ADDIS ABABA UNIVERSITY

College of Natural and Computational Sciences

Control Message Evaluation Model for Routing Protocol Low power lossy Network to Determine DIO Int Min value

Senait Desalegn Merne

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

December 2019

Addis Ababa University

College of Natural and Computational Sciences

Senait Desalegn Merne

Adivisor: Dagmawi Lemma (PhD)

This is to certify that the thesis prepared by Senait Desalegn, titled: *Control message evaluation model for Routing Protocol Low power lossy network* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	Signature	Date
Advisor:			
Examiner:			
Examiner:			

Abstract

Routing protocol has a duty for forwarding traffic and responsible for routing decisions. If the routing choices are not smart, more re-transmissions will occur throughout the network. Trickle algorithm uses default Destination Information Object Interval Minimum (DIO Int Min) value so it requires further analysis to determine the interval of sending message in Routing Protocol for Low power Lossy Network (RPL).

We experimented control messages transmission interval time by using DIO Int Min value which is one of trickle algorithm parameter that is used to determine the transmission interval time of control message. In this experimental evaluation we use Contiki (OS) and Cooja simulator and then try to suggest DIO Int Min values in terms of energy consumption, Packet Delivery Ratio, latency and control traffic overhead. We used 20, 50 and 80 motes in random topology. The results of DIO Int Min values in trickle algorithm will help to provide better Energy Consumption, decreased Control Traffic overhead, lower Latency and high PDR which helps in general to increase the performance of RPL in Low-Power Lossy Network (LLN).

We have done our experiment with in depth investigation of RPL control messages by considering the four performance metrics. From this study we get good understanding into different RPL settings and suitable for many application areas

Keywords: RPL, Cooja, performance metric, parameters, Low Power and Lossy Network

Dedication

To my families

Acknowledgments

Frist of all I would like to thank the almighty God for helping through all challenges .As well as thanks to my advisor, Dr. Dagmawi Lemma, for his day to day encouraging, support, guidance in carrying out this study. I would like to express gratitude and indebtedness for his valuable advice and guidance.

My deepest gratitude goes to my families I am grateful for the love, care and lessons they gave me in my life. Without their assistance, tolerance, understanding, sacrifice, kindness, unremitting support and inspiration, none of these would have happened. I will be grateful for their love forever.

Table of Contents

List of Tables

List of Figure

Acronyms and Abbreviations

BLIP	Berkeley Low-Power IP
CMEM	Control Message Evaluation Model
CTP	Collection Tree Protocol
DAG	Directed Acyclic Graph
DAO	Destination Advertisnment Object
DIO	Destination Information Object
DIS	Destination Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graph
DV	Distance Vector
DIO Int Min	Destination Information Object Interval Minimum
ETX	Expected Transmission Count
ICMPv6	Internet Control Message Protocol Version 6
IEEE	Institute of Electrical and Electronics Engineering
IETF	Internet Enginerring Task Force
IoT	Internet of Things
IPv6	Internet Protocol version 6
LBR	LLN Boarder Router
LLN	Low-Power Lossy Network
LPM	Low Power Mode
MAC	Medium Access Control
MoP	Mode of Operation
MRHOF	Minimum Rank with Hystersis Objective Function
OCP	Objective Code Point
OF	Objective Function

PDR	Packet Delivery Ratio
PER	Packet Error Rate
QoS	Quality of Service
RPL	Routing Protocol for Low-Power Lossy Network
ROLL	Routing Over Low-Power Lossy Network
RS	Router Solicitation
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WSN	Wireless Sensor Network

Chapter 1: Introduction

1.1 Background

Energy efficient technologies are becoming pertinent solutions in most area of computing, especially in areas where devices are remotely deployed and unable to have a sustainable power source. For example, Internet of Things (IoT) devices often battery-powered hence efficient use of the limited energy is important. As a result, the use of low-power wireless devices is becoming part of the design consideration.

Low-power and lossy networks (LLNs) are made up of many embedded devices, such IoT devices, with limited power, memory, and processing capability while being low cost [1].

IoT devices require energy mainly for two purposes: for internal operation as electronic device and for communication. In the latter case, the devices use significant amount of energy for routing and message relaying in the network.

For example, the devices use power for routing, which is for selecting the path for traffic in their network to communicate their neighboring devices and across multiple networks. This is especially the basic process in IP networks and it's the most important factor influencing interconnection between devices and performance of information exchange. Regarding the power conservation, the routing protocol and its quality in implementation improve the performance of the LLN [1].

The Routing Protocol for Low-Power and Lossy Networks (RPL) is meant to make the IoT into reality by bringing solution for power conservation. This protocol can quickly create network routes, share routing knowledge and adopt the topology in an efficient way [2].

The performance of RPL need to be measured to determine routing quality for LLN which helps the process of measuring and optimizing the service quality of LLN by considering different performance metric such as

- latency:- the time spent on the transmission of a message between the source and destination.
- packet Delivery Ratio (PDR): which is the quantitative relation between received packet and sent packet in the pair of nodes.

- energy utilization: which refers to a measure of energy consumed to transfer message from source towards the destination.
- Control traffic overhead:- refers to bandwidth utilization due to control packets mainly depends on topology and data traffic.

In RPL, routing from a node towards the sink is established by forwarding control messages. These messages are disseminated over the dynamically formed network topology called Destination Oriented Directed Acyclic Graph (DODAG) by set of Control messages [1]. Here, the main idea is building a non-looping table where each sensor is able to communicate with at least one node and to create a path in which each node is reachable the following control messages are used [1]:

- DODAG Information Object (DIO): is a multicast downward message initiated by a node that lets other nodes to know about its existence while inviting other nodes which are interested to join the network.
- DODAG Information Solicitation (DIS): message on the other hand is made when no announcement is heard, while the node wants to join a DODAG it sends a control message, for that it wants to know if any DODAG exists. So the message which it sends is mainly to request of find if there any DODAG.
- DODAG Advertisement Object (DAO): a request sends by a child to sink these messages responses to allow the child to join to a DODAG. Its' response can either be a Yes or No [1].

The DIO message defines and maintains upward routes containing information about the rank, the objective function, the node ID and so on.On the other hand, the DAO message advertises prefix reach-ability towards the client nodes of a DODAG to enable downward traffic.

The DIS message is proactively solicit the DODAG related information from neighboring nodes. The key role in RPL is played by the sink node which acts as a bridge between the local sensor network and the Internet. Information captured by sensors is delivered to the sink node [8].

Figure 1.1 from [1] illustrates a simple example of DODAG building process. Among the three client nodes, node 3 is beyond the radio range of the sink node. The sink node starts building a network topology by broadcasting a DIO control message with its rank and ID to client nodes. In Figure 1.1, client node 1 and node 2 are to represent nodes within the radio range of the sink. Such nodes respond with DAO messages to the sink for joining the DODAG. Client node 3, on the other hand represents a node that cannot hear from the sink node due to its location at a distant. Such node starts to send DIS message proactively to solicit DIO from neighbor nodes within some interval predefined time.

Suppose client node 2 receives the DIS message, it will then forward the DIO message received earlier to client node 3. Up on receiving this DIO message, client node 3 sends back a DAO message to client node 2, which will be forwarded to the sink node. In this way, client node 3 joins the DODAG, completing the construction process [1].

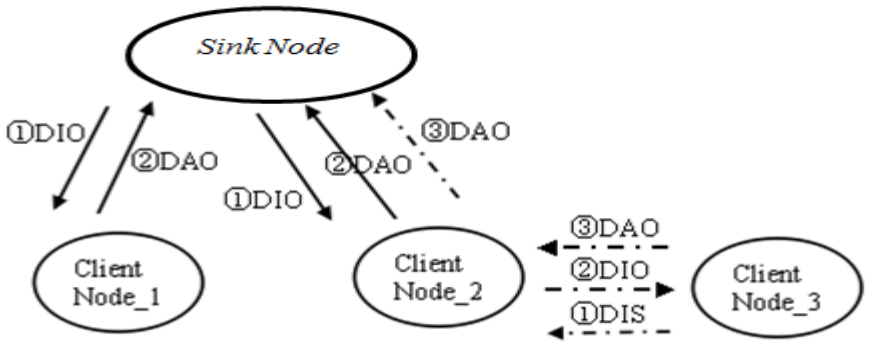


Figure 1.1: DODAG construction

A sensor node, also known as a mote, is a node in a sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected motes in the network. We use the term mote in this study since, our experiment is done with Contiki Operating System (OS) and Cooja simulator.

1.2 Motivation

The IoT has become an alternative for embedded systems in providing connectivity for less resourceful devices and it foresees a future in which information systems will be seamlessly integrated with smart objects. In due course, IoT applications are expected to penetrate our daily lives in areas as diverse as e-health (e.g., remote patient monitoring), smart home (e.g., smart lighting and heating), and smart city (e.g., smart traffic and parking applications) [2].

As the result, there is an exponential growth in the number of smart devices interconnected through mobile Internet. The devices are embedded with intelligence due to the rapid developments in sensors and other processing hardware technology.

In 2050, it is expected to have 50 billion devices Connected to the Internet [5]. Therefore, the original design and architecture of the Internet we know may not be able to satisfy the requirements of IoT. Hence, it is important to find solution at the various layers of the Internet protocol, so as to address special requirements coming along with the technology.

IoT aims to connect a number of devices over various networks, including IPv6 network [2]. Routing is one of the basic processes of an IPv6 network. However, routing demands power and processing resources, which can be an issue for low-resourced devices such as IoT devices and to alleviate the problem RPL is provided.

RPL offers Routing State Propagation, Spatial diversity and Expressive link and mote metrics. The performance of RPL has to be evaluated with respect to different performance metrics to observe the behavior of the network performance [4]. The three types of control messages (DAO, DIO and DIS) used in RPL specification are important to have communication between motes but at the same time they may have effect on RPL performance. How these control messages affect the performance of RPL motivates to work on RPL. Especially, this work is motivated based on recommended future works in [2], which suggested that there is a gap on the effect of control message on RPL performance.

Furthermore, the performance of RPL together with efficient and effective power utilization is believed to be a factor in determining the success of IoT in general.

1.3 Statement of the Problem

RPL performance determines the routing quality in LLN. According to S.Umamaheswari and A. Negi [2] the performance metrics for evaluating RPL include Packet Delivery Ratio, Control traffic Overhead, Energy consumption and Latency.

These performance metrics have been evaluated by considering different network size from different perspective such as number of motes, control message sending interval time, transmission range, motes arrangement and distance between motes. However, the effect of DIO messages sending interval time on RPL performance is suggested for further analysis [2]. It has to be noted that the DIO messages sending interval time shall be selected with care. For IoT, using smaller interval time means consuming more power though the response time can be improved. Alternatively, power can be efficiently utilized by using larger values for the message sending interval time. This could be ideal to save power since the mote needs to wake up once in a while to send the DIO message, however, this could result higher latency.

While [2] applied simulation tool for the RPL performance measurement, the tool uses trickle algorithm in the evaluation. Trickle algorithm controls the amount of routing traffic in the form of DIO's that enter to the network. It also controls the amount of time that a mote listens for new information and how often it sends out its current information to its neighbor motes [4].

Trickle algorithm uses DIO Interval Minimum (DIO Int Min) value, which is transmission interval time of control message .Yet the effect of DIO message requires further analysis [2] and it is important to determine the DIO Int Min value based on experimental value so as to define the optimal message sending interval time in relation to RPL performance.

Therefore, it is important to create a control message evaluation model to experimentally evaluate the RPL performance measurement as well as to determine the DIO Int Min value, which is dominant in the control message to determine the performance of RPL [3].

1.4 Objectives

General Objective

The general objective of this thesis is to develop a model for analyzing the effect of control message on RPL performance by using DIO Int Min from Trickle algorithm.

Specific Objectives

The following specific objectives will help to achieve the general objective stated above

- Carry out background study and literature review about the RPL protocol and routing metrics.
- Analyze the behavior of RPL in COOJA simulator.
- Evaluating the performance of RPL with respect to control traffic overhead, energy consumption, Latency and PDR.
- Identifying the impact trickle algorithm Parameters namely DIO Int Min on RPL performance.
- Measuring the results of the performance through simulation and analyzing their implication in the performance of RPL.

1.5 Methods

In this thesis experimental research is conducted with the goal of proposing a model that will help to understand the effect of control messages on RPL performance. The following listed methods help to accomplish our goal.

- **Literature review**

In the first phase background of study and literature review of the state of the art protocols and operating system for LLN and different technique available will be studied. Also, research papers on LLN, RPL performance, trickle algorithm of control message and others related papers will be reviewed.

- **Design and development of the model**

Based on the finding of the literature review, our model will be designed and experimentation to control message evaluation model will be done.

- **Evaluation**

In the evaluation phase several experiment will be done to calculate latency, Energy consumption, packet delivery ratio and control traffic overhead of control message.

- **Research Tools**

This research will be done by using Contiki operating system. It consists of the kernel, libraries, the program loader, and a set of processes. It is used in networked embedded systems and smart objects and its simulator COOJA is selected for studying the RPL protocol.

It is a flexible simulator designed for simulating networks of sensors running the Contiki operating system [8]. For constructing diagrams, tools like Microsoft Visio builder will be used.

1.6 Scope and Limitations

This research work will focus on the effect of control messages on RPL performance metrics by using trickle algorithm that will help to compute Energy consumption, PDR, latency and control traffic overhead and suggest a solution by using a model. Because of time limitation this work will not include performance evaluation of RPL by considering the distance between motes to the sink (source).

1.7 Application of Results

The use of low-power wireless devices is becoming part of our daily life. It has many application areas like smart grid, industrial automation, wireless sensor network, intelligent transport system, home automation, security, etc. The devices in these sensor networks are resource constrained because they are small, low-power, and low cost so routing is an

important factor influencing interconnection between devices and performance of information exchange [1].

The RPL will make the IoT into reality and offers Routing State Propagation, Spatial diversity and Expressive link and mote metrics [2]. The performance of RPL is affected by different performance metric and one of them is control messages. The effect of these control messages impacts interconnection between devices and performance of information exchange [8]. In other words the performance of RPL affects the communication between IoT devices so by increasing the effectiveness of RPL and by reducing the impacts of different performance metric we can make IoT into reality as it is expected.

1.8 Organization of the Rest of the Thesis

The rest of this thesis work is organized as follows: Chapter 2 discusses literature review. In Chapter 3, the details of various approaches that are related to our work will be discussed. Design of control message evaluation model to determine DIO Int Min value presented in Chapter 4. The experiment and evaluation conducted on the model are discussed in Chapter 5. Finally, chapter 6 is conclusion.

Chapter 2: Literature Review

2.1 Overview of RPL Protocol

According to [9] Internet Engineering Task Force (IETF) chartered the routing over low-power and lossy networks (RoLL) working group in 2008 to standardize a practical IPv6 routing protocol. The main characteristics of LLN are discussed as follows

• LLN comprises thousands of constrained motes that have limited processing power, memory, and sometimes energy (when they are battery operated).

• These constrained motes are interconnected by lossy links that are usually unstable and typically support only low data rates [9].

After additional 3 years efforts, in 2012, RoLL finalized RPL standardization to fulfill the aforementioned requirements. IoT consists in connecting static or mobile objects and devices equipped with communication, sensors, and actuators modules through Internet [11]. The data transportation and routing with quality of service (QoS) and security are key issues in IoT. RPL is an open routing protocol standardized by the IETF ROLL working group in 2008 [10].

Routing protocols are designed with the goal of enabling effective communication between different motes. So, different features and applications of the target network should be taken into consideration while designing the routing protocol [9]. RPL was designed for LLNs these networks use devices with scarce resources (limited energy power, limited computational power, limited memory, etc.) and are subject to high Packet Error Rates (PER).

RPL is a Distance-Vector (DV) and a source routing protocol which operates at the IP level. Hence, RPL can operate over different types of link layers such as Institute of Electrical Electronics Engineering (IEEE) 802.15.4 PHY and MAC layers [10]. RPL make a tree like topology also called Destination Acyclic Graph (DAG). Each mote in a RPL network has a

preferred parent which acts as a gateway for that mote. If a mote does not have an entry in its routing table for a packet, the mote simply forwards it to its preferred parent and so on until it either reaches the destination or a common parent which further forwards it down the tree towards the destination [14].

Based on [16] RPL is distance vector based on IPv6 routing protocol for LLNs that specifies how to construct a DODAG using an Objective Function (OF) which helps in selection of the route towards the sink and a set of metrics.

LLNs do not have a predefined topology. It is thus the job of RPL to find links and choose peers. To do so, RPL organizes the network as a DAG with the sink located at the root and uses an OF to minimize the cost of reaching the root from any mote in the network. Furthermore, the selected metrics should have the capacity for combining different routing metrics together [9].

RPL the IPv6 routing protocol for LLNs was designed to be suitable for resource-constrained devices in industrial, home, and urban environments [9]. The main goal of RPL is to provide IPv6 connectivity to a large number of battery-operated embedded wireless devices that use low-power radios to communicate and deliver their data over multiple hops. Specifically, RPL was designed to meet the requirements of several applications in the WSN and IoT domain and is considered a critical component that links the low-power network connectivity to application layers in the IETF protocol suite for LLNs [10].

2.2 RPL Control Messages

RPL control messages are new ICMPv6 (Internet Control Message Protocol version 6) control messages. They start with an ICMPv6 header followed by a message body and possibly followed by some options. The header contains a type field, a code field and checksum [13].

RPL can build DAGs based on selected routing metrics and constraints. This DAG structure is adopted to efficiently support upstream dominant traffic patterns with resource constrained motes. The basis of RPL is to build a routing topology, called DODAG rooted at

one or more LLN border router and support bi directional IPv6 communication between network devices [10].

