

RPL-BASED ENERGY-EFFICIENT ROUTING PROTOCOL FOR THE INTERNET OF THINGS

M.TECH THESIS REPORT

submitted in partial fulfillment of the requirements

for the award of the degree of

Master of Technology

in

COMPUTER SCIENCE AND ENGINEERING

By

**MADHURJYA SAHA
(320519019)**

Under the supervision of:

DR. SURAJEET GHOSH



**Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology, Shibpur
HOWRAH – 711103 WEST BENGAL (INDIA)**

July 2021

CERTIFICATE

I hereby certify that the work which is being presented in the M.Tech. Thesis report entitled “**RPL-Based Energy-Efficient Routing Protocol for the Internet of Things**” in partial fulfillment of the requirements for the award of the degree **Master of Technology in Computer Science and Engineering** is an authentic record of my own work carried out during a period from Aug, 2020 to July, 2021 under the supervision of **Dr. Surajeet Ghosh**, Computer Science and Technology Department.

The matter presented in this thesis has not been submitted for the award of any other degree elsewhere.

Signature of Candidate

Madhurjya Saha

320519019

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of Supervisor

Date:

Dr. Surajeet Ghosh

Acknowledgements

I would like to express a deep sense of gratitude towards my project supervisor and mentor Dr. Surajeet Ghosh, Department of Computer Science & Technology, IIST Shibpur, for providing excellent guidance and inspiration, and for giving me the opportunity to work under his guidance. He has given help, motivation and taught me how to present the ideas in the term paper. I convey my sincere gratitude to him. I am thankful to Prof. Sekhar Mondal, former Head, Dept. of Computer Science and Engineering, and Prof. Asit Kumar Das, Head, Dept. of Computer Science and Engineering, IIST, Shibpur for extending all facilities to carry out the present study. I am also thankful to my classmates for their valuable inputs and discussions during the preparation of this project. Last but not the least; I would also like to thank my beloved parents for their love and constant support.

Abstract

The Internet of Things (IoT) is growing at a very rapid rate, and total connected devices in IoT is projected to be around 22 billion units by the year 2025. Thus a reliable and efficient routing protocol is needed to support this growth in various applications. Routing protocol in LLNs(RPL) was designed by IETF to meet the different requirement of IoT. A lot of research is going on in this area to modify the standard RPL protocol to make it more suitable for applications in different areas. In this project, we explore the limitations of LLNs, and features of RPL. We explore some limitations of RPL and propose to enhance the objective function of RPL using a composite metric of link quality index and battery depletion index in the nodes; and switch parents based on these values. Also evaluate its performance using Contiki Cooja Simulator; and compare them with standard RPL.

Contents

Abstract	4
List of Figures	6
List of Tables	7
1. Introduction	8
1.1 Introduction to IoT	8
1.2 IoT Applications	8
1.3 Architecture of IoT	10
1.4 IoT Protocol stack	11
2. RPL: The Routing Protocol	13
2.1 Introduction to RPL	13
2.2 Objective Functions	15
2.3 RPL control messages and network topology	15
2.4 RPL network management	21
3. Problem with existing approach.....	24
4. Literature Survey.....	26
5. Proposed Solution	28
5.1 Metrics of Interest	28
5.2 Objective Function design using ETX and BDI as composite routing metric	30
6. Simulation Results Evaluation	32
6.1 Overview of simulation environment	33
6.2 Simulation and Network Setup	38
6.3 Simulation results	39
7. Conclusion and Future Scope	44
References	45

List of Figures

Figure Number	Figure Title	Page
Figure 1.1	The three-layer IoT architecture	10
Figure 1.2	IoT Protocol Stack	11
Figure 2.1	An RPL network with three DODAGs in two instances	14
Figure 2.2	RPL control message packet format	15
Figure 2.3	The DIO message format	16
Figure 2.4	DAG Metric container	17
Figure 2.5	DODAG configuration option	17
Figure 2.6	The DAO message format	18
Figure 2.7	Traffic patterns supported by RPL	19
Figure 2.8	P2P message forwarding using Mode Of Operation MOP 1 and MOP 2	20
Figure 3.1	An Example of DODAG construction and parent node selection	24
Figure 6.1	The different files of RPL module implemented in Contiki	31
Figure 6.2	Code snippets from rpl-mrhof.c file	32
Figure 6.3	Code snippets from udp-client.c file	33
Figure 6.4	Code snippets from tcp-ip.c file	33
Figure 6.5	Code snippets from rpl-icmp6.c file	34
Figure 6.6	Code snippets from uip6.c file	34
Figure 6.7	Code snippets from rpl-conf.h file	35
Figure 6.8	Code snippets from udp-client.c file	35
Figure 6.9	Code snippets from udp-server.c file	36
Figure 6.10	Code snippets from rpl-private.h file	36
Figure 6.11	RPL random topology with 25 nodes	38
Figure 6.12	RPL random topology with 50 nodes	39
Figure 6.13	Packet Delivery Ratio as function of varying number of nodes	40
Figure 6.14	Average Energy consumption of the nodes as a function of varying number of nodes	41
Figure 6.15	End-to-end delay as a function of varying number of nodes	41

