Dissertation on

"Secure and Efficient Routing Mechanism for Healthcare Networks"

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

Bachelor of Technology
in
Computer Science & Engineering

UE19CS390B – Capstone Project Phase - 2

Submitted by:

Balaji BV	**PES2UG19CS079**
Bhoomika P Bhavimath	**PES2UG19CS091**
Durgalakshmi.V	**PES2UG19CS120**
Rahul B	**PES2UG19CS313**

Under the guidance of

Prof. Animesh Giri
Designation
PES University

June - Nov 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Secure and Efficient Routing Mechanism for Healthcare Networks'

is a bonafide work carried out by

Balaji BV	**PES2UG19CS079**
Bhoomika P Bhavimath	**PES2UG19CS091**
Durgalakshmi.V	**PES2UG19CS120**
Rahul B	**PES2UG19CS313**

In partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE19CS390B) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June 2022 – Nov. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature	Signature	Signature
Prof. Animesh Giri	Dr.Sandesh B J	Dr. B K Keshavan
Asst. Professor	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. ____________________________	____________________________
2. ____________________________	____________________________

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"Secure and Efficient Routing Mechanism for Healthcare Networks"** has been carried out by us under the guidance of Prof.Animesh Giri, Asst. Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June – Nov. 2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES2UG19CS079	**Balaji BV**	***Balaji BV***
PES2UG19CS091	**Bhoomika P Bhavimath**	***Bhoomika P Bhavimath***
PES2UG19CS120	**Durgalakshmi.V**	***Durgalakshmi.V***
PES2UG19CS313	**Rahul B**	***Rahul B***

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Animesh Giri, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE19CS390B -Capstone Project Phase –2.

I am grateful to the Capstone Project Coordinator, Dr. Sarasvathi V, Professor and Dr. Sudeepa Roy Dey, Associate Professor, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. SandeshB J, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr.Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The demand for healthcare services is increasing with the growing population. To cater to this demand, hospitals are updating their facilities and treatment approaches. Integration of Internet of Things (IoT) in the healthcare industry is a key move in this direction. Real-time patient tracking can now be realized with sensory data enabling improved diagnosis and treatment process. Routing protocols form a key aspect of a network. The three principal elements of routing protocols deployed in a healthcare sector include compliance with the constrained nature of devices, the ability to handle heterogeneous traffic, and ensuring the security of data. In this study, the "*IPv6 Routing Protocol for Low Power and Lossy Networks*" (RPL) is being explored in detail. RPL groups connected nodes with a common objective into an instance. This paper compares the usage of Multiple RPL Instances and Multi Sink (MI+MS) with Single RPL Instance and Single Sink (SI+SS) against latency, control traffic overhead, energy consumption and Packet Delivery Ratio (PDR) in Cooja Simulator. The results obtained illustrate that the MI+MS approach performs better in all the metrics. Thus, the use of both multiple instances and multiple sinks helps to effectively manage heterogeneous traffic. Further, cryptographic mechanisms are added to ensure data confidentiality. Data from sinks are transferred to the internet via COAP (Constrained Application Protocol) server and border router to the backend server hosting a website.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Internet of Things

In this era of digital transformation, the "*Internet of Things*" (IoT) is one of the main driver technologies. This computing paradigm helps to connect our everyday objects or "things" to the internet. It is formally defined as "*a network of things, with clear element identification, embedded with software intelligence, sensors, and ubiquitous connectivity to the Internet*" [1].

These IoT devices agree with the IEEE 802.15.4 standard making them constrained by nature; i.e they have limited power, energy, low bit rate, memory, storage, and processing power. Such wireless nodes form a network termed "*Low-Power Wireless Personal Area Networks*" (LoWPANs). Thus, the conventional Internet Protocol (IP) cannot be implemented directly on these devices. The "*Internet Engineering Task Force*" (IETF) thus developed a solution to make Internet Protocol Version 6 (IPv6) to communicate with LoWPANs with strategies like compression of IPv6 headers and addition of an adaptation level/layer between IP and Data Link layer [6] [7]. Figure 1.1 displays the protocol stack of IoT devices.

"Low-Power and Lossy Networks" (LLNs) comprise resource-constrained nodes attached through links that are highly unstable and prone to packet losses. One of the major considerations for successful implementation in such scenarios becomes building a suitable routing protocol. The protocol must deal with the less energy and processing power of nodes along with providing reliability given the lossy nature of the network [2].

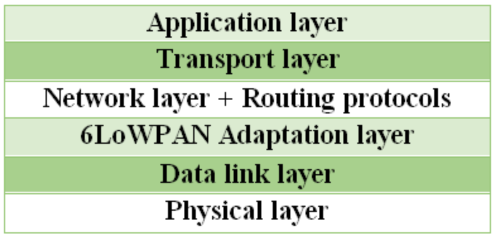


Figure 1.1: IoT protocol stack

1.2 RPL

The IPv6 Routing Protocol for LLNs (RPL) is a routing protocol explicitly designed for LLNs by the ROLL Working Group in the IETF. It is a proactive; "*distance-vector*" protocol that constructs a graph termed "*Destination Oriented Directed Acyclic Graph*" (DODAG) [8]. It is a graph with no cycles and all directed to one node termed as DODAG root or sink node (figure 1.2).

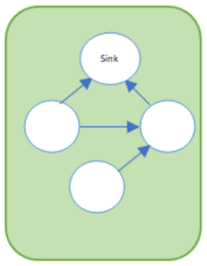


Figure 1.2: Example of a DODAG

This protocol splits the given network into one or more instances. Each instance consists of at least one tree-like structure termed DODAG. A DODAG is uniquely recognized using DODAG ID and RPL Instance ID. Every RPL instance is linked to a single Objective Function (OF) that helps in the selection and optimization of paths within it [9] [13].

The OF helps a node to take primary decisions such as

- Creating a potential parent list
- Choosing a parent over the others with an additional metric
- Choose a DODAG to join
- Rank calculation to determine its distance from the root node [30].

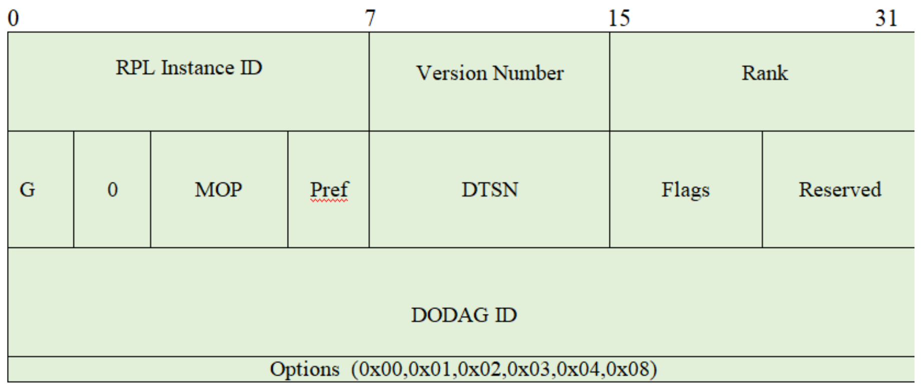
1.2.1 Control messages

This section provides an overview of the different control messages transmitted between RPL nodes to construct and maintain the routing paths.

1.2.1.1 DODAG Information Object message (DIO)

This message is initially sent by the root to its neighbor nodes to form a new DODAG. Further, every intermediate node transmits this message to its neighboring child nodes to complete the DODAG structure. Figure 1.3 illustrates a DIO packet.

This packet allows a node to learn about its parents, their rank values, and form and maintain DODAGs. It is mainly used to find and maintain upward paths [24].



0				7	15	31
RPL Instance ID				Version Number	Rank	
G	0	MOP	Pref	DTSN	Flags	Reserved
DODAG ID						
Options (0x00,0x01,0x02,0x03,0x04,0x08)						

G: Grounded flag
MOP: Mode of Operation
Pref: DODAG Preference
DTSN: Destination Advertisement Trigger Sequence Number

Figure 1.3: DIO message format

1.2.1.2 DODAG Information Solicitation message (DIS)

This message is optionally sent by nodes to join a DODAG / request for a DIO message from its neighboring nodes. Figure 1.4 shows the traversal of DIO and DIS messages [25].

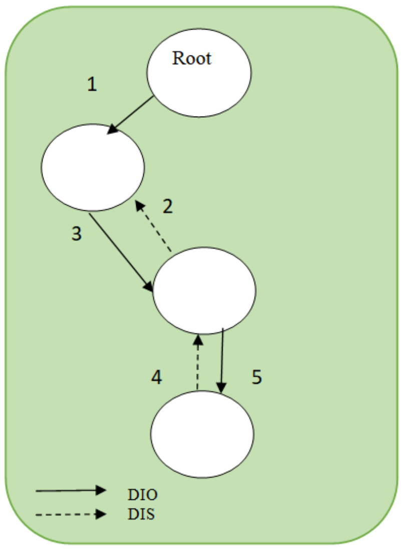


Figure 1.4.: Traversal of DIO and DIS

1.2.1.3 DODAG Advertisement Object message (DAO)

This packet is mainly used to find and maintain downward paths. It is sent by all intermediate and child nodes. It allows parent nodes to learn about addresses and prefixes of its child nodes. Figure 1.5 displays a DAO packet [25].

1.2.1.4 DODAG Advertisement Object Acknowledgement message (DAO-ACK)

It is an optional acknowledgment message. It is unicasted by the receiver of a DAO message to the DAO sender. Figure 1.6 displays a DAO-ACK packet [9].

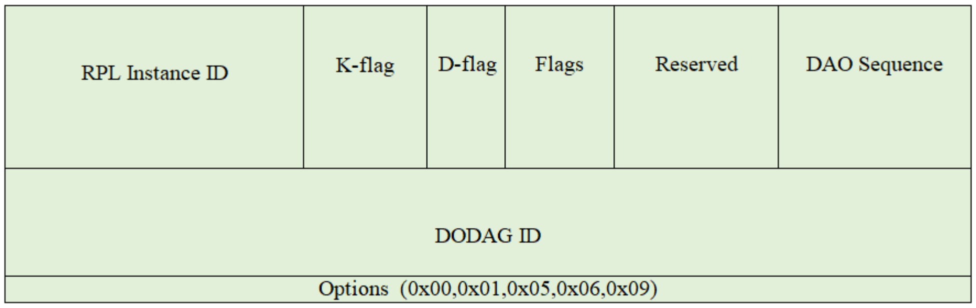


Figure 1.5: DAO message format

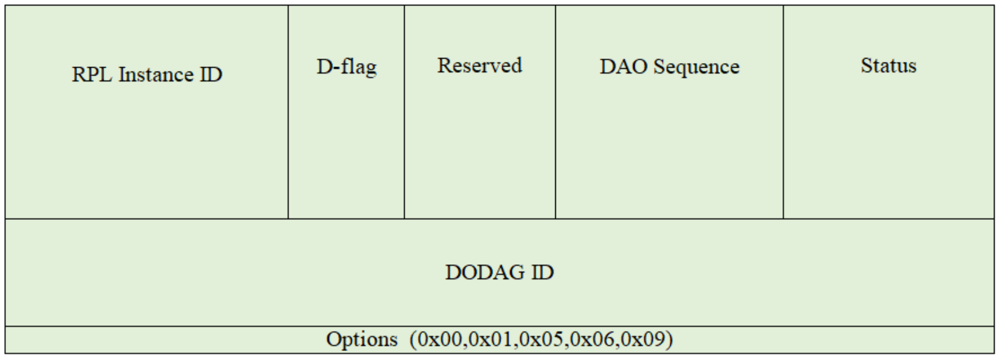


Figure 1.6: DAO-ACK message format

1.2.2 Objective function 0 (OF0)

It is one of the standardized and basic OF. The aim is to help a node join a DODAG that provides adequate connectivity to a specific group of nodes. The metric used is Hop Count (HC). Based on the hop-count value to the root, every node computes its rank. Initially, the root has the least rank value. It broadcasts its rank along with other information using the DIO message. All nodes in the transmission range of the root receive this message, consider the root as its parent node, compute the rank (increments by one), and further broadcast DIO messages with its rank information. This process is continued till all child nodes join the network. In a case where a node receives two DIO messages, the one with the lesser rank value is chosen as a parent. Since, hop-count forms the heart of the process, this OF is also termed as minimum hop-count OF [14].

1.2.3 Minimum Rank with Hysteresis Objective Function (MRHOF)

This OF finds the ideal path using path costs. This involves two techniques '*minimum- rank and hysteresis*'. First, it determines the path with the minimum rank (considered as the minimum path cost). Second, a new path is selected if this path cost lesser than the given minimum one by a defined threshold. Given that routing goal is to reduce a specific routing metric, MRHOF can be used with an additional metric. In the present implementation, this metric is ETX (Expected Transmission Count). An ETX value shows the number of retransmissions needed for the successful transmission of a packet to the next designated node (along with an acknowledgment). Thus, it allows taking into account the quality of communication links for path selection [16].