RPL control messages can briefly be described as follows

- DIS Control Message: It is sent from a requesting mote to other motes to solicit them to send to the requesting mote a DIO message, so the requesting mote can explore its neighborhood for nearby DODAGs [13].
- DIO Control Message: It plays the basic role by helping motes in discovering different RPL Instances with their configuration parameters and in constructing a DODAG. It can be sent as a reply to DIS or triggered by the root when constructing a DODAG. It contains information about the RPL InstanceID, DODAGID, DODAG Version Number, the Rank, and the objective code point (OCP) which identifies the objective function the DODAG is using and so on [13].
- DAO Control Message: it is used to propagate destination information to trace visited motes upward along the DODAG to the root [13].

Figure 2.1[11] illustrates RPL's control messages and parent selection process with the simplest network structure, a single DODAG in the forest. Each mote in RPL advertises routing metrics and constraints through DIO messages. Upon receiving DIO messages from its neighbors a mote chooses routing parents according to its OF and routing information in DIO messages (e.g., RANK, DODAG ID), and then constructs a routing topology (i.e., DODAG) [11].Note that a RPL mote can have multiple parent motes to achieve reliable packet delivery through path diversity. DIO messages are transmitted based on the Trickle Timer [12] to achieve a balance between control overhead (energy consumption) and fast convergence/recovery, and minimize parameter configurations. DIO messages are also transmitted upon request when a DODAG information solicitation (DIS) message is received.

RANK is defined and used by the OF to represent the routing distance from a mote to LLN Boarder Router (LBR)and link mote metrics (e.g., Expected Transmission Count (ETX)) are used for RANK calculation and parent selection [11].

Once each mote selects routes towards an LBR, RPL uses DAO messages for reverse route construction, which advertise routing information on how other motes can reach various destinations and prefixes within a RPL network when traveling down the RPL DODAG. How a DAO message is processed by each mote and the LBR depends on which mode-of-operation (MOP) is used for downward routing: 'storing mode' (table-driven routing) or non-storing mode' (source routing). In 'storing mode', each mote stores the downward routing information for all descendant motes in its sub tree, where as in 'non-storing mode', only the root mote (LBR) stores that information for all motes in its network. In both cases, basic idea is for the ancestor motes to process and store the information in DAO messages to create routing entries for the motes in the sub tree [11]

Figure 2.1 [11] shows a mote that does not have a route (e.g., newly joined mote) may use the DIS message to solicit a DIO from RPL mote. Its use is analogous to that of a Router Solicitation (RS) as specified in IPv6 Neighbor Discovery; a mote may use DIS to probe its neighborhood for nearby DODAGs.

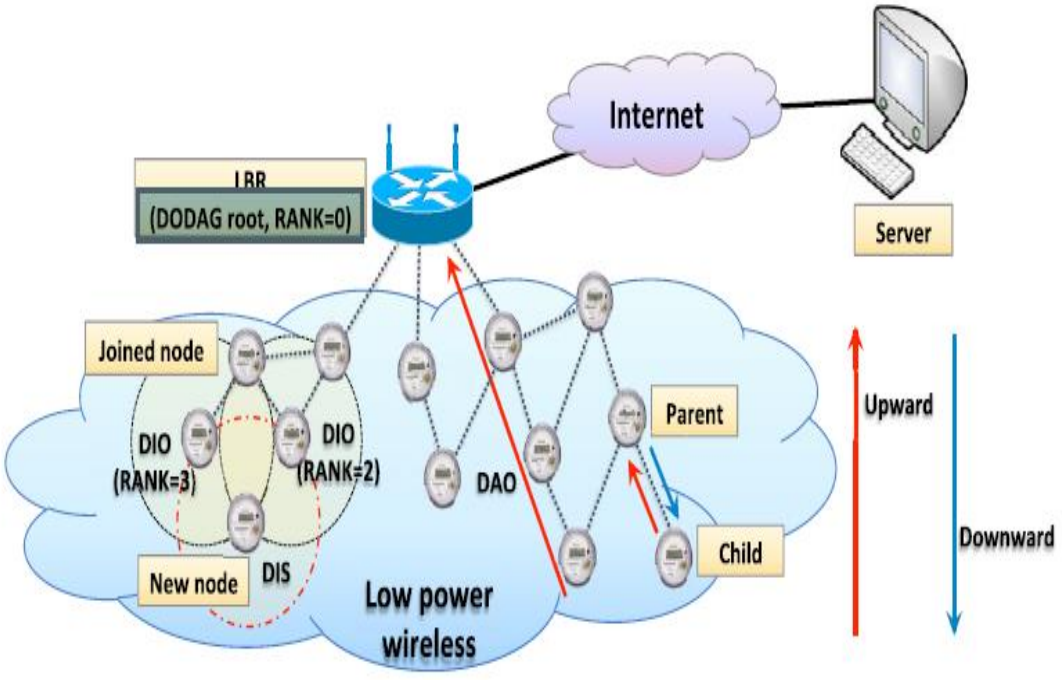


Figure 2.1: Operation of RPL with a single DODAG and a single RPL instance

2.3 Factors that make IoT resource constraint

The appearance of cheap embedded computing devices capable of wireless communications is leading to the emergence of an IoT. The ability to network embedded devices opens up opportunities to develop new applications. From home automation to an energy balanced smart electricity grid, IoT is expected to introduce new computing services that integrate existing software services already available on the Internet with the control and data gathering capabilities of embedded devices [16].

IoT devices are likely to be deployed in large numbers in highly competitive markets. Technology improvements will most likely be used to make embedded devices cheaper, smaller and more energy efficient but not necessarily more powerful. Typical embedded IoT devices are equipped with 8- or 16-bit microcontrollers that possess very little RAM and storage capacities. Resource constrained devices are often equipped with an IEEE 802.15.4 radio, which enables low-power low-data-rate wireless personal area networks (WPANs) with data rates of 20-250 kbps and frame sizes of up to 127 octets [14].

The IoT device is constrained by the entity which is in physically monitoring state and the position of the entity is frequently changing without access to power [15].

Most of the devices in the IoT are having considerable small size and are not fixed. Due to their size and frequently changing location property devices are not able to access the power all the time. So the low energy consumption is universal constraint of the IoT. Either they use battery technologies or they can use some techniques for taking power from their environment using other devices. Therefore, there is a need of design in which such energy consumption techniques or low energy consumption schemes are made with long lasting life of devices [15].

The other issue addressed in [15] is the design issue to be addressed as a requirement of such modular approach that subtracts the need to make a separate chip for each and every application because it is not feasible to create a chip for each and every application. Such high modular approach will combine existing chips within size and power constraint

2.4 RPL Performance Metrics

According to [1] WSN performance is the quality of services given by sensor network in general. Some of the performance metrics evaluated in previous studies include signaling overhead, latency, energy consumption and so on, which are vital to the overall performance of a wireless sensor network. Their work is the first effort spanning across the whole life cycle of wireless sensor networks, including both the network construction process and the functioning stage.

2.4.1 The analysis of latency

Latency is the time interval between a state message sent by one mote and received by another one. A number of factors have an impact on the latency and convergence time, including the time spent on packet processing Low-Power Wireless Personal Area Networks (LoWPAN), ContikiMAC protocol, Radio model which are underlying modules of Contiki, radio duty cycle, network density and the number of DODAGs etc [15].

Contiki MAC provides an asynchronous, sender initiated radio duty cycle mechanism. The feature of radio duty cycle is that, a packet is sent repeatedly until it gets an acknowledgment from the receiver. The receiver periodically wakes up to listen on packet transmissions from neighbors [13]. Once a packet transmission starts, the receiver keeps its radio on until the transmission completes [1].

According to the authors in [1] a potential concern of this mechanism is the delay and energy consumption, as the the sender keeps sending packets continuously until the receiver wakes up and responds to the sender. On the other hand, if we want to save power of the sender and shorten the time of packet transmission by letting the receiver wake up frequently, we may end up wasting energy at the receiver side when it needlessly wakes up without data to receive. To help make a reasonable trade-off, it is important to study the relationship between the latency and the load of packet transmission in the WSN.

Figure 2.3 [1] presents results on this relationship. We can see that the latency for transmitting a single packet does not increase with the overall load of packets. The reason is that, with less packets, the motes often go to radio-off state for saving energy, and it will

take extra timing overhead switching back to radio-on state for packet transmission. Whereas with higher load of packets, the timing overhead on mote competing for each packet is alleviated by less mote state switching time [1].

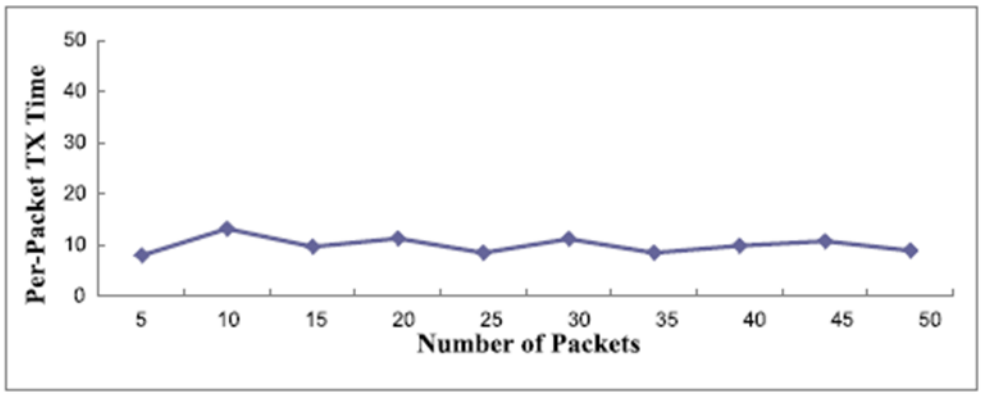


Figure 2.2: Relationship between latency and number of packets

2.4.2 The analysis of signaling overhead

From the background section [1] we get an idea that the control messages transmitted for DODAG construction may flood the network. This affects the RPL performance and the construction stage may take a long time. To validate this assumption and to provide guidelines for potential optimization of this process, the authors perform a quantitative analysis on this signaling overhead. Figure 2.3 [1] shows the statistics on control messages and data messages in terms of transmitted network packets sent by WSN. Clearly, control messages dominate most of the construction procedure, whereas UDP and radio data messages are negligible compared to control messages.

The reason leading to this result is that in the RPL implemented network, the DODAG root mote starts building network topology by broadcasting RPL control messages to all motes in its radio range, and a large number of messages responding this broadcast are generated and forwarded to the root mote. On the other hand, UDP messages are sent as a confirmation only upon the establishment of a link path to the DODAG root mote as discussed in the literature [1].

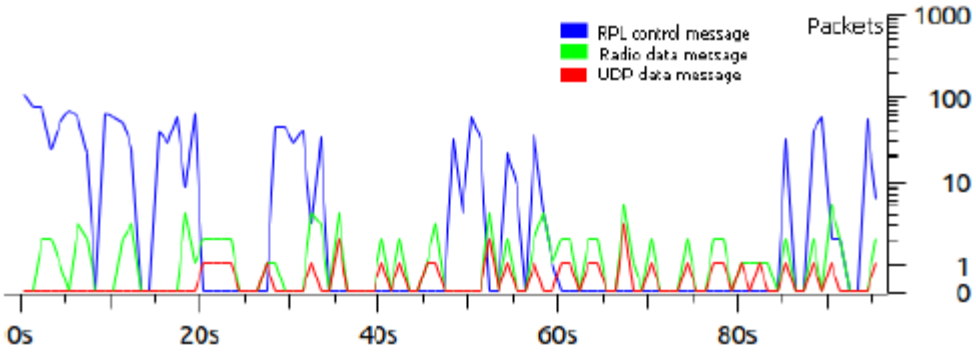


Figure 2.3: Control message and data message

The radio message is the IEEE 802.15.4 type message along with UDP message, due to the UDP message transmission via IEEE 802.15.4 frame data [17]. From Figure 2.3 we can see that it takes nearly 60 seconds for the control messages to disappear, which indicates the completion of the network construction stage. Note that the spikes of control messages after 80 seconds are caused by new motes joining WSN.

From above analysis, the main overhead of the WSN construction stage is RPL signaling. Another conclusion is that RPL is sensitive to motes changes (indicated by the spikes of blue lines after 80 seconds). To get more insights, they break the control messages into three groups, namely the DIO, DAO, and DIS messages. This breakdown is presented in Figure 2.4 [1].

Firstly, the scarce of DIS messages can be easy to understand, as in most cases, only client motes beyond the radio range of the root mote need to generate DIS messages asking for DIO messages from neighbor motes. For DIO and DAO messages the DAO messages are used to maintain downward traffic from the root mote to the client motes the DIO messages are for upward traffic. The DAO messages have to be forwarded up to the root mote from all client motes which are willing to join or already in the DODAG. The DIO messages are broadcasted only at one hop distance from one mote to its parent. Thus the overhead of DAO transmission is much higher than DIO transmission. To study the scalability of RPL with respect to the scale of the network, the authors evaluated the increase on the number of control messages (in terms of network packets) with the increase on the number of WSN

motes. Figure 2.5 [1] shows that the RPL protocol has a reasonably good scalability, in which the number of control messages tend to increase at a lower speed than the number of WSN motes, especially when the scale of WSN becomes large.

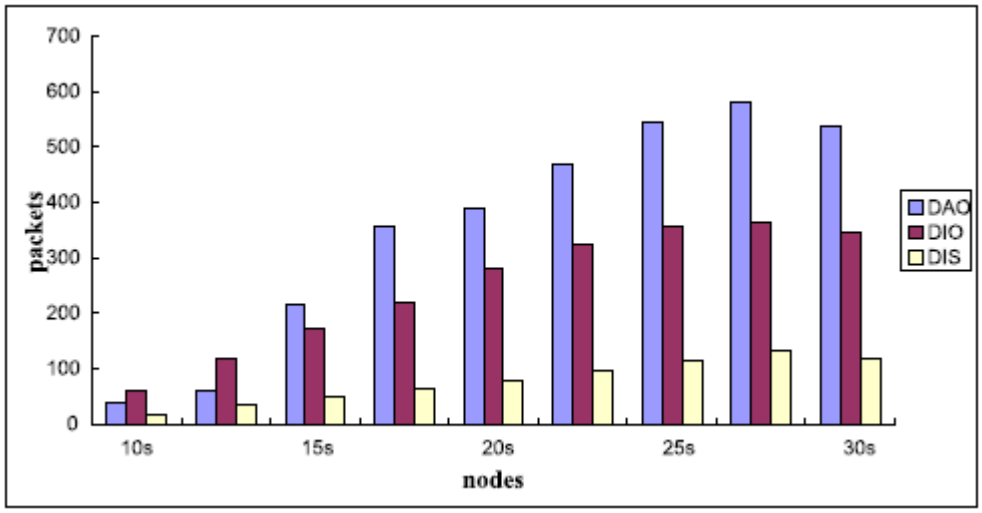


Figure 2.4: Break down of RPL control message

2.4.3 The analysis of energy consumption

Energy consumption is the first order constraints of WSN, and it is indispensable to evaluate the energy consumption of a WSN running RPL [5]. The authors in [1] conduct this evaluation work in two aspects. Firstly, they evaluate the Energy consumption of the whole network to get an overall picture on power. Then the power consumption of individual motes to get more insights. The difference of energy consumptions between client motes and sink motes is also evaluated and analyzed.

Earlier research suggests that radio transceiver is the major source of power consumption in a WSN mote. For example, the Energy consumption by the radio is three orders of magnitude larger than that of the CPU for the Tmote Sky platform [18]. Because of this, they focus on radio transmission to measure the Energy consumption. As mentioned above, Contiki MAC is the default radio duty cycle protocol in the contiki operating system and the majority of energy is spent on idle listening, repeated packet sending and reception.

Therefore, the authors use the time of radio on as an equivalent metric of Energy power consumption instead of using absolute Energy consumption in joules.

In [1] the overall usage of radio in the network is 80.28%. Which means the radio keeps on most of time, thus the Energy consumption is significant during this period. This suggests that a protocol revision that reduces the time of active radio would make an effective reduction on power consumption in this period.

2.5 Control messages on RPL topology

2.5.1 Message Structure of DIO

According to [5] DIO message is the most fundamental control message in RPL's topology construction process. Figure 2.5 [5] outlines its structure.

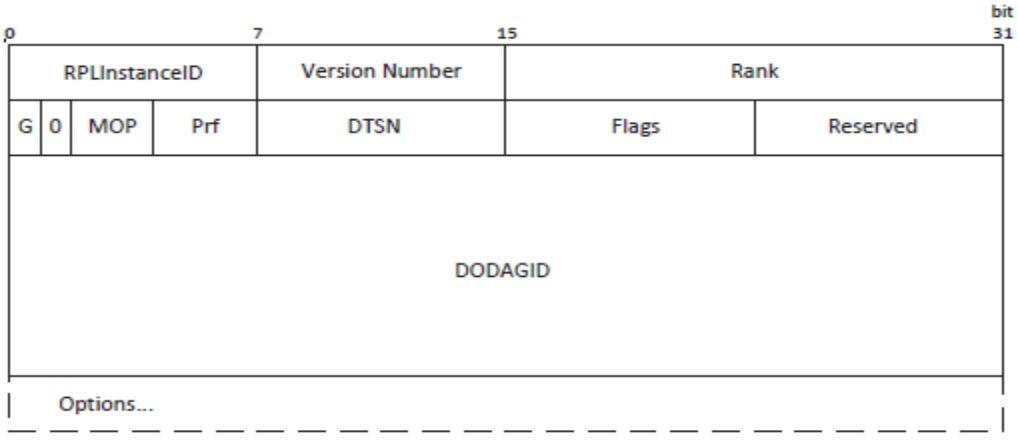


Figure 2.5: DIO message format

DIO carries information that allows a mote to discover RPL instance, learn its configuration parameters, select a DODAG parent set, and maintain the DODAG [5]. The main fields of DIO message include Version Number, DODAGID and RPL Instance ID. Version Number is used within a DODAG, DODAGID is used within a RPL Instance. The 'G' flag indicates whether the DODAG is grounded or floating.

