DYNAMIC ROUTING AND POWER CONTROL IN WSNs WITH K-MEANS AND Q-LEARNING USING ARTIFICIAL INTELLIGENCE

PROJECT EVALUATION I REPORT

Submitted by

REENA S

SHREE HARINI S

SOUNDARYA AL

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND BUSINESS SYSTEMS

K. RAMAKRISHNAN COLLEGE OF

ENGINEERING

(AUTONOMOUS)

SAMAYAPURAM, TRICHY

ANNA UNIVERSITY

CHENNAI 600 025

NOVEMBER 2025

DYNAMIC ROUTING AND POWER CONTROL IN WSNs WITH K-MEANS AND Q-LEARNING USING ARTIFICIAL INTELLIGENCE

UCB1713 PROJECT EVALUATION I REPORT

Submitted by

REENA S (8115U22CB044)

SHREE HARINI S (8115U22CB050)

SOUNDARYA AL (8115U22CB052)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND BUSINESS SYSTEMS

Under the Guidance of

Mrs. P. UMA MAHESHWARI, M.E.,

COMPUTER SCIENCE AND BUSINESS SYSTEMS

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this project report **“DYNAMIC ROUTING AND POWER CONTROL IN WSNs WITH K-MEANS AND Q-LEARNING USING ARTIFICIAL INTELLIGENCE”** is the Bonafide work of **“REENA S (8115U22CB044), SHREE HARINI S (8115U22CB050), SOUNDARYA AL (8115U22CB052)”**who carried out the project work under my supervision.

Dr. J. SASIDEVI, M.E., Ph.D.,	**Mrs. P. UMA MAHESHWARI, M.E.,**
Associate Professor	Assistant Professor
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Computer Science and Business Systems	Department of Computer Science and Business Systems
K. Ramakrishnan College of Engineering, (Autonomous)	K. Ramakrishnan College of Engineering, (Autonomous)
Samayapuram, Trichy-621112.	Samayapuram, Trichy-621112.

SIGNATURE OF INTERNAL EXAMINER	**SIGNATURE OF EXTERNAL EXAMINER**
NAME :	**NAME :**
DATE :	**DATE :**

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

ACKNOWLEDGMENT

We thank the almighty GOD, without whom it would not have been possible for us to complete our project.

We wish to address our profound gratitude to **Dr. K. RAMAKRISHNAN,** Chairman, K. Ramakrishnan College of Engineering (Autonomous), who encouraged and gave us all help throughout the course.

We express our hearty gratitude and thanks to our honorable and grateful Executive Director **Dr. S. KUPPUSAMY, B.Sc., MBA., Ph.D.,** K.Ramakrishnan College of Engineering (Autonomous).

We are glad to thank our Principal **Dr. D. SRINIVASAN, M.E., Ph.D., FIE., MIIW., MISTE., MIS, AE., C.Eng.,** K.Ramakrishnan College of Engineering (Autonomous), for giving us permission to carry out this project

We wish to convey our sincere thanks to **Dr. J. SASIDEVI, M.E., Ph.D.,** Head of the Department of Computer Science and Business Systems, K. Ramakrishnan College of Engineering (Autonomous), for giving us constant encouragement and advice throughout the course.

We are grateful to **Mrs. P. UMA MAHESHWARI, M.E.,** Assistant Professor in Computer Science and Business Systems, K. Ramakrishnan College of Engineering (Autonomous), for her guidance and valuable suggestions during the course of study.

We sincerely thank, **Mrs. B. SATHIYA, M.TECH.,** for outstanding leadership and dedication throughout coordinating the project.

Finally, we sincerely acknowledge in no less terms all our staff members, colleagues, parents and friends for their cooperation and help at various stages of this project work.

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

DECLARATION BY THE CANDIDATE

I declare that to the best of my knowledge the work reported here in has been composed solely by myself and that it has not been in whole or in part in any previous application for a degree.

Submitted for the project Viva-Voce held at K. Ramakrishnan College of Engineering on __________

SIGNATURE OF THE CANDIDATE

(REENA S)

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

DECLARATION BY THE CANDIDATE

I declare that to the best of my knowledge the work reported here in has been composed solely by myself and that it has not been in whole or in part in any previous application for a degree.

Submitted for the project Viva-Voce held at K. Ramakrishnan College of Engineering on __________

SIGNATURE OF THE CANDIDATE

(SHREE HARINI S)

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI

DECLARATION BY THE CANDIDATE

I declare that to the best of my knowledge the work reported here in has been composed solely by myself and that it has not been in whole or in part in any previous application for a degree.

Submitted for the project Viva-Voce held at K. Ramakrishnan College of Engineering on __________

SIGNATURE OF THE CANDIDATE

(SOUNDARYA AL)

ABSTRACT

Wireless Sensor Networks are used in environmental monitoring, healthcare, industrial automation, and smart cities. Since sensor nodes have limited battery power, low communication range, and minimal processing capability, managing energy efficiently is crucial for longer network lifetime. This research introduces an AI-based approach that combines K-means clustering with Q-learning for intelligent routing and dynamic power control. K-means forms energy-efficient clusters to reduce unnecessary communication, while Q-learning learns the best routing paths and adjusts transmission power based on network conditions and battery levels. This makes the network adaptive and capable of improving its performance over time. When compared with traditional methods like LEACH and its multi-hop versions, as well as recent energy-efficient techniques, the proposed method shows major improvements in active node count, energy usage, remaining battery power, and successful packet delivery to the base station. The advantages are more noticeable in large networks with many nodes. Overall, the combination of machine learning and classical communication techniques creates a more stable, efficient, and long-lasting Wireless Sensor Network for future smart environments, with strong adaptability to dynamic topology changes and varying traffic loads, reduced packet loss under harsh conditions, high scalability as node density increases, minimal computational overhead suitable for low-power devices, and strong potential for deployment in complex and evolving ecosystems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	**EXPANSION**
WSN / WSNs	Wireless Sensor Network(s)
LEACH	Low Energy Adaptive Clustering Hierarchy
DMHT-LEACH	Distributed Multi-Hop LEACH
EDMHT-LEACH	Enhanced Distributed Multi-Hop LEACH
TEEN	Threshold Sensitive Energy Efficient Sensor Network Protocol
MPSO	Modified Particle Swarm Optimization
API	Application Programming Interface
PIL	Python Imaging Library
SciPy	Scientific Python
Numpy	Numerical Python
Pandas	Python Data Analysis Library

CHAPTER 1

INTRODUCTION

1.1 WIRELESS SENSOR NETWORKS (WSNs)

Wireless Sensor Networks (WSNs) are distributed systems made up of many low-power sensor nodes that can sense, process, and transmit data over wide areas. They are widely used in fields such as environmental monitoring, industrial automation, smart agriculture, disaster management, and military surveillance. A typical WSN includes sensor nodes, cluster heads, and a base station that receives and processes the collected information.

However, WSNs face major challenges due to limited battery power, short communication ranges, and low processing capability. Since sensor nodes are battery-operated and often deployed in remote or hazardous areas, replacing or recharging them is difficult. Therefore, energy efficiency becomes the most important factor in determining network lifetime. To address this, intelligent clustering and energy-aware routing strategies are needed. While traditional protocols like LEACH offer basic energy-saving features, they struggle to adapt to dynamic network conditions, highlighting the need for scalable, adaptive, and learning-based solutions.

1.2 TECHNOLOGIES USED

1.2.1 K-Means Clustering

Used for grouping sensor nodes into clusters based on their physical location. It helps minimize intra-cluster communication distance and balances energy usage.

1.2.2 Q-Learning (Reinforcement Learning)

A learning-based routing strategy that enables cluster heads to choose energy-efficient paths dynamically. The model improves routing decisions through continuous learning.

1.2.3 Wireless Communication Protocols

Energy models and communication standards define how nodes transmit, receive, and aggregate data, ensuring efficient network operation.

1.2.4 Simulation Frameworks

Tools such as Python, MATLAB, or NS-3 enable modeling, visualization, and performance evaluation of clustering and routing algorithms.

1.2.5 Energy Models

The first-order radio model calculates energy consumed during transmission, reception, and data aggregation, helping evaluate system performance.

1.3 APPLICATIONS

Energy-efficient clustering and routing in WSNs have broad real-world applications:

- **Smart Agriculture**

  Monitoring soil moisture, temperature, and crop growth efficiently over long durations.

- **Environmental Monitoring**

  Tracking pollution levels, forest fires, water quality, and wildlife movements.

- **Industrial Automation**

  Supervising equipment health, machine vibrations, and gas leak detections.

- **Smart Cities**

  Managing street lighting, traffic systems, and waste management with low-energy sensors.

- **Military and Border Security**

  Monitoring strategic zones where sensor replacement is difficult.