1.3 Healthcare and IoT

Most healthcare organizations are leveraging the IoT technology to transform into 'Smart Healthcare' [5]. Earlier patient-doctor interactions were restricted to text, telephonic communications, or physical meetings. Today, the need for personalized/customized care and treatment has risen among patients. A holistic approach is expected where the well-being of patients is considered the main priority [3]. Remote real-time monitoring of patients could be a key aspect to realize this expectation.

Use of various IoT devices such as Bluetooth enabled coagulation tester (figure 1.7) and wall-mount temperature sensor (figure 1.8) help to collect and analyze, transfer data about patients and hospital environment efficiently to a central database The immediate effects include better decision making, avoidance of disasters, remote ubiquitous access to information through the internet [4].

Figure 1.7: Bluetooth-enabled coagulation tester

Figure 1.8: Wall-mount temperature sensor

1.4 Heterogeneous Traffic in Healthcare Networks

Sensors in healthcare could be broken down into two classes:

• Sensors part of patient body monitoring vital signs (forms Body Area Network -BAN), could be static or dynamic

• Static sensors monitoring environmental conditions of the hospital (placed on walls).

More often BAN sensors represent critical data with higher sending intervals. It is thus observed that different sensors within the healthcare environment generate different types of traffic that need to be handled in a varied manner. In summary, IoT devices in the healthcare environment generate heterogeneous traffic.

1.5 Heterogeneous traffic and RPL

In a heterogeneous traffic environment, each type of traffic requires different QoS (Quality of Service) parameters. For instance, in a healthcare application, periodic data sent from labs (monitoring environment of labs) would need a guaranteed delivery while critical data sent from Intense Care Unit (ICU) would require less delay [17].

An RPL instance is a collection of one or more DODAGs all operating with the same OF. Thus, all nodes in an instance is said to have one objective (in terms of how to optimize their routes). A heterogeneous traffic context would need multiple routing objectives. Thus, forming multiple instances is a way to handle this requirement in RPL [12] [36].

1.7 Multiple Sinks

Most networks consist of multiple sender nodes connected to a single sink node. The sender nodes farther from the sink (not in the transmission range) transmit their information to the nodes closer to the sink thereby following a multi-hop approach. Thus, the nodes near the sink tend to become "*hotspots*" that lead to drastic energy consumption [33].

Further in most practical scenarios, the reception ability of sinks would be less than 100%. Thus, with a single sink approach, the packet loss ratio tends to rise.

To tackle both these issues, a multi-sink approach was proposed. This approach improves the overall performance of the network and helps to reduce the energy consumption of nodes [31] [34] [35]. Figure 1.9 displays the single and multi-sink approaches.

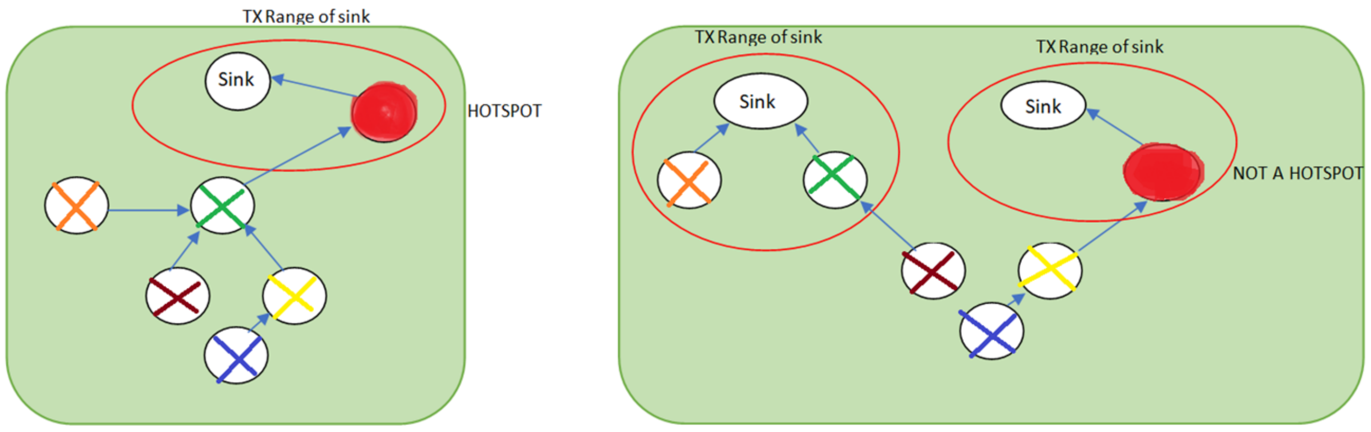


Figure 1.9: Single and multi-sink

CHAPTER 2

PROBLEM STATEMENT

There has been a sharp rise in IoT initiatives in the field of healthcare. Some of the use cases could include tracking of staff and patients, and automating patient care workflow. It is observed that the traffic generated by sensors deployed in this environment is heterogeneous in nature.

The RPL protocol has evolved to become one of the standard protocols to be used in IoT. However, the current implementation does have its prime focus on homogenous traffic. RPL creates an instance consisting of many DODAGs all with the same objective function (e.g hop count or ETX (expected transmission count)). Thus, a single instance can route homogenous traffic effectively.

Our problem statement thus shall focus on building an efficient routing mechanism for healthcare sector with heterogeneous traffic. To facilitate this, RPL is being modified to support the formation of multiple instances within a single network. Further, given that sensors deployed in a healthcare environment generate sensitive information related to patients and hospital management, it is integral to also identify and analyze threats on such networks and routing protocols.

CHAPTER 3

LITERATURE SURVEY

3.1 '*Using Multiple RPL Instances to Enhance the Performance of New 6G and Internet of Everything (6G/IoE)-Based Healthcare Monitoring Systems-2020*' [12]

3.1.1 Introduction

This paper demonstrates the performance of multiple instances in a 6G/ IoE-based sample health monitoring system. The system is assumed to be employed at King Abdullah University Hospital (KAUH), Jordan. Sensors were deployed at every floor of the hospital, each segregated into departments such as labs and Intensive Care Unit (ICU). Thus, each floor would generate different traffic types, namely high-critical, medium-critical and periodic.

3.1.2 Implementation

Sending intervals and packet sizes were different for each type. Requirements included minimal delay and high reliability for critical data (data from ICU), considerable delay for medium-critical data and least priority in terms of delay and reliability for periodic data (data from general wards). Three floors (1,9,11) were simulated in Cooja simulator. Average PDR and average latency was tested for three reception ratios (100,85,70%).

3.1.3 Results

PDR values have improved significantly in two out of three floors. Only in one floor, single instance RPL outperformed multiple instance (reference values for PDR are 100% and 85% respectively). Average latency is significantly decreased in the proposed approach thereby making this technique suitable to transmit/carry highly critical data.

3.1.4 Limitations

Other metrics such as convergence time, energy consumption were not compared. The number of RPL instances was static and predefined.

3.2 '*On providing differentiated service exploiting multi-instance RPL for industrial low-power and lossy networks*' [26]

3.2.1 Introduction

The authors considered an LLN of 60 nodes deployed to monitor an industrial setup. Further, four different traffic types varied based on reliability and latency requirements were chosen.

3.2.2 Proposed approach

The mechanism used to handle the different QoS requirements is to have four different RPL instances. This proposed solution termed '*MI-RPL*' (Multiple-Instance RPL solution for industrial LLN) uses a composite routing parameter (formed by a combination of many metrics) for every instance and defines a new OF that determines routing paths based on instance requirements.

3.2.3 Implementation details and Results

In this paper, MI-RPL was compared against a single instance approach (MRHOF with ETX as the only routing metric) under PDR, average energy consumption, and average latency. The experiments were conducted at 20-100 % RX ratios and 5-30 ppm loads in the Cooja simulator.

It was observed that latency, PDR, and energy consumption were improved at all RX ratios and traffic loads in the proposed approach.

3.3 ‘*Using multiple RPL instances for enhancing the performance of IoT-based systems*’ [10]

3.3.1 Objective and Implementation

The authors in this paper considered an IoT system deployed in a floor (of a hospital) with four wards with three kinds of traffic, namely critical, medium-critical and periodic. Sending intervals and packet sizes were modified for each. Requirements included minimal delay and high reliability for critical data, considerable delay for medium-critical data and least priority in terms of delay and reliability for periodic data. The authors evaluated the performance of both single and multiple instance RPL (three instances in total) with respect to average PDR and average latency with three reception ratios (100,85,70%). The experiment was conducted using Cooja simulator.

3.3.2 Results

The results obtained illustrated that multiple RPL instances outperformed at all reception ratios. Considering RX ratio 100%, an overall latency of 21.2 seconds was obtained with single instance. In contrast, with multiple instances, an average latency of 0.04 seconds was observed. Similarly, average PDR was 28% with single instance and 93% with multiple instance RPL. However, only two metrics were evaluated against single instance RPL.

3.4 *'Performance evaluation of RPL objective functions for multi-sink'*[31]

3.4.1 Objective

The authors in this paper have evaluated the performance of multi-sink in a network topology of randomly spaced 35 sender nodes. Cooja Simulator was used to contrast the performance of 1 sink, 5 sinks, and 10 sinks against PDR, energy consumption, number of lost packets, and throughput. Additionally, the results were compared with all nodes running with OF0 and MRHOF separately.

3.4.2 Results and Discussion

The results obtained proved that the multi-sink approach improved the performance in all four aspects. Here every node tries to form a DODAG with the nearest sink thereby decreasing the number of packets lost, energy consumption, and increasing PDR and throughput. Further, MRHOF (ETX) outperformed OF0 (HC) in all cases.

3.5 '*Performance evaluation of multiple RPL routing tree instances for Internet of Things applications'* [11]

3.5.1 Introduction

The authors built a network of 60 nodes generating critical and regular/periodic type data traffic. Main idea was to contrast the performance of single and multiple instance RPL.

3.5.2 Implementation details

Sending intervals and packet sizes were modified for each traffic type. They evaluated the performance of both single and multiple instance RPL (two instances in total) with respect to average PDR, average convergence time and average latency with three reception ratios (100,85,70%). The simulations were conducted in Cooja simulator.

3.5.3 Results

It was observed that average routing tree convergence time was a little higher in multiple instance by a margin of 4-5 seconds at all reception ratios. This is because every node must form two DAGs with the sink node as opposed to one in single instance. Additionally, both types of data traffic had improved performance in latency and PDR at all RX ratios.

3.5.4 Limitations

Parameters like energy consumption and control traffic overhead were not explored in this paper.

3.6 ‘*Quality of service differentiation for smart grid neighbor area networks through multiple RPL instances*’ [29]

3.6.1 Objective

The authors have examined three variations of RPL for a Smart Grid application namely ‘*standard RPL or single instance RPL with ETX*’, ‘*multiple instance RPL or RPL*-M’, and ‘*multiple instance RPL with prioritized channel access termed RPL-M+*’. RPL-M+ has an add-on feature of prioritization of traffic against RPL-M. The OFs used in multiple instance RPL are OF0 (HC) and MRHOF (ETX).

3.6.2 Implementation details

The simulation was performed in OMNET++ (Objective Modular Network Testbed in C++) with two types of traffic critical and periodic each having varying packet sizes, sending intervals, reliability, priority, and latency requirements.

3.6.3 Results and Discussion

The results obtained showed that RPL-M performed the best in terms of latency and PDR. The standard single instance version could not handle the varied traffic classes while RPL-M+ had modified the back-off access methods thereby having early channel access and creating large delays.

CHAPTER 4

PROJECT REQUIREMENTS SPECIFICATION

4.1 Product Perspective

Our project aims to develop an IoT architecture system in the healthcare industry that can handle heterogeneous traffic generated by the numerous IoT sensor nodes. Heterogeneous traffic is commonly observed across various smart systems. Thus, there is a need to identify and establish the best routing mechanism that satisfies the nature of the system.

4.1.1 Product Features

- An efficient and secure routing protocol for IoT sensor nodes.
- Achieve better patient care, experience, and overall efficiency in healthcare.
- Ensure reliable and secure data transfer.
- Heterogeneous traffic to be handled in a single network system.