DODAG Preference is used when there exist multiple DODAG and defines the preferable root. Rank indicates the DODAG rank of the mote when sending the DIO. In [5]

experiments, the authors defined the DIO in an independent structure. The basic fields of DIO format are included in the DIO structure definition. Some fields, such as Flags and Reserved, are not included in their definition since they are not used in their simulation.

In [5] experiment the routing metric data is also declared in the DIO structure in their simulation. The DODAG Configuration Option is an important option in DIO structure. It is used to report the configuration information through the DODAG. Therefore, it contains the trickle timer configuration parameters of message mechanism, including DIOIntMin (Imin), DIOIntDoubl (Imax) and DIORedun (k). The rank calculation parameters, including MaxRankIncrease and MinHopIncrease, are also contained in this option. Generally, this option is usually configured and only allowed to be modified by DODAG root. Unlike the main fields in DIO format which can be changed in the simulation, DODAG Configuration Option is kept unchanged through the simulation.

2.5.2 Message structure of DAO

Figure 2.6 [9] represents the structure of DAO message. Similar to the DIO message, the DAO message includes an RPL Instance ID. This is the same one with the mote that learned from a received DIO. The next field is the Flags field where only the first two bits are used. The first one is the'K' flag which indicates whether the sender of the DAO expects to receive a DAO-ACK in response. The second one is the 'D' flag which indicates if the DODAGID field is present. Due to the fact that the DODAGID field represents the IPv6 address of the root it may be omitted if there is only one root mote present.

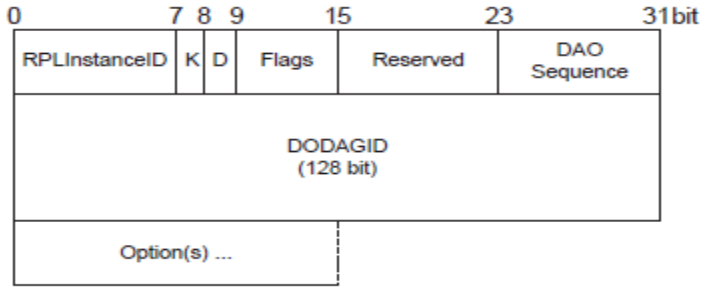


Figure 2.6: DAO message format

Chapter 3: Related work

A large number of researchers strongly reveal that RPL is a well suited protocol for LLNs and the performance metrics for evaluating RPL were suggested. For example, [18, 19] Routing in LLN is challenging not only because of lossy nature of the radio medium and the constrained resources of the sensor motes but also due to the various routing requirements and high flexibility of the RPL configurations. A variety of RPL metrics were used in the previous studies to evaluate its performance in various network scenarios and it has been observed that a number of optimizations can be added to RPL network.

The paper in [2] focuses on understanding of the architecture and protocol stack of RPL and its place in IoT. The assessment of the performance of RPL protocol is done in Cooja simulator under Contiki operating system. The performance of RPL is carried out for a Smart Health environment scenario. The Contiki MAC radio duty cycling built up with a lesser rate facilitates RPL to offer better energy efficiency and PDR and lesser control traffic overhead.
The network building procedure has been analyzed by measuring the metrics like signaling overhead, PDR, latency and energy utilization. The experimental results reveal that RPL is a well suited protocol for LLNs.

The authors in [1] evaluate and analyze the performance of the IETF RPL routing protocol using COOJA simulator under Contiki operating system for WSN. As they mentioned it is the first effort covering the whole life cycle of RPL enabled WSN. Analyzes the performance of network construction process by measuring several important metrics, and then they investigate the performance and possible situations during the functioning stage of the WSN. The results indicate that RPL is a pretty robust protocol for WSN. But there are still several aspects to be improved, like signaling overhead, latency and so on. In addition, their work provides guidelines for the design of future Internet of Things with IPv6 networking enabled.

The authors in [19] proposed a modified RPL routing protocol for IoT. As they mentioned RPL protocol may suffer from unreliability problem in some conditions specially because its

lack of enough knowledge of the quality of links. They have shown it via simulation using Tinyoperating system by means of a new link parameter instead of number of retransmission to calculate ETX improves the performance by selecting the links with higher reliability. Finally by modifying the calculation of ETX as a typical objective function in RPL protocol they can improve the performance of a defined network in terms of PDR, number of re-transmission, throughput, and end to end delay. As a future work, they will consider mote metrics in addition to link metrics as routing metrics for rank computation in RPL protocol in order to achieve the minimum power consumption in a specific situation that motes have energy constraints.

3.1 RPL Design Over view

The paper [6] presents the IPv6 Routing Protocol, which has been designed to overcome routing issues in LLNs. It implements measures to reduce energy consumption such as dynamic sending rate of control messages and addressing topology inconsistencies only when data packets have to be sent. The protocol makes use of IPv6 and supports not only traffic in the upward direction, but also traffic flowing from a gateway mote to all other network participants. The paper focuses on the employment of RPL in WSNs which gives a brief overview of the protocol's performance and the author mentioned the performance of RPL with respect to control messages but still didn't mentioned the effect of control message on RPL performance.

Another important fact that is addressed by [24] J. P. Vasseur group is about the protocol's design. They describe it as maintenance of the topology. Since most of devices in a LLN are typically battery powered, it is crucial to limit the amount of sent control messages over the network. Many routing protocols broadcast control packets at a fixed time interval which causes energy to be wasted when the network is in a stable condition. Thus, RPL adapts the sending rate of DIO messages by extending the Trickle algorithm [9]. Finally the mentioned authors in [24] conclude that in a network with stable links the control messages will be rare whereas an environment in which the topology changes frequently will cause RPL to send control information more often. Their experiment is done in Contiki operating system and Cooja simulator.

3.2 RPL performance evaluation

Gnawali et al. [21] presented a RPL implementation, called Tiny RPL. In their study they use the Berkeley Low-power IP (BLIP) stack in Tiny OS 2.x which interrelates with their protocol implementation. More concretely, the control plane of Tiny RPL communicates with the BLIP stack which gives a forwarding plane implementation. In this way, during one test run numerous transport protocol settings can be assessed and compared. In addition to this Tiny RPL is further compared with the Collection Tree Protocol (CTP) [21], it is de-facto routing protocol standard for Tiny OS.

Also in [22] a 51mote TelosB testbed setup is constructed where only one mote is acting as a root. Overall, they use two network setting one that has a data packet group interval of 5 seconds and another that has a data packet time interval of 10 seconds. For each of them the protocols are verified against each other and an estimation of the packet reception ratio and the average number of control packets is made.

Each test bed run lasts 24 hours. Since CTP uses ETX for topology maintenance, the OF of TinyRPL is set to use the similar method for metric calculation. In order to make a fair assessment the downstream routing options are disabled since the standard CTP configuration does not define a mechanism similar to the one realized with control messages.

In [21, 22] both experiments runs to achieve a reception rate of approximately 99 %. The amount of control traffic reported also stays at the same level since both protocols make use of the trickle timer and are evaluated in static scenarios. When we generalizes their study it is focus on modification of RPL protocol and main trouble in these work are the researchers didn’t clearly evaluate RPL with respect to control message the main problem of RPL protocol in LLN is performance one criteria to increase the performance of RPL is to limit the interval of control message sending time but the authors didn’t include control messages in their modified RPL protocol.

The authors in [20] investigate OF and the most influential parameters on RPL performance in Sparse Wireless Sensor Network in the Cooja simulator of Instant Contiki and evaluated RPL performance in terms of Energy, Packet Delivery Ratio, Average Transmitted and Received Power and Convergence Time for the Sparse Wireless Sensor Network**.** As they run and studied the simulation in sparse formation i.e. the number of links are less than the maximum nos. of links they can make. It is proposed to reduce the power consumption in Wireless Sensor Networks. But the main limitation in their work is the convergence time for only 25 motes is approximately 14 seconds which is much higher as compare to the convergence time for 80 mote network and that was approx. 15-16 seconds with DIO minimum.

The authors in [14] run several experiments using Contiki operating system and Cooja simulator by varying a smaller and larger number of motes. Their objective was to overviews the most used objective functions then proposes a modification on the Minimum Rank with Hysteresis Objective Function (MRHOF) which takes into consideration two metrics instead of one, to get more reliable and optimized routes. The obtained result shows while measuring packet delivery ratio, average power consumption, and the churn respectively while changing the values of Reception Success Ratio(RX%) which is 20% and 80% using random topologies of 15 and 35 motes. They stated that, in general, PDR decreases as the number of the motes in the network increases reaching as low as 87% with 35 motes while using the MRHOF with the Energy metric. Moreover, their experiment indicates that PDR increases as the values of RX increases while the number of motes was 15, and even when the RX% is set to 20%, which is considered very low, the PDR remained higher than 99.5% for all cases and reached 100% when RX was set to 40% and higher. Also, the power consumption of the network in general increases as they increase the number of motes.

To finalize their work an Objective Function that minimizes one metric only is not sufficient and an Objective Function that takes into consideration more than one metric will be more efficient and effective. The short coming in their experiment was they only take 15 and 35 no. of motes but what about dense network they should have to say something.

Shimaa et al. [3] proposed a real and simulated implementation of RPL behavior with proper modifications .their objective was to support the Advanced Metering Infrastructure (AMI) based WSN routing requirements. They evaluate RPL performance using 140 motes from the wireless sensor testbed (IoT-LAB) and 1000 motes using Cooja simulator measure RPL performance within medium and high-density networks.

They adopted two routing metrics for path selection the First one was HOP Count (HC) and the second was ETX to evaluate RPL performance in terms of packet delivery ratio, network latency, control traffic overhead and power consumption. Their results illustrate that routes with ETX calculations in low and medium network densities outperform routes using HC and the performance decreases as the network becomes dense. However, Cooja implementation results provides an average reasonable performance for AMI with high-density networks still many RPL motes suffering from high packet loss rates, network congestion and many retransmissions due to the selection of optimal paths with highly unreliable links.

The authors in [23] focus on solving the problem of Internet IP. Then they studied the RPL Routing Protocol which is suitable for this requirement, through simulation verified it. In the simulation they used 20 motes to test, forming the DAG, RPL's control messages were verified. In order to verify the RPL they collected the DIO, DAO and DIS messages and observed its networking process in the hardware experiments. The result of their simulation was to verify the feasibility and reliability of the RPL.

J.Ko et al. In [24] proposes to investigate and evaluate the performance of two independent IPv6 implementations for LLNs, one for Contiki and one for Tiny OS. Their implementation was able to run networks with Tiny RPL and Contiki RPL They present their experiences with the Contiki and Tiny OS implementations of the IPv6 stack for low-power and lossy (LLN) networks including the IETF 6LoWPAN adaptation layer and the IETF RPL protocol. Their results show two independent implementations, that have good performance on their own, can have a substandard performance in a mixed network configuration and that small implementation differences can lead to large-scale differences in behavior. This suggests that IPv6 stack implementations for LLNs need to be tested not just for correctness

but also for performance. The main shortcoming in their work was they use the OF0 objective function, the simplest of the objective function that RPL networks should support so what about other objective functions

The authors in [5] provide an in-depth review of current research activities to achieve large scale simulation development and performance evaluation under various objective functions and routing metrics are pioneering works in RPL study. The results of their finding was the first attempt to build a large scale network using RPL and provide analytical results under different objective functions and serve as a reference for evaluating the effectiveness of routing solutions in large scale IoT. Use cases experiment the routing metric data is also declared in the DIO structure in their simulation. They make an investigation on network simulation tools and select OMNeT++ as the ideal one for a large scale RPL simulation.
In [5] the authors discuss the format of DIO in depth but didn't consider other types of control messages
The authors in [12] stated it is obvious that any routing mechanism involves significant control overhead in a network. Their objective was to develop an effective message control mechanism which is significantly important in reducing network overhead and balancing limited network resources. The result of their finding was Configuring the trickle timer with appropriate parameters to understand control messages with RPL since it will influence the network reliability and stability as well as performance Trickle timer mechanism which is mainly used by DIO, has been emphasized as an important part of message control mechanism. But their paper lacks the detail implementation of control message and the effect of DIO message requires further analysis

Chapter 4: Design of CMEM on RPL Performance

This Chapter presents the design of Control Message Evaluation Model (CMEM) on RPL performance. The design conducted in this research study involves deep analysis of the protocol behavior, analysis the effect of different parameters (latency, PDR, control traffic overhead and energy consumption) on RPL performance. It also needs a large set of data and take significant amount of time to perform the experiment and to get reliable and statistically correct result.

In section 4.1 CMEM to determine DIO Int Min value presented then in section 4.2 flow chart of the evaluation process presented.

4.1 CMEM to determine DIO Int Min value

Trickle algorithm is used to limit the number of control packets sent. The algorithm uses important parameter namely DIO Int Min. The DIO Int Min is used to determine the initial interval of the control packet transmission its default interval value in Contiki is 12.

- **DIO Int Min**

This parameter controls the rate of DIO transmission and therefore crucial for control traffic overhead, energy consumption, PDR, latency and etc. The more rapidly the DIOs are transmitted the more rapidly the network gets converged but at the expense of control traffic overhead and more energy consumption etc. This parameter is influential on the performance of the whole protocol performance. In Contiki this parameter is set by RPL_DIO_INT_MIN.

- **Imin**

The transmission of DIO is controlled by a timer called trickle timer whose minimum value is Imin. Trickle starts its first interval by setting I to a value from the range Imin, usually it sets the first interval to a value of Imin [25]. This parameter gives the minimum amount of

time between two DIOs. DIOs are transmitted periodically to reduce the redundant Control Traffic and use the limited resources more optimally. The value of Imin is determined by the RPL parameter DIO Int Min [25].

In the trickle algorithm the default DIO Int Min value is 12 (see Annex A) which is referred in [12] and shown in equation (1)

$$\begin{aligned} \text{Imin} &= 2^{\text{DIO Int Min}} \\ &= 2^{12} \text{ ms} \\ &= 4096 \text{ ms} \\ &= 4.096 \text{ s} \end{aligned} \qquad (1)$$

The value of trickle timer starts sending message with in the interval of 4.096 s.

Our model includes the equation for latency, packet delivery ratio and control traffic overhead. Let us see each of them.

To compute the total latency of motes subtract the time interval between messages sent by motes from the time interval received by another motes as given in equation (2).

$$\left(\sum_{e=1}^{n} \text{Recv DIOt}_e + \sum_{o=1}^{Z} \text{Recv DAOt}_o + \sum_{X=0}^{y} \text{Recv DISt}_x\right) - \left(\sum_{e=1}^{n} \text{SentDIOt}_e + \sum_{o=1}^{Z} \text{SentDAOt}_o + \sum_{X=0}^{y} \text{SentDISt}_x\right) \qquad (2)$$

where Recv = Receiving

t = time

e =is the number DIO message if there is a communication between motes at least one DIO message should have to transmitted and increasing up to ‘n’ number of message

n = up to ‘n’ number of DIO messages sent

z= up to ’z’ number of DAO messages sent

x= If all motes are in the radio range of the root mote there is no DIS message so the DIS message value will be x=0 but if all motes are not in the radio range of the root mote the number of DIS message increase up to ‘y’ number of messages

y= up to ’y’ number of DIS messages sent

To calculate average packet delivery ratio we compute the number of successfully received packets of the mote and divide it by the number of sent then multiply by 100 as shown in equation (3).

$$\left(\sum_{e=1}^{n} \text{RecvDIOp}_e + \sum_{o=1}^{Z} \text{RecvDAOp}_o + \sum_{X=0}^{y} \text{RecvDISp}_x\right) / \left(\sum_{e=1}^{n} \text{sentDIOp}_e + \sum_{o=1}^{Z} \text{sentDAOp}_o + \sum_{X=0}^{y} \text{sentDISt}_x\right) * 100 \quad (3)$$

where P = packets

The other variables have similar meaning as in equation (2).

To calculate the control traffic overhead every mote will print out the message DIO, DAO, DIS sent based on the kind of control message. We gather these control messages for each mote and sum up for the overall RPL network control traffic overhead see equation (4)

$$\sum_{e=1}^{n} \text{DIOp}_e + \sum_{o=1}^{Z} \text{DAOp}_o + \sum_{X=0}^{y} \text{DISp}_x \quad (4)$$

All variables have similar meaning as in equation (2) and (3).

To measure the energy consumption we use the mechanism of Powertrace system existing in Contiki. Powertrace is a system for network-level power profiling for low-power wireless networks which estimates the energy consumption for CPU processing, packet transmission and listening [26].

