], total_wt = 0, total_tat = 0;
 findWaitingTime(processes, n, bt, wt, quantum);
 findTurnAroundTime(processes, n, bt, wt, tat);
 cout << "Processes "<< " Burst time "
       << " Waiting time " << " Turn around time\n";
 for (int i=0; i<n; i++)
 {
       total_wt = total_wt + wt[i];
       total_tat = total_tat + tat[i];
       cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
              << wt[i] <<"\t\t " << tat[i] <<endl;
 }

 cout << "Average waiting time = "
       << (float)total_wt / (float)n;
 cout << "\nAverage turn around time = "
       << (float)total_tat / (float)n;
}
int DeepReinforcementscheduler::mainpro()
{
 int processes[] = { 1, 2, 3};
 int n = sizeof processes / sizeof processes[0];
 int burst_time[] = {10, 5, 8};
 int quantum = 2;
 findavgTime(processes, n, burst_time, quantum);
 return 0;
}
}
```

Appendix 4: Anti-Colony source codes

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */

#include "aco.h"
#include <fstream>
#include <vector>
#include <cmath>
#include <cassert>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <vector>
#include <math.h>
#include <assert.h>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
using namespace std;
namespace ns3 {
ACO::ACO (int nAnts, int wsnnodes,double alpha, double beta, double q, double ro,
 double taumax,int initCity) {
 NUMBEROFANTS = nAnts;
 NUMBEROFnodes  = wsnnodes;
 ALPHA              = alpha;
 BETA               = beta;
 Q                  = q;
 RO                 = ro;
 TAUMAX                     = taumax;
 INITIALCITY        = initCity;

 randoms = new Randoms (21);
}
ACO::~ACO () {
 for(int i=0; i<NUMBEROFnodes; i++) {
        delete [] GRAPH[i];
        delete [] nodes[i];
        delete [] PHEROMONES[i];
        delete [] DELTAPHEROMONES[i];
        if(i < NUMBEROFnodes - 1) {
               delete [] PROBS[i];
        }
 }
 delete [] GRAPH;
 delete [] nodes;
 delete [] PHEROMONES;
 delete [] DELTAPHEROMONES;
 delete [] PROBS;
}
void ACO::init () {
 GRAPH                 = new int*[NUMBEROFnodes];
 nodes                 = new double*[NUMBEROFnodes];
 PHEROMONES            = new double*[NUMBEROFnodes];
 DELTAPHEROMONES = new double*[NUMBEROFnodes];
 PROBS                 = new double*[NUMBEROFnodes-1];
 for(int i=0; i<NUMBEROFnodes; i++) {
       GRAPH[i]                  = new int[NUMBEROFnodes];
       nodes[i]                  = new double[2];
       PHEROMONES[i]             = new double[NUMBEROFnodes];
       DELTAPHEROMONES[i]        = new double[NUMBEROFnodes];
       PROBS[i]                  = new double[2];
       for (int j=0; j<2; j++) {
              nodes[i][j] = -1.0;
              PROBS[i][j]  = -1.0;
       }
       for (int j=0; j<NUMBEROFnodes; j++) {
              GRAPH[i][j]                   = 0;
              PHEROMONES[i][j]              = 0.0;
              DELTAPHEROMONES[i][j]         = 0.0;
       }
 }
 ROUTES = new int*[NUMBEROFANTS];
 for (int i=0; i<NUMBEROFANTS; i++) {
       ROUTES[i] = new int[NUMBEROFnodes];
       for (int j=0; j<NUMBEROFnodes; j++) {
              ROUTES[i][j] = -1;
       }
 }
 BESTLENGTH = (double) INT_MAX;