List of Tables

Table Number	Table Title	Page
Table 6.1	Simulation Parameters	37
Table 6.1	Results of simulation for network with 25 nodes	38
Table 6.1	Results of simulation for network with 50 nodes	39

Chapter 1

Introduction

1.1 Introduction to IoT

Internet of Things (IoT) is one of the main communication developments in recent years. It makes our everyday objects (e.g. health sensors, industrial equipments, home appliances, vehicles, clothes, etc.) connected to each other and to the Internet. The basic concept behind IoT is the pervasive presence of various wireless technologies such as Radio-Frequency Identification (RFID) tags, sensors, actuators and mobile phones, in which computing and communication systems are seamlessly embedded.

The concept of IoT involves the management of sensors or devices distributed around the network, so as to recognize and notify users instantly about real-time events. These devices, having basic computational skills, are called smart objects. IoT devices are mainly characterized by their constrained resources in terms of power, processing, memory, and bandwidth. Due to this fact, traditional protocols concerning network operations and security cannot be implemented in IoT specific environment, with their current form, therefore IoT specific protocols are required to handle the low-power, low data rate devices.

1.2 IoT Applications

IoT can find its application in almost every aspect of our everyday life. Some of its common applications are summarized below:

Smart Home: Smart home is one of the major applications of IoT, which has achieved great popularity in the last few years. Sensors and actuators are attached to home appliances such as air conditioning, lighting, heating, TVs, computers, refrigerators, security and camera systems etc. These appliances are connected using a Home Area Network (HAN); hence they are capable of communicating with each other and controlled and monitored remotely through internet using a Smartphone application by the owner.

Wearables: Wearable devices are installed with sensors and software which collect data and information about the users. This data is later pre-processed to extract essential insights about

user. These devices broadly cover fitness, healthcare and entertainment requirements. These wearable IoT applications need to be highly energy efficient or ultra-low power and small sized.

Smart City: Smart city offers intelligent, sustainable environments that reduce environmental impact and offer citizens a high-quality life. It comprises of a number of applications spread out in a city which include- efficient water supply, smart traffic management, reliable public transportation, energy-efficient buildings, smart parking, smart street lighting, waste management etc.

Smart Grids: Smart grid system offers global/integrated bi-directional communication between service providers and consumers, involving power generation, transmission, distribution and utilization systems. Smart grids employ various devices for the monitoring, analysis and control of the grid, deployed at power plants, distribution centers and in consumers' premises (e.g, Smart meters) in a very large number, which are connected by IoT. These create data on the consumption and load patterns; data analytics is then used to get insights that help in power generation and transmission decisions.

Industrial IoT: IIoT is used for industrial purposes such as manufacturing, supply chain monitor and management system. IIoT uses more sensitive and precise sensors and actuators including more location-aware technologies with sophisticated advanced control and analytics. It automatically monitors the industrial applications and generates alerts / alarms or takes intelligent decisions using predictive maintenance. IIoT provides improved safety, time and cost savings and other operational efficiencies for industrial organizations.

Connected Car: Connected car technology use an extensive network of sensors, antennas, embedded software and communication technologies to assist in safe navigation, pollution control and traffic management. It also monitors and provides essential real-time information about car conditions to the driver. Vehicle -to-infrastructure (V2I) communication can stream diagnostic data to the car service centre, and help us to locate and reserve vacant parking spots. Vehicle-to-vehicle (V2V) communication, powered by high-speed in-vehicle networks, cameras and radar, will allow cars to locate each other and avoid collisions and streamline traffic.