- **Healthcare Monitoring**

  Wearable sensors for patient health tracking with prolonged battery life.

1.4 DISADVANTAGES OF EXISTING METHODS

1. Static Routing Decision

Traditional protocols like LEACH use predefined rules that do not adapt to changing energy levels or network topology**.**

2. Uneven Energy Consumption

Some nodes become overburdened, resulting in early energy depletion and reduced network lifetime.

3. Limited Scalability

Existing algorithms struggle when node count increases, causing performance degradation.

4. Lack of Learning-Based Optimization

No mechanism exists to learn from previous routing decisions and improve future communication paths.

5. High Packet Loss

Long-distance communication without multi-hop optimization leads to data loss and inefficiency.

1.5 FEATURE ENGINEERING IN WSN OPTIMIZATION

The clustering and routing rely largely on algorithmic operations, certain features must be extracted or computed for efficient functioning

- **Distance Calculation**
  Used to assign nodes to clusters and estimate transmission energy.
- **Residual Energy Tracking**
  Helps determine eligible cluster heads and route-quality metrics.
- **Node Density and Neighbor Count**
  Influence cluster size and routing efficiency.
- **State Representation for Q-Learning:**
  Converts raw measurements (energy, distance, hops) into meaningful states.

1.6 HYPERPARAMETER OPTIMIZATION

1.6.1 Learning Rate (α) in Q-Learning

Determines how quickly the routing agent updates its knowledge.

1.6.2 Discount Factor (γ)

Balances immediate energy savings with long-term network benefits.

1.6.3 Exploration Rate (ε)

Controls the trade-off between exploring new routes.

1.6.4 Number of Clusters (K)

Impacts cluster size, energy consumption, and communication overhead.

1.6.5. Cluster Interval:

Determines how frequently clusters should be rebuilt to adapt to network.

1.7 NORMALIZATION

Normalization ensures uniform scaling and stability of data used in clustering and routing:

- **Coordinate Normalization:** Scaling node positions ensures that clustering remains consistent across varying network sizes.
- **Energy Normalization:** Residual energy values are often normalized between 0 and 1 to simplify decision-making.
- **Routing State Normalization:** Input features for Q-learning (distance, hops, energy) are normalized to maintain balanced learning.

1.8 ADVANTAGES OF THE PROPOSED SYSTEM

- **Higher Energy Efficiency**

  Adaptive learning reduces unnecessary transmissions and balances load among nodes.

- **Improved Network Lifetime**

  Effective CH rotation, multi-hop routing, and Q-learning ensure longer sensor operation.

- **Scalable Architecture**

  Suitable for dense and large-scale sensor deployments.

- **Better Packet Delivery Rate**

  Intelligent routing leads to reduced packet drops and improved reliable.

- **Dynamic Adaptability**

  Model learns from network behavior, adapting routes based on energy.

CHAPTER 2

LITERATURE SURVEY

2.1 Q-Learning-Based Data-Aggregation-Aware Energy-Efficient Routing Protocol for Wireless Sensor Networks

AUTHORS: WAN-KYU YUN AND SANG-JO YOO

YEAR OF PUBLICATION: 2021

A Q-learning-based, data-aggregation-aware, energy-efficient routing protocol for Wireless Sensor Networks (WSNs), where energy consumption critically impacts network lifetime. To reduce redundant transmissions, the method applies sensor-type-dependent data aggregation and uses reinforcement learning to select optimal routes based on aggregation efficiency, communication cost, and residual energy. By maximizing these rewards, the protocol avoids low-energy nodes and identifies effective forwarding paths. Simulation results show that the proposed approach significantly reduces data volume and improves overall network lifetime compared to conventional energy-aware routing schemes.

2.2 EEMLCR: Energy-Efficient Machine Learning-Based Clustering and Routing for Wireless Sensor Networks

AUTHORS: MUHAMMAD AKRAM, SIBGHAT ULLAH BAZAI, MUHAMMAD IMRAN

YEAR OF PUBLICATION: 2025

An Energy-Efficient Machine Learning-Based Clustering and Routing method for Wireless Sensor Networks (WSNs). Due to the limited power, communication range, and processing capability of sensor nodes, clustering and efficient routing are essential for reducing energy consumption. EEMLCR applies K-means clustering and Q-learning to automate cluster formation and routing path selection. The method is evaluated against LEACH and its multi-hop variants (DMHT-LEACH and EDMHT-LEACH), showing significant improvements in alive node count, energy usage, remaining battery power, and packet reception after 600 rounds in a 400-node network. Comparisons with recent algorithms such as EECDA and CMML further confirm its competitive or superior performance in terms of network lifetime and energy efficiency. Overall, EEMLCR demonstrates how machine learning can optimize clustering and routing, leading to more energy-efficient and longer-lasting WSN deployments.

2.3 Integration of artificial intelligence (AI) with sensor networks: Trends, challenges, and future directions

AUTHORS: SALIM EL KHEDIRI, AWATEF BENFRADJ

YEAR OF PUBLICATION: 2024

Artificial Intelligence (AI) can enhance Wireless Sensor Networks (WSNs), which are traditionally configured statically and difficult to adapt dynamically. Integrating AI, including machine learning and reinforcement learning, can improve energy management, reduce waste, and extend the lifespan of WSN and IoT devices. The paper reviews supervised, unsupervised, and reinforcement learning techniques applied to WSNs , offering researchers a clear overview of recent advancements. It also analyzes the strengths and limitations of current AI-based approaches, highlights ongoing challenges, and outlines future directions for achieving more sustainable and adaptable WSN solutions.

2.4 An Energy-Efficient Scheduling and Routing Protocol Based on Q-Learning for WSN

AUTHORS: HUDA SALIH ABD, ASAAD S DAGHUL

YEAR OF PUBLICATION: 2025

Nodes make forwarding decisions using local energy, distance, and link-quality information, while sleep mode reduces unnecessary communication. Compared with LEACH and the proposed method performs better in all metrics, proving the value of combining learning-based routing with adaptive clustering for longer WSN lifetime. This work presents an energy-efficient WSN routing protocol that integrates dynamic K-means clustering, Q-learning routing, and adaptive sleep scheduling.

2.5 Reinforcement Learning for Tackling Energy-saving and Energy-balance Dilemma of Cluster-based Routing Protocols in WSNs

AUTHORS: YAN WANG, KE DENG, YONGLI REN, JEFFREY CHAN

YEAR OF PUBLICATION: 2024

Tackles the long-standing energy-saving and energy-balance dilemma in cluster-based routing protocols for Wireless Sensor Networks (WSNs). Instead of introducing a new routing protocol, the authors employ reinforcement learning to enhance existing ones by making two key decisions each round: whether reclustering is necessary and which cluster-heads should be excluded from multi-hop relaying. These adaptive decisions help reduce energy consumption while evenly distributing energy usage across nodes. Experimental results demonstrate that the reinforcement learning–based approach significantly improves energy efficiency and energy balance compared to traditional methods, thereby extending the operational lifespan of WSNs