4.3 Flow Chart of Control Message Evaluation process

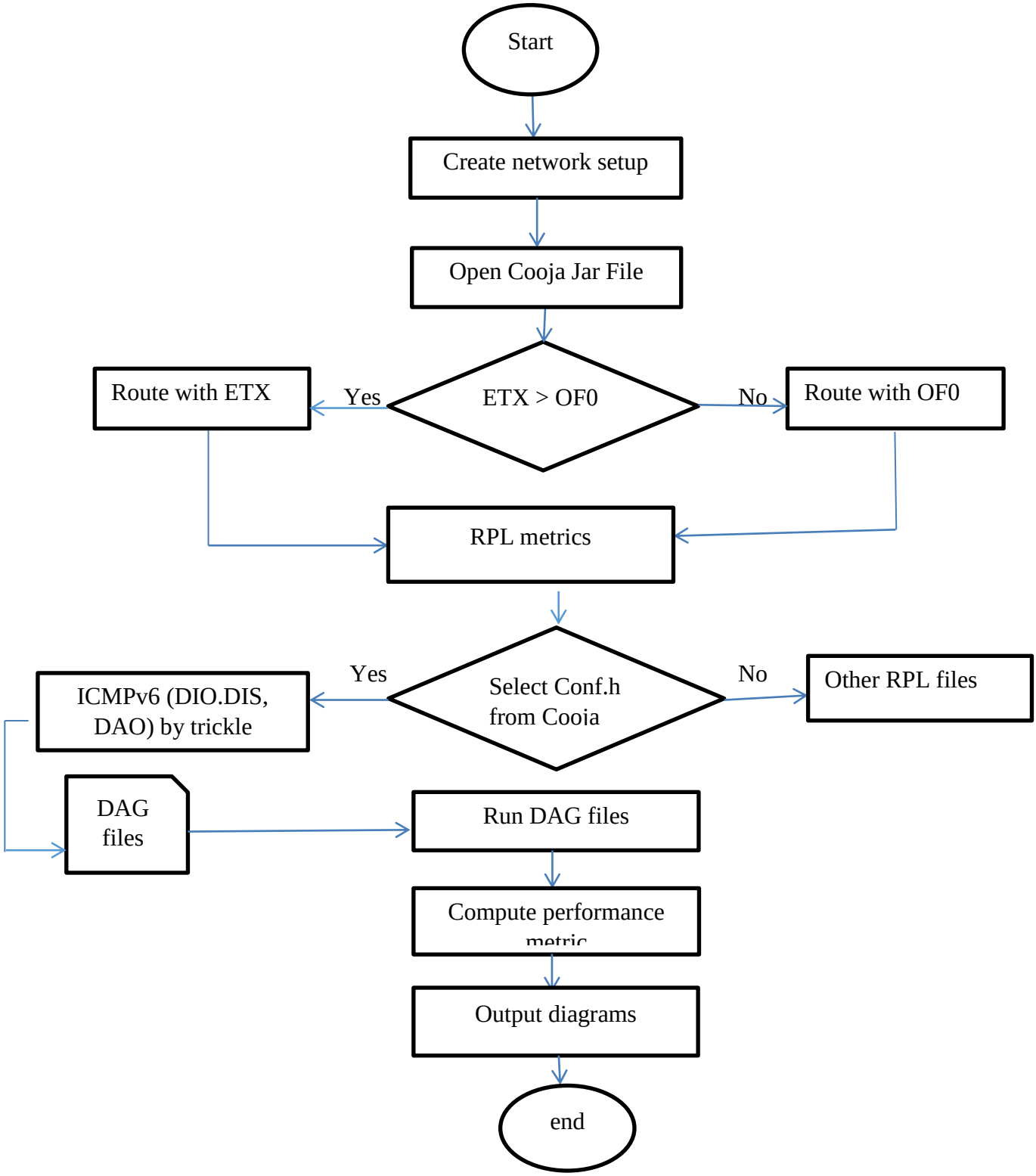


Figure 4.1: flow chart of control message evaluation process

As Figure 4.1 depicts the process starts by running Contiki OS and Cooja simulator then we will create of network setup in our case 20, 50 and 80 motes with random topology created. Then to start the communication between motes we will open the Cooja jar file which is our code located.

In Cooja all the interactions with the simulated motes are performed via plugins like Simulation Visualizer, Timeline, and Radio logger. It stores the simulation in an XML file with extension Cooja simulation configuration 'csc'. This file contains information about the simulation environment, plugins, the motes and its positions, random seed and radio medium etc. So after initial process starts the implementation COOJA simulator will begin to evaluate the behavior of the Contiki RPL then select of the objective function we can select either OF0 or ETX for this experiment we select ETX because ETX provide better route than OF0 [25].

OF0 is designed for determining the parent mote among the contending neighbors which have minimum rank with respect to the root mote. Initially the root mote is assigned with the minimum rank and this information is multicast to the other motes using control message. Now all those motes who receive the control message will make the root mote as a preferred parent and will assign themselves with a rank which is incremented by one with respect to the rank of the root mote. This process is repeated again when these motes further transmit there DIO forward to other motes in the network and ranks are incremented depending on the preferred parent. Now whenever there is more than one mote contending to be a preferred parent for a particular mote, in that case the mote with the lowest rank will be selected as the preferred parent. Therefore this Objective Functions helps in selection of the route towards the sink but ETX uses expected number of transmissions required to successfully transmit a packet on the link.

ETX provides better routes than OF0 by taking into account the link characteristics which consequently provides better Network Latency because the ETX selects the best routes which has better path ETX than the path selected by OF0. The best paths ensure less re-transmissions and radio collisions across the network and this provides better Latency, PDR and Energy consumption

After running the objective function we will get the best path to the sink then there are different RPL metrics which helps us to understand different RPL characteristics.

Our files are placed in the directory of Contiki application which is run by Cooja simulator called “project-conf.h” file which provides the ability to change RPL parameters in one place.

Then the Contiki OS provides modules for different tasks (layers). It provides the routing modules in a separate directory “contiki/core/net/rpl” and consists of a number of files. These files are separated logically based on the functionalities they provide for instance rpl-dag.c contains the functionality for Directed Acyclic Graph (DAG) formation, rpl-icmp6.c provides functionality for packaging ICMP messages etc.

Also inside of RPL, we will get a timer which uses the Trickle Timer algorithm to control the updating and construction of the DODAG. The DODAG contains the information for forwarding the packets every mote receives. The Trickle algorithm controls the amount of routing traffic in the form of control message that enter the network [12]. It also controls the amount of time that motes listen for new information and how often it sends out its current information to its neighbor motes

After running the DAG files our aim is to measure the performance metric the first performance metric is energy Consumption. The ICMPv6 control message energy consumption on RPL will be computed in order To make good energy estimation we use percent radio on time of the radio which dominates the energy usage in sensor motes. Furthermore we take the average percent radio on time for all the motes in the whole network setup.

The second metric is Packet Delivery Ratio (PDR) it is computed based on the trickle algorithm and is defined as the number of received messages at the sink to the number of sent message to sink. Since it will be computed by taking the average PDR of all the packets received successfully at sink

The third performance metric of interest in the study is packet latency. The three most dominant control message latency will be measured as previously discussed in the previous section. The latency is defined as the amount of time taken by a packet from mote to reach the sink and is the average of the latencies of all the packets in the network from all the motes.

The fourth performance metric is Control Traffic Overhead for the network. This includes DIO, DIS and DAO messages generated by each mote and it is imperative to limit the Control Traffic keeping in mind the scarce resources in LLN. The aim of this metric is to analyze the effect of the stated parameters on the Control Traffic overhead.

Chapter 5: Experimentation and Evaluation

This section discusses the experiment and evaluation conducted on the CMEM to determine DIO Int Min value. In this Experiment, emphasis is given to the overall procedures that help to measure the performance of Control Message in LLN and the results gained after completing the experiment.

The first section focuses on the test bed. Then in section 5.2 configuration and value used and discussion about the experimental procedure after this in section 5.3 ,5.4 and 5.5 emphasis is given on CMEM experimentation process also Control message information exchange process discussed finally performance evaluation of Control message with their suggested result will be made.

5.1 Test Bed

The Figure 5.1 [27] includes IoT layers and implementation of RPL. Iot layers adopted from [27] and the shaded part implementation of RPL by Contiki using cooja simulator added for our experiment. Let us discuss each of them.

5.1.1 IoT layers

The perception layer is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment [27].

The network layer is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data [27].

The application layer is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health [27].

The processing layer is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data. It can manage and provide a diverse set of services to the

lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules [27]

The business layer manages the whole IoT system, including applications, business and profit models, and users' privacy [27]

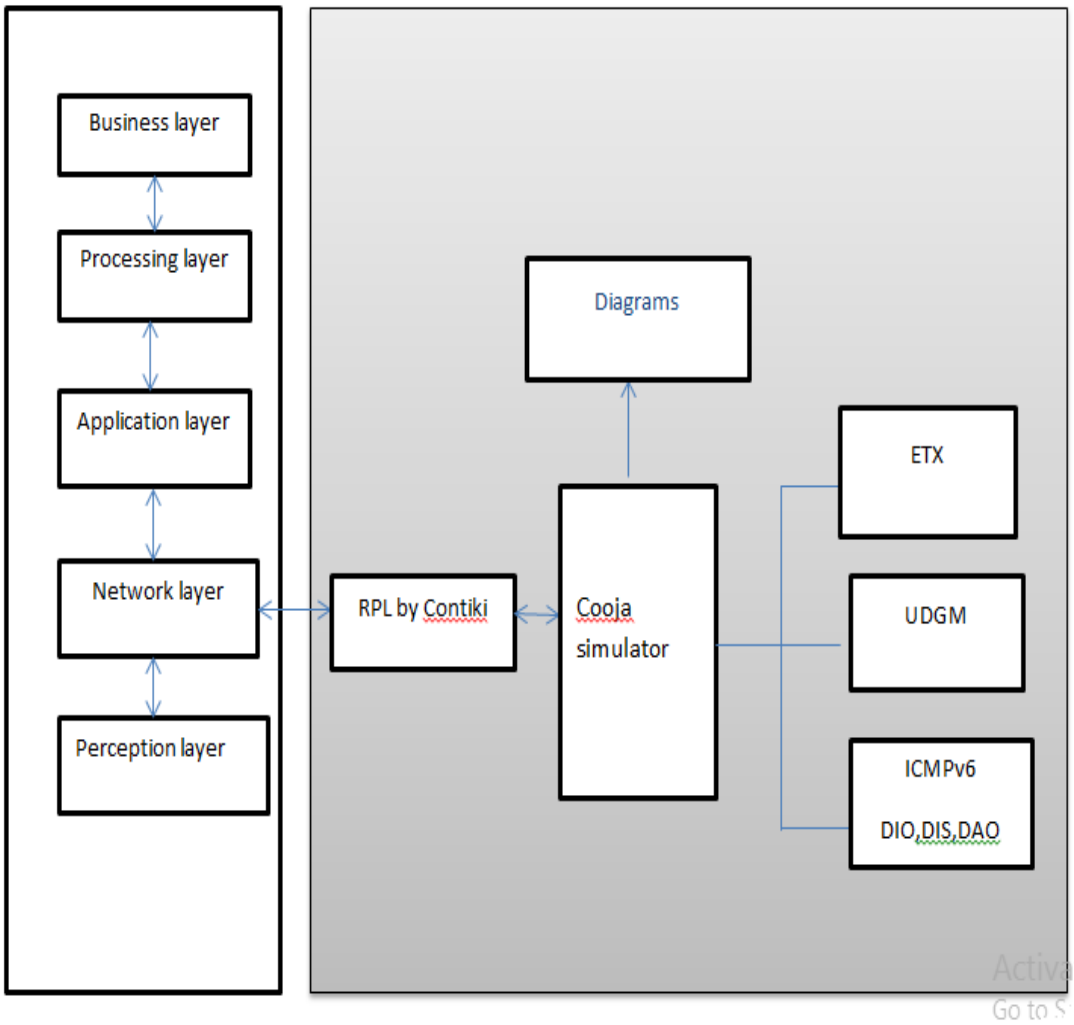


Figure 5.1: Test Bed

5.1.2 RPL by Contiki

The choice of Contiki OS is because of its optimized IPv6 stack, which includes 6LowPAN as an adaptation layer to support routing over the link and network layer. It implements a light weight network stack called Rime, which provides protocols for data collection and route discovery to a destination mote [18].

Contiki OS implements different module for a specific tasks. The contiki/core/net/ module contains specific folder that implements different layers of the protocol stack, which includes the MAC, Rime and RPL. Some of the important files in rpl folder are rpl.c, rpl-dag.c, mrhof.c, OF0, icmp6.c.

The rpl.c file contains implementation of IPv6 Routing Protocol for RPL. The rpl-dag.c file contains logic implementation use in constructing DAG in RPL. Mrhof.c contains logic implementation of minimum rank hysteresis objective function, which uses ETX as a routing metric. OF0.c implements objective function zero which uses hop count as a routing metric and finally the icmp6.c file contains functionalities for RPL controls.

5.2 Configurations and values used in the experiment

Table 5.1: Parameters and values

Configuration on Contiki Cooja	**Value**
Start Delay	65s
OF	ETX
TX Range	50m
RX Ratio	80%
TX Ratio	80%
Interference Range	60m
Client motes	20,50,80
Server mote	1
Test Time	3hr
Radio medium model	Unit Disc Graph Medium (UDGM)
Moto type	Sky mote
Positioning	Random position

DIO Int Min is set to Contiki RPL default value. We select DIO Int Min which is the most dominant control message in RPL [5]. As Table 5.1 refers that the Transmission Ratio (TX) range is set to 50m and interference range to 60m. The Reception Ratio (RX) denotes how lossy the radio medium and is set in percentages through the continuous iterations of the experiment. The TX is set to 80% since we consider that both the source and destination sides are lossy. We run the three experiments for a total of 3 hour for 20, 50 and 80 client motes and one server mote in the simulation

5.3 CMEM Experimentation: 1

We design a sample network in the Cooja simulator containing 20 client motes and 1 server mote performing as root of the DODAG as shown in the Figure 5.2. In order to explain lossyness in wireless medium we use the Cooja Unit Disk Graph Medium which familiarizes lossyness by means of relative distances of motes in the Radio Medium. It uses two different ranges one for transmission and one for interference.

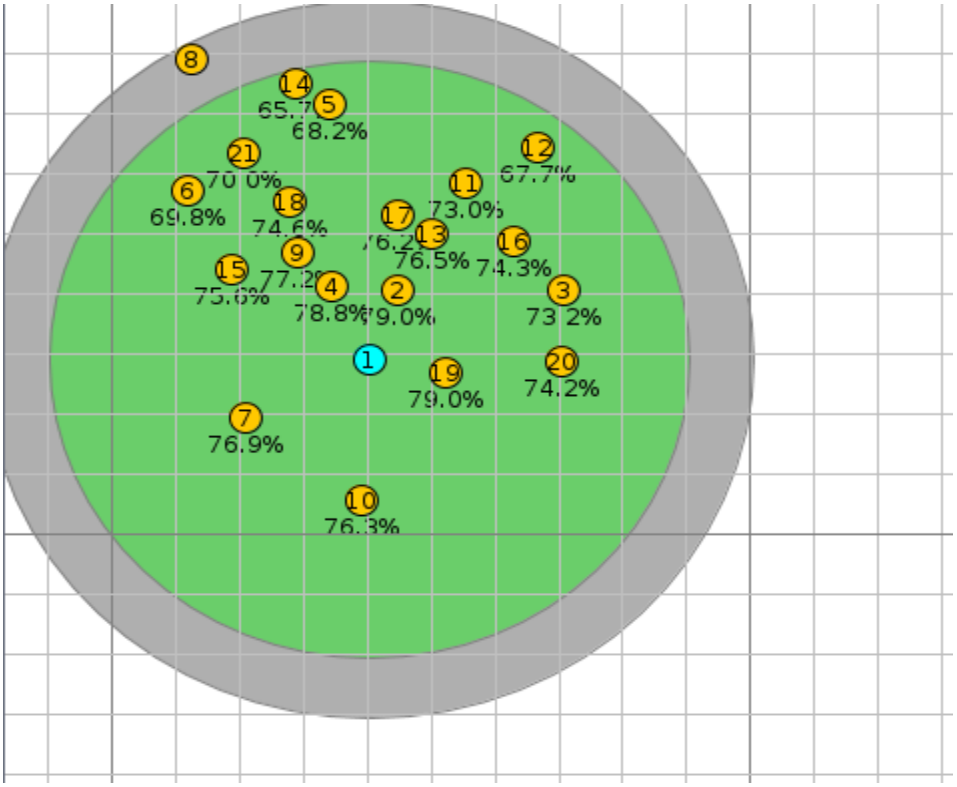


Figure 5.2: RPL network setup process

RPL Network is Setup for 20 client motes and 1 sink mote in COOJA Simulator. The blue mote is the sink, and all other motes with orange circles are the client motes and send packets to the sink. The larger green circle indicates the part of transmission range while the gray circle specifies the area of collision. The number as percentages express the reception ratio.