Connected Healthcare: From personal fitness sensors to surgical robots, IoT in healthcare brings new tools updated with the latest technology in the ecosystem that helps in delivering better healthcare. The IoT device collect and transfer health data: blood pressure, oxygen and blood sugar levels, weight, ECG etc. These data are stored in the cloud and can be shared with a doctor and health insurance company. Real-time monitoring of the patient's condition can save lives in event of a medical emergency.

Smart Farming: Smart farming uses smart agriculture vehicles, drones, autonomous robots and actuators as well as sensor-based systems for monitoring crop, soil moisture, field, weather conditions etc. These systems create important data about the condition of crops; and with data analytics, visualization and management systems, it provides vital insights to farmers. Hence,

IoT in agriculture promises increased productivity, cost efficiency and reducing crop wastage using automation and data driven processes.

1.3 Architecture of IoT

The high-level IoT network architecture consists of three-layers: Perception Layer, Network Layer, and Application Layer [10]. Figure 1.1 below shows this architecture.

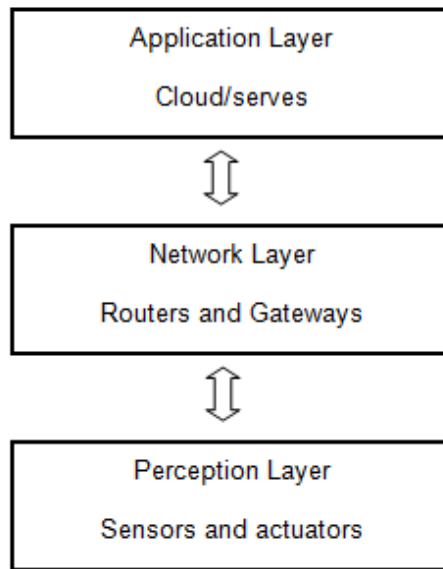


Fig 1.1: The three-layer IoT architecture

Perception Layer: The perception layer represents the physical sensors of the IoT system. It is responsible for collecting information about the environment and processing this information. This layer consists of sensors and actuators that perform different functions such as querying location, temperature, weight, motion, vibration, acceleration, humidity etc. This information is then digitized and transmitted to the Network Layer.

Network Layer: The Network Layer does the processing of the received data from Perception Layer, makes decisions and delivers the required service over the network to the Application Layer. Huge quantities of data will be carried by the network.

Hence, it is crucial to provide a sound middleware to store and process this massive amount of data. To reach this goal, cloud computing is the primary technology in this layer.

Application Layer: The application Layer uses the processed data received from the Network Layer to provide services requested by the customers. This Layer constitutes the front end of the IoT system. The Application Layer also provides various tools for building applications for Smart Home, Smart building, transportation, industrial automation and Smart Healthcare etc.

1.4 IoT Protocol Stack

The TCP/IP protocol suite is applicable to internet applications. However, because of the limitations of nodes in Low power and Lossy networks (LLN) in terms of limited power, memory, bandwidth etc, the same cannot be applied to IoT networks. To meet IoT specific requirements, the Internet Engineering Task force (IETF) has worked on standardizing the corresponding communication protocols for each layer of the communication stack. Namely, IEEE 802.15.4 for the data link and physical layers, IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) as a lightweight addressing scheme, Routing Protocol for Low Power and Lossy Networks (RPL) as a routing protocol, and Constrained Application Protocol (CoAP) to be adopted in the application layer. Figure 1.2 below shows the IoT protocol stack.