 BESTROUTE  = new int[NUMBEROFnodes];
 for (int i=0; i<NUMBEROFnodes; i++) {
       BESTROUTE[i] = -1;
 }
}
void ACO::connectnodes (int cityi, int cityj) {
 GRAPH[cityi][cityj] = 1;
 PHEROMONES[cityi][cityj] = randoms -> Uniforme() * TAUMAX;
 GRAPH[cityj][cityi] = 1;
 PHEROMONES[cityj][cityi] = PHEROMONES[cityi][cityj];
}
void ACO::setCITYPOSITION (int city, double x, double y) {
 nodes[city][0] = x;
 nodes[city][1] = y;
}
void ACO::printPHEROMONES () {
 cout << " PHEROMONES: " << endl;
 cout << "  | ";
 for (int i=0; i<NUMBEROFnodes; i++) {
        printf("%5d   ", i);
 }
 cout << endl << "- | ";
 for (int i=0; i<NUMBEROFnodes; i++) {
        cout << "--------";
 }
 cout << endl;
 for (int i=0; i<NUMBEROFnodes; i++) {
        cout << i << " | ";
        for (int j=0; j<NUMBEROFnodes; j++) {
                if (i == j) {
                        printf ("%5s   ", "x");
                        continue;
                }
                if (exists(i, j)) {
                        printf ("%7.3f ", PHEROMONES[i][j]);
                }
                else {
                        if(PHEROMONES[i][j] == 0.0) {
                                printf ("%5.0f   ", PHEROMONES[i][j]);
                        }
                        else {
                                printf ("%7.3f ", PHEROMONES[i][j]);
                        }
                }
        }
        cout << endl;
 }
 cout << endl;
}
double ACO::distance (int cityi, int cityj) {
 return (double)
        sqrt (pow (nodes[cityi][0] - nodes[cityj][0], 2) +
                    pow (nodes[cityi][1] - nodes[cityj][1], 2));
}
bool ACO::exists (int cityi, int cityc) {
 return (GRAPH[cityi][cityc] == 1);
}
bool ACO::vizited (int antk, int c) {
 for (int l=0; l<NUMBEROFnodes; l++) {
        if (ROUTES[antk][l] == -1) {
                break;
        }
        if (ROUTES[antk][l] == c) {
                return true;
        }
 }
 return false;
}
double ACO::PHI (int cityi, int cityj, int antk) {
 double ETAij = (double) pow (1 / distance (cityi, cityj), BETA);
 double TAUij = (double) pow (PHEROMONES[cityi][cityj],   ALPHA);
 double sum = 0.0;
 for (int c=0; c<NUMBEROFnodes; c++) {
        if (exists(cityi, c)) {
                if (!vizited(antk, c)) {
                        double ETA = (double) pow (1 / distance (cityi, c), BETA);
                        double TAU = (double) pow (PHEROMONES[cityi][c],   ALPHA);
                        sum += ETA * TAU;
                }
        }
 }
 return (ETAij * TAUij) / sum;
}
double ACO::length (int antk) {
 double sum = 0.0;
 for (int j=0; j<NUMBEROFnodes-1; j++) {
        sum += distance (ROUTES[antk][j], ROUTES[antk][j+1]);
 }
 return sum;
}
int ACO::city () {
 double xi = randoms -> Uniforme();
 int i = 0;
 double sum = PROBS[i][0];
 while (sum < xi) {
        i++;
        sum += PROBS[i][0];
 }
 return (int) PROBS[i][1];
}
void ACO::route (int antk) {
 ROUTES[antk][0] = INITIALCITY;
 for (int i=0; i<NUMBEROFnodes-1; i++) {
        int cityi = ROUTES[antk][i];
        int count = 0;
        for (int c=0; c<NUMBEROFnodes; c++) {
                if (cityi == c) {
                        continue;