5.3.1 Control message information exchange process

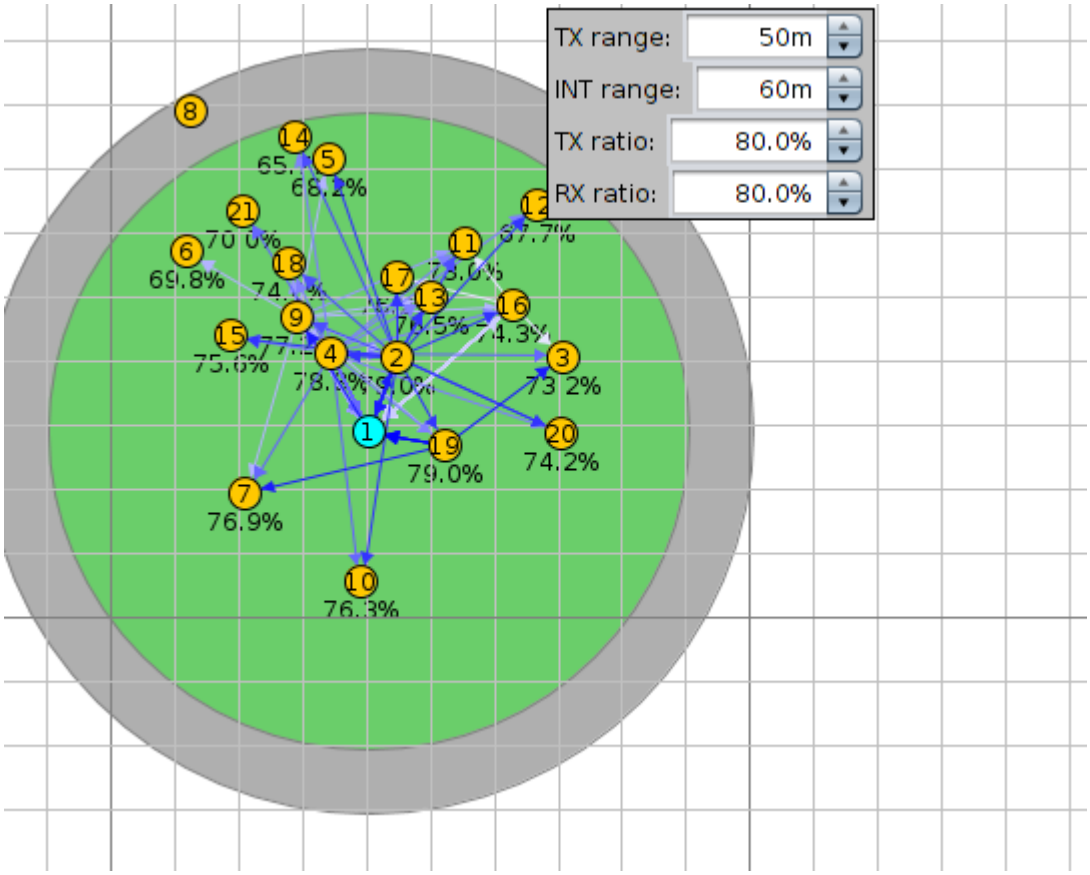


Figure 5.3: Control Message information exchange process

Figure 5.3 shows RPL's control messages information exchange process with the simplest network arrangement, a particular DODAG in the structure. Every mote in RPL promotes routing metrics and constraints over DIO messages. Upon receiving of DIO messages from its neighbors, a mote selects routing parentages based on it's OF and routing info in DIO messages (e.g., RANK, DODAG ID), and then builds a routing topology.

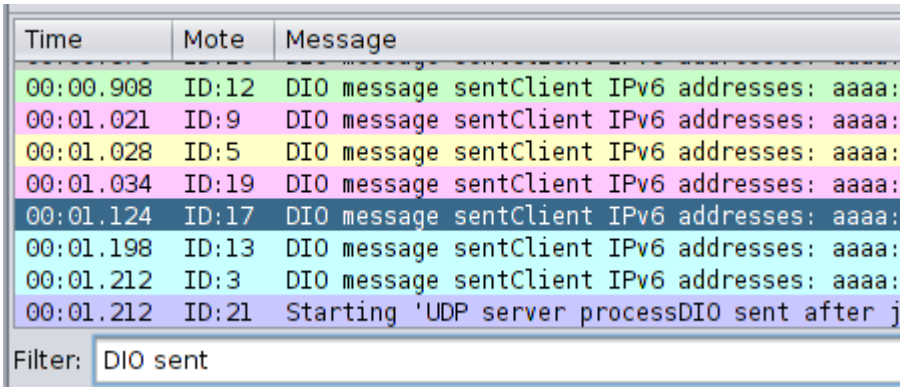


Figure 5.4: Filtering of Control message

Once the mote links the DAG after receiving of DIO it sends out totally the information like DAG-ID, message etc. so that other motes will accept it and joins the DAG after computing its rank as shown in Figure 5.4. This procedure carry's on till the farthest mote in the topology tree links the DAG.

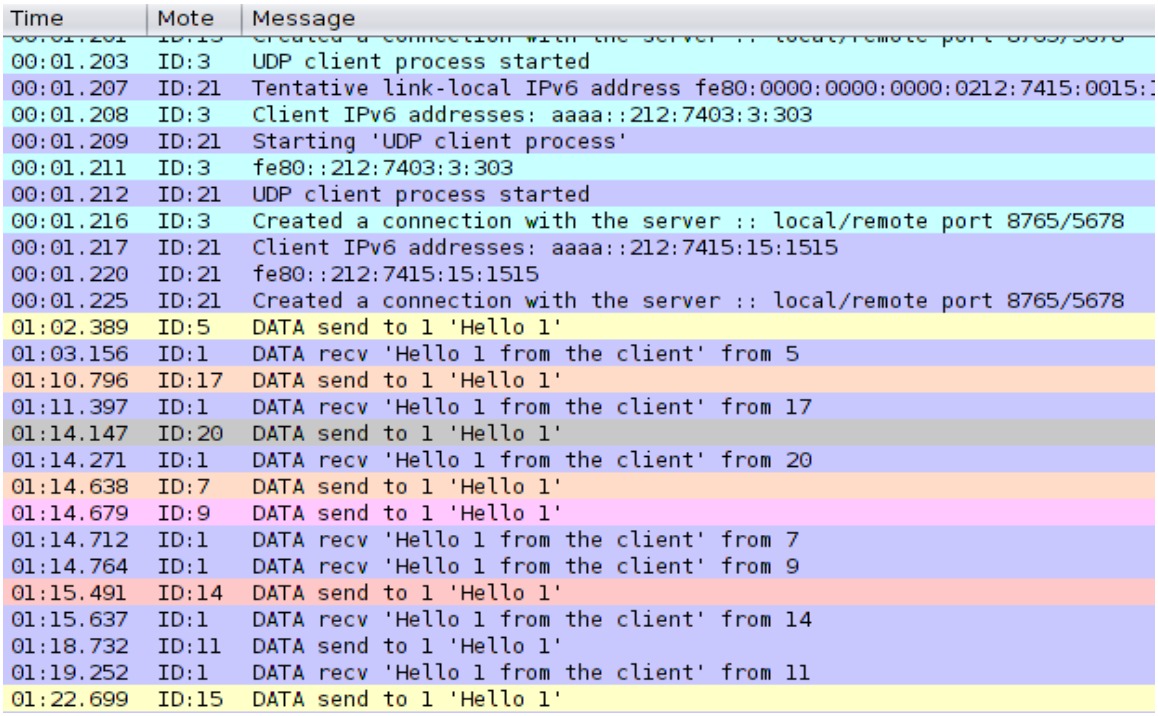


Figure 5.5: Sample UDP packets

Figure 5.5 has 20 client motes and one server (sink) mote each mote sends a sample UDP packet ("Hello N", where N is the series number) to the server (sink) after every Send Interval (10 seconds) and after an initial Start Delay (65 s).

The client output a message 'Hello N sent to Server' quickly as it sends a packet and this allows us to remind the sending time. Likewise Server output a message 'Hello N received from Client M' and this permits us to note the receiving time of the packet at server. The print messages are kept in log file.

5.3.2 Control message performance evaluation

We note that DIO messages are the dominant control messages between all RPL messages as they are used to update and construct the routing tables. The increasing of control messages reflects the network instability and the increasing of the collision and congestion between packets [3].

- **Energy Consumption**

To calculate the energy consumption estimates the energy consumption for packet transmission and listening. This mechanism maintains a table for the time interval a component like CPU, radio transmitter was on. Based on this computation we calculate the percentage of radio on time. We compute the power consumption for radio transmission and listening as these are the most energy consuming components [1].

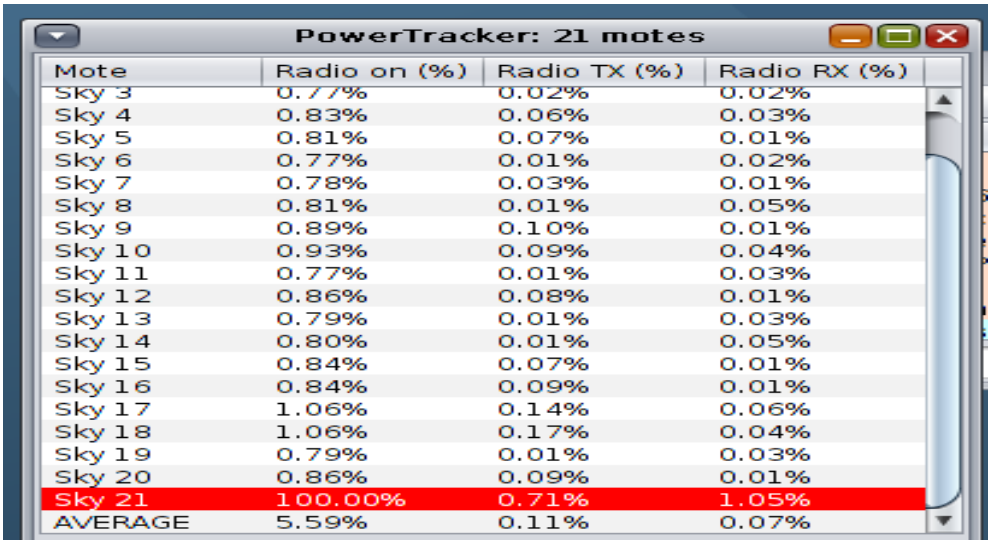


Figure 5.6: Sample UDP packets

As Figure 5.6 shows the RPL network consumes a lot of energy about around 6% Radio ON time. The rapid generations of control packets not only increases control packets but it similarly raises collisions so more retransmissions are essential to effectively send a packet and the energy Consumption increases accordingly.

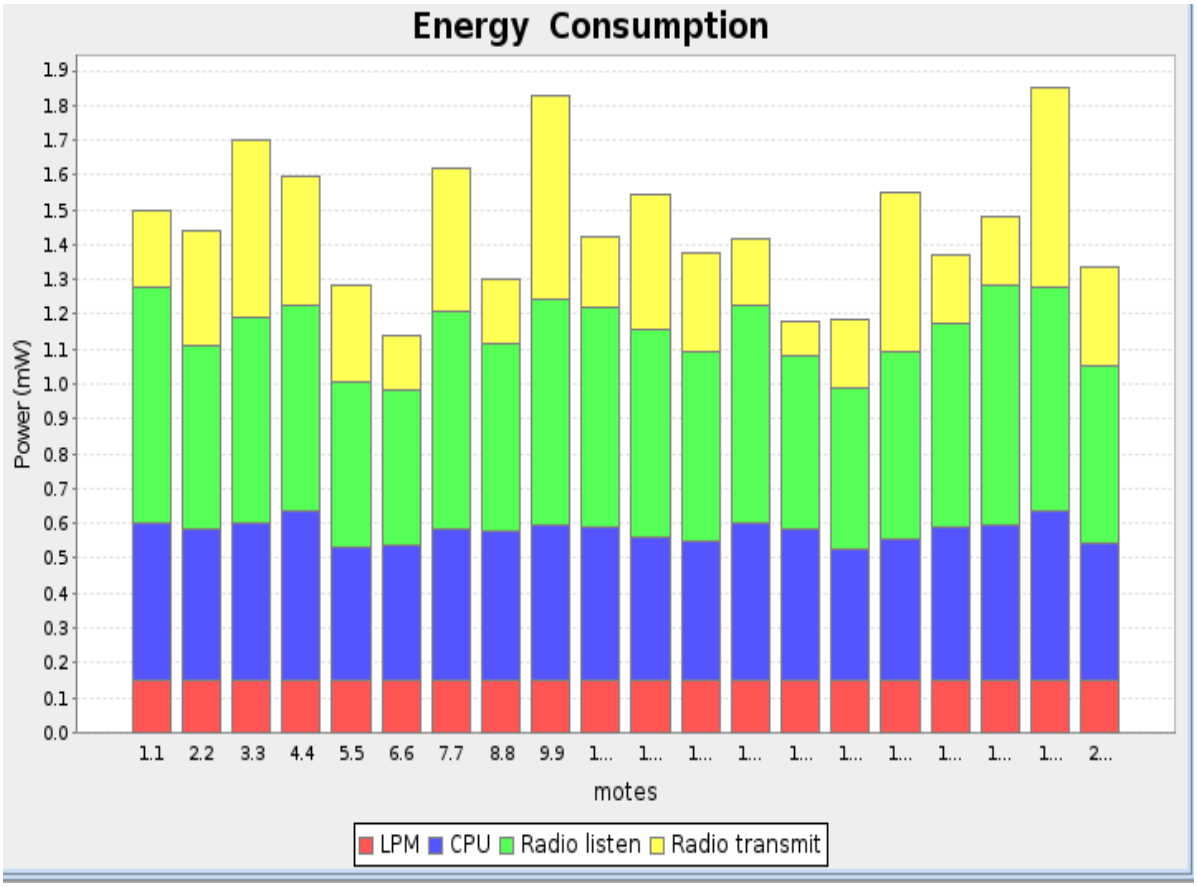


Figure 5.7: Graphical representation of Control message energy consumption

Figure 5.7 shows energy consumption of 20 motes the graph has four parts the red color indicates Low Power Mode (LPM) average energy consumption, the blue color refers CPU power consumption during the full power mode, the green color defined as the power consumption on reception process of radio signals at motes and yellow color defined as the power consumption on transmitting the radio signals. Table 5.2 shows numerical representation of the individual motes energy consumption.

Table 5.2: Numerical Representation of Control message energy consumption

Node	Received	Lost	CPU Power	LPM Power	Listen Power	Transmit Power
1.1	16	0	0.430	0.150	0.467	0.066
2.2	16	0	0.403	0.151	0.409	0.094
3.3	15	0	0.377	0.152	0.412	0.071
4.4	16	0	0.446	0.150	0.421	0.071
5.5	14	2	0.337	0.153	0.393	0.066
6.6	16	0	0.370	0.152	0.399	0.045
7.7	16	0	0.369	0.152	0.416	0.092
8.8	16	0	0.405	0.151	0.423	0.031
9.9	15	1	0.387	0.152	0.431	0.138
10.10	16	0	0.428	0.151	0.451	0.076
11.11	14	1	0.379	0.152	0.410	0.092
12.12	16	0	0.386	0.152	0.420	0.079
13.13	16	0	0.424	0.151	0.415	0.036
14.14	16	0	0.434	0.150	0.424	0.029
15.15	13	2	0.351	0.153	0.399	0.061
16.16	16	0	0.354	0.153	0.408	0.099
17.17	16	0	0.415	0.151	0.487	0.085
18.18	16	0	0.398	0.151	0.454	0.078
19.19	16	0	0.430	0.150	0.433	0.114
20.20	15	0	0.349	0.153	0.397	0.057
21.21	0	0	0.000	0.000	0.000	0.000
Avg	**15.500**	**0.300**	**0.394**	**0.152**	**0.423**	**0.074**

For DIO Int Min , between 3 and 7 the Energy Consumption keeps on declining in a linear style as Figure 5.8 shows. But the finest energy Consumption can be observed for DIO Int Min of 8 and above. This indicates that best value for DIO Int Min according to Energy Consumption is between 8-16.

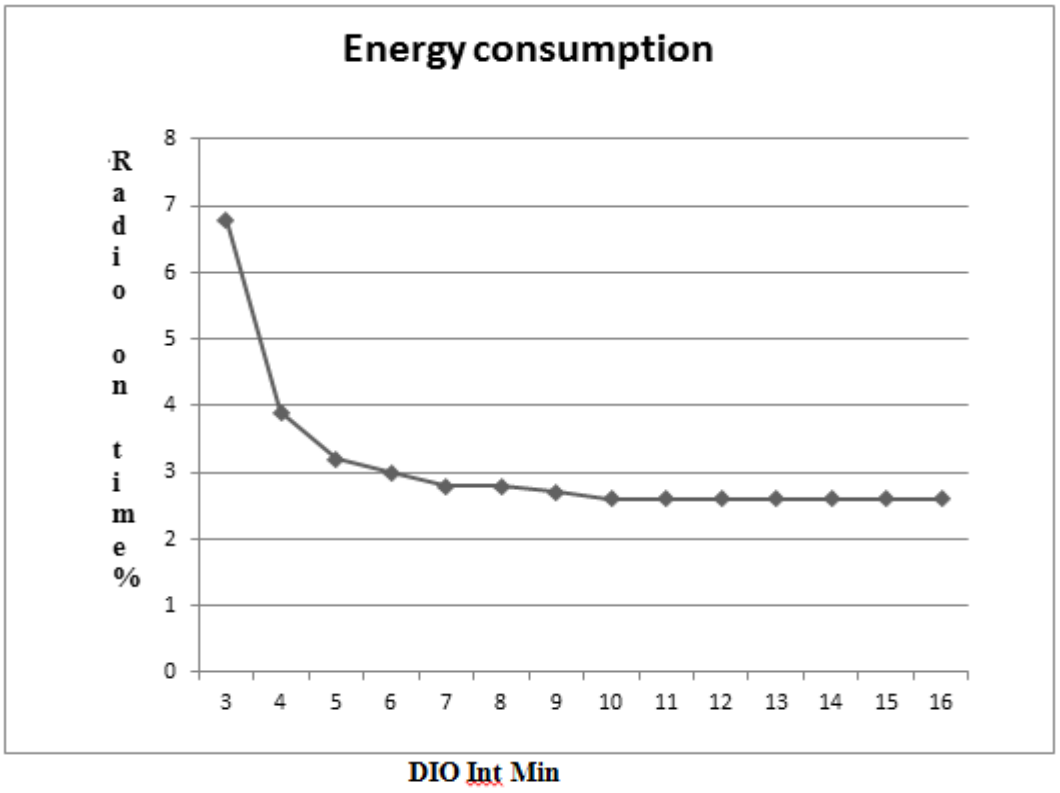


Figure 5.8: Energy consumption with DIO Int Min

- **Packet Delivery Ratio (PDR)**

To calculate the Packet Delivery Ratio we measure the number of sent packets from all motes to the sink and divide it by the number of effectively received packets at the sink. From the Figure 5.9 we note that between 3-5 PDR is low which means due to high control overhead the RPL network suffer collisions and therefore the PDR is poor but PDR increasing between 6-14 and the best PDR observed because the interval of sending DIO message increase at the same time collision decreases. From this we conclude best value for DIO Int Min according to Packet Delivery Ratio is between 6-14.