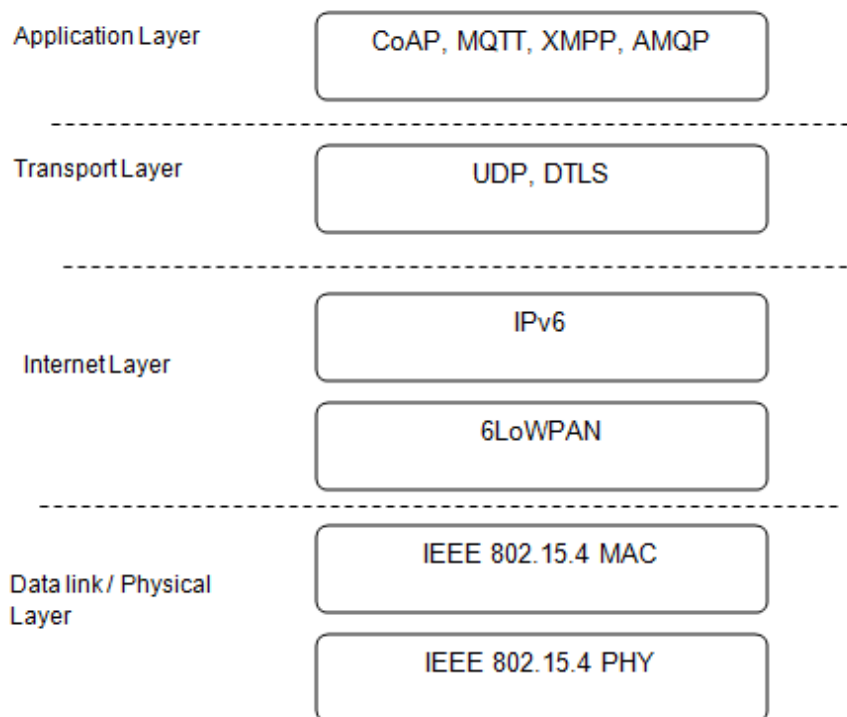


Fig 1.2: IoT Protocol Stack

6LoWPAN adaptation Layer: 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Network) [12] is an open standard defined in RFC 6282 by the IETF. 6LoWPAN was developed to connect IEEE 802.15.4-based resource-constrained IoT devices with IPv6 networking. Hence it enabled the Low-Power Wireless Personal Area Networks (LPWANS) full interoperability with the internet.

The 6LoWPAN protocol is primarily focused on applying IPv6 over MAC and PHY layer of IEEE 802.15.4. The key challenge here was that the maximum frame size allowed by IEEE 802.15.4 is only 127 bytes, while the minimum value of MTU for IPv6 is 1280 bytes. In addition, use of larger packets has a direct impact on battery life of these low-power devices. 6LoWPAN solves this problem by introducing an adaptation layer between the data-link layer and network layer. It compresses IP headers to avoid transmitting repetitive information not needed on the 802.15.4 subnet. It also provided very efficient fragmentation mechanisms for transmission and reassembly of IP packets that did not fit within a single 802.15.4 packet.

Chapter 2

RPL: The Routing Protocol

Different application areas of IoT require different networking requirements. Industry related IoT applications require latency delay to be less, as these are event-driven applications. For Smart Health related applications, network reliability is more important. Whereas Smart farming, forest monitoring and smart city applications need minimum power consumption due to critical condition of environment and remote location.

Since the nodes of low-power and lossy networks (LLNs) are very resource constrained i.e., low-power, limited processing and memory capacity. Therefore, the routing protocols for ad-hoc networks are unable to fulfill the requirements of LLNs. In order to cope with the limitations of the IoT systems, a routing protocol should match the traffic pattern of heterogeneous nodes, and it needs to be resourceful in terms of energy consumption as well as provide support for mobility and security. So IETF working group standardized a routing protocol called IPv6 over Low-power and Lossy Networks (RPL) [1] for LLNs.

2.1 Introduction to RPL

RPL is a distance–vector (DV) and a source-routing protocol that is designed to operate on top of several link-layer mechanisms including IEEE 802.15.4 PHY and MAC layers. LLNs are networks with very limited resources in terms of energy, computation and bandwidth turning them highly exposed to packet losses. RPL has been specifically designed to provide a routing solution to LLNs. This protocol was designed to be highly adaptive to network conditions and to provide alternate routes, whenever default routes are inaccessible.

RPL is based on the topological concept of Directed Acyclic Graphs (DAGs). RPL organizes nodes as Destination-Oriented DAGs (DODAGs), where the most popular destination node providing a default route to the Internet (i.e. sink/gateways) act as the roots of the DAGs.

A network may consist of one or several DODAGs, which together form an RPL instance identified by a unique ID, called RPLInstanceID. A network may run multiple RPL instances concurrently; but these instances are logically independent. A node may join multiple RPL instances, but must only belong to one DODAG within each instance. Figure 2.1 below shows an RPL network with 3 DODAG in 2 instances.

An RPL-based network topology is inherently hierarchical as it forces underlying nodes to self-organize as one or several DODAGs, based on parent-to-child relationship. On the other hand, RPL supports the mesh topology, as it allows routing not only through