                }
                if (exists (cityi, c)) {
                        if (!vizited (antk, c)) {
                                PROBS[count][0] = PHI (cityi, c, antk);
                                PROBS[count][1] = (double) c;
                                count++;
                        }
                }
        }

        // deadlock
        if (0 == count) {
                return;
        }
        ROUTES[antk][i+1] = city();
 }
}
int ACO::valid (int antk, int iteration) {
 for(int i=0; i<NUMBEROFnodes-1; i++) {
        int cityi = ROUTES[antk][i];
        int cityj = ROUTES[antk][i+1];
        if (cityi < 0 || cityj < 0) {
                return -1;
        }
        if (!exists(cityi, cityj)) {
                return -2;
        }
        for (int j=0; j<i-1; j++) {
                if (ROUTES[antk][i] == ROUTES[antk][j]) {
                        return -3;
                }
        }
 }
 if (!exists (INITIALCITY, ROUTES[antk][NUMBEROFnodes-1])) {
        return -4;
 }
 return 0;
}
void ACO::updatePHEROMONES () {
 for (int k=0; k<NUMBEROFANTS; k++) {
        double rlength = length(k);
        for (int r=0; r<NUMBEROFnodes-1; r++) {
                int cityi = ROUTES[k][r];
                int cityj = ROUTES[k][r+1];
                DELTAPHEROMONES[cityi][cityj] += Q / rlength;
                DELTAPHEROMONES[cityj][cityi] += Q / rlength;
        }
 }
 for (int i=0; i<NUMBEROFnodes; i++) {
        for (int j=0; j<NUMBEROFnodes; j++) {
                PHEROMONES[i][j]    =    (1    -    RO)    *    PHEROMONES[i][j]    +
 DELTAPHEROMONES[i][j];
                DELTAPHEROMONES[i][j] = 0.0;
        }
 }
}
void ACO::optimize (int ITERATIONS) {
 for (int iterations=1; iterations<=ITERATIONS; iterations++) {
        cout << flush;
        cout<< "ITERATION " << iterations << " HAS STARTED!" << endl << endl;
        for (int k=0; k<NUMBEROFANTS; k++) {
                cout<< " : ant " << k << " has been released!" << endl;
                while (0 != valid(k, iterations)) {
                        cout<< "  :: releasing ant " << k << " again!" << endl;
                        for (int i=0; i<NUMBEROFnodes; i++) {
                                ROUTES[k][i] = -1;
                        }
                        route(k);
                }
                for (int i=0; i<NUMBEROFnodes; i++) {
                        cout << ROUTES[k][i] << " ";
                }
                cout << endl;

                cout << "  :: route done" << endl;
                double rlength = length(k);
                if (rlength < BESTLENGTH) {
                        BESTLENGTH = rlength;
                        for (int i=0; i<NUMBEROFnodes; i++) {
                                BESTROUTE[i] = ROUTES[k][i];
                        }
                }
                cout<< " : ant " << k << " has ended!" << endl;
        }
        updatePHEROMONES ();
        cout<< " done!" << endl << endl;
        for (int i=0; i<NUMBEROFANTS; i++) {
                for (int j=0; j<NUMBEROFnodes; j++) {
                        ROUTES[i][j] = -1;
                }
        }
 }
}
}
```