4.1.2. Operating Environment

To simulate the network:

Contiki OS:

It is an open-source and lightweight Operating System (OS), specifically built for constrained nodes. It is developed using the C language and is highly portable. Further, the uIP (micro IP) library facilitates the functionality of a complete TCP/IP stack. The advantages include:

- Protothreads: it is a programming abstraction implemented in C that is stackless and has less memory overhead. All processes are protothreads.
- Saves power and time for modification by allowing to unload and load specific applications at runtime
- Commonly used to analyze RPL protocol
- Support for TCP/IP stack makes the constrained networks communicate with conventional networks [18] [19]

The above reasons make this OS best-suited for our architecture.

Version – Latest (Contiki-NG)
Operating System – Linux
Source – https://github.com/contiki-ng/contiki-ng

COOJA Network Simulator:

It is a powerful cross-level network simulator in Contiki OS developed using Java language. It supports different kinds of motes/hardware devices such as Tmote Sky, Z1, and MicaZ all developed in C. The advantages include:

- Enables integration with external tools
- Both detailed and high-level emulation of motes is supported
- Ability to calculate various network metrics like latency and average power consumption.
- Support for RPL [18] [20]

Hence, our proposed approach shall be tested using the Cooja simulator.

To capture the network traffic:

<u>Wireshark</u>:

It is an open-source tool used to analyze data packets/ packets transmitted via any network. It is essentially a sniffer tool. Wireshark tool will be used to analyze pcap files in this project.

Version – Stable release 3.4.3

Operating System – Linux

Source - https://www.wireshark.org/download.html

4.1.3. General Constraints, Assumptions and Dependencies

<u>Dependencies:</u> Cooja simulator, Contiki OS Mote's simulation code

<u>Assumptions:</u> The simulations will have to run in the Cooja simulator, in Contiki OS. Wireshark and python will be used for analyzing the created simulations.

<u>Constraints:</u> All the codes for the simulations of nodes have been coded in C.

4.1.4. Risks

- Cooja simulations will start lagging once the number of motes in the network reaches more than 70- 80.
- IoT Dataset collection/preparation might be hard.
- The Cooja Collect View might not be functionally accurate every time.

4.2 Functional Requirements

Initially, set up an IoT network with different kinds of sensors (used in the healthcare environment) in Cooja simulator and configure different parameters of the network like transmission range, interference range along with traffic pattern of each sensor like sending intervals, ppm, etc.

- The main function is to route the traffic effectively.
- Test against various routing metrics like PDR, latency, convergence time, and reliability
- Enhance the features and continue testing, till the threshold set for these metrics is not reached. Further, identify various security threats and implement best-suited algorithms to mitigate threats.
- The routing mechanism must be resilient against various attacks considered and must ensure all security principles (CIA) are achieved.

4.3 Non-Functional Requirements

4.3.1. Performance Requirements

The routing protocol proposed must be efficient in terms of speed and power. The routing decisions must be taken quickly with minimum delay. This requirement aids the system to be handle critical traffic (in case of a medical emergency). Further, catering to the limited resources in the IoT devices, the power consumption must be less.

4.3.2. Security Requirements

Patient data must not get tampered with while in transit, hence suitable mechanisms like encryption must be applied. Attacks on routing protocols, like unauthorized modification of routing tables must be addressed. Access to patient data stored in the backend must be done through mechanisms like authentication. Further, attacks on the various layers of the uIP protocol stack must be mitigated.

4.3.3 Other Requirements:

Scalability is a vital factor keeping in mind the exponential growth of IoT networks. Routing protocols must be able to efficiently route traffic as the number of sensors scale up. The routing mechanism proposed must also satisfy Maintainability. It must be open and adaptable to change as the requirements of the systems and nature of traffic varies. Reusability of the proposed routing mechanism could be considered when the nature of the system is similar to the healthcare environment.

CHAPTER 5

SYSTEM DESIGN

5.1 Design Considerations

5.1.1 Design goals

The design of routing protocol for communication in smart health care must be:

- Highly efficient in terms of performance and energy consumption
- It should support low power sensors used in health care devices.
- Reliable and reduce packet loss during transmission.
- Maintainable, should not go down easily or unbreakable.
- Portable from one device architecture (hardware and OS) to other.
- Secure to attacks and prevent leakage of sensitive data of patients

A detailed description of each goal is provided in section 5.4

5.2 High-level System design

Figure 5.1 displays the architecture of our project.

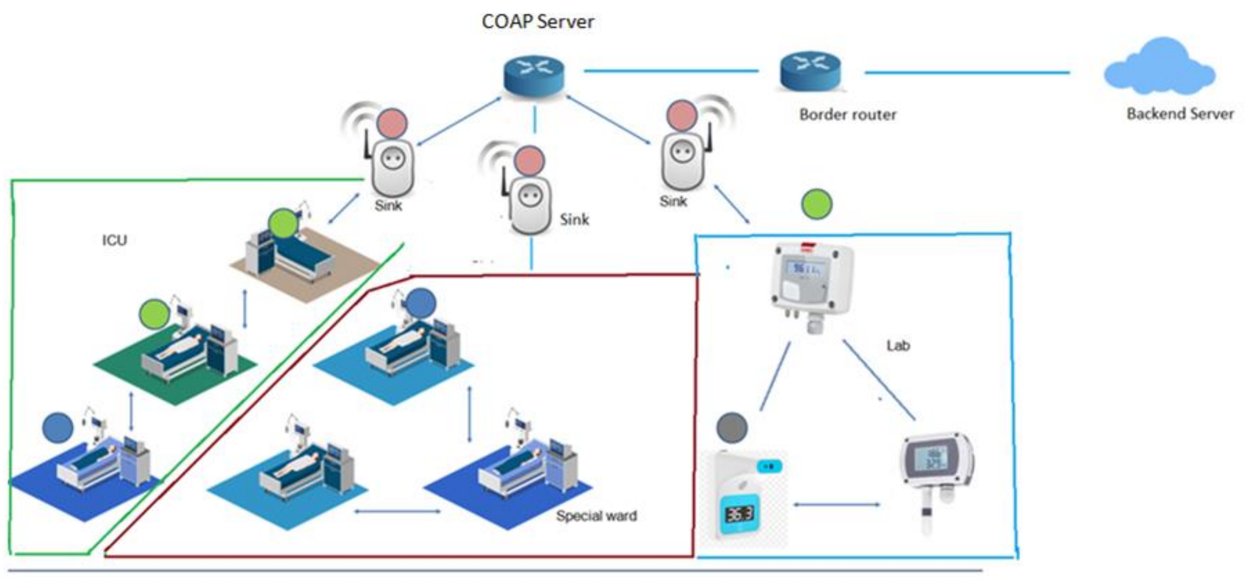


Figure 5.1: High-level design

In this paper, we have assumed an IoT system to be deployed on a hospital floor with three departments, ICU, a Special ward, and a laboratory. Sensors are placed on patients in Special wards and ICUs to closely track their health statistics while sensors are placed on walls of laboratories to get environmental statistics. Information from the ICU and Special ward sections are regarded as critical traffic and medium-critical traffic respectively. Data from labs are classified as periodic traffic (to be sent after a fixed period) considering it measures values such as temperature and humidity of the room.

The chosen network topology is mesh. The layers in this architecture are:

Layer 0: Sensors deployed at different hospital departments.

Layer 1: The information from these sensors is given to other wireless sensor nodes

Layer 2: Sensory information from the ICU is collected by a single sink node

Layer 3: Sensory information from the Special ward is collected by a dedicated sink node

Layer 4: Sensory information from labs is collected by a single sink node

Layer 5: Sinks send information to the COAP (Constrained Application Protocol) server

Layer 6: The COAP server relays information to the backend server via internet with the help of a border router and a proxy server.

5.3 Design Description

5.3.1 Use case diagram

It is a model to highlight the interactions between the users of the system and the functionalities provided by the system.

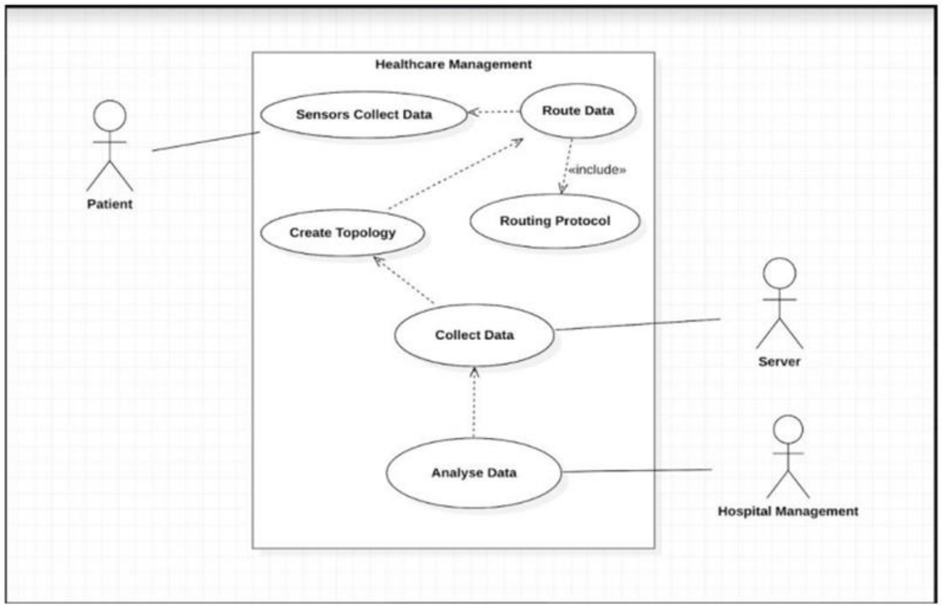


Figure 5.3: Use case diagram

The patients have sensors deployed on them (measuring their vital signs) that produce information (use case: Sensors Collect Data). The generated information shall be routed to the sink (use case: Route data). A routing protocol is needed for routing information (<include> tag used). The routing protocols form various structures to connect nodes for routing traffic (use case: Create topology). The data is further collected in the COAP server (use case: Collect Data). It is then used by the backend server. The analysis is performed and useful insights are consumed by hospital management (use case: Analyze Data).

5.3.2 Data flow diagram

It is a visual depiction of the flow of data within a system. It elaborates on the inputs and outputs of every unit in the system.

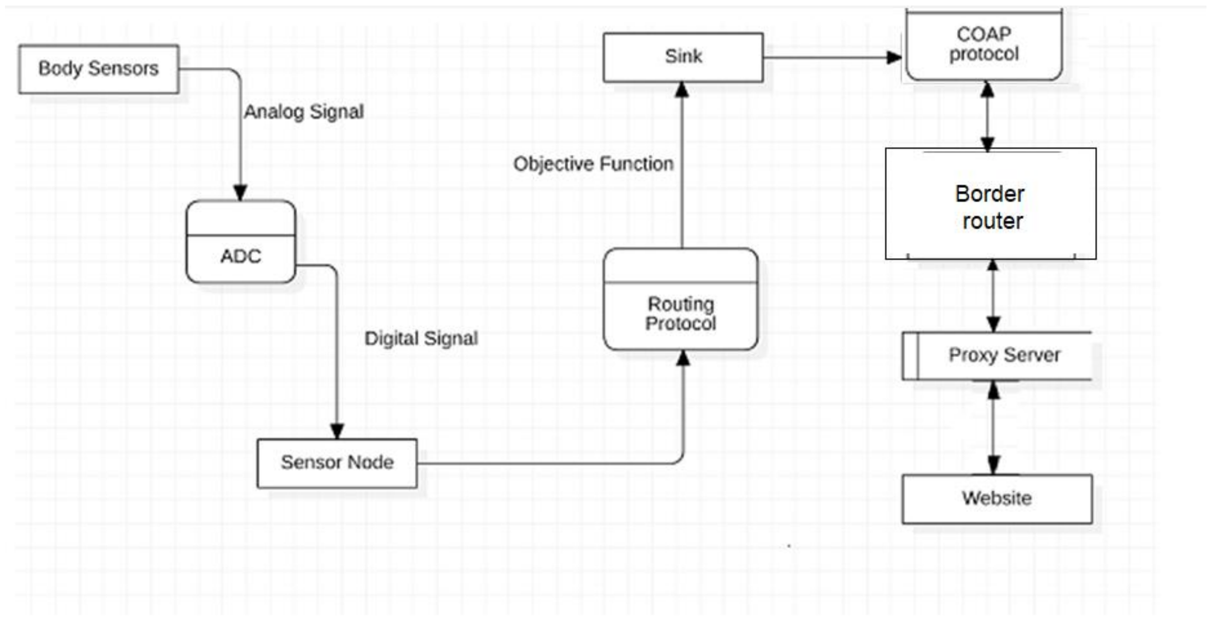


Figure 5.4: Data flow diagram

The body sensors produce an analog signal which is given to an analog signal convertor. The produced analog signal is transmitted to multiple sensor nodes as per routes created by the routing protocol. These routes are formed with the help of an Objective Function (OF). The generated information then reaches the sink node which is then transmitted to the COAP server. The COAP server gives the information to the proxy server where it gets stored via the border router. This information gets exchanged with backend server via the SSE (Server-Sent-Events) service.