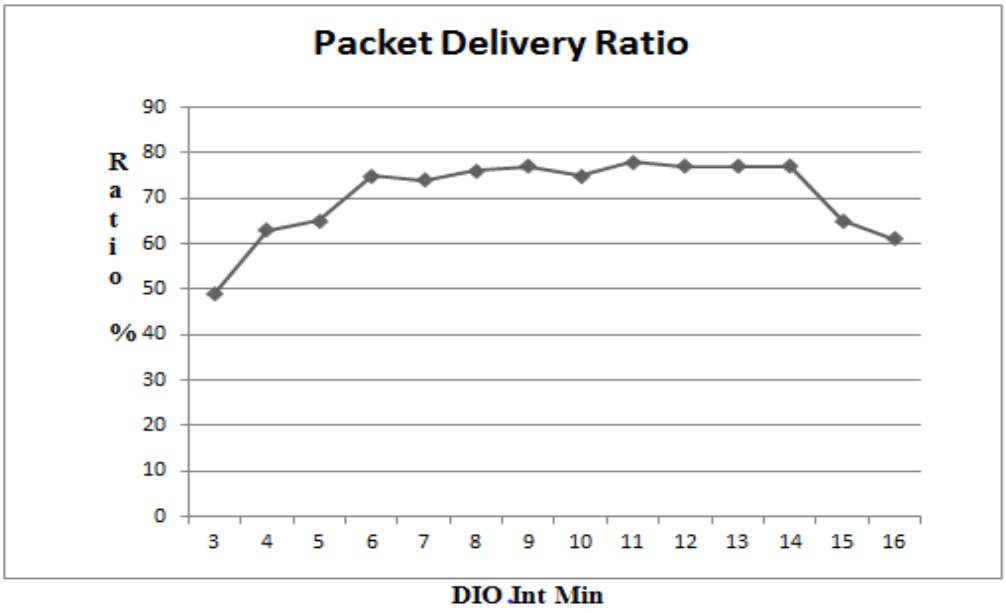


Figure 5.9: Packet Delivery Ratio with DIO Int Min

- **Latency**

Latency is the time interval between message sent by one mote and received by another one [1]. From the sending time and receiving time we calculate the Latency for all the packets (Annex B) the timing information is provided by Cooja Simulator.

The Network Latency falls from 3.5s to 1.8 s as we see from Figure 5.10 for DIO Int Min between3 - 6. This unexpected reduction is because of lower values DIO Int Min generated heavy Control Traffic.

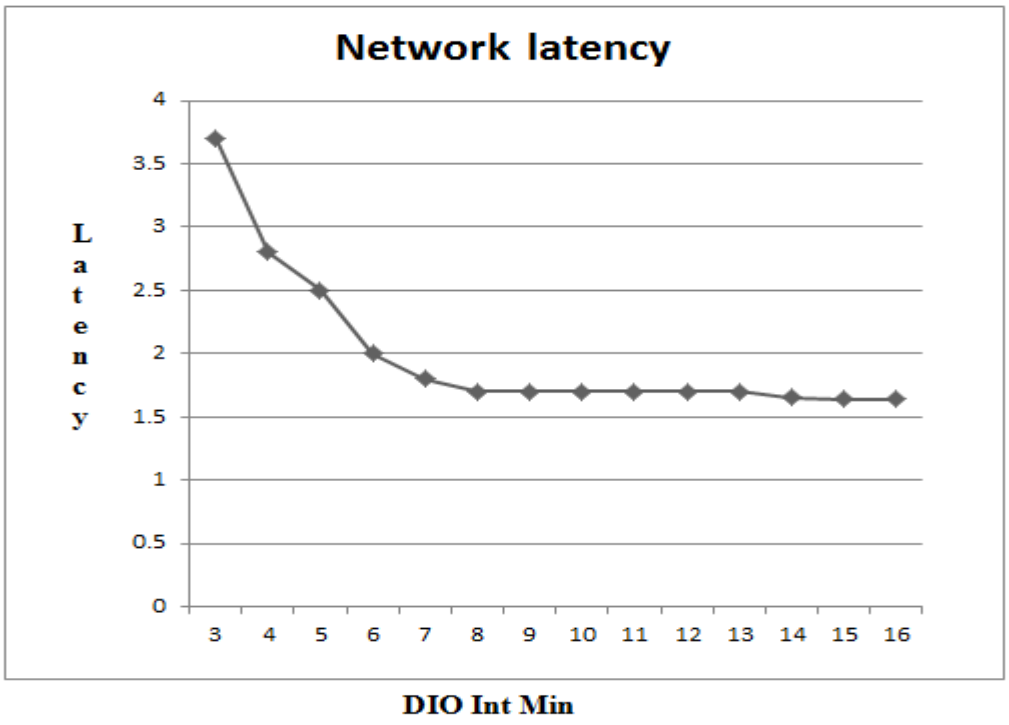


Figure 5.10: Network latency with DIO Int Min

- **Control Traffic Overhead**

Control traffic overhead is very high for lower DIO Int Min as shown in Figure 5.11. Likewise we observe that the control overhead decreases when DIO Int Min increases. As we increase the sending interval time of control message the traffic between packets decreases. Also the amount of lost packet decreases. From this observation we conclude that DIO Int Min in the range (8-16) is the most optimum set for Control Traffic overhead.

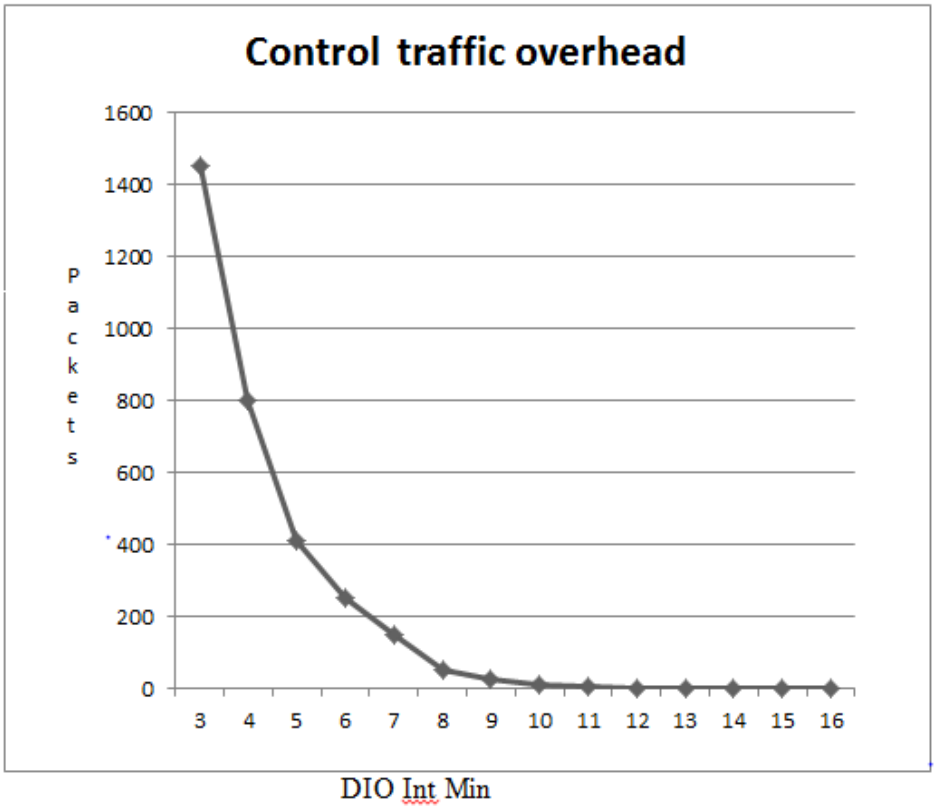


Figure 5.11: Control Traffic overhead with DIO Int Min

5.4 CMEM Experimentation: 2

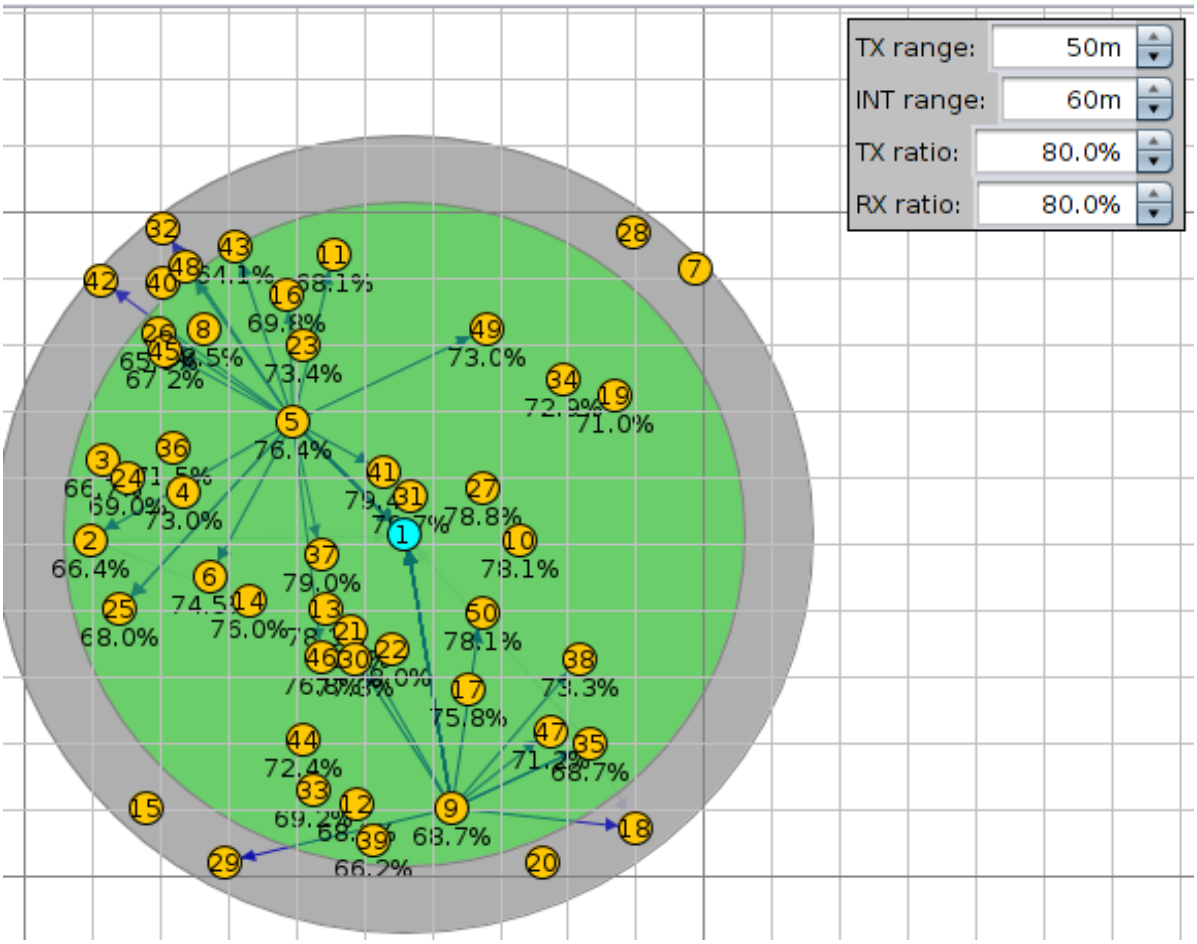


Figure 5.12: Control message information exchange process

The Figure 5.12 shows RPL's control messages information exchange process with 50 client motes and 1 sink mote.

- **Energy consumption**

Figure 5.13 indicates DIO Int Min between 3 and 7 the energy consumption keeps on declining in a linear fashion which shows similar result that is observed in experiment one. The best energy consumption can be observed for DIO Int Min of 8 and above. But Radio on time increases as we increase our network size which means it consumes more energy

than experiment one. From experiment two we observed that the best DIO Int Min value according to energy consumption is between 8-16.

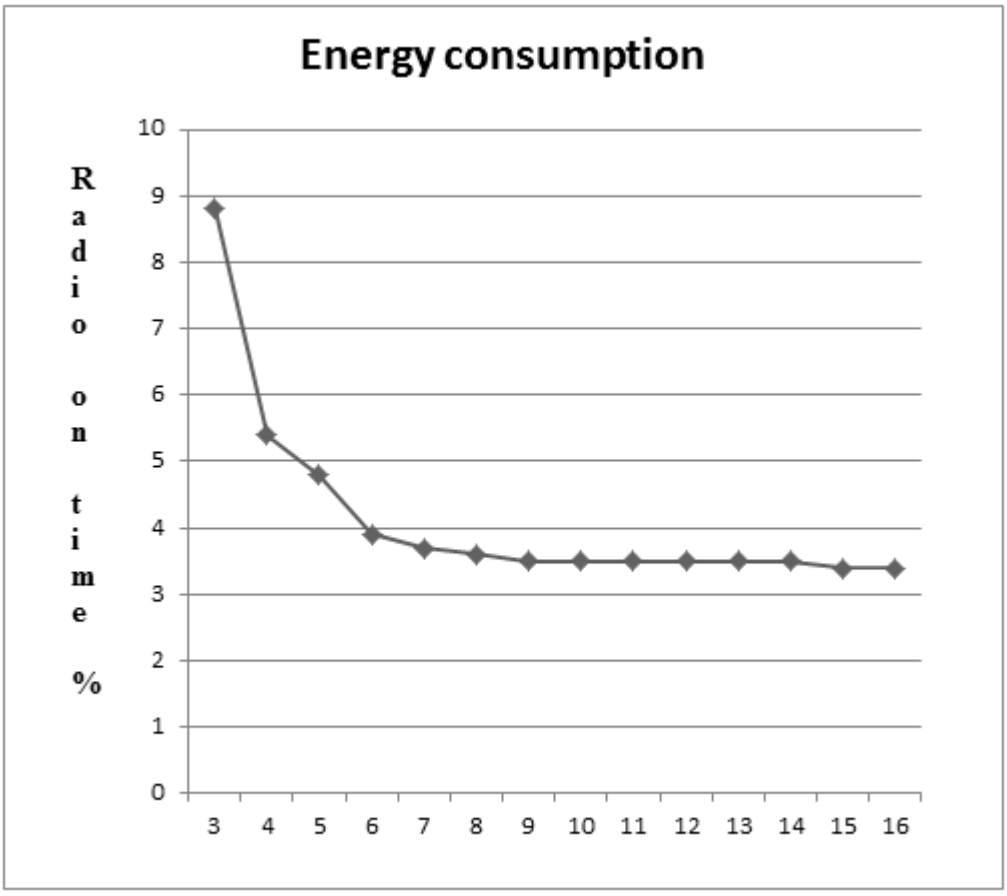


Figure 5.13: Energy consumption with DIO Int Min

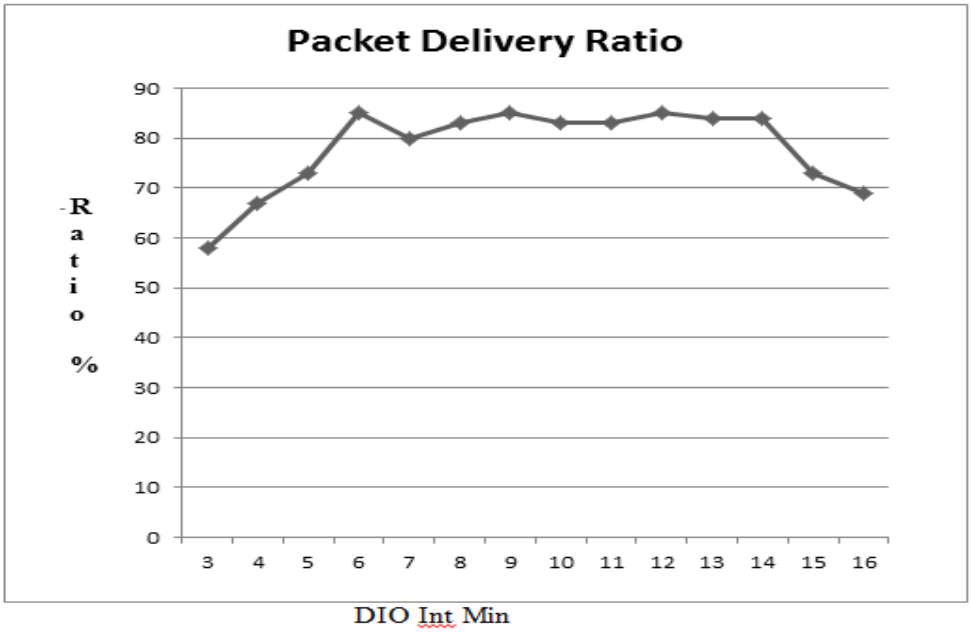


Figure 5.14: Packet Delivery Ratio with DIO Int Min

In Figure 5.14 the PDR is below 75% at the start, for DIO Int Min of 3 to 5 which means due to high control overhead the RPL network suffer collisions and therefore the PDR is poor. However as we increase the DIO Int Min 6 to 14 RPL provides a good Packet Delivery Ratio. Also PDR decreases for DIO Int Min of 15 and 16 because DIO Int Min value higher than 14 does not provide a quick network convergence

- **Latency**

In network latency we observe similar result with experiment two. latency falls from 3.8s to 2s as we see from the Figure 5.15 for DIO Int Min between 3 - 7. The dense control traffic causes more collisions. As soon as the control traffic decreases for DIO Int Min of higher than 8 the packet buffering decreases as a result the packet reaches the destination quickly.