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

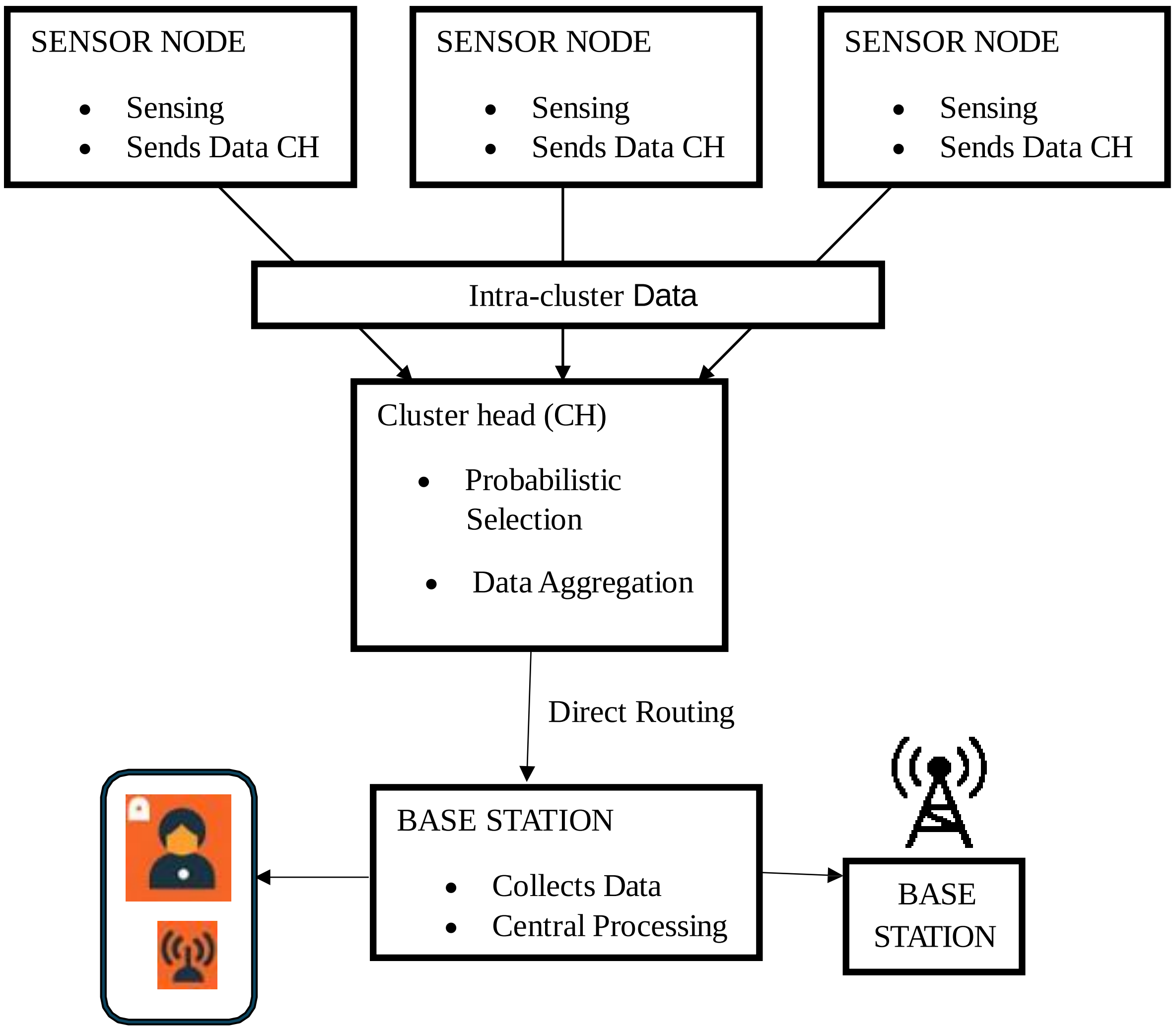

Fig 3.1. EXISTING SYSTEM ARCHITECTURE

• Wireless Sensor Networks (WSNs) are widely used in environmental monitoring, industrial automation, military systems, and smart cities. Although protocols like

LEACH, DMHT-LEACH, and EDMHT-LEACH aim to save energy through clustering, they rely on static or probabilistic rules that do not adjust to changing energy levels, network density, or communication distances.

• Lack of intelligent and adaptive decision-making leads to inefficient cluster formation and routing. Low-energy nodes may still be chosen as cluster heads, causing early energy depletion, uneven load distribution, and reduced network lifetime. Traditional methods also fail to optimize routing paths using learning-based techniques.

• Limited support for adaptive multi-hop routing results in suboptimal communication, higher energy usage, and greater packet loss in large-scale deployments. This causes static behaviour, inefficient energy usage, and poor scalability in current WSN systems.

3.2 PROPOSED SYSTEM

• The proposed system, EEMLCR (Energy-Efficient Machine Learning-Based Clustering and Routing), introduces a hybrid machine learning-based approach to overcome the limitations of existing WSN protocols. This system integrates K-means clustering and Q-learning-based routing to intelligently manage energy consumption and extend network lifetime.

• In this system, K-means clustering is used to form energy-balanced clusters by grouping sensor nodes based on their spatial positions. This ensures that clusters are well distributed across the network. After cluster formation, cluster heads (CHs) are selected based on proximity to cluster centroids and residual energy, improving fairness and load distribution.

• Once the clusters are formed, Q-learning, a reinforcement learning technique, is used to determine the most energy-efficient routing paths from cluster heads to the base station. The learning agent continuously updates its routing policy based on energy consumption,

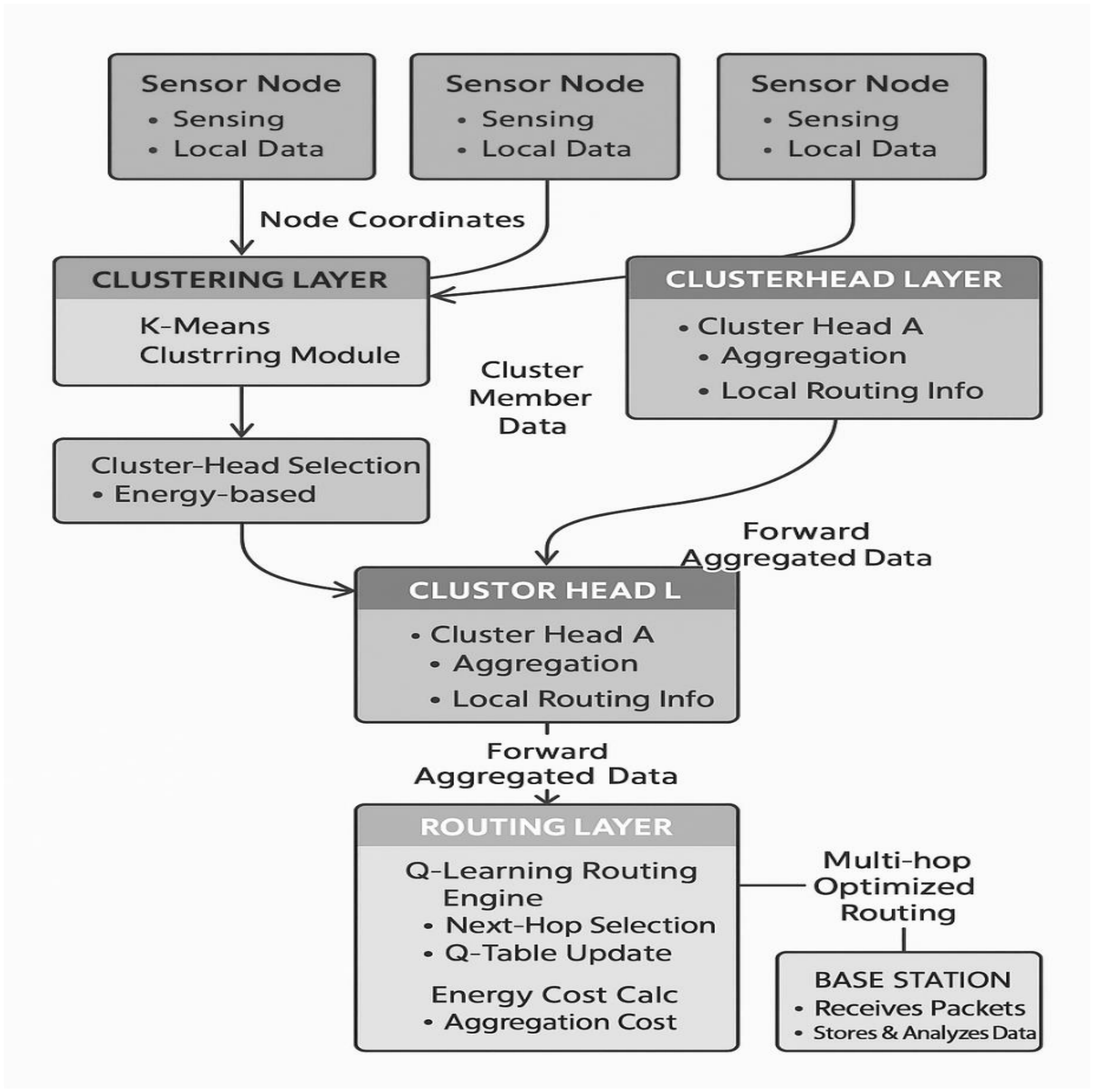

Fig 3.2. PROPOSED SYSTEM BLOCK DIAGRAM

• Successful packet delivery, and hop count. This adaptive routing allows the network to identify optimal multi-hop paths while minimizing power usage.

• Compared to LEACH and its multi-hop variants, EEMLCR dynamically adapts to changing network conditions, resulting in increased alive nodes, reduced average energy consumption, improved packet reception rates, and longer network lifetime. Moreover, the proposed system is evaluated against modern algorithms such as EECDA and CMML, demonstrating comparable or superior performance.

• Overall, the proposed system provides an intelligent, energy-aware, and scalable solution that significantly enhances the efficiency and lifetime of Wireless Sensor Networks.

3.3 ARCHITECTURE DIAGRAM

The architecture diagram of the proposed EEMLCR system illustrates the complete workflow of clustering and routing in Wireless Sensor Networks using machine learning. The process begins with the deployment of sensor nodes across the monitoring area. Each node is initialized with its position, unique ID, and energy level.