Appendix 5: Temperature and humidity sensor Arduino codes

```
/* How to use the DHT-22 sensor with Arduino uno
   Temperature and humidity sensor
*/

//Libraries
#include <DHT.h>;
#include <ArduinoJson.h>;
//Constants
#define DHTPIN 7     // what pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302)
DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz Arduino
//Variables
int chk;
float hum;  //Stores humidity value
float temp; //Stores temperature value

void setup()
{
  Serial.begin(9600);
  dht.begin();

}

void loop()
{
    //Read data and store it to variables hum and temp
    hum = dht.readHumidity();
    temp= dht.readTemperature();

delay(1000); //Delay 2 sec.

  StaticJsonDocument<32> root;
  root["Id"] = "EN1";

  JsonArray data = root.createNestedArray("sensor");
  data.add(hum);
  data.add(temp);

  serializeJson(root, Serial);

    Serial.println();
}
```

Appendix 6: Evaluation source codes

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/csma-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
#include "ns3/basic-energy-source.h"
#include "ns3/simple-device-energy-model.h"
#include "ns3/config-store-module.h"
#include "ns3/ipv4-static-routing-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/rtt-estimator.h"
#include "ns3/aodv-module.h"
#include <ns3/wsn-module.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("Sleep_Scheduling");
AnimationInterface *anim;
std::string NETWORK_X1 = "900.0";
std::string NETWORK_Y1 = "900.0";
int NoOfSensor = 50;
int Sink=1;
double pk=0;
double packetSize = 1024;
double numPackets = 200;
double MsgSize = packetSize* numPackets;
double interval = 0.1;
double simTime = 10.0;
double distance1=1;
std::string rate="10Mbps";
std::string rate1="20Mbps";
int nodeSpeed = 1;
int rttval;
double kbs=0;
int noofcluster=12;
int noofnodeincluster=int(NoOfSensor/noofcluster);
double Trange=80;
double Nsleep=10;
double Initialenergy=100;
double trans_recv=50;
Time interPacketInterval = Seconds (interval);
NodeContainer SensorNodes;
NodeContainer SinkNode;
int bytesTotal;
double ds=100.0;
int energy=100.0;
int cHead;
double Initialtime=0;
double PICO=0.000001;
double NANO =0.001;
int dis=10;
int status=1;
int backoff=10;
int pz=50;
double q=5;
double ro=2;
double taumax=2;
int initCity=5;
uint32_t packetsReceived;
struct rgb {
uint8_t r;
uint8_t g;
uint8_t b;};
struct rgb colors [] = {
{ 255, 0, 0 },
{ 0, 255, 0 },
{ 0, 0, 255 }};
void GetDistance_From (NodeContainer node1, NodeContainer node2){
Ptr<MobilityModel> model1 = node1.Get(0)->GetObject<MobilityModel>();
Ptr<MobilityModel> model2 = node2.Get(10)->GetObject<MobilityModel>();
distance1 = model1->GetDistanceFrom (model2);}
void ReceivePacket (Ptr<Socket> socket){
Ptr<Packet> pckt;
while (pckt = socket->Recv ()){
rate="512Mbps";
Ptr<MobilityModel> model1 = SinkNode.Get(0)->GetObject<MobilityModel>();
Ptr<MobilityModel> model2 = SensorNodes.Get(10)->GetObject<MobilityModel>();
distance1 = model1->GetDistanceFrom (model2);
pk=pk+1;}}
void compare_Minimum(double dis){
if(ds>dis){
ds=dis;}}
void getNearbynode(NodeContainer nod,double x1,double y1){
int nn;
for(uint32_t i=0;i<nod.GetN ();i++){
Ptr<ConstantPositionMobilityModel>            FCMob            =            nod.Get(i)-
 >GetObject<ConstantPositionMobilityModel>();
Vector m_position = FCMob->GetPosition();
double x=m_position.x;
double y=m_position.y;
double xx=x1-x;
double yy=y1-y;
double x2=(xx*xx);
double y2=(yy*yy);
double sx=sqrt(x2);
double sy=sqrt(y2);
double dis=(sx+sy);
compare_Minimum(dis);
if(ds==dis){
nn=i;}}
std::cout<<"minimum Distance:" <<nn<<std::endl;}
void energyReceive(int bits,int mystatus){
double en;
if(mystatus==status){
en=(double)bits*50.0*NANO;
energy=energy-en;}}
void energyTransmit(int bits,double dist,int mystatus){
double en;
if(mystatus==status){
en=bits*10*PICO*dist+(double)bits*50.0*NANO;
energy=energy-en;}}
void energyDataAggr(int signals,int mystatus){
double en;
if(mystatus==status){
en=5*NANO*signals;
energy=energy-en;}}
void finish(){
std::cout<<"energy: "<<energy<<"\n";}
static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,uint32_t pktCount, Time
 pktInterval ){
if (pktCount > 0){
socket->Send (Create<Packet> (pktSize));
int b=pktCount*512;
energy=(double)trans_recv;
energyReceive(b,1);
energyTransmit( b,ds,1);
energyDataAggr(1*pktCount ,1);
Simulator::Schedule (pktInterval, &GenerateTraffic,socket, pktSize,pktCount-1,
 pktInterval);
}else{
socket->Close ();}}
void Getcolor(NodeContainer d){
for (uint32_t i = 0; i < SensorNodes.GetN (); ++i){
uint32_t sensor1img =anim->AddResource("/home/research/ns-allinone-3.26/netanim-
 3.107/img1/sensor1.png");
Ptr<Node> wid1= SensorNodes.Get (i);
uint32_t nodeId1 = wid1->GetId ();
anim->UpdateNodeImage (nodeId1, sensor1img);
if(i<12){
anim->UpdateNodeDescription (SensorNodes.Get (i), "CH" );
anim->UpdateNodeColor (SensorNodes.Get (i), 0, 0, 255);}
else{
```