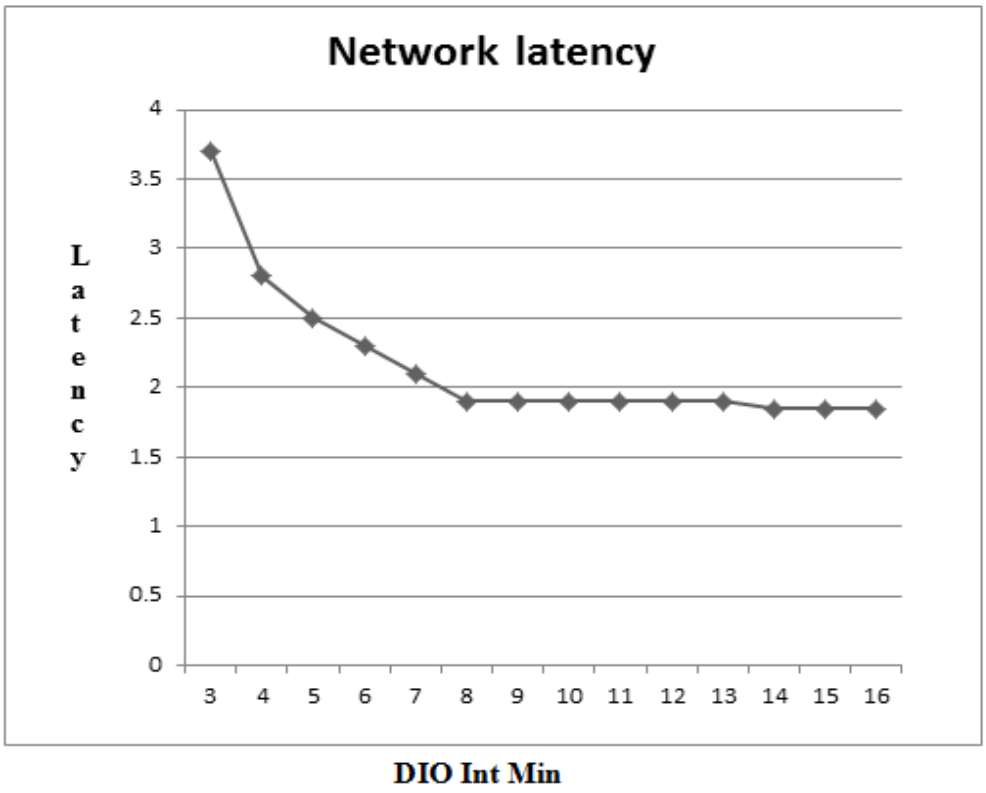


Figure 5.15: Network latency with DIO Int Min

- Control traffic overhead

As we observe from Figure 5.16 control traffic overhead is higher for DIO Int Min of 3 which decrease further to 50 Packets for Interval of 16. We conclude a subset of DIO Int Min in the range of 8 to 16 is the most optimum set for Control Traffic overhead

.

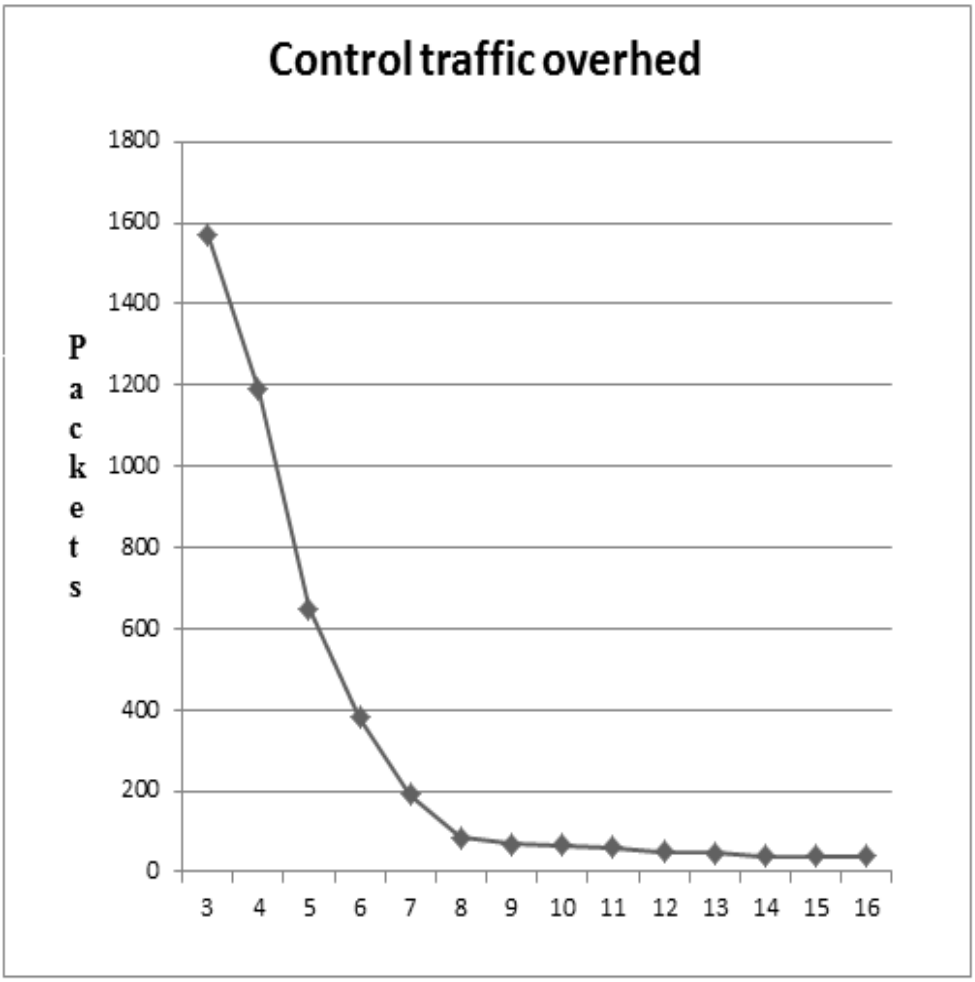
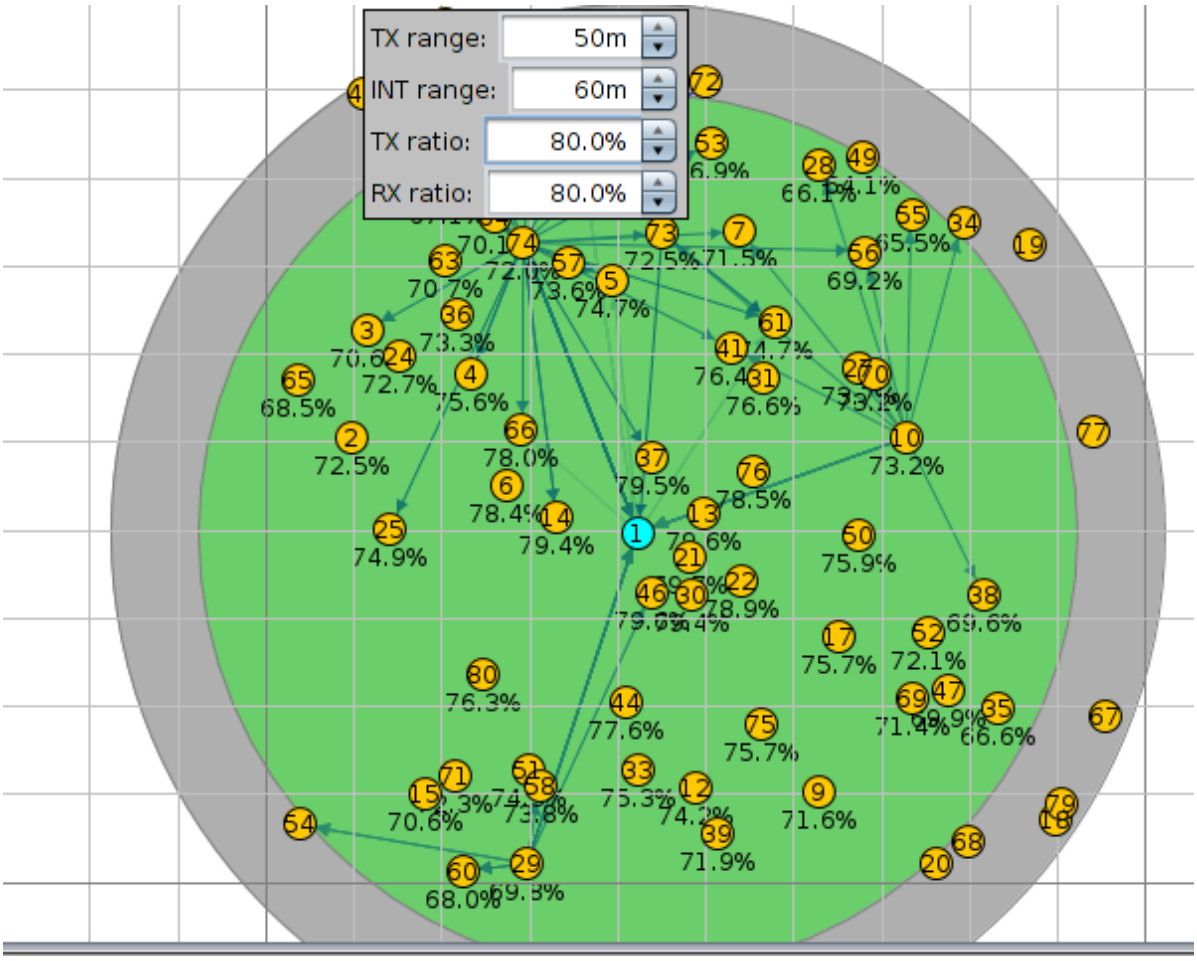


Figure 5.16: Control traffic overhead with DIO Int Min

5.5 CMEM Experimentation: 3



.

Figure 5.17: Control traffic overhead with DIO Int Min

Figure 5.17 shows RPL's control messages information exchange process with 80 client motes and 1 sink mote

- Energy consumption

Figure 5.18 indicates the RPL network consumes a lot of energy about 10% Radio ON time when DIO interval is very small.as we increases DIO Int Min value radio on time decreases as a result it will consume less energy.

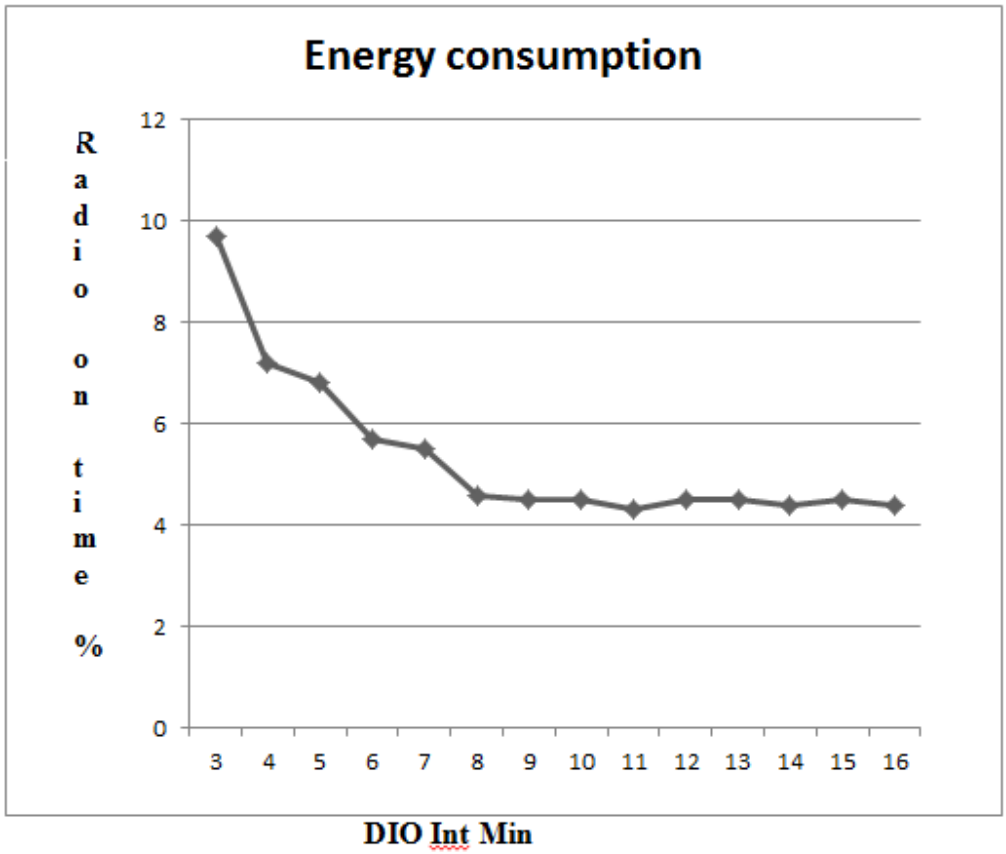


Figure 5.18: Energy consumption with DIO Int Min

In Figure 5.19 the PDR is below 85% at the start, for DIO Int Min of 3 to 5. However as we increase the DIO Int Min 6 to 14 RPL provides a good PDR. The result is similar result with previous observations for 20, 50 and 80 client motes.

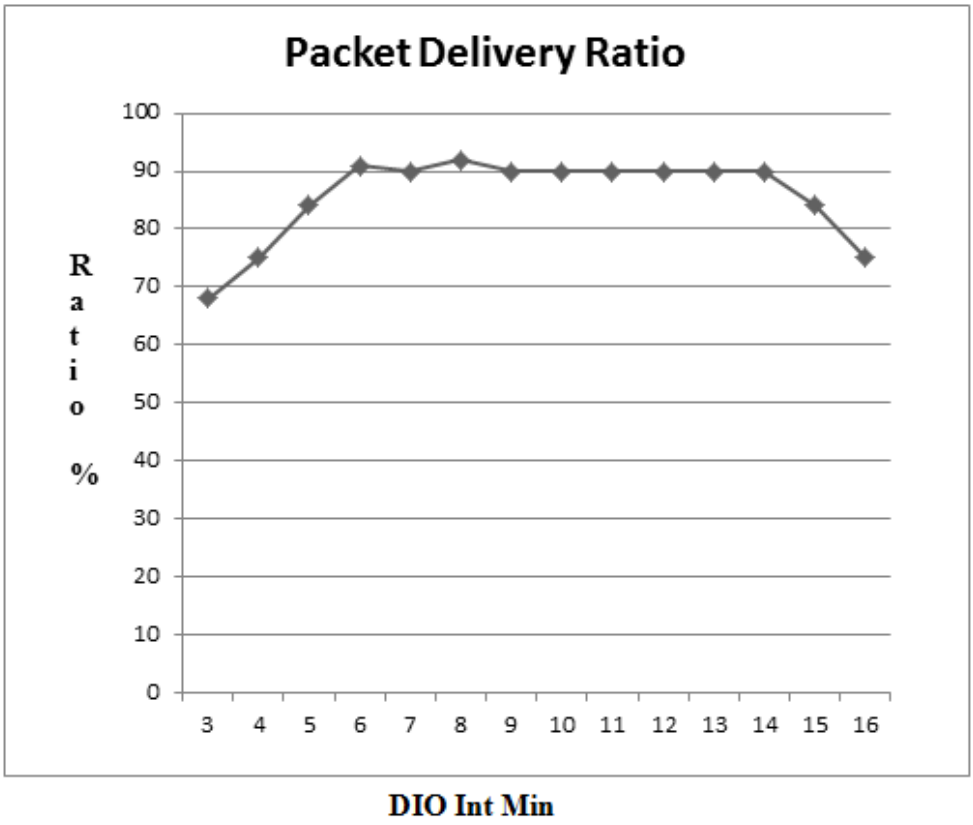


Figure 5.19: Packet Delivery Ratio with DIO Int Min

Figure 5.20 shows network latency decreases with the increase in DIO Int Min for 80 client motes.