The system first performs data preprocessing, where node attributes such as coordinates, neighbor lists, and residual energy are collected. These values are passed to the K-means clustering module, which groups nodes into optimal clusters. After clustering, the Cluster Head (CH) Election Module selects a CH for each cluster based on centroid closeness and available energy.

Once the CHs are selected, the system applies the Q-learning routing module. For each CH, a reinforcement learning agent evaluates possible next-hop nodes and selects the most energy-efficient route toward the base station.

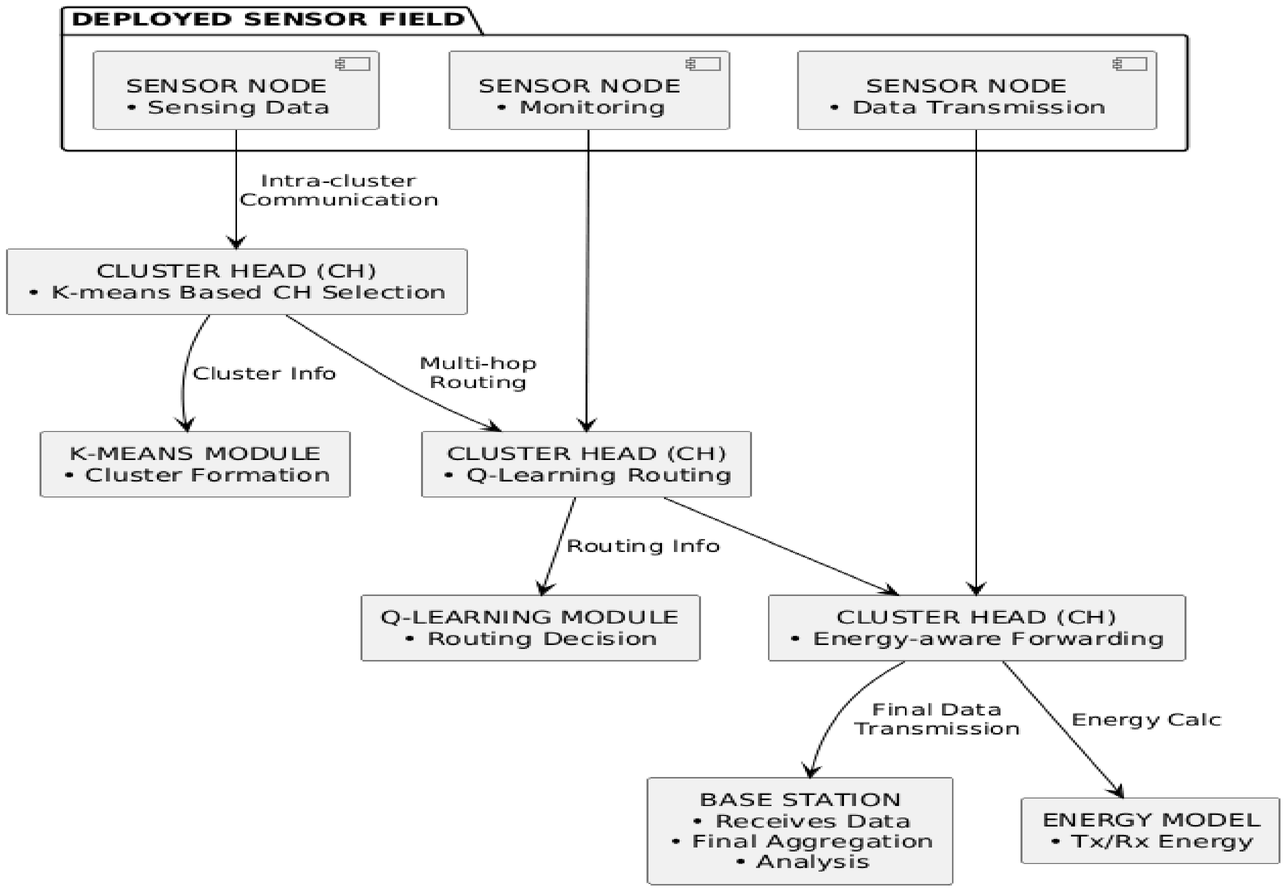
DEPLOYED SENSOR FIELD
SENSOR NODE • Sensing Data
SENSOR NODE • Monitoring
SENSOR NODE • Data Transmission
Intra-cluster Communication
CLUSTER HEAD (CH) • K-means Based CH Selection
Cluster Info
Multi-hop Routing
K-MEANS MODULE • Cluster Formation
CLUSTER HEAD (CH) • Q-Learning Routing
Routing Info
Q-LEARNING MODULE • Routing Decision
CLUSTER HEAD (CH) • Energy-aware Forwarding
Final Data Transmission
Energy Calc
BASE STATION • Receives Data • Final Aggregation • Analysis
ENERGY MODEL • Tx/Rx Energy

Fig 3.3. ARCHITECTURE DIAGRAM

During data transmission, nodes send their sensed data to their CHs. The CH aggregates the data and forwards it through the Q-learning-optimized route. The Energy Model continuously updates the energy levels of nodes based on transmission and reception activities.

The final output consists of performance metrics such as total energy consumption, number of alive nodes, packet reception rate, and remaining energy after a defined number of rounds. The architecture integrates clustering, learning-based routing, energy modeling, and performance evaluation, forming a complete intelligent WSN communication framework

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

- Python
- Anaconda
- Jupyter Notebook

4.2 HARDWARE REQUIREMENTS

• Processor	**:**	Intel Core i5
• RAM	**:**	8GB
• OS	**:**	Windows

4.3 SOFTWARE DESCRIPTION

Python

Python is a general-purpose, high-level, interpreted, interactive, and object-oriented programming language developed by Guido van Rossum between 1985 and 1990. Its source code is available under the GNU General Public License (GPL). Known for its simple syntax and readability, Python uses English keywords instead of heavy punctuation, making it easy to learn and write. This tutorial provides a basic understanding of the Python programming language. Python supports both object-oriented and procedural programming and is currently one of the most widely used languages. Python programs are usually shorter and more readable than those written in languages like Java due to its clean indentation rules.

It is used by major companies such as Google, Amazon, Facebook, Instagram, Dropbox, and Uber. One of Python's greatest strengths is its extensive standard library, which supports machine learning, GUI development (Kivy, Tkinter, PyQt), web frameworks (Django), image processing (OpenCV, Pillow), web scraping (Scrapy, Beautiful Soup, Selenium), testing, multimedia, and scientific computing.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following :

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, Beautiful Soup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing

Tensor Flow

Tensor Flow is an end-to-end open-source platform for machine learning. It offers a flexible ecosystem of tools, libraries, and community support that helps researchers advance ML techniques and enables developers to build and deploy ML-powered applications easily. It provides intuitive, high-level APIs for beginners and experts to create machine learning models in multiple programming languages. Tensor Flow models can be deployed on servers, cloud platforms, mobile and edge devices, web browsers, and various JavaScript environments, making it easy to move from model development to deployment.

Tensor Flow provides a collection of workflows with intuitive, high-level APIs for both beginners and experts to create machine learning models in numerous languages. Developers have the option to deploy models on a number of platforms such as on servers, in the cloud, on mobile and edge devices, in browsers, and on many other JavaScript platforms. This enables developers to go from model building and training to deployment much more easily.

Keras

Keras is a deep learning API written in Python that runs on top of TensorFlow. It is designed for fast experimentation and provides a user-friendly interface for building deep learning models. Keras supports both CPU and GPU execution and includes built-in modules for convolutional networks, recurrent networks, and combinations of both. It also allows the creation of complex architectures such as multi-input or multi-output models and shared layers, making it suitable for a wide range of deep learning applications.

• Allows the same code to run on CPU or on GPU, seamlessly.

• User-friendly API which makes it easy to quickly prototype deep learning models.

• Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.

• Supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, etc.

• This means that Keras is appropriate for building essentially any deep learning model, from a memory network to a neural Turing machine.

Pandas

Pandas is a powerful and flexible open-source library used for data analysis and manipulation in Python. It provides efficient data structures like DataFrames that make working with labeled or tabular data simple and intuitive. Pandas supports importing data from formats like CSV, JSON, Excel, SQL, and Parquet. It offers rich functionalities for data cleaning, merging, reshaping, selecting, and transforming, and is built on top of NumPy, bringing many R-like DataFrame features into Python.

Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. The development of pandas introduced into Python many comparable features of working with Data frames that were established in the R programming language. The panda's library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on Data frames.

NumPy

NumPy, short for Numerical Python, is a core library for numerical computing. It provides high-performance multidimensional arrays and a collection of mathematical and logical functions to operate on them. Many scientific and machine learning libraries in Python are built on top of NumPy, making it a fundamental tool for array-based computations.