5.4 Design Details

This simulation of the proposed network topology would be done using Contiki OS and Cooja simulator (implemented using C programming language). In order to program and control the working of IoT devices in the network, these source files shall be modified and recompiled. To analyze the packets sent, Wireshark tool would be utilized. It is an open-source sniffer tool.

5.4.1 Novelty

Our study deals with a **new** network topology with sensors deployed in the healthcare sector. These sensors generate heterogeneous traffic and need an efficient and secure routing methodology. Most implementations of RPL deal with homogenous traffic. The **integration of COAP and SSE service** into the system is a new concept.

5.4.2 Performance

The routing protocol must be highly efficient with respect to performance and energy. The main work of these protocols is to relay or transmit information from sensors on patients (including patients in ICU). This kind of information could be critical and could demand immediate action or attention making good/ high performance/less delay necessary to be factored in. Suitable Objective functions must be utilized to achieve good performance.

5.4.3 Security

The data carried by the protocol is sensitive (consists of patient health and personal information). Therefore, security is a must to protect against unauthorized access and modification to data. The protocol must be resistant to various attacks to preserve the confidentiality and integrity of data.

5.4.4 Reliability

The routing protocol must deliver packets successfully with minimal packet loss. The 'Packet Delivery Ratio' (PDR) = (num of packets delivered)/ (total num of packets sent) must be high. The project aims to have a PDR of a minimum of 85-90% for the entire system to be reliable.

5.4.5 Maintainability

The routing protocol must be easily readable and maintainable. It must be open to changes and enhancements. Heterogeneous traffic is being generated in various other domains like the military, smart cities, and so on. Therefore, the protocol must have the ability to be modified to suit other domains.

5.4.6 Portability

The routing protocol must be portable on different architectures (different hardware and OS). RPL is known to work with different LLN (Low Power and Lossy network) devices (e.g. Zmote, Skymote – motes/sensor nodes in Cooja simulator with different hardware architectures). It is also known to work with different light-weight OS like Contiki, TinyOS, and T-Kernel.

5.4.7 Reusability

The routing protocol must have the ability to be reused in sectors with similar needs as that of healthcare. RPL is being used in many domains that employ LLN's.

CHAPTER 6

PROPOSED METHODOLOGY

6.1 Approach

Figure 6.1 illustrates the layout of sensor deployment at the hospital. Three departments are considered, the Intensive Care Unit (ICU), Special Ward (SW), and Laboratories.

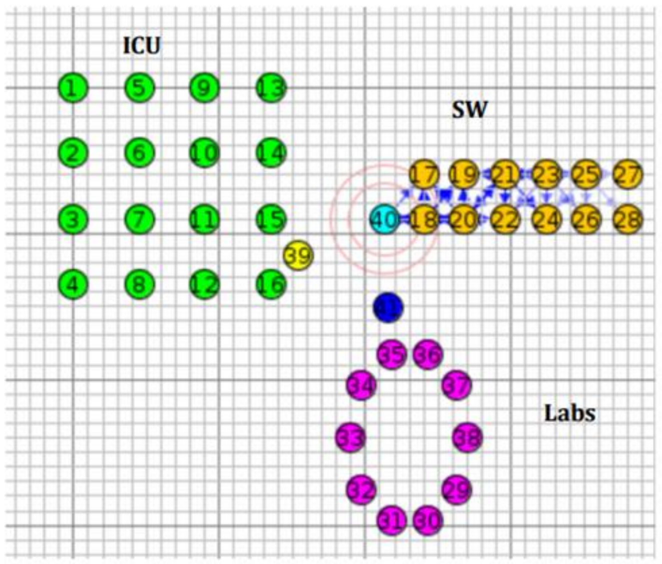


Figure 6.1: Sensor Deployment

A **multiple instance approach** is proposed to route traffic efficiently in this scenario. Data from ICU and Special ward is considered as one instance while traffic from labs forms the second instance. This is because both critical and medium-critical data share characteristics such as high sending intervals and less delay. While periodic data is sent after a large fixed interval. Here heterogeneity is only defined based on the QoS parameters requirement.

Further, a multi-sink approach is proposed. Each department is provided with a **separate sink** to relay information. Thus, three DODAGs would be formed in total with two of them being part of one instance (critical traffic) and the other forming a separate instance (periodic traffic). Figure 6.2 illustrates the proposed mechanism.

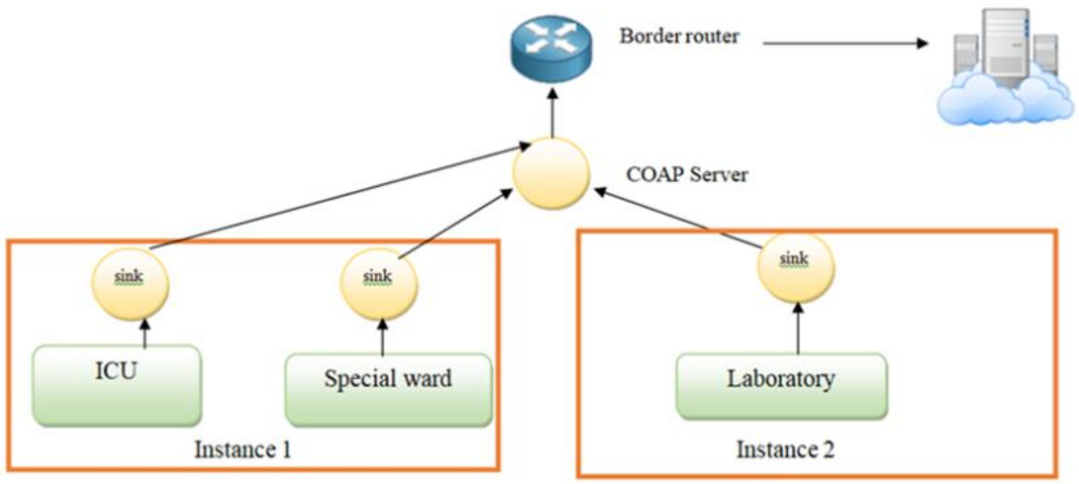


Figure 6.2: Proposed Approach

This paper aims to compare and contrast the single-instance and single-sink approach with the multiple-instance and multi-sink [with 2 and 3 sinks] approach for the proposed architecture.

A single OF is used when all the nodes have the same QoS (Quality of Service) requirement. In a heterogeneous environment, a Multiple Instance approach (DODAGs running on different OFs to create routes) shall be deployed [4][5]. The proposed multi-instance approach is shown in the figure below.

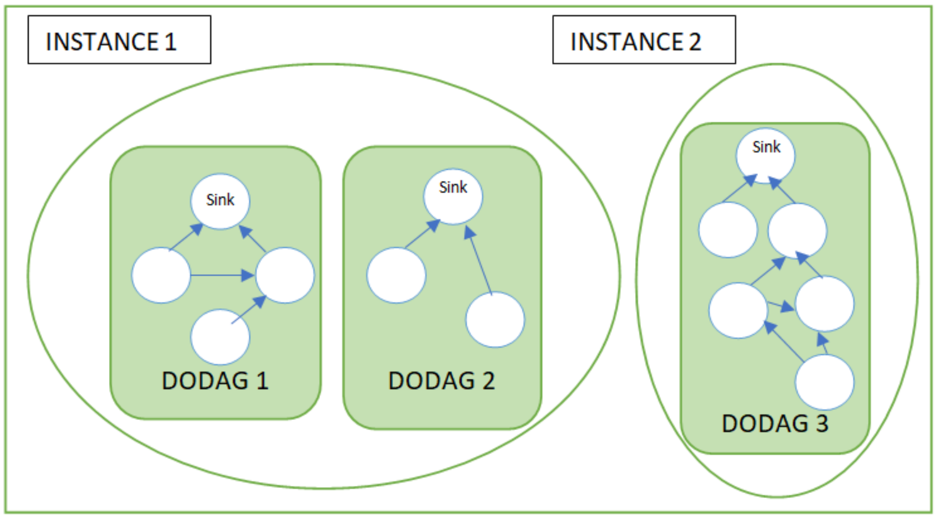


Figure 6.3: Multiple Instance

6.2 Simulation Parameters

Critical, medium-critical, and periodic traffic is distinguished based on sending intervals and payload sizes. Critical data from ICU would be sent regularly at short intervals. While medium-critical data would also be sent regularly. Periodic data will be sent after large interval [11]. Further, the payload length will be in the order critical, medium-critical and periodic as of smallest to largest. Table 6.1 and figure 6.4 highlight the different values.

	Critical traffic	**Medium-critical traffic**	**Periodic traffic**
Sending interval	15 sec	30 sec	200 sec

Table 6.1: Sending interval

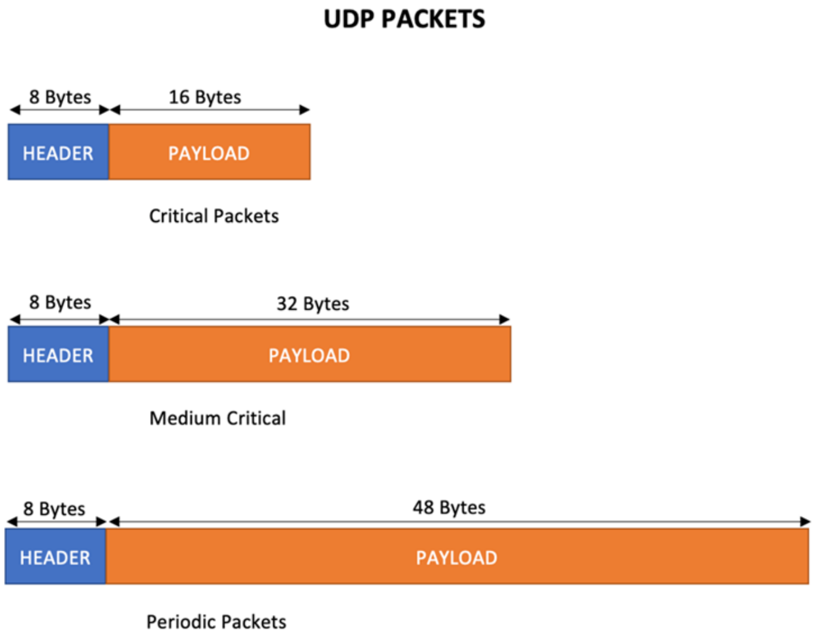


Figure 6.4: Payload length

Figures 6.5, 6.6 and 6.7 display the topology for multi-sink, multi-instance, and single-sink, single-instance approaches. Multiple Instance with 2 sinks has a sink common for ICU and Special ward and a dedicated sink for Lab sensors.