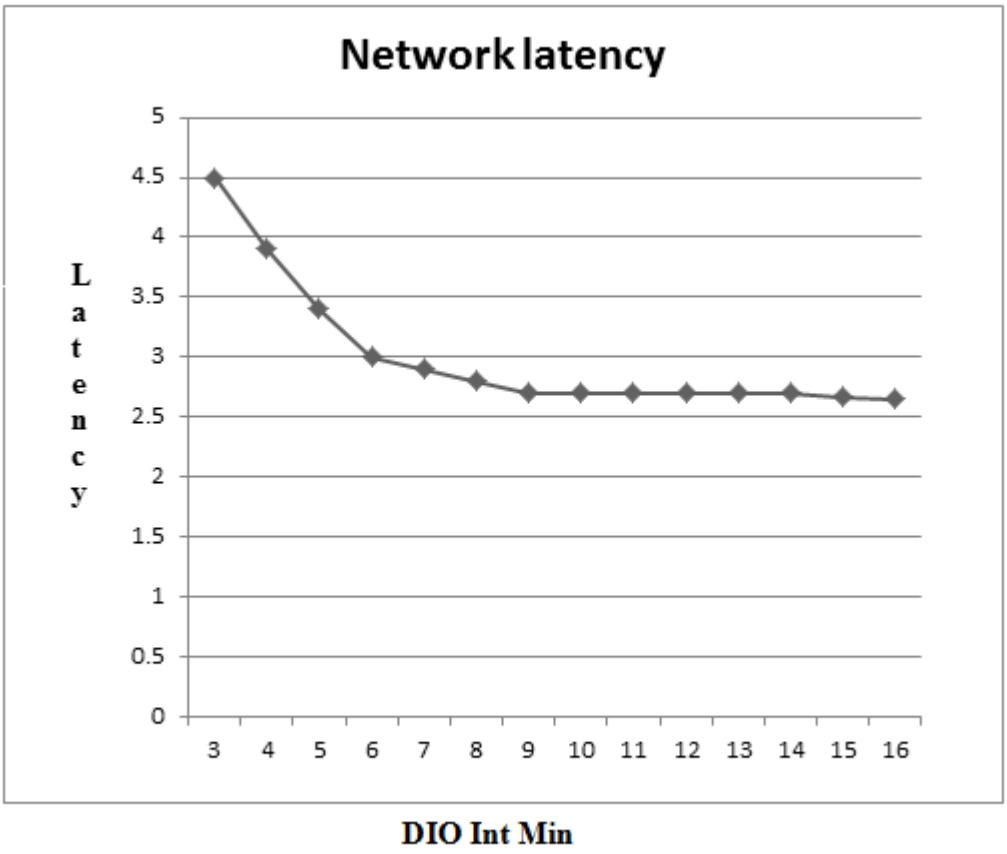


Figure 5.20: Network latency with DIO Int Min

Figure 5.21 shows control traffic overhead decreases with increase in DIO Int Min for 80 client motes.

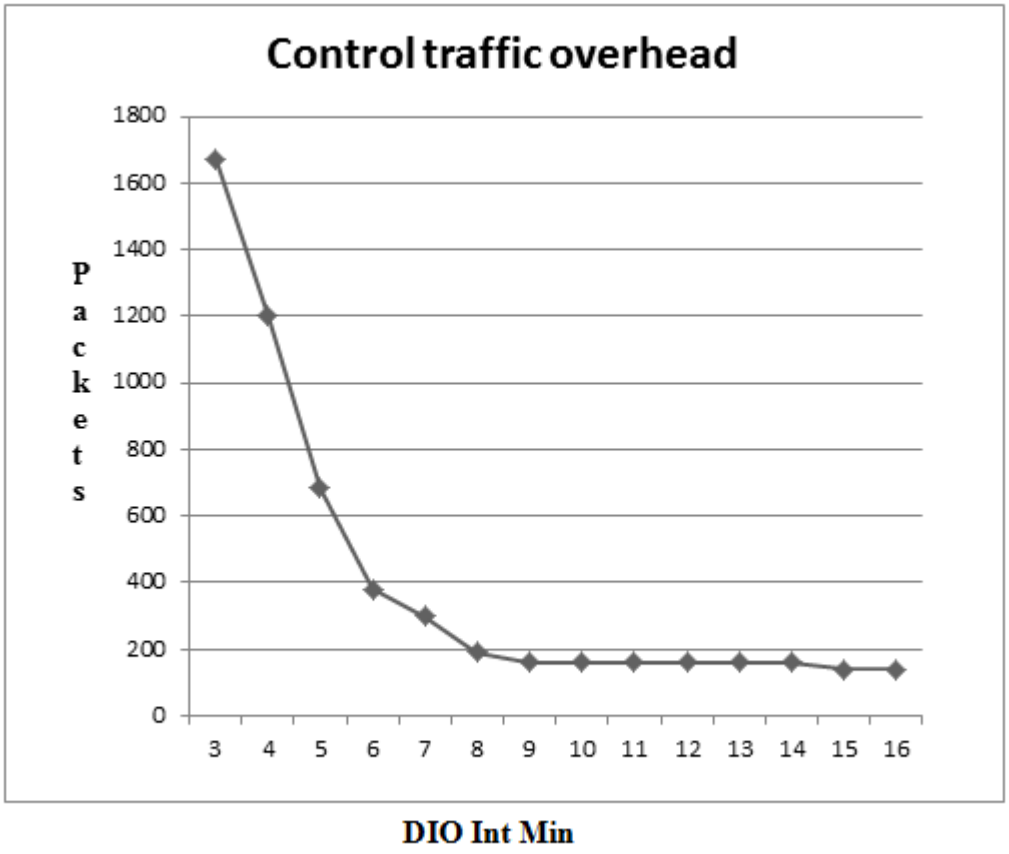


Figure 5.21: Control traffic overhead with DIO Int Min

.

Chapter 6: Conclusion

The trickle algorithm was targeted to limit the number of control messages transmitted in LLN and to use the limited resources in LLN more optimistically. This process is controlled by DIO Int Min. It is the most influential since it controls the generation of Control message (DIO) directly. This parameter causes a tradeoff between the performance metrics and we observed carefully the effect of every possible change of this parameter on the performance metrics of interest.

We observed that a set of RPL parameters are crucial for its better performance in vast areas of sensors applications. Trickle timer is an important component of RPL in order to utilize the scarce resources of LLN. Our experiment evaluated with three observations for 20, 50 and 80 client motes. Instead of using the algorithm default value we can use the experimented based values which are summarized in Table 6.1. These values will help to provide better Energy Consumption, decreased Control Traffic overhead, lower Latency and high PDR by using it in the trickle algorithm.

Table 6.1: Suggested control message DIO Int Min values

Performance Metric	**Control Message DIO Min Interval**
Energy Consumption	8-16
Latency	8-16
Packet Delivery Ratio	6-14
Control Traffic	8-16

References

[1] T.Zhang and X. Li “Evaluating and Analyzing the Performance of RPL in Contiki,” Engineering Lab on Intelligent Perception for Internet of Things(ELIP), August 11, 2014.

[2] S.Umamaheswari and A. Negi .“Internet of things and RPL routing protocol, International

Conference on Computer Communication and Informatics IEEE, Jan. 05 – 07, 2017 .pp. 1-7.

[3] A.Shimaa ,Abdel Hakeem and HyungWon Kim “Routing Protocol Performance in Smart Grid Applications Based Wireless Sensors,” Experimental, 2019.

[4] A.Parasuram , D. Culler and R.Katz“An Analysis of the RPL Routing Standard for Low Power Lossy Networks,” Technical Report No.UCB/EECS , California, May 14,2016.

[5] X. Liu, Z. Sheng, C. Yin, F. Ali, D. Roggen , “ Performance Analysis of Routing Protocol for low -power lossy network in large scale network,” IEEE, 2012, pp. 1-14.

[6] T.Tsvetkov, “ IPv6 Routing Protocol for low power lossy network,” Network Architectures and Services, July 2011.

[7] R.Hevner,T.March,J.Park and S.Ram , “Design science in information system research” MIS Quarterly Vol.28 No.1, /march/2014,pp.75-105.

[8] N.Nasser, L.Karim, A.Ali and M.Anan, “Routing in internet of things,”Conference GLOBECOM, IEEE ,December 2017.

[9] T. Winter, “RPL: IPv6 Routing Protocol for Low-Power and LossyNetworks,” [Online]. Available:https://rfc-editor.org/rfc/rfc6550.txt [Accessed 18-May- 2019]

[10] Shimaa A., Anar A. and HyungWon Kim, " RPL Rouing Protocol performance in smart grid applications based wirless sensor experiment and simulated analysiiss",ERI, 5-Feb-2019

[11] Hyung-sin kim, Jeonggil Ko,E.Culler,Jeongyeup Paek, " Challanging the ipv6 routing protocol for low power and lossy networks", IEEE Communications Surveys & Tutorials, Volume 19, NO. 4, Fourth Qarter 2017

[12] T. Clausen, O. Gnawali, J. Ko, P. Levis, and J. Hui, "The Trickle Algorithm." [Online]. Available: http://tools.ietf.org/html/rfc6206. [Accessed: 24-April-2019]

[13] A. Dunkels , "The ContikiMAC Radio Duty Cycling Protocol,Swedish Institute of Computer Science", Tech. Rep.T2011:13, Dec. 2011.

[14] John Abied, Haidar Safa and Wassim El-hajj , " enhancing routing protocol for low power lossy network".IWCMC,IEEE,2017

[15] Yibo Chen, Kun-Mean Hou, Haiying Zhou. " A survey in Wireless Communications, Networking and Moblie Computing".WiCOM, 7th InternationalConference on. IEEE, 2011, pp. 1-4.

[16] Asma Haroon , Wajeeha Naeem , Munam Ali Shah, Muhammad Kamran, Yousra Asim and Qaisar Javaid , " Constraints in the IoT. *International Journal of Advanced Computer science Applications*",*IJACSA,Volume 7, No. 11, 2016*

[17] Z.Sheng ,S.Yang,V.Vasilakos,A.Julie and K.Kin "A survey on the IETF protocol Suite for The Internet-of-Things: Standards, Challenges and Opportunities" ,IEEE, 2013

[18] Moteiv, Tmote Sky - ultra low power IEEE 802.15.4 compliant wireless sensor module. Data sheet.

[19] Zahra Aslani and Hadi Sargolzaey," Improving the Performance of RPL Routing Protocol for Internet of Things",Journal of Computer & Robotics, 16 October 2017

[20] Loven Kalyan and Karamjit Kaur ,"An Analysis of Performance of Routing Protocol for Low Power and Lossy Network in Sparse Wireless Sensor Network", IJARCET, Volume 4 Issue 8, August 2015

[21] J. Ko, S.Dawson-Haggerty, O.Gnawali, D.Culler, and A.Terzis, “Evaluating the performance of RPL and 6LoWPAN in TinyOS”, In Proceedings of the Workshop on Extending Internet to Low power and Lossy Networks (IPSN), .2014.

[22] J. P. Vasseur, N. Agarwal, J. Hui, Z. Shelby, P. Bertrand, and C. Chauvenet, “ RPL: IPv6 Routing Protocol for Low power and Lossy Networks”, IPSO Alliance, 2014.

[23] Jinghan Wang and Hongxin Li,"Researching and Hardware Implementation of RPL Routing Protocol Based on the Contiki Operating System", International Journal of Future Computer and Communication, Vol. 3, No. 6, December 2014

[24] J. Ko, Jo. Eriksson, N.Tsiftes, S. Dawson-Haggerty, A. Terzis, A.Dunkels and D. Culler,"ContikiRPL and TinyRPL: Happy Together",IPSN’11, April 14, 2011

[25] Xian Ni, Kun-chan ,Robert .M,"on the performance of Expected Transmission Count (ETX)", ISBN ,October 2012

[26] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, “Powertrace: Network-level Power Profiling for Low-power Wireless Networks”, Mar. 2012.

[27] Pallavi Sethi and Smruti R. Sarangi ,Sarangi, “Internet of Things: Architectures, Protocols, and Applications”, Journal of Electrical and Computer Engineering, Volume 2017, Article ID 9324035, 25 pages

Annex A: Default DIO Int Min value in trickle algorithm

```
#define RPL_CODE_SEC_DIS               0x80   /* Secure DIS
*/
#define RPL_CODE_SEC_DIO               0x81   /* Secure DIO
*/
#define RPL_CODE_SEC_DAO               0x82   /* Secure DAO
#define RPL_CODE_SEC_DAO_ACK           0x83
#ifdef RPL_CONF_DIO_INTERVAL_MIN
#define RPL_DIO_INTERVAL_MIN        RPL_CONF_DIO_INTERVAL_MIN
#define RPL_DIO_INTERVAL_MIN        12
static void
handle_dio_timer(void *ptr)
{
  rpl_instance_t *instance;

  instance = (rpl_instance_t *)ptr;

  PRINTF("RPL: DIO Timer triggered\n");
  if(!dio_send_ok) {
    if(uip_ds6_get_link_local(ADDR_PREFERRED) != NULL) {
      dio_send_ok = 1;
    } else {
      PRINTF("RPL: Postponing DIO transmission since link
local address is not ok\n");
      ctimer_set(&instance->dio_timer, CLOCK_SECOND,
&handle_dio_timer, instance);
      return;
```

```
    }
  }

  if(instance->dio_send) {
    /* send DIO if counter is less than desired redundancy */
    if(instance->dio_counter < instance->dio_redundancy) {
#if RPL_CONF_STATS
      instance->dio_totsend++;
#endif /* RPL_CONF_STATS */
      dio_output(instance, NULL);
    } else {
      PRINTF("RPL: Supressing DIO transmission (%d >= %d)\n",
             instance->dio_counter, instance-
>dio_redundancy);
    }
    instance->dio_send = 0;
    PRINTF("RPL: Scheduling DIO timer %lu ticks in future
(sent)\n",
           instance->dio_next_delay);
    ctimer_set(&instance->dio_timer, instance-
>dio_next_delay, handle_dio_timer, instance);
  } else {
    /* check if we need to double interval */
    if(instance->dio_intcurrent < instance->dio_intmin +
instance->dio_intdoubl) {
      instance->dio_intcurrent++;
      PRINTF("RPL: DIO Timer interval doubled %d\n",
instance->dio_intcurrent);
    }
    new_dio_interval(instance);
  }
}
```

```
void
rpl_reset_periodic_timer(void)
{
  next_dis = RPL_DIS_INTERVAL / 2 +
    ((uint32_t)RPL_DIS_INTERVAL * (uint32_t)random_rand()) /
RANDOM_RAND_MAX -
    RPL_DIS_START_DELAY;
  ctimer_set(&periodic_timer, CLOCK_SECOND,
handle_periodic_timer, NULL);
}
/* Resets the DIO timer in the instance to its minimal
interval. */
void
rpl_reset_dio_timer(rpl_instance_t *instance)
{
#if !RPL_LEAF_ONLY
  /* Do not reset if we are already on the minimum interval,
     unless forced to do so. */
  if(instance->dio_intcurrent > instance->dio_intmin) {
    instance->dio_counter = 0;
    instance->dio_intcurrent = instance->dio_intmin;
    new_dio_interval(instance);
  }
#if RPL_CONF_STATS
  rpl_stats.resets++;
#endif /* RPL_CONF_STATS */
#endif /* RPL_LEAF_ONLY */
}
static void
handle_dao_timer(void *ptr)
{
```

```
  rpl_instance_t *instance;

  instance = (rpl_instance_t *)ptr;

  if(!dio_send_ok && uip_ds6_get_link_local(ADDR_PREFERRED)
== NULL) {
    PRINTF("RPL: Postpone DAO transmission\n");
    ctimer_set(&instance->dao_timer, CLOCK_SECOND,
handle_dao_timer, instance);
    return;
  }

  /* Send the DAO to the DAO parent set -- the preferred
parent in our case. */
  if(instance->current_dag->preferred_parent != NULL) {
    PRINTF("RPL: handle_dao_timer - sending DAO\n");
    /* Set the route lifetime to the default value. */
    dao_output(instance->current_dag->preferred_parent,
instance->default_lifetime);
  } else {
    PRINTF("RPL: No suitable DAO parent\n");
  }
  ctimer_stop(&instance->dao_timer
void
rpl_schedule_dao(rpl_instance_t *instance)
{
  clock_time_t expiration_time;

  expiration_time = etimer_expiration_time(&instance-
>dao_timer.etimer);

  if(!etimer_expired(&instance->dao_timer.etimer)) {
```

```
    PRINTF("RPL: DAO timer already scheduled\n");
  } else {
    expiration_time = RPL_DAO_LATENCY / 2 +
      (random_rand() % (RPL_DAO_LATENCY));
    PRINTF("RPL: Scheduling DAO timer %u ticks in the
future\n",
           (unsigned)expiration_time);
    ctimer_set(&instance->dao_timer, expiration_time,
               handle_dao_timer, instance);
  }
```

Annex B: CMEM sample code

```
struct latency_structure{ // Structure to save the time when
the message was send

   rtimer_clock_t timestamp;// Time when the message was send

};
/
PROCESS(example_broadcast_process, "Broadcast example");
AUTOSTART_PROCESSES(&example_broadcast_process);

//Receive a broadcast with the timestamp and compute the
latency
static void
broadcast_recv(struct broadcast_conn *c, const linkaddr_t
*from)
{
  struct latency_structure msg;
  rtimer_clock_t latency;
```

```
  packetbuf_copyto( &msg ); // Copy the message from the
packet buffer to the structure called msg

#if TIMESYNCH_CONF_ENABLED
  latency = time rece_time() - msg.sentime;
#else
  latency = 0;
#endif

  printf("broadcast message received from %d.%d with latency
%lu ms\n",
         from->u8[0], from->u8[1], (1000L * latency) /
RTIMER_ARCH_SECOND );
}
static const struct broadcast_callbacks broadcast_call =
{broadcast_recv};
static struct broadcast_conn broadcast;
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  struct latency_structure *msg;

  static struct etimer et;

  PROCESS_EXITHANDLER(broadcast_close(&broadcast);)

  PROCESS_BEGIN();

  broadcast_open(&broadcast, 129, &broadcast_call);

  while(1) {

    /* Delay 2-4 seconds */
```

```
    etimer_set(&et, CLOCK_SECOND * 4 + random_rand() %
(CLOCK_SECOND * 4));

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

#if TIMESYNCH_CONF_ENABLED
      msg->timestamp = timesynch_time();
#else
      msg->timestamp = 0;
#endif

    // Do not send the normal/original 'hello' message.
Instead, send a message with the timestamp.
    packetbuf_copyfrom(msg,   sizeof(*msg) ); // This message
msg includes the timestamp
    //packetbuf_copyfrom("Hello", 6);
    broadcast_send(&broadcast);
    printf("broadcast message sent\n");
  }

  PROCESS_END();
}
```

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Senait Desalegn Merne

Signature: _____

Date: September 30,2019

Confirmed by advisor:

Name: _____

Signature: _____

Date: _____