NumPy is a powerful Python library widely used for numerical and scientific computing. It provides efficient multi-dimensional arrays (nd arrays) that are faster and more memory-efficient than Python lists. With its vast collection of mathematical and statistical functions, NumPy enables operations like vectorization, which avoids slow Python loops and greatly boosts performance. It also supports linear algebra, Fourier transforms, and random number generation, making it essential for data science, machine learning, and scientific applications. Additionally, NumPy serves as the foundation for many major libraries such as Pandas, SciPy, and TensorFlow, making it a core tool in the Python ecosystem.

Matplotlib

Matplotlib is a comprehensive visualization library in Python used to create static, animated, and interactive plots. It works closely with NumPy and supports embedding graphs into applications built with GUI frameworks like Tkinter, Qt, wxPython, or GTK. Matplotlib is widely used for creating line plots, bar charts, scatter plots, histograms, and many other graphical representations.

This is a widely used plotting library for Python. Matplotlib enables the creation of static, animated, and interactive visualizations in Python. It provides a flexible and powerful framework for generating various types of plots, including line plots, scatter plots, bar charts, histograms, and more, making it essential for data exploration and presentation.

Scikit Learn

Scikit-learn is a widely used machine learning library built on top of NumPy and SciPy. It provides tools for classification, regression, clustering, and dimensionality reduction, offering algorithms such as SVM, Random Forest, Gradient Boosting, K-Means, and DBSCAN. It is simple to use and integrates well with other scientific computing libraries.

This is an open-source machine learning library for Python. It provides a wide range of tools for data analysis and modeling, including algorithms for classification, regression, clustering, dimensionality reduction, and more. Scikit-learn is built upon other scientific Python libraries like NumPy and SciPy, and it integrates well with Matplotlib for data visualization.

Pillow

Pillow is a popular Python imaging library that acts as a friendly fork of the original PIL (Python Imaging Library). It provides easy-to-use functions for loading, displaying, and manipulating images, including creating new images and applying transformations.

This library is a fork of the Python Imaging Library (PIL) and provides extensive image processing capabilities. Pillow allows for opening, manipulating, and saving various image file formats. It is commonly used for tasks such as resizing, cropping, rotating, applying filters, and color manipulation in image

OpenCV

OpenCV is an open-source computer vision library used to help machines interpret and understand digital images and videos. It provides tools for face recognition, object detection, image processing, and many other vision-based tasks. OpenCV is widely used in AI, robotics, and real-time image analysis applications.

OpenCV is a comprehensive library for computer vision and machine learning. It offers a vast collection of algorithms and functions for image and video analysis, including object detection, facial recognition, image segmentation, and real-time computer vision applications.

Features of python

There are many features in Python, some of which are discussed below –

Easy to code: Python is a high-level programming language. Python is very easy to learn thelanguage as compared to other languages like C, C#, Java script, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

Free and Open Source: Python language is freely available at the official website and can download it easily.

Object-Oriented Language: One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

High-Level Language: Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

Extensible feature: Python is an Extensible language. We can write us some Python code into C or C++ language and also, we can compile that code in C/C++ language.

Python is Portable language: Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

Python is Integrated language: Python is also an Integrated language because we can easily integrated python with other languages like C, C++, etc.

Interpreted Language: Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode .

Large Standard Library: Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc

Dynamically Typed Language: Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

Advantages of Python

Integration Feature: Python integrates the Enterprise Application Integration that makes it easy to develop Web services by invoking COM or COBRA components. It has powerful control capabilities as it calls directly through C, C++ or Java via Python. Python also processes XML and other mark-up languages as it can run on all modern operating systems through same byte code.

Improved Programmer’s Productivity: The language has extensive support libraries and clean object-oriented designs that increase two to tenfold of programmer‟s productivity while using the languages like Java, VB, Perl, C, C++ and C#.

Productivity: With its strong process integration features, unit testing framework and enhanced control capabilities contribute towards the increased speed for most applications and productivity of applications. It is a great option for building scalable multi-protocol network applications.

Domain

Artificial Intelligence (AI) refers to the ability of machines and computer systems to perform tasks that typically require human intelligence. These tasks include learning from data, recognizing patterns, making decisions, and solving problems. AI techniques—such as machine learning, deep learning, and reinforcement learning—allow systems to improve their performance over time without being explicitly programmed for every scenario. Today, AI is widely used in industries like healthcare, transportation, communication, and smart environments, enabling automation, increased efficiency, and intelligent decision-making.

Jupyter

JupyterLab is the most recent interactive development environment for notebooks, code, and data on the web. Users may create and arrange workflows in datascience, scientific computing, computational journalism, and machine learning using its versatile interface. A modular design encourages additions to enhance and increase functionality.

Jupyter Notebook is an open-source, web-based interactive platform for creating and sharing documents including live code, mathematical equations, images, maps, charts, visualizations, and narrative prose. It is compatible with a wide range of programming languages, including Python, PHP, R, C#, and others.

Jupyter Notebook, as you may know, is an open-source web-based interactive environment that integrates code, text, photos, videos, mathematical equations, charts, maps, graphical user interface, and widgets into a single document.

FRONT END

Anaconda is a free and open distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

Conda

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-languages. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository

The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

BACK END

Kernel

A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

Notebook Dashboard

The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used toopen notebook documents, and to manage the running kernels (visualize and shutdown). The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

CHAPTER 5

MODULE DESCRIPTION

MODULES

5.1 Data Collection & Initialization Process

5.2 K-Means Clustering

5.3 Q-Learning-Based Routing Modules

5.4 Energy Consumption Analysis

5.5 Performance Evaluation Module

5.6 Results Visualization & Analysis Module

5.1 Data Collection & Initialization Process

This module collects the initial parameters required to set up the Wireless Sensor Network (WSN). It initializes sensor node positions, energy levels, communication range, number of clusters, and base station location. The module ensures that all nodes are properly deployed and ready for clustering and routing operations.

Functions:

- Deploy sensor nodes randomly or in a grid
- Assign initial energy to each node
- Set simulation parameters (rounds, area size, threshold values)
- Define base station position

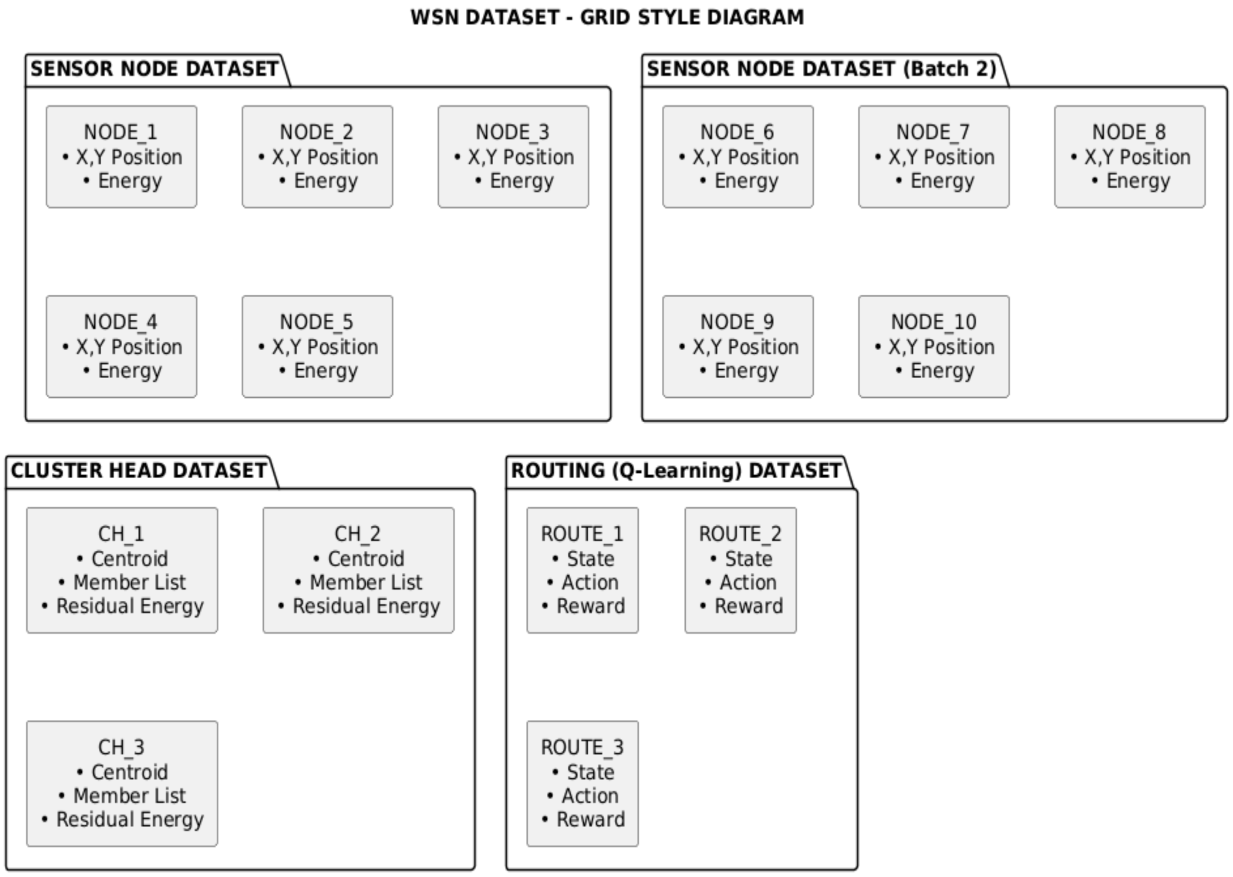
WSN DATASET - GRID STYLE DIAGRAM
SENSOR NODE DATASET
NODE_1 • X,Y Position • Energy
NODE_2 • X,Y Position • Energy
NODE_3 • X,Y Position • Energy
NODE_4 • X,Y Position • Energy
NODE_5 • X,Y Position • Energy
SENSOR NODE DATASET (Batch 2)
NODE_6 • X,Y Position • Energy
NODE_7 • X,Y Position • Energy
NODE_8 • X,Y Position • Energy
NODE_9 • X,Y Position • Energy
NODE_10 • X,Y Position • Energy
CLUSTER HEAD DATASET
CH_1 • Centroid • Member List • Residual Energy
CH_2 • Centroid • Member List • Residual Energy
CH_3 • Centroid • Member List • Residual Energy
ROUTING (Q-Learning) DATASET
ROUTE_1 • State • Action • Reward
ROUTE_2 • State • Action • Reward
ROUTE_3 • State • Action • Reward