```
anim->UpdateNodeDescription (SensorNodes.Get (i), "Sensor" );
anim->UpdateNodeColor (SensorNodes.Get (i), 255, 0, 0); }}}
void DividePartition(NodeContainer At){
for(  uint32_t i=12;i<At.GetN ();i++){
Ptr<RandomWaypointMobilityModel>          FCMob          =          At.Get(i)-
 >GetObject<RandomWaypointMobilityModel>();
Vector m_position = FCMob->GetPosition();
double ux=m_position.x;
double uy=m_position.y;
if( (ux>1 && ux<450) && ( uy>1 && uy<450)){
if(ux>300 && ux<600){
anim->UpdateNodeDescription (At.Get (i), "Partition-1 , Corona-1 , Z1 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>150 && ux<300) || ( uy>600 && uy<750)){
anim->UpdateNodeDescription (At.Get (i), "Partition-1 , Corona-2 , Z2 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>0 && ux<150) || ( uy>750 && uy<900)){
anim->UpdateNodeDescription (At.Get (i), "Partition-1 , Corona-3 , Z3 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}}
else if( (ux>450 && ux<900) && ( uy>1 && uy<450)){
if(ux>300 && ux<600){
anim->UpdateNodeDescription (At.Get (i), "Partition-2 , Corona-1 , Z1 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>150 && ux<300) || ( uy>600 && uy<750)){
anim->UpdateNodeDescription (At.Get (i), "Partition-2 , Corona-2 , Z2 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>0 && ux<150) || ( uy>750 && uy<900)){
anim->UpdateNodeDescription (At.Get (i), "Partition-2 , Corona-3 , Z3 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}}
else if( (ux>1 && ux<450) && ( uy>450 && uy<900)){
if(ux>300 && ux<600){
anim->UpdateNodeDescription (At.Get (i), "Partition-3 , Corona-1 , Z1 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>150 && ux<300) || ( uy>600 && uy<750)){
anim->UpdateNodeDescription (At.Get (i), "Partition-3 , Corona-2 , Z2 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>0 && ux<150) || ( uy>750 && uy<900)){
anim->UpdateNodeDescription (At.Get (i), "Partition-3 , Corona-3 , Z3 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}}
else if( (ux>450 && ux<900) && ( uy>450 && uy<900)){
if(ux>300 && ux<600){
anim->UpdateNodeDescription (At.Get (i), "Partition-4 , Corona-1 , Z1 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>150 && ux<300) || ( uy>600 && uy<750)){
anim->UpdateNodeDescription (At.Get (i), "Partition-4 , Corona-2 , Z2 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}
else if( (ux>0 && ux<150) || ( uy>750 && uy<900)){
anim->UpdateNodeDescription (At.Get (i), "Partition-4 , Corona-3 , Z3 ");
anim->UpdateNodeColor (At.Get (i), 0,0, 0);}}}}
void PKTtrans(NodeContainer c , NodeContainer d){
```