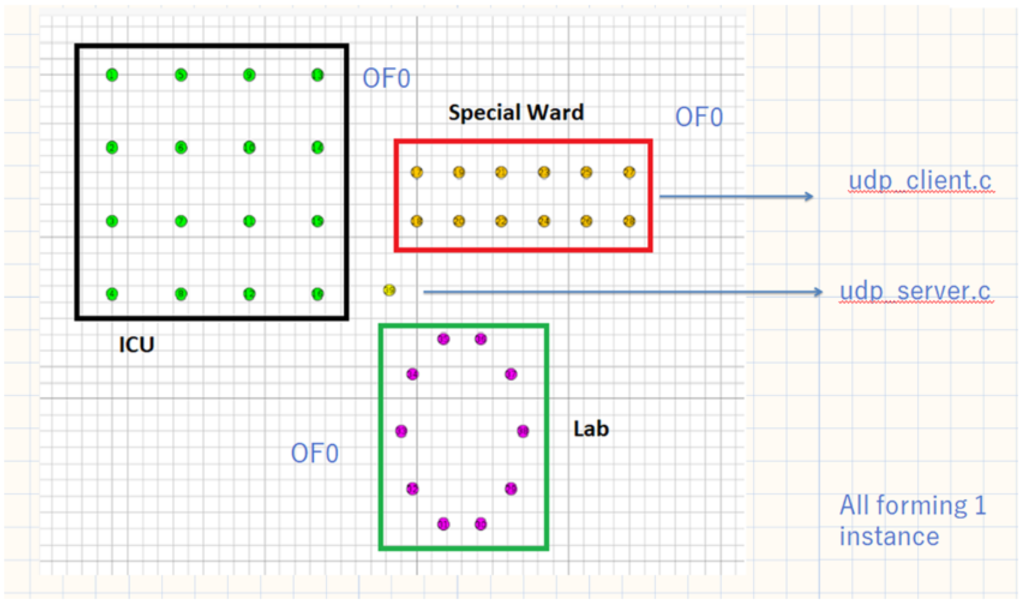


Figure 6.6: Single Instance with Single sink

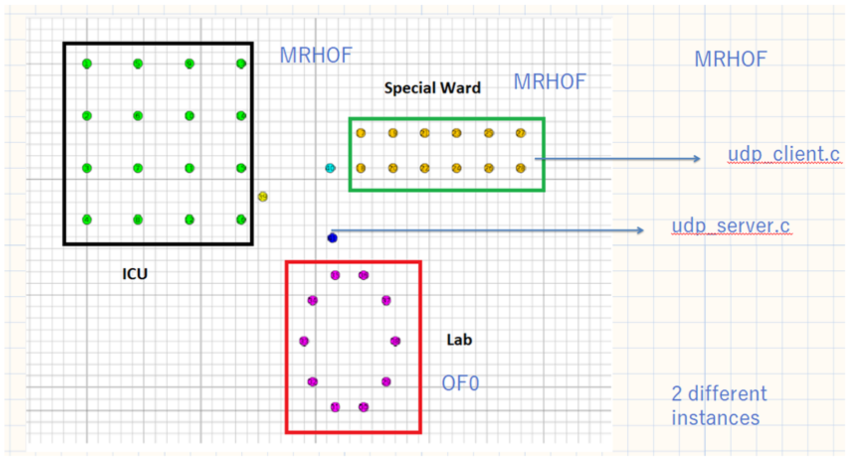


Figure 6.5: Multiple Instance with Multi (3) sinks

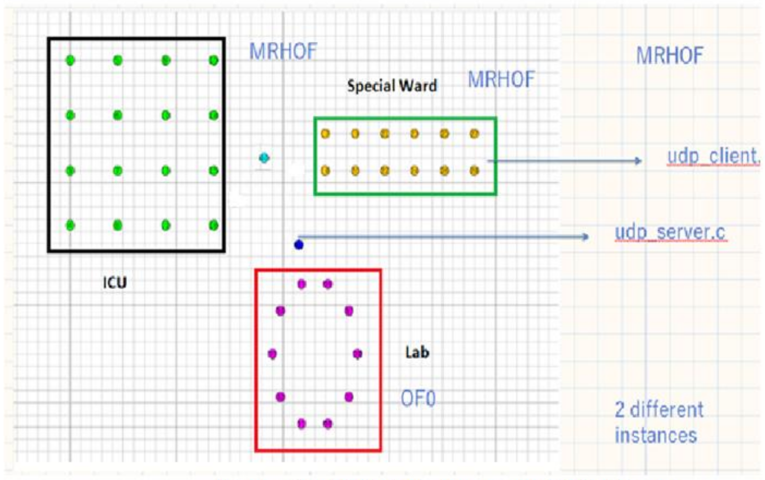


Figure 6.6: Multiple Instance with Multi (2) sinks

6.3 Metrics

The comparison of the approaches shall be done over the below mentioned metrics/ performance indicators.

6.3.1 Packet Delivery Ratio (PDR)

It is defined as the ratio of the total number of packets received at a node and the total number of packets sent to that node, i.e

$$\frac{Total\ number\ of\ packets\ recieved}{Total\ number\ of\ packets\ sent} * 100 \qquad (1)$$

In this paper, we calculate the PDR for each type of traffic. E.g. total number of critical packets received at the sink to the total number of critical packets to the sink.

It displays the degree of network reliability with a direct proportionality [21].

In this paper, this metric was calculated using JavaScript code in the Cooja Script editor (APPENDIX B).

6.3.2 Latency

Delay/Latency is defined as the time difference between when a packet reaches its destination to when it was sent by the sender. It is calculated for each packet and an average is taken (termed average latency) [21]. In this paper, this metric was calculated using JavaScript code in the Cooja Script editor (APPENDIX B).

$$\frac{\sum_{i=1}^{n\,(number\ of\ recieved\ packets)} (Time\ at\ which\ pkt(i) was\ recieved - Time\ at\ which\ pkt(i) was\ sent)}{n} \quad (2)$$

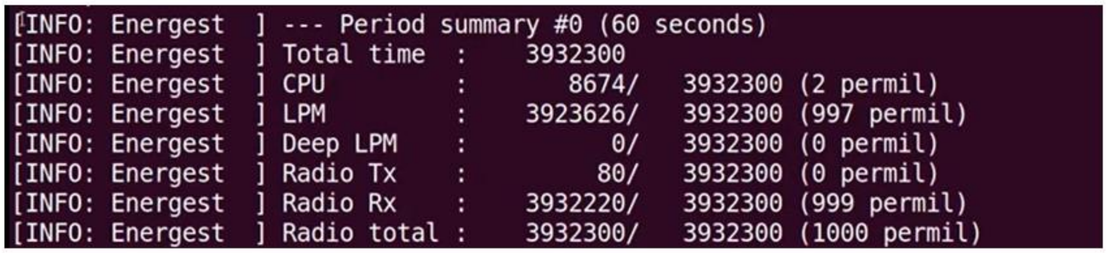
6.3.3 Control traffic overhead

Messages exchanged between nodes to set up DODAGs form and update routes are termed control messages. DIS, DIO, and DAO messages come under this category. Control traffic overhead is defined as the total number of control messages transmitted by all the nodes in the given simulation time. This metric is representative of how much energy is used by nodes to build the network and paths. Thus, a smaller value indicates the goodness of the routing protocol [23]. In this paper, this metric was calculated by analysing the resultant pcap file in Wireshark.

$$\boxed{\sum_{i=1}^{m\,(number\ of\ nodes\ except\ root)} DIS(i)} + \boxed{\sum_{j=1}^{n\,(number\ of\ nodes\ except\ leaf\ nodes)} DIO(j)} + \boxed{\sum_{k=1}^{p} DAO(k)} \quad (4)$$

6.3.4 Average Energy consumption

Energy consumption is defined as the total amount of energy utilized by the nodes in transmitting packets (both data and control) to other nodes in the network. It is a key aspect to measure as these devices are constrained. The energy consumption of each node shall be calculated using the "Energest module" in Contiki-NG. This library helps to track the components of the node and prints those details as part of mote output. Figure 6.7 illustrates this output. The log file (indicating simulation output) is given as an input to a python script written to calculate the energy consumption with the outputted values. The code is provided in APPENDIX C. The average energy consumption of all nodes shall be taken and contrasted against the proposed and existing approach.



```
[INFO: Energest  ] --- Period summary #0 (60 seconds)
[INFO: Energest  ] Total time  :    3932300
[INFO: Energest  ] CPU         :       8674/   3932300 (2 permil)
[INFO: Energest  ] LPM         :    3923626/   3932300 (997 permil)
[INFO: Energest  ] Deep LPM    :          0/   3932300 (0 permil)
[INFO: Energest  ] Radio Tx    :         80/   3932300 (0 permil)
[INFO: Energest  ] Radio Rx    :    3932220/   3932300 (999 permil)
[INFO: Energest  ] Radio total :    3932300/   3932300 (1000 permil)
```

Figure 6.8: Energest Module Output

CHAPTER 7

IMPLEMENTATION AND PSEUDOCODE

Figure 7.1 demonstrates data flow between different layers.

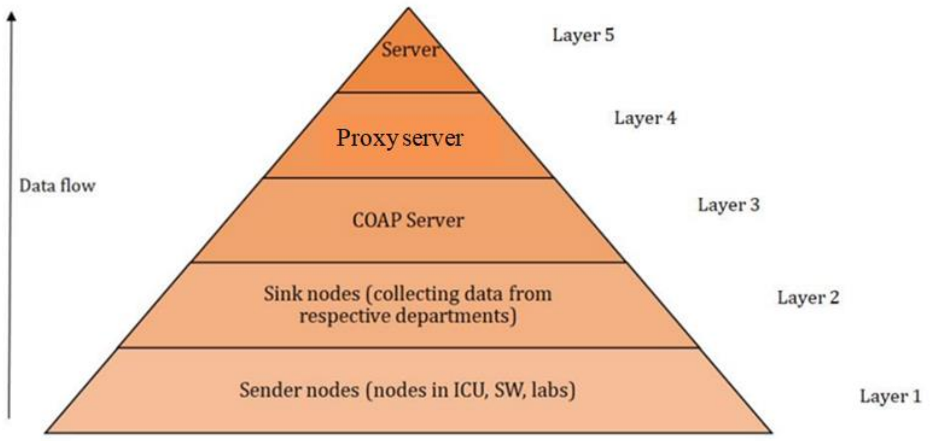

Server
Layer 5
Proxy server
Layer 4
Data flow
COAP Server
Layer 3
Sink nodes (collecting data from respective departments)
Layer 2
Sender nodes (nodes in ICU, SW, labs)
Layer 1

Figure 7.1: Flow of data through layers

At layer 1, the sensors are deployed to collect data. This information is routed to the respective sink nodes (forming layer 2). At layer 3, the COAP server is active and queues all packets from all three sinks.

The process between Layer 3, 4, 5 is elaborated in figure 7.2. The backend server (indicated as client in the figure) establishes a SSE connection with the Proxy server (shown as server in the figure). Thus, the client/ backend server continues to listen on this connection.

The proxy server periodically sends a COAP request to the COAP server. The entire queue (maintained by the COAP server) is sent back as a COAP response to the proxy server. On receiving data, the Server/ Proxy server sends it to the client/Backend server via the SSE connection to update the website.

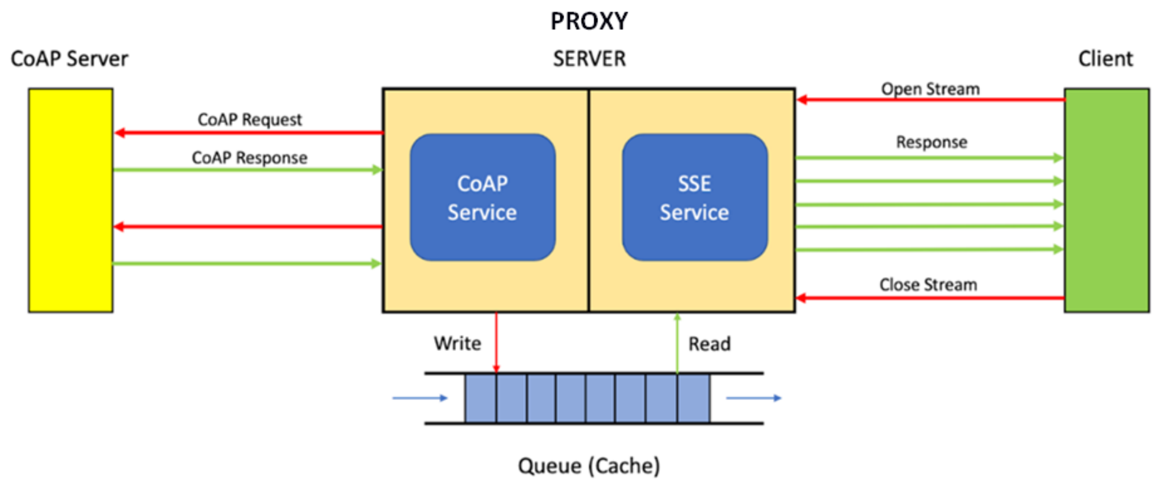


Figure 7.2: COAP-Proxy-Backend Server communications

RPL protocol works at layer 1. Figure 7.3 shows the sequence of activities undertaken by the protocol to form routes.

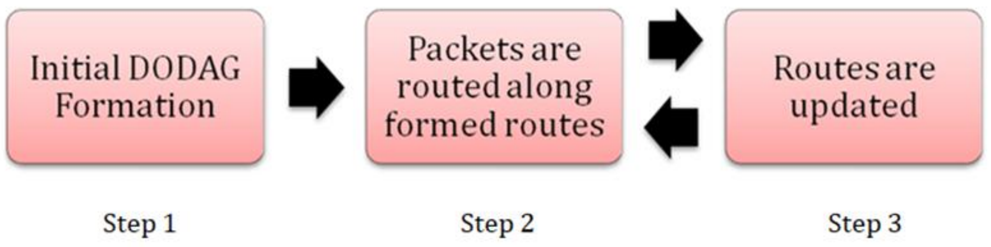


Figure 7.3: Working of RPL protocol

Step 1 of figure 7.3 s elaborated in figure 7.4. A series of control messages are exchanged between the nodes to form the DODAG.

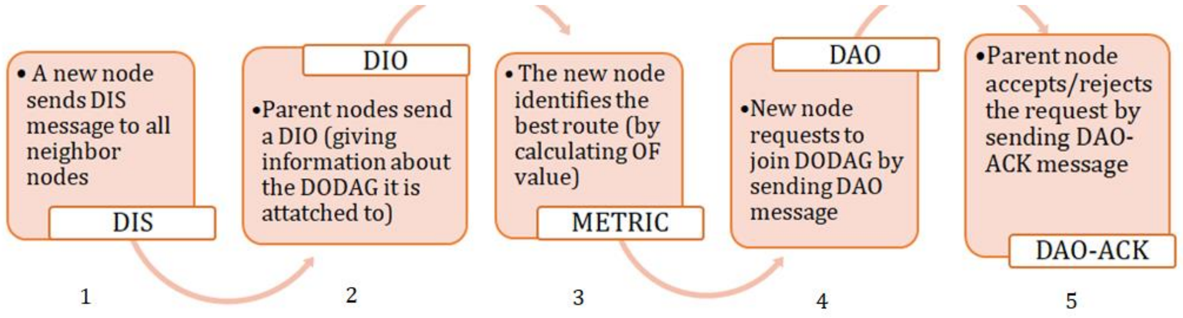

• A new node sends DIS message to all neighbor nodes
DIS
1
DIO
•Parent nodes send a DIO (giving information about the DODAG it is attatched to)
2
• The new node identifies the best route (by calculating OF value)
METRIC
3
DAO
•New node requests to join DODAG by sending DAO message
4
•Parent node accepts/rejects the request by sending DAO-ACK message
DAO-ACK
5

Figure 7.4: DODAG Formation

All the nodes in ICU and SW run on a single OF i.e. MRHOF (The Minimum Rank with Hysteresis Objective Function) while sensors at Labs run on OF0 (Objective Function 0).

The pseudo-code for OF0 and MRHOF is explained below:

```
hp=[] //hop count array

for i in (all parents that sent a DIO message)
        hp[i]= hop_count of node to sink node


rank= min(hp)  // minumum hop count
parent node to new node = i
```

Figure 7.5: Pseudo code for OF0

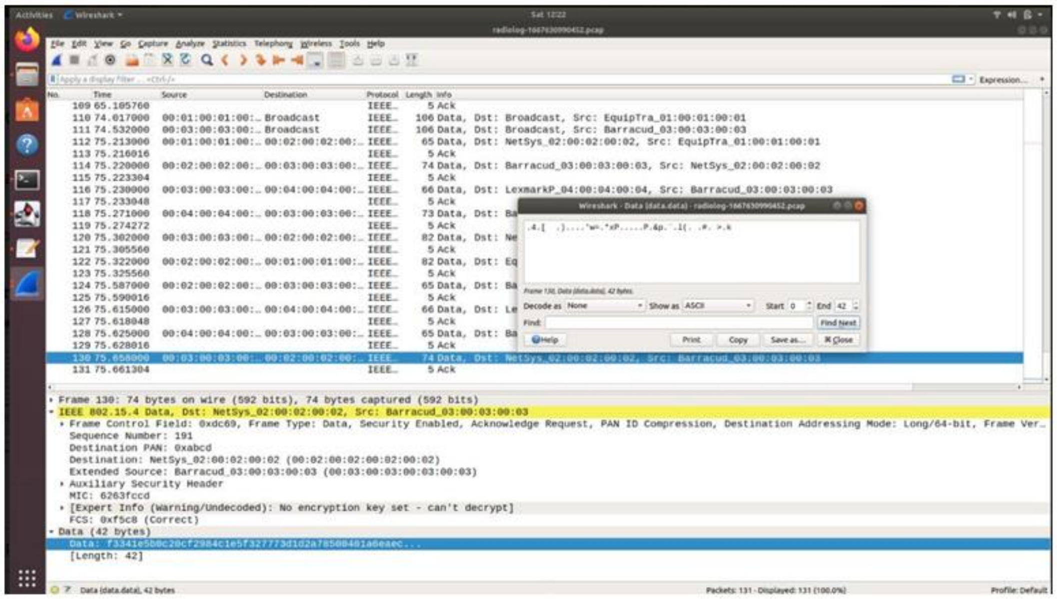
```
etx=[] //etx array- is initialized for all possible routes with the formula
//dxy= probability that a message sent from node x to y reaches node y
//dyx= probability that the ACK message sent from y reaches x
//etx = 1/(dxy*dyx)


for i in (all parents that sent a DIO message):
        rank[i] = rank(i) + etx[new_node, i]


rank= min(rank array)  // minumum rank value becomes rank of new node
parent node to new node = i
```

Figure 7.6: Pseudo code for MRHOF

Certain measures have been taken to ensure secure data transfer within the simulation. Contiki-NG provides link-layer security for IEEE 802.15.4 TSCH (Time slotted Channel Hoping). It maintains 2 cryptographic keys for data transfer and control/management messages. Link-layer security secures frames as it travels across the medium. Thus, it secures all communication protocol layers including routing protocols like RPL.

Further DTLS (Datagram Transport Layer Security) is used in this project. It is a security layer built over UDP that ensures all CIA properties are maintained. It has both crpytographic and authentication mechanism all resolved within a short run of time thereby not adding to latency [39].

The image of the encrpyted packet is shown below (capstured in Wireshark on running the simualtion.


Activities
Wireshark
Sat 12:22
radiolog-1667630990452.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Wireshark · Data (data.data) · radiolog-1667630990452.pcap
Decode as None
Show as ASCII
Start 0
End 42
Find:
Find Next
Help
Print
Copy
Save as...
Close

```
No.  Time       Source               Destination          Protocol Length Info
 109 65.105760                                            IEEE…      5 Ack
 110 74.017000  00:01:00:01:00:…     Broadcast            IEEE…    106 Data, Dst: Broadcast, Src: EquipTra_01:00:01:00:01
 111 74.532000  00:03:00:03:00:…     Broadcast            IEEE…    106 Data, Dst: Broadcast, Src: Barracud_03:00:03:00:03
 112 75.213000  00:01:00:01:00:…     00:02:00:02:00:…     IEEE…     65 Data, Dst: NetSys_02:00:02:00:02, Src: EquipTra_01:00:01:00:01
 113 75.216016                                            IEEE…      5 Ack
 114 75.220000  00:02:00:02:00:…     00:03:00:03:00:…     IEEE…     74 Data, Dst: Barracud_03:00:03:00:03, Src: NetSys_02:00:02:00:02
 115 75.223304                                            IEEE…      5 Ack
 116 75.230000  00:03:00:03:00:…     00:04:00:04:00:…     IEEE…     66 Data, Dst: LexmarkP_04:00:04:00:04, Src: Barracud_03:00:03:00:03
 117 75.233048                                            IEEE…      5 Ack
 118 75.271000  00:04:00:04:00:…     00:03:00:03:00:…     IEEE…     73 Data, Dst: Ba
 119 75.274272                                            IEEE…      5 Ack
 120 75.302000  00:03:00:03:00:…     00:02:00:02:00:…     IEEE…     82 Data, Dst: Ne
 121 75.305560                                            IEEE…      5 Ack
 122 75.322000  00:02:00:02:00:…     00:01:00:01:00:…     IEEE…     82 Data, Dst: Eq
 123 75.325560                                            IEEE…      5 Ack
 124 75.587000  00:02:00:02:00:…     00:03:00:03:00:…     IEEE…     65 Data, Dst: Ba
 125 75.590016                                            IEEE…      5 Ack
 126 75.615000  00:03:00:03:00:…     00:04:00:04:00:…     IEEE…     66 Data, Dst: Le
 127 75.618048                                            IEEE…      5 Ack
 128 75.625000  00:04:00:04:00:…     00:03:00:03:00:…     IEEE…     65 Data, Dst: Ba
 129 75.628016                                            IEEE…      5 Ack
 130 75.658000  00:03:00:03:00:…     00:02:00:02:00:…     IEEE…     74 Data, Dst: NetSys_02:00:02:00:02, Src: Barracud_03:00:03:00:03
 131 75.661304                                            IEEE…      5 Ack
```

```
.4.[  .}....'w=.*xP.....P.&p.`.l(. .#. >.k
Frame 130, Data (data.data), 42 bytes
```

```
▸ Frame 130: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▾ IEEE 802.15.4 Data, Dst: NetSys_02:00:02:00:02, Src: Barracud_03:00:03:00:03
  ▸ Frame Control Field: 0xdc69, Frame Type: Data, Security Enabled, Acknowledge Request, PAN ID Compression, Destination Addressing Mode: Long/64-bit, Frame Ver…
    Sequence Number: 191
    Destination PAN: 0xabcd
    Destination: NetSys_02:00:02:00:02 (00:02:00:02:00:02:00:02)
    Extended Source: Barracud_03:00:03:00:03 (00:03:00:03:00:03:00:03)
  ▸ Auxiliary Security Header
    MIC: 6263fccd
  ▸ [Expert Info (Warning/Undecoded): No encryption key set - can't decrypt]
    FCS: 0xf5c8 (Correct)
▾ Data (42 bytes)
    Data: f3341e5b0c20cf2984c1e5f327773d1d2a78500401a6eaec...
    [Length: 42]
```

```
Data (data.data), 42 bytes          Packets: 131 · Displayed: 131 (100.0%)          Profile: Default
```

Figure 7.7: Encryption

CHAPTER 8

RESULTS AND DISCUSSION

8.1 Packet Delivery Ratio (PDR)

The PDR values are drastically improved in the proposed approach for all types of traffic (at all reception ratios).

For critical traffic, Single Instance and Single Sink (SI+SS) approach provides lesser values compared to Multiple Instance + Multi Sink [3 sinks] (MI+MS) approach at all RX values. This difference becomes large at RX ratios below 80%. At 50%, SI+SS approach provides only 83.6734% while this was improved to 99.6817% in the MI+MS approach.

For medium-critical traffic, the SI+SS approach gives only 98.84% and 47.4137% at 85 and 50% RX ratios. While these values are increased to 100% and 100% in the MI+MS approach. Similarly, values jump are seen at the other reception ratios.

For periodic traffic, SI+SS approach presents 0 at 50% RX ratios. While this value is increased to 97.0588% in the MI+MS approach. The rest of the values are 100 in both approaches.

Generally, at a 50% RX ratio, the PDR values will be considerably low because of the low reception ability of the sink node thereby leading to high data loss. In the SI+SS approach, all kinds of data traffic packets are aimed at one node making it congested. Critical packets are sent at a 4ppm rate and medium critical with a 2ppm rate. Periodic packets are sent at a 0.33 ppm rate thereby around 18-19 packets are sent in the entire simulation. With a single congested sink, none of the periodic packets get a chance to reach the sink node making the PDR 0%. In contrast, in the MI+MS approach, each kind of traffic has its sink node. Therefore, periodic traffic gets a better PDR value of 97.0588%.

The reduction in congestion is one of the main reasons for the improvement in PDR in all traffic kinds.

Another reason for the change in PDR for critical and medium-critical traffic is due to the use of ETX (in MI+MS). ETX considers the quality of the communication links and can rapidly respond to dynamic network conditions, a feature that is missing in OF0 [29]. This feature helps us to meet high-reliability requirements for critical and medium-critical traffic.

Further, the variation in PDR in SI+SS approach with RX values is large compared to the MI+MS method thereby making it a better choice in realistic scenarios where RX ratios are less than 100%.

The reason for small differences between SI+SS and MI+MS approach could be attributed to the small number of nodes in the network.

The same experiment was conducted for MI+MS (with 2 sinks) and the results are shown in figure 8.1. It is seen that MI+MS (3 sinks) outperforms all the approaches.

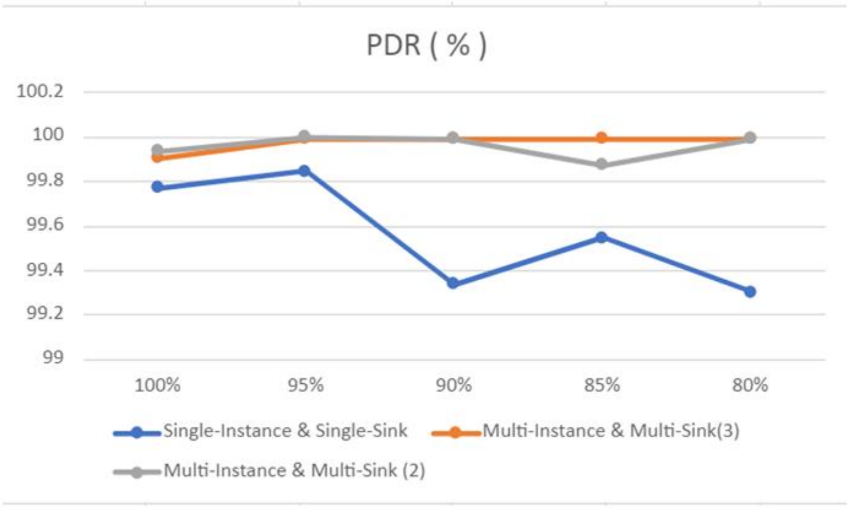


Figure 8.1: PDR values

8.2 Latency

Figure 8.2 displays the overall latency results obtained. Here latency is calculated as delay of every packet received to the total number of packets received irrespective of the traffic type. The MI+MS[with 3 sinks] approach gives better performance at almost all RX ratios.
The multi-sink approach gives lower latency due to the presence of closely placed dedicated sink. Further it is proved in multiple papers that ETX has a lesser hop count than OF0 thereby giving way for lesser delay [32].
Furthermore, as the RX ratios decrease, latency values increase in both the methods. But the variation is reduced in MI+MS approach. This is because MRHOF was used for critical and medium critical traffic and it considers data related to link quality such as ETX while forming routes.

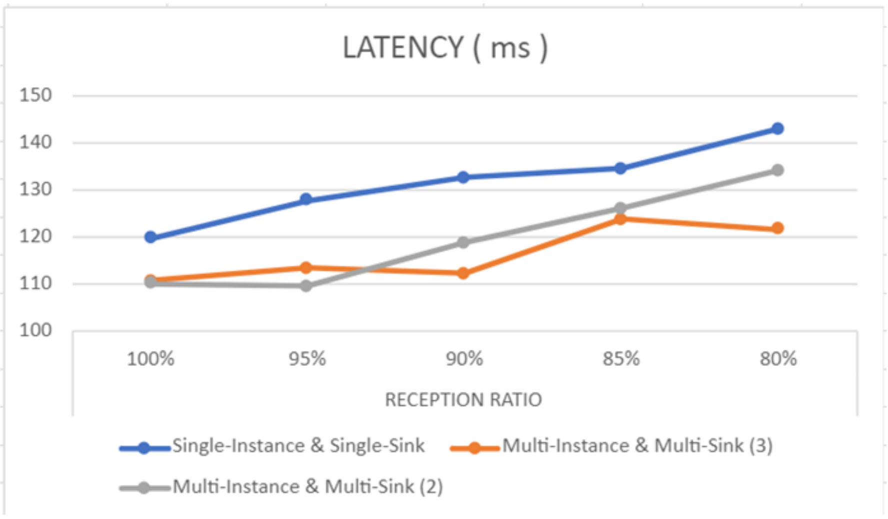


Figure 8.2: Latency values

8.3 Control Traffic Overhead

Figure 8.3 displays the results obtained. Control traffic overhead is calculated as total number of control messages transmitted throughout the simulation time to form and maintain routes. It is observed that MI+MS [3 sinks] has lesser control traffic overhead among all the approaches at all RX ratios. Further, as the reception levels decrease, there is an increase seen in the control messages passed, but the variation is less in our proposed approach. Higher the overhead, higher is the network traffic and energy drain [38].

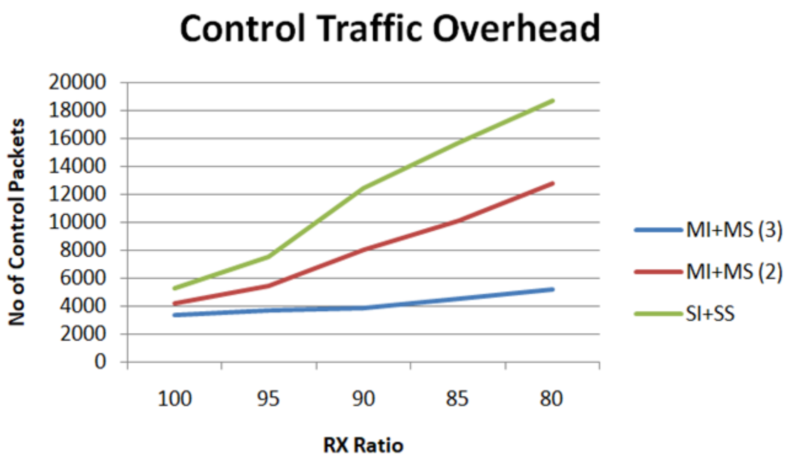


Figure 8.3: Control Traffic Overhead values

8.4 Average Energy Consumption

Figure 8.4 displays the results obtained. Average energy consumption of all nodes is the least in proposed approach of MI+MS [3 sinks] at all RX ratio. This is attributed to the presence of dedicated sinks. This avoids normal sender nodes to be relay nodes (nodes that transmit information of other nodes to the sink), thereby reducing the energy consumption. Further, the energy consumption of MRHOF is lesser than OF0 because it measures link quality while forming routes thereby creating better routes, allowing lesser re-transmissions and saving energy [40].

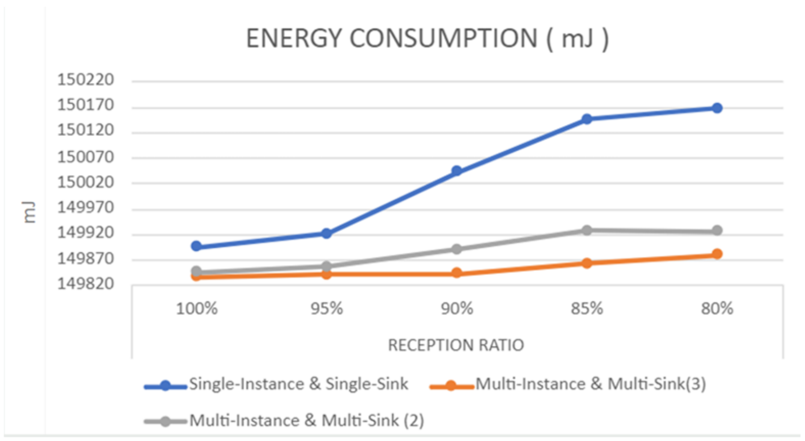


Figure 8.4: Energy consumption values

CHAPTER 9

CONCLUSION AND FUTURE WORK

In summary, our project proposes the use of Multiple Instance RPL to handle heterogeneous traffic within a single network. The instances were formed based on variations sending interval, QoS requirements and payload sizes. Thus, each department was made into a separate instance. Further, a multi sink[3 sinks – one for each department] approach was also chosen to avoid hotspots within the network. The proposed approach was compared with MI+MS[2 sinks] and SI+SS approach in Cooja Simulator Results showed that the proposed approach is better at all reception ratios in terms of PDR, latency, Control Traffic Overhead and energy consumption.

As part of future implementation, the current topology can be upgraded with multiple departments and floors. The proposed routing mechanism shall be tested on the aforementioned network. The routing mechanism could be altered by making the number of instances dynamic rather than pre-defined. Additional security mechanisms (such as Firewall, Intrusion Detection System) can be incorporated to ensure all security principles are maintained. Our project aimed at securing data transfer within the simulation bounds, further, attacks on COAP and SSE could be tackled.

REFERNCES

[1] Rayes, A. and Salam, S., 2017. Internet of things from hype to reality. *The road to Digitization*, *2*.

[2] Martocci, J., De Mil, P., Riou, N. and Vermeylen, W., 2010. *Building automation routing requirements in low-power and lossy networks* (No. rfc5867).

[3] Zhao, W., Luo, X. and Qiu, T., 2017. Smart healthcare. *Applied Sciences*, *7*(11), p.1176.

[4] Kodali, R.K., Swamy, G. and Lakshmi, B., 2015. An implementation of IoT for healthcare. In 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS)(pp. 411-416).

[5] Dautov, R., Distefano, S. and Buyya, R., 2019. Hierarchical data fusion for smart healthcare. *Journal of Big Data*, *6*(1), pp.1-23.

[6] Kumar, V. and Tiwari, S., 2012. Routing in IPv6 over low-power wireless personal area networks (6LoWPAN): A survey. *Journal of Computer Networks and Communications*, *2012*.

[7] Kushalnagar, N., Montenegro, G. and Schumacher, C., 2007. IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals.

[8] Witwit, A.J. and Idrees, A.K., 2018, October. A comprehensive review for RPL routing protocol in low power and lossy networks. In *International conference on new trends in information and communications technology applications* (pp. 50-66). Springer, Cham.

[9] Winter, T., 2011. RPL: IPv6 routing protocol for low power and lossy networks. *RFC6550*.

[10] Al-Abdi, A., Mardini, W., Aljawarneh, S. and Mohammed, T., 2019, December. Using multiple RPL instances for enhancing the performance of IoT-based systems. In *Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems* (pp. 1-5).

[11] Banh, M., Mac, H., Nguyen, N., Phung, K.H., Thanh, N.H. and Steenhaut, K., 2015, October. Performance evaluation of multiple RPL routing tree instances for Internet of Things applications. In *2015 international conference on advanced technologies for communications (ATC)* (pp. 206-211). IEEE.

[12] Mardini, W., Aljawarneh, S. and Al-Abdi, A., 2021. Using multiple RPL instances to enhance the performance of new 6G and Internet of Everything (6G/IoE)-based healthcare monitoring systems. *Mobile Networks and Applications*, *26*(3), pp.952-968.

[13] Abdel Hakeem, S.A., Hady, A.A. and Kim, H., 2019. RPL routing protocol performance in smart grid applications based wireless sensors: Experimental and simulated analysis. *Electronics*, *8*(2), p.186.

[14] Thubert, P., 2012. *Objective function zero for the routing protocol for low-power and lossy networks (RPL)* (No. rfc6552).

[15] Gnawali, O. and Levis, P., 2012. *The minimum rank with hysteresis objective function* (No. rfc6719).

[16] Gnawali, O. and Levis, P., 2010. The ETX Objective Function for RPL," draft-gnawali-roll-etxof-01. *URL https://tools. ietf. org/html/draft-gnawali-roll-etxof-00*.

[17] Kim, H.S., Ko, J., Culler, D.E. and Paek, J., 2017. Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey. *IEEE Communications Surveys & Tutorials*, *19*(4), pp.2502-2525

[18] Contiki: The Open Source Operating System for the Internet of Things. (Accessed on 16/7/2022). [Online] Available: http://www.contiki-os.org/.

[19] Stan, A., 2007. Porting the Core of the Contiki operating system to the TelosB and MicaZ platforms. *Computer Science, International University Bremen, Campus Ring, Bremen, Germany*, *1*, p.28759.

[20] Roussel, K. and Zendra, O., 2016, February. Using Cooja for WSN simulations: Some new uses and limits. In *EWSN 2016—NextMote workshop* (pp. 319-324). Junction Publishing.

[21] Ali, H., 2012. A performance evaluation of rpl in contiki.

[22] Jyothi, S.A., Singla, A., Godfrey, P.B. and Kolla, A., 2016, November. Measuring and understanding throughput of network topologies. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 761-772). IEEE.

[23] Charles, A.J. and Kalavathi, P., 2018. QoS measurement of RPL using Cooja simulator and Wireshark network analyser. *International Journal of Computer Sciences and Engineering*, *6*(4), pp.283-291.

[24] Xie, H., Zhang, G., Su, D., Wang, P. and Zeng, F., 2014, June. Performance evaluation of RPL routing protocol in 6lowpan. In *2014 IEEE 5th International Conference on Software Engineering and Service Science* (pp. 625-628). IEEE.

[25] Gaddour, O., Koubaa, A., Chaudhry, S., Tezeghdanti, M., Chaari, R. and Abid, M., 2012, March. Simulation and performance evaluation of DAG construction with RPL. In *Third international conference on communications and networking* (pp. 1-8). IEEE.

[26] Monowar, M.M. and Basheri, M., 2020. On providing differentiated service exploiting multi-instance RPL for industrial low-power and lossy networks. *Wireless Communications and Mobile Computing*, *2020*.

[27] Peyrard, A., Kosmatov, N., Duquennoy, S. and Raza, S., 2018, February. Towards formal verification of Contiki: Analysis of the AES–CCM* modules with Frama-C. In *RED-IOT 2018-Workshop on Recent advances in secure management of data and resources in the IoT*.

[28] CCM Mode. xilinx.github.io, https://xilinx.github.io/Vitis_Libraries/security/2019.2/guide_L1/internals/ccm.html. Accessed 24 July 2022.

[29] Rajalingham, G., Gao, Y., Ho, Q.D. and Le-Ngoc, T., 2014, September. Quality of service differentiation for smart grid neighbor area networks through multiple RPL instances. In *Proceedings of the 10th ACM symposium on QoS and security for wireless and mobile networks* (pp. 17-24).

[30] Lamaazi, H., Benamar, N. and Jara, A.J., 2016, May. Study of the impact of designed objective function on the RPL-based routing protocol. In *International Symposium on Ubiquitous Networking* (pp. 67-80). Springer, Singapore.

[31] Zaatouri, I., Alyaoui, N., Guiloufi, A.B. and Kachouri, A., 2017, December. Performance evaluation of RPL objective functions for multi-sink. In *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)* (pp. 661-665). IEEE.

[32] Yassein, M.B., Flefil, A., Krstic, D., Khamayseh, Y., Mardini, W. and Shatnawi, M., 2019, April. Performance evaluation of RPL in high density networks for internet of things (IoT). In *Proceedings of the 2019 8th International Conference on Software and Information Engineering* (pp. 183-187).

[33] Farooq, M.O., Sreenan, C.J., Brown, K.N. and Kunz, T., 2015, October. RPL-based routing protocols for multi-sink wireless sensor networks. In 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) (pp. 452-459). IEEE.

[34] Yoo, H., Shim, M., Kim, D. and Kim, K.H., 2010, June. GLOBAL: A Gradient-based routing protocol for load-balancing in large-scale wireless sensor networks with multiple sinks. In The IEEE symposium on Computers and Communications (pp. 556-562). IEEE.

[35] Mottola, L. and Picco, G.P., 2011. MUSTER: adaptive energy-aware multi-sink routing in wireless sensor networks. IEEE Transactions on Mobile Computing.

[36] Nassar, J., Berthomé, M., Dubrulle, J., Gouvy, N., Mitton, N. and Quoitin, B., 2018. Multiple instances QoS routing in RPL: Application to smart grids. Sensors, 18(8), p.2472.

[37] Lamaazi, H., Benamar, N. and Jara, A.J., 2018. RPL-based networks in static and mobile environment: A performance assessment analysis. Journal of King Saud University-Computer and Information Sciences, 30(3), pp.320-333.

[38] Musaddiq, A., Zikria, Y.B. and Kim, S.W., 2020. Routing protocol for Low-Power and Lossy Networks for heterogeneous traffic network. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), pp.1-23.

[39] Rescorla, E. and Modadugu, N., 2012. Datagram transport layer security version 1.2 (No. rfc6347).

APPENDIX A: DEFINITIONS, ACRONYMS AND ABBREVIATIONS

IoT – Internet of Things

IEEE 802.15.4 - A wireless access technology with a low data transmission rate specifically for low-power devices.

LoWPANs - Low-Power Wireless Personal Area Networks

IP- Internet Protocol

IPv6 - Internet Protocol Version 6

IETF - Internet Engineering Task Force

LLN- Low-Power and Lossy Networks

RPL- IPv6 Routing Protocol for LLNs

DODAG- Destination-Oriented Directed Acyclic Graph

OF- Objective Function

DIO -DODAG Information Object message

DIS- DODAG Information Solicitation message

DAO- DODAG Advertisement Object message

DAO-ACK - DODAG Advertisement Object Acknowledgement message

OF0 -Objective function 0

HC -Hop Count

MRHOF- Minimum Rank with Hysteresis Objective Function

ETX -Expected Transmission Count

BAN- Body Area Network

QoS - Quality of Service

ICU - Intense Care Unit

KAUH- King Abdullah University Hospital

MI-RPL - Multiple-Instance RPL solution for industrial LLN

OMNET++ - Objective Modular Network Testbed in C++

OS- Operating System

uIP - micro IP

Contiki-NG- Contiki –Next Generation

COAP -Constrained Application Protocol

SSE - Server-Sent-Events

PDR- Packet Delivery Ratio

MI+MS- Multiple Instance + Multi Sink

SI+SS- Single Instance and Single Sink

TSCH - Time slotted Channel Hoping

DTLS - Datagram Transport Layer Security

APPENDIX B: JS CODE TO CALCULATE LATENCY AND PDR

```
TIMEOUT(900000); // simulation duration in milliseconds

//PDR
num_messages_tx_c = 0;
num_messages_rx_c = 0;

num_messages_tx_m = 0;
num_messages_rx_m = 0;


num_messages_tx_p = 0;
num_messages_rx_p = 0;

created_rpl_dag = 0;


//latency
motes = [];
motesInfo = new Object();
```

```
while(true){
if(msg){
        if(msg.startsWith("DATA send")){

        if(1<=id<=16){ num_messages_tx_c+=1}

        else if(17<=id<=28){ num_messages_tx_m+=1}

        else if(29<=id<=38) { num_messages_tx_p+=1}
```

```
            if(motes.indexOf(id) == -1){
               motes.push(id);
               motesInfo[id.toString()] = {"sent_time": time , "latency": 0 };
            }
            motesInfo[id.toString()].sent_time = time;
         }


         if(msg.startsWith("DATA recv")) {


            sentMote = msg.substr(-2);
            arr = sentMote.split("");
            if(arr.indexOf(" ") != -1) sentMote = arr[1];

            s=parseInt(sentMote);
```

```
          if(1<=s<=16){ num_messages_rx_c+=1}

          else if(17<=s<=28){ num_messages_rx_m+=1}

          else if(29<=s<=38) { num_messages_rx_p+=1}
```

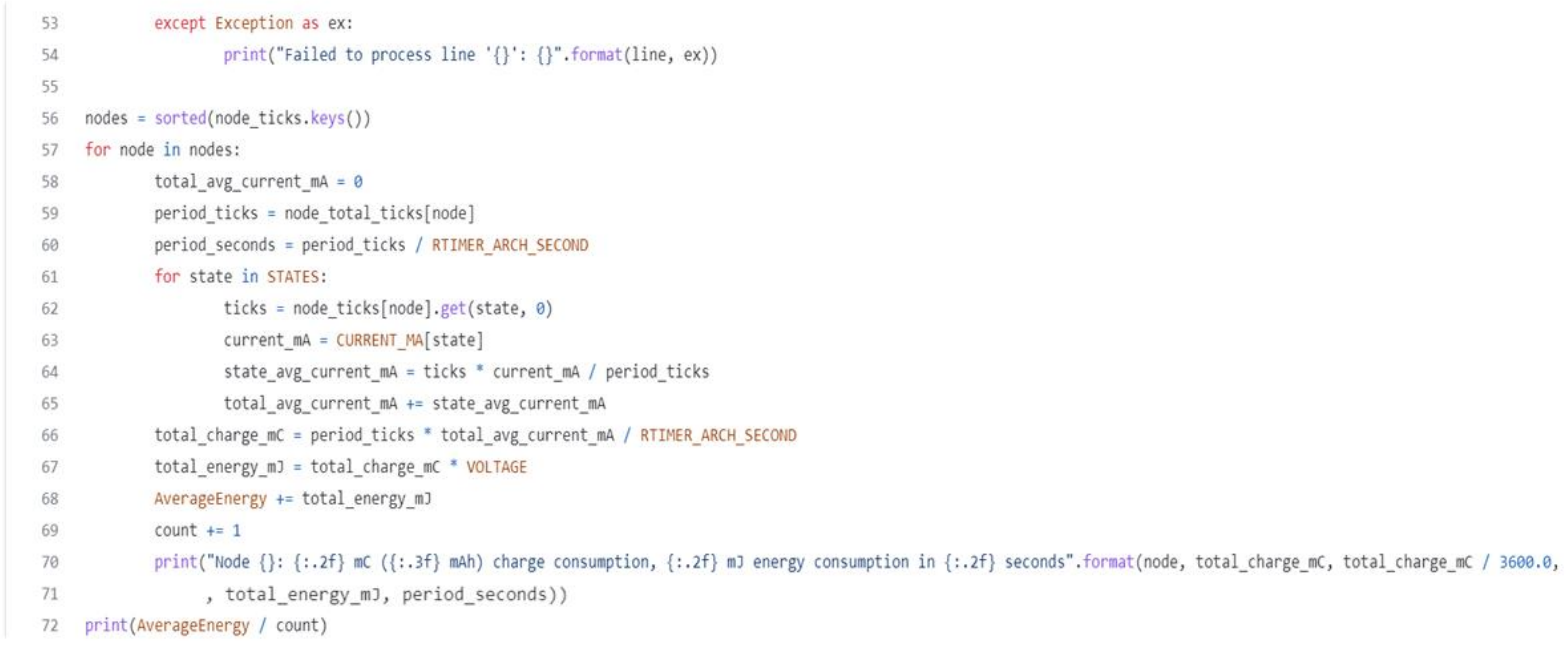
```
             if(motesInfo[sentMote.toString()]){
                sent_time = motesInfo[sentMote.toString()]["sent_time"];
                        latency = time - sent_time;
                        old_latency = motesInfo[sentMote.toString()]["latency"];
                        if(old_latency === 0)
                           motesInfo[sentMote.toString()]["latency"] = latency;
                        else{
                           latency = (latency + old_latency) / 2
                           motesInfo[sentMote.toString()]["latency"] = latency;
                        }

             }
          }



          if(msg.startsWith("created a new RPL dag")) {
             created_rpl_dag += 1;

          }

       }

       YIELD();
    }
```

```
timeout_function = function () {
    log.log("Script timed out.\n");

    log.log("Total Messages transmitted: " + (num_messages_tx_c+   num_messages_tx_m + num_messages_tx_p)  + " \n");

    log.log("Total Messages received: " + (num_messages_rx_c +    num_messages_rx_m +  num_messages_rx_p)  + " \n");

    log.log("PDR for critical traffic:    " + ((num_messages_rx_c/num_messages_tx_c)*100) + " \n");

    log.log("PDR for medium critical traffic:    " + ((num_messages_rx_m/num_messages_tx_m)*100) + " \n");


    log.log("PDR for periodic traffic:    " + ((num_messages_rx_p/num_messages_tx_p)*100) + " \n");


    log.log("Number of DAGS created  :  " + created_rpl_dag + " DAGs \n");

    total = 0;

    for(x in motesInfo) {
        log.log("mote " + x + " with latency " + motesInfo[x]["latency"] / 1000 + " ms \n");

        total += motesInfo[x].latency / 1000
    }
    log.log("Average latency: " + total/motes.length + " ms \n");


    log.testOK();
}
```

APPENDIX C: PYTHON SCRIPT TO CALCULATE ENERGY CONSUMPTION

```python
INPUT_FILE = "loglistener.txt"

# From Z1 node datasheet
CURRENT_MA = {
        "CPU" : 10,
        "LPM" : 0.023,
        "Deep LPM" : 0, # not used by Z1 nodes
        "Radio Rx" : 18.8,
        "Radio Tx" : 17.4,
}

STATES = list(CURRENT_MA.keys())

VOLTAGE = 3.0 # assume 3 volt batteries
RTIMER_ARCH_SECOND = 32768

node_ticks = {}
node_total_ticks = {}

AverageEnergy = 0
count = 0;

f = open(INPUT_FILE, "r")
```

```
for line in f:
        if "INFO: Energest" not in line:
                continue
        fields = line.split()
        try:
                node = int(fields[1][3:])
        except:
                continue

        if node not in node_ticks:
      # initialize to zero
                node_ticks[node] = { u : 0  for u in STATES }
                node_total_ticks[node] = 0
        try:
                state_index = 5
                state = fields[state_index]
                tick_index = state_index + 2
                if state not in STATES:
                        state = fields[state_index] + " " + fields[state_index+1]
                        tick_index += 1
                        if state not in STATES:
                                # add to the total time
                                if state == "Total time":
                                        node_total_ticks[node] += int(fields[tick_index])
                                continue
                # add to the time spent in specific state
                ticks = int(fields[tick_index][:-1])
                node_ticks[node][state] += ticks
```



```
        except Exception as ex:
                print("Failed to process line '{}': {}".format(line, ex))

nodes = sorted(node_ticks.keys())
for node in nodes:
        total_avg_current_mA = 0
        period_ticks = node_total_ticks[node]
        period_seconds = period_ticks / RTIMER_ARCH_SECOND
        for state in STATES:
                ticks = node_ticks[node].get(state, 0)
                current_mA = CURRENT_MA[state]
                state_avg_current_mA = ticks * current_mA / period_ticks
                total_avg_current_mA += state_avg_current_mA
        total_charge_mC = period_ticks * total_avg_current_mA / RTIMER_ARCH_SECOND
        total_energy_mJ = total_charge_mC * VOLTAGE
        AverageEnergy += total_energy_mJ
        count += 1
        print("Node {}: {:.2f} mC ({:.3f} mAh) charge consumption, {:.2f} mJ energy consumption in {:.2f} seconds".format(node, total_charge_mC, total_charge_mC / 3600.0,
              , total_energy_mJ, period_seconds))
print(AverageEnergy / count)
```