Fig 5.1. DATASET

5.2 K-Means Clustering Module

This module performs the clustering process using the K-means algorithm. It groups sensor nodes based on their spatial distribution to form energy-efficient and balanced clusters. The clustering aims to reduce communication distance and minimize intra-cluster energy consumption.

Functions:

- Calculate optimal number of clusters
- Assign nodes to nearest cluster centers

- Update cluster centroids iteratively
- Select cluster heads
- Form stable and energy-balanced clusters

5.3 Q-Learning-Based Routing Module

This module uses Q-learning to determine the most energy-efficient routing path from each cluster head to the base station. It allows cluster heads to learn and adapt their routing strategies based on network conditions.

Functions:

- Initialize Q-table for each cluster head
- Define states (energy level, distance, neighbor status)
- Define actions (choose next-hop node)
- Apply reward function based on energy saving & successful delivery
- Update Q-values using reinforcement learning
- Select optimal multi-hop routing path

5.4 Energy Consumption Calculation Analysis

This module calculates the energy consumed during data transmission, reception, aggregation, and routing. It uses the radio energy model to accurately measure energy usage across the network.

Functions:

- Compute transmission energy
- Compute reception energy

- Apply data aggregation cost at cluster heads
- Update remaining energy of all nodes
- Track total energy consumption per round

5.5 Performance Evaluation Module

This module evaluates the efficiency of the proposed EEMLCR method. It compares the performance with benchmark algorithms such as LEACH, DMHT-LEACH, EDMHT-LEACH, EECDA, and CMML.

Performance Metrics:

- Number of alive nodes
- Average remaining energy
- Energy consumption per round
- Packet delivery ratio
- Network lifetime (stability period & instability period)

5.6 Results Visualization & Analysis Module

This module generates graphs, statistics, and visualization plots to analyze the performance .

Outputs:

- Alive nodes vs. rounds graph
- Remaining energy vs. rounds graph
- Packet received vs. rounds graph
- Comparison graphs with LEACH family & state-of-the-art algorithms

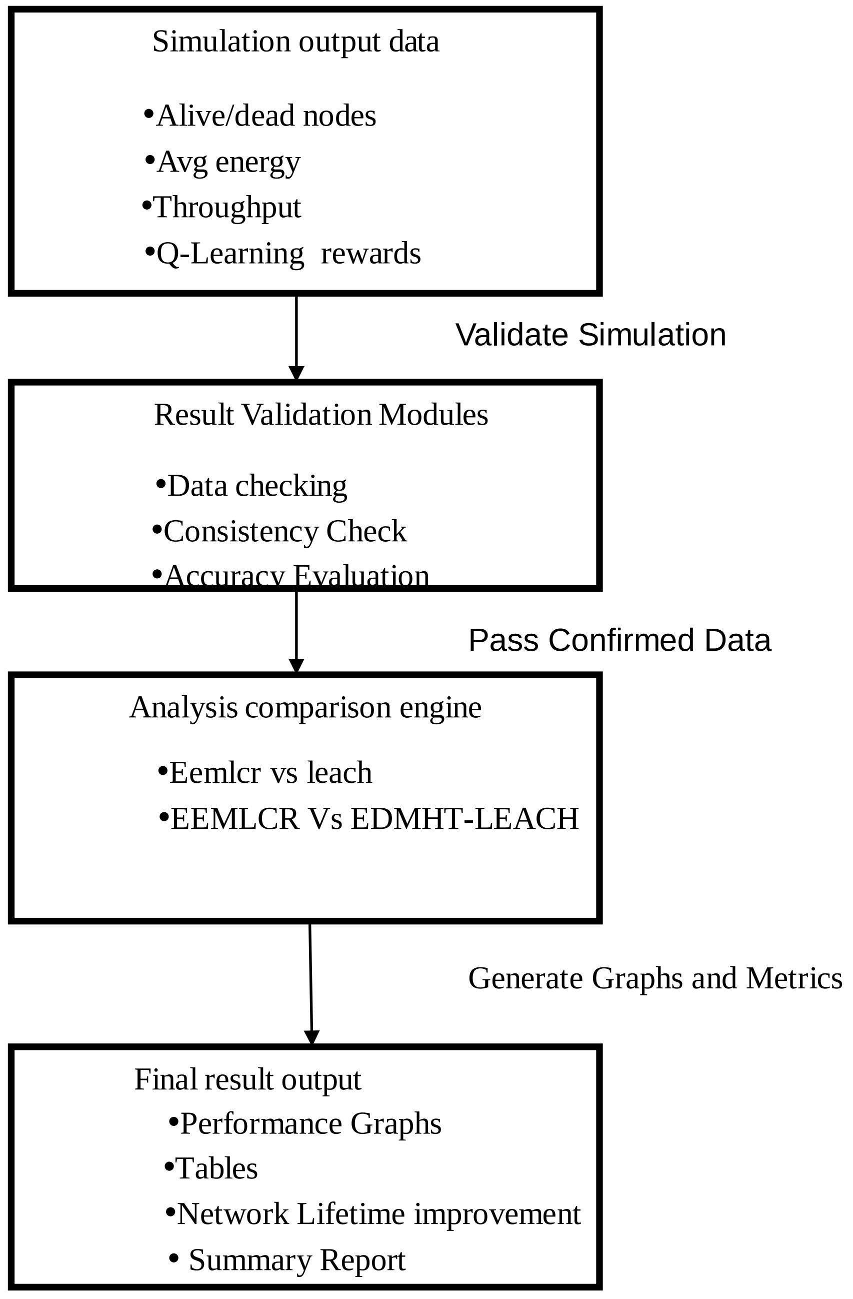

Fig 5.6. RESULT VALIDATION AND ANALYSIS

Simulation Output Data

This stage represents the raw results produced after running the network simulation. The simulation generates several key metrics that describe the system's behavior and performance, including the number of alive and dead nodes, the average residual energy remaining in the network, the throughput achieved, and the Q-learning reward values when reinforcement learning is used. These parameters collectively reflect how energy-efficient, stable, and reliable the simulated protocol is. At this point, the data is not yet verified, but it provides the essential foundation on which further analysis will be built.

Validate Simulation

Before analysis, the simulation results must undergo a validation step to ensure they accurately represent the intended experiment. This stage checks that the simulation ran correctly, that all expected output metrics are present, and that no unexpected behavior occurred due to bugs or incorrect parameters. Validation ensures the trustworthiness of the simulation environment and prevents the propagation of errors into later stages of analysis. Only after the simulation is confirmed to be performed correctly can the results move to deeper verification.

Result Validation Modules

This block performs detailed verification using three internal procedures: data checking, consistency check, and accuracy evaluation. Data checking ensures the presence and completeness of all output values, while consistency checking verifies that the data follows logical and expected trends—such as gradual energy depletion and progressive node failure over time.