```
Ptr<UniformRandomVariable> x = CreateObject<UniformRandomVariable> ();
x->SetAttribute ("Min", DoubleValue (1));
x->SetAttribute ("Max", DoubleValue (8));
int value = x->GetValue ();
for(uint32_t i=12;i<c.GetN ();i++){
if((i%value) == 0){
anim->UpdateNodeColor (c.Get (i), 0, 255, 255);
uint32_t    sensor2img    =anim->AddResource("/home/research/ns-allinone-3.26/netanim-
 3.107/img1/sensor2.png");
Ptr<Node> wid1= c.Get (i);
uint32_t nodeId1 = wid1->GetId ();
anim->UpdateNodeImage (nodeId1, sensor2img);
}else{
uint32_t    sensor1img    =anim->AddResource("/home/research/ns-allinone-3.26/netanim-
 3.107/img1/sensor1.png");
anim->UpdateNodeColor (c.Get (i), 255, 0, 255);
Ptr<Node> wid1= c.Get (i);
uint32_t nodeId1 = wid1->GetId ();
anim->UpdateNodeImage (nodeId1, sensor1img);
TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket (d.Get (0), tid);
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));
Ptr<Socket> source = Socket::CreateSocket (c.Get (i), tid);
InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"), 80);
source->SetAllowBroadcast (true);
source->Connect (remote);
Simulator::ScheduleWithContext    (source->GetNode    ()->GetId    (),Seconds    (0.1),
 &GenerateTraffic,source, packetSize, numPackets,interPacketInterval);}}}
void SleepTransaction(NodeContainer c , NodeContainer d){
for (uint32_t i = 0; i < Nsleep; ++i){PKTtrans(c,d);}}
int main (int argc, char *argv[]){
printf("\nEnter the Number Of Sensor Nodes ( n > 50 ) :");
scanf("%d",&NoOfSensor);
CommandLine cmd;
cmd.Parse (argc,argv);
NodeContainer allNodes;
SensorNodes.Create (NoOfSensor);
allNodes.Add (SensorNodes);
SinkNode.Create (Sink);
allNodes.Add (SinkNode);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
channel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
channel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());
double txp=80;
affinity_propagation obj;
std::string phyMode ("DsssRate11Mbps");
```

```
phy.Set ("TxPowerStart",DoubleValue (txp));
phy.Set ("TxPowerEnd", DoubleValue (txp));
Config::SetDefault ("ns3::OnOffApplication::PacketSize",StringValue ("64000"));
Config::SetDefault ("ns3::OnOffApplication::DataRate",StringValue (rate));
Config::SetDefault      ("ns3::WifiRemoteStationManager::NonUnicastMode",StringValue
 (phyMode));
Config::SetDefault ("ns3::OnOffApplication::DataRate",StringValue (rate1));
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
wifi.SetRemoteStationManager
 ("ns3::ConstantRateWifiManager","DataMode",StringValue
 (phyMode),"ControlMode",StringValue (phyMode));
WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer sensorDevices;
sensorDevices = wifi.Install (phy, mac, SensorNodes);
mac.SetType ("ns3::ApWifiMac","Ssid", SsidValue (ssid));
NetDeviceContainer sinkDevices;
sinkDevices = wifi.Install (phy, mac, SinkNode);
MobilityHelper mobility;
int nodePause = 0;
int64_t streamIndex = 0;
ObjectFactory pos;
pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
pos.Set     ("X",     StringValue     ("ns3::UniformRandomVariable[Min=50.0|Max="+
 NETWORK_X1 +"]"));
pos.Set     ("Y",     StringValue     ("ns3::UniformRandomVariable[Min=40.0|Max="+
 NETWORK_Y1 +"]"));
Ptr<PositionAllocator> taPositionAlloc = pos.Create ()->GetObject<PositionAllocator> ();
streamIndex += taPositionAlloc->AssignStreams (streamIndex);
mobility.SetPositionAllocator(taPositionAlloc);
std::stringstream ssSpeed;
ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed << "]";
std::stringstream ssPause;
ssPause << "ns3::ConstantRandomVariable[Constant=" << nodePause << "]";
mobility.SetMobilityModel          ("ns3::RandomWaypointMobilityModel","Speed",
 StringValue (ssSpeed.str ()),"Pause", StringValue (ssPause.str ()),
"PositionAllocator", PointerValue (taPositionAlloc));
MobilityHelper mobility1;
mobility1.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility1.Install (SinkNode);
for (uint32_t i = 0; i < SensorNodes.GetN (); ++i){
if(i < 12){
mobility1.Install (SensorNodes.Get (i));
}else{
mobility.Install (SensorNodes.Get (i));}}
obj.Position(SensorNodes);
AnimationInterface::SetConstantPosition (SinkNode.Get (0), 450, 450);
Time interPacketInterval = Seconds (interval);
```

```
AodvHelper aodv;
Ipv4StaticRoutingHelper staticRouting;
Ipv4ListRoutingHelper list;
list.Add (staticRouting, 0);
list.Add (aodv, 1);
InternetStackHelper internet;
internet.SetRoutingHelper (list);
internet.Install (SensorNodes);
internet.Install (SinkNode);
Ipv4AddressHelper ipv4;
NS_LOG_INFO ("Assign IP Addresses.");
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer sen = ipv4.Assign (sensorDevices);
Ipv4InterfaceContainer sin = ipv4.Assign (sinkDevices);
Simulator::Schedule(Seconds(0.2), &DividePartition,SensorNodes);
Simulator::Schedule(Seconds(0.5), &Getcolor, SensorNodes);
Simulator::Schedule(Seconds(1.1), &SleepTransaction, SensorNodes,SinkNode);
Simulator::Schedule(Seconds(1.5), &Getcolor, SensorNodes);
Simulator::Schedule(Seconds(2.1), &SleepTransaction, SensorNodes,SinkNode);
Simulator::Schedule(Seconds(2.5), &Getcolor, SensorNodes);
Simulator::Stop (Seconds (simTime));
anim= new AnimationInterface ("Sleep_Scheduling.xml");
tracewsn w;
w.stTracematrices( NoOfSensor,simTime);
anim->SetBackgroundImage                    ("/home/research/ns-allinone-3.26/netanim-
 3.107/img1/bg.png", -1055, -510, 5.50, 3.50, 1.0);
uint32_t    resourceId    =anim->AddResource("/home/research/ns-allinone-3.26/netanim-
 3.107/img1/sink.png");
uint32_t    sensor1img    =anim->AddResource("/home/research/ns-allinone-3.26/netanim-
 3.107/img1/sensor1.png");
for (uint32_t i = 0; i < SensorNodes.GetN (); ++i){
Ptr<Node> wid1= SensorNodes.Get (i);
uint32_t nodeId1 = wid1->GetId ();
anim->UpdateNodeSize (nodeId1, 150.0,150.0);
anim->UpdateNodeImage (nodeId1, sensor1img);
anim->UpdateNodeDescription (SensorNodes.Get (i), "Sensor" );
anim->UpdateNodeColor (SensorNodes.Get (i), 255, 0, 0); }
anim->UpdateNodeDescription (SinkNode.Get (0), "Sink" );
anim->UpdateNodeColor (SinkNode.Get (0), 200, 200, 200);
Ptr<Node> wid2= SinkNode.Get (0);
uint32_t nodeId2 = wid2->GetId ();
anim->UpdateNodeSize (nodeId2, 225.0,225.0);
anim->UpdateNodeImage (nodeId2, resourceId);
Ptr<RttMeanDeviation> myrtt = CreateObject<RttMeanDeviation> ();
Simulator::Run ();
Simulator::Destroy ();
system("gnuplot 'Throughput.plt'");
system("gnuplot 'Energy_Consumption_Nodes.plt'");
system("gnuplot 'Energy_Consumption_Simtime.plt'");
system("gnuplot 'Network_Lifetime_Nodes.plt'");
system("gnuplot 'Network_Lifetime_Simtime.plt'");
system("gnuplot 'Delay.plt'");
return 0;}
```

Appendix 7: Python codes for sensor data analysis

```
import time
import json
import serial
import requests

ser=serial.Serial("/dev/ttyUSB0", 9600)

while True:
   if ser.in_waiting > 0 :
      data=ser.readline()
      time.sleep(.2)
      if(len(data)> 10):
         j=data.decode('utf-8')
         print(j)

         jo=json.loads(j)
         if(jo['Id']=="EN1"):

 requests.get("https://api.thingspeak.com/update?api_key=34Y70BIO0PLFCAIJ&field2={}
 &field1={}".format(jo['sensor'][0],jo['sensor'][1]))
         if(jo['Id']=="EN2"):

 requests.get("https://api.thingspeak.com/update?api_key=OSEA2427GR888NUN&field2={
 }&field1={}".format(jo['sensor'][0],jo['sensor'][1]))

         print("Sensor Id= {}, Temprature = {} C, Humidity= {} %".format(jo['Id'],
 jo['sensor'][1],jo['sensor'][0] ))
```