Analysis Comparison Engine

After the data is confirmed, it is fed into the analysis comparison engine, where the performance of the proposed protocol is compared against benchmark methods. The flowchart highlights comparisons such as EEMLR versus LEACH and EEMLCR versus EDMHT-LEACH. These comparisons allow researchers to assess improvements in network lifetime, energy efficiency, and throughput under the same simulated conditions. By evaluating the proposed method against both classical and advanced protocols, this stage demonstrates the relative advantages and effectiveness of the new approach

Generate Graphs and Metrics

Once comparative analysis is complete, the system generates visual and numerical outputs to represent performance trends. This includes graphs that illustrate metrics such as node survival, energy usage, and throughput over simulation rounds. It may also produce statistical summaries and computed performance indicators. These visuals and metrics make it easier for readers to interpret results, identify trends, and appreciate the improvements offered by the proposed method. This step prepares the information for reporting and documentation.

Final Result Output

The final stage compiles all the validated data, analysis results, graphs, and metrics into a structured output package. This includes performance graphs, summary tables, documented improvements in network lifetime, and a concise summary report explaining the findings. The finalized materials are then ready for inclusion in the thesis or publication.

CHAPTER 6

SYSTEM DESIGN

6.1 FLOWCHART

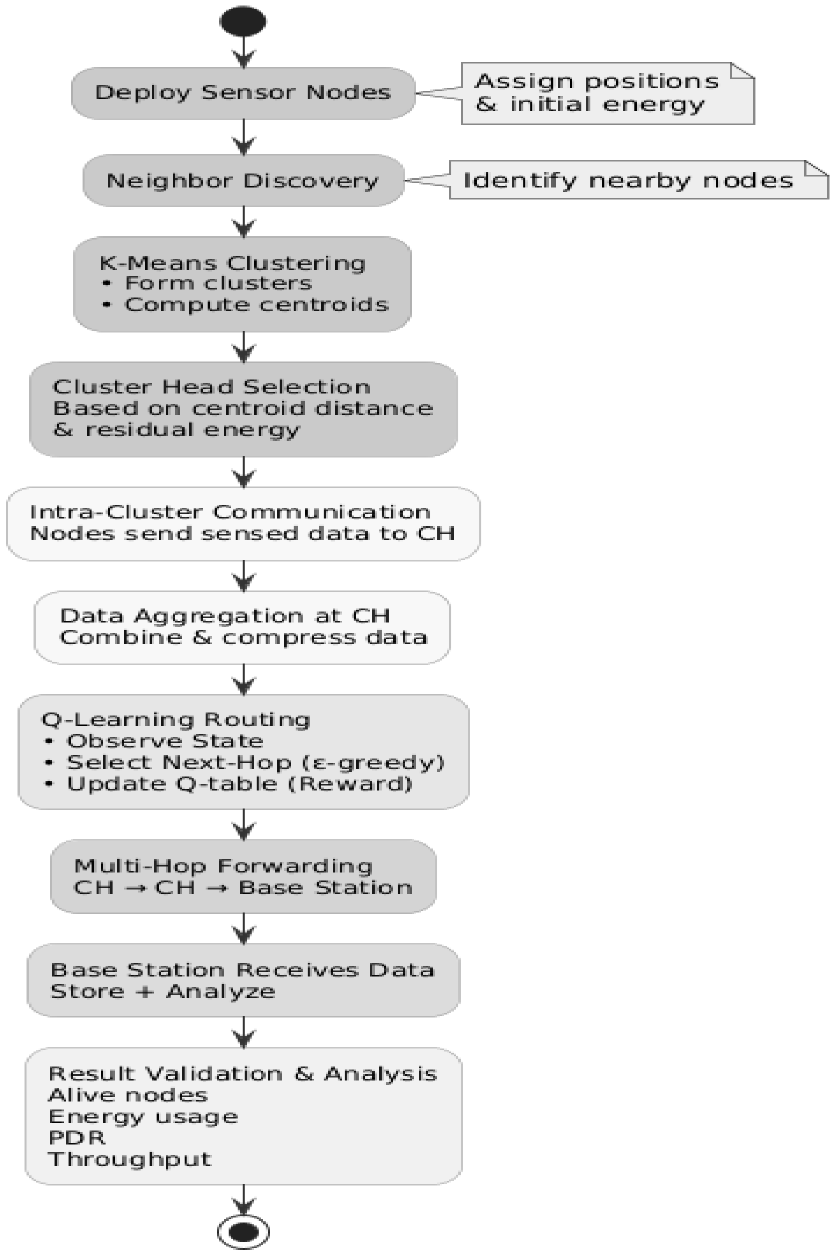
Deploy Sensor Nodes
Assign positions & initial energy
Neighbor Discovery
Identify nearby nodes
K-Means Clustering • Form clusters • Compute centroids
Cluster Head Selection Based on centroid distance & residual energy
Intra-Cluster Communication Nodes send sensed data to CH
Data Aggregation at CH Combine & compress data
Q-Learning Routing • Observe State • Select Next-Hop (ε-greedy) • Update Q-table (Reward)
Multi-Hop Forwarding CH → CH → Base Station
Base Station Receives Data Store + Analyze
Result Validation & Analysis Alive nodes Energy usage PDR Throughput

Fig 6.1. FLOWCHART

The flowchart represents the working process of the proposed system. The process begins with the deployment of sensor nodes in the network area. After deployment, nodes perform

Neighbor discovery, and the K-means algorithm is used to form clusters. Once clusters are created, the node closest to each centroid with sufficient energy is selected as the Cluster Head (CH).

Sensor nodes then sense data and send it to their CH. The CH aggregates the received data and uses a Q-learning algorithm to select the most energy-efficient path toward the base station. Packets are forwarded through multiple CHs until they reach the base station. After each round, node energy is updated, and performance metrics are recorded. This cycle continues until the simulation ends.

6.2 CLASS DIAGRAM

The class diagram shows the main components used in the system. The Node class contains attributes such as ID, position, and energy. The Cluster class manages cluster members and CH selection. The Cluster Head class extends the Node class and contains additional functions for aggregation and Q-learning.

The K-Means Module handles cluster formation, while the Q-Learning Agent chooses the next hop for routing. The Energy Model calculates energy spent during transmission and reception. The Base Station receives packets, and the Logger stores performance metrics. These classes interact together to implement EEMLCR efficiently.

A class diagram is a static structure diagram in the Unified Modeling Language (UML) that describes a system's structure by showing its classes, their attributes, operations, and relationships. It serves as a blueprint for object-oriented software, visualizing the static view of a system and providing a high-level model for design communication and code construction

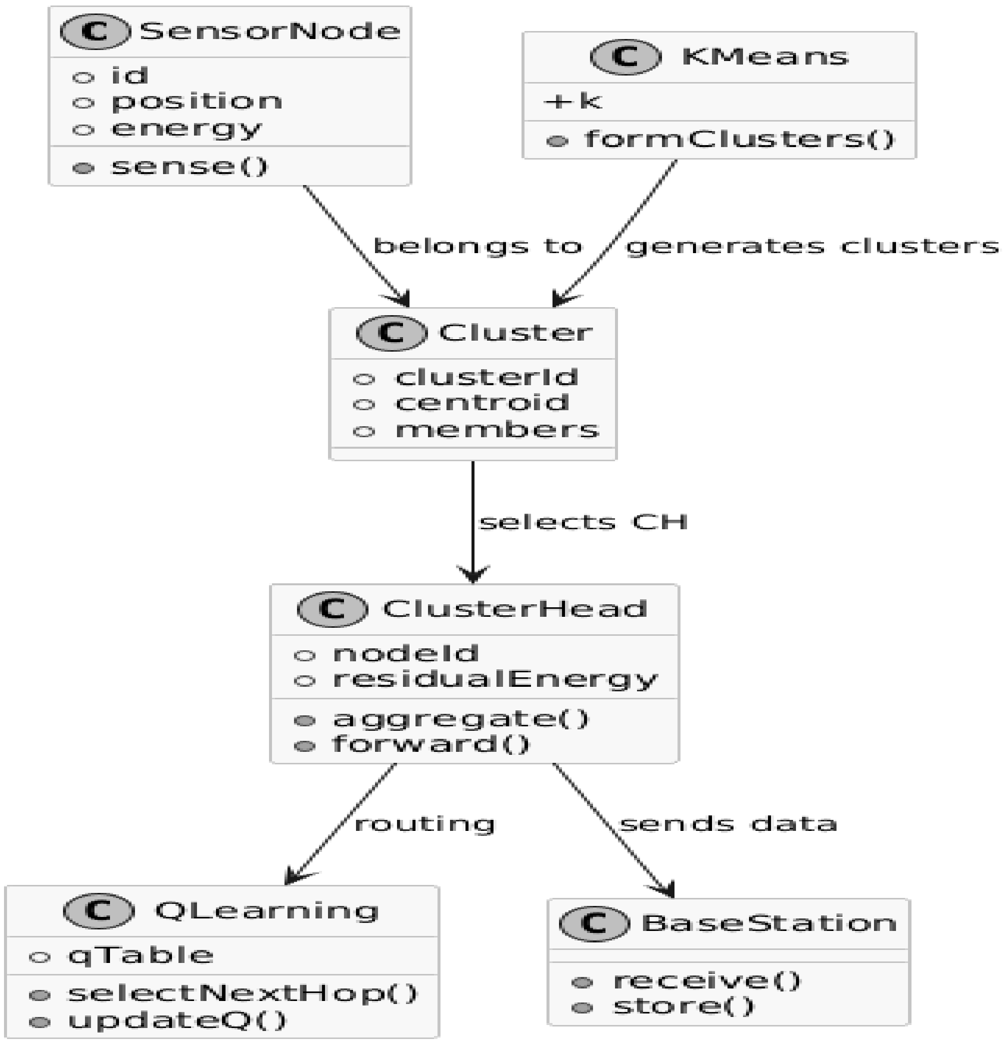

Fig 6.2. CLASS DIAGRAM

6.3 SEQUENCE DIAGRAM

The sequence diagram explains the order of operations in a single simulation round. The simulator calls the K-means module to form clusters. Each cluster elects its CH Sensor nodes sense data and send it to the CH. The CH aggregates the data and requests the Q-learning agent to select the next hop.

A sequence diagram is a type of Unified Modeling Language (UML) diagram that visually represents the interactions between objects or system components in a chronological order to carry out a specific function or scenario. Time progresses from top to bottom, and messages are exchanged horizontally between vertical lifelines.

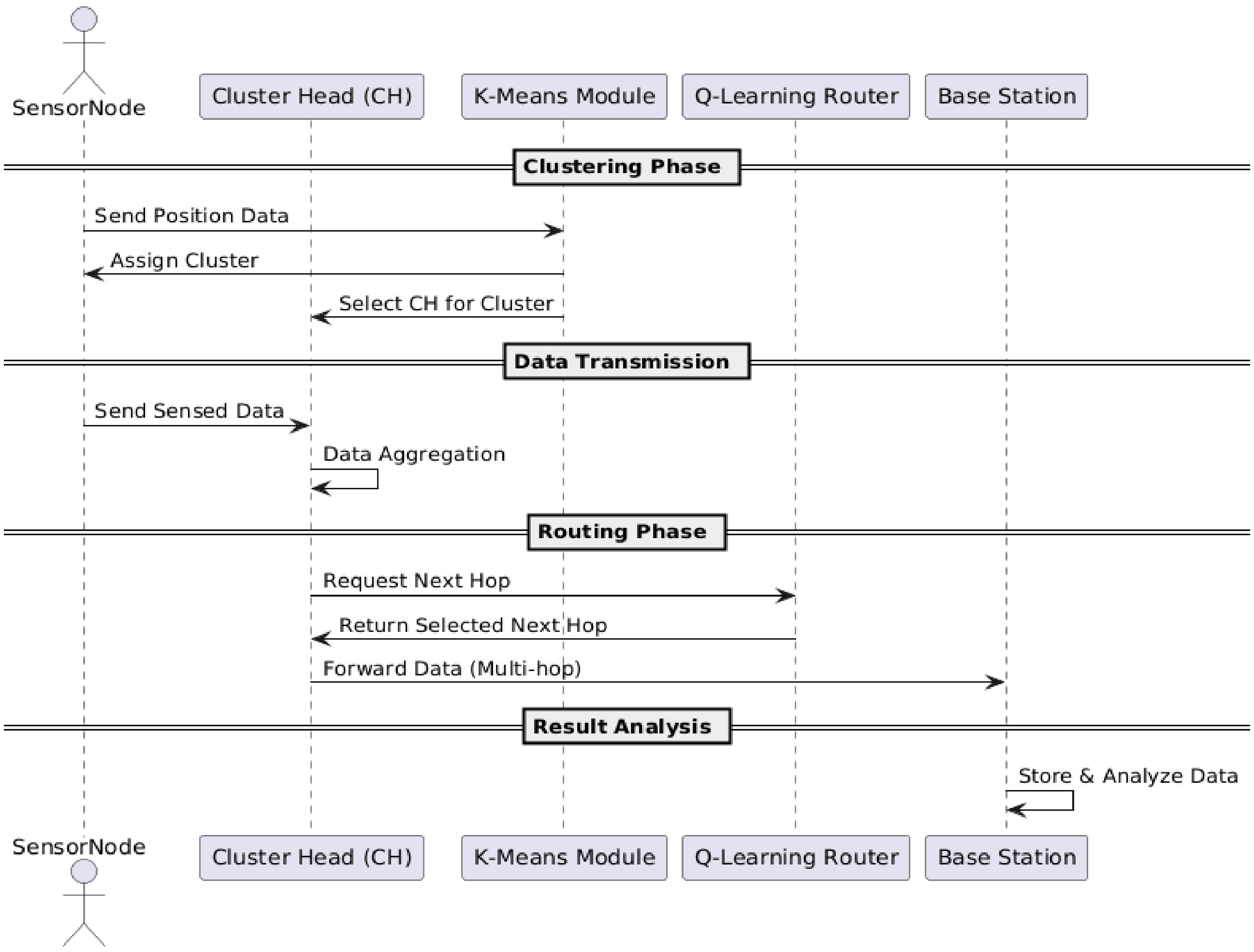

Fig 6.3. SEQUENCE DIAGRAM

CHAPTER 7

CONCLUSION

The proposed Energy-Efficient Machine Learning-Based Clustering and Routing method combines K-means clustering with Q-learning-based routing to effectively address the challenges of energy consumption and network longevity in wireless sensor networks. By integrating intelligent cluster formation with adaptive routing decisions, the method offers a dynamic and scalable solution for improving communication in resource-limited sensor environments. Implemented in MATLAB and evaluated across different network sizes, it demonstrates significant improvements over traditional routing techniques such as LEACH and its multi-hop variants, showing reduced energy usage, a higher number of active sensor nodes, and enhanced overall network stability. The results also indicate that the approach performs even better in larger deployments, extending network lifetime and improving operational efficiency, which confirms the value of applying machine learning techniques to wireless sensor communication. This research shows that learning-based methods can substantially improve sensor network performance by enabling more efficient clustering and smarter routing. The gains achieved in energy conservation and operational durability highlight the method's potential as a reliable and effective solution for modern sensor-based applications. Future work may further enhance this approach by exploring additional learning models, incorporating node mobility, or integrating real-time environmental factors to achieve even greater optimization.

CHAPTER 8

REFERENCES

[1]Wan-Kyu Yun and Sang-Jo Yoo, ‘‘Q-Learning-Based Data-Aggregation-Aware Energy-Efficient Routing Protocol for Wireless Sensor Networks”, IEEE International conference,2021.

[2]Muhammad Akram, Sibghat Ullah Bazai, Muhammad Imran, “EEMLCR: Energy-Efficient Machine Learning-Based Clustering and Routing for Wireless Sensor Networks”, IEEE International conference,2025.

[3]Salim El Khediri, Awatef Benfradj, “Integration of artificial intelligence (AI) with sensor networks: Trends, challenges, and future directions”, Journal of King Saud University - Computer and Information Sciences,2024.

[4]Huda Salih Abd, Asaad S Daghul, “An Energy-Efficient Scheduling and Routing Protocol Based on Q-Learning for WSN”, Asian Journal of Computing and Engineering Technology,2025.

[5]Yan Wang, Ke Deng, Yongli Ren, Jeffrey Chan, “Reinforcement Learning for Tackling Energy-saving and Energy-balance Dilemma of Cluster-based Routing Protocols in WSNs”, IEEE International conference,2024.

[6]F. A. Khan, M. Khan, M. Asif, A. Khalid, and I. U. Haq, ‘‘Hybrid and multi-hop advanced zonal-stable election protocol for wireless sensor networks,’’ IEEE Access,2019.

[7]X. Li, J. Peng, J. Niu, F. Wu, J. Liao, and K. R. Choo, ‘‘A robust and energy efficient authentication protocol for industrial Internet of Things’’, Research Gate,2018.

[8]T.Hemalatha, M.V.Ramesh, and V.P.Rangan,‘‘Effective and accelerated forewarning of landslides using wireless sensor networks and machine learning,’’ IEEE Access,2025.

[9]S. M. Zahid, M. A. Jan, F. Khan, and M. Alam, ‘‘Energy-efficient cluster-based routing using machine learning techniques for wireless sensor networks,’’ IEEE Access, 2021.

[10]Y. Wang, K. Deng, Y. Ren, G. Tian, and C. S. Jensen, ‘‘Reinforcement learning for improving energy balance in cluster-based routing protocols of wireless sensor networks,’’ IEEE Access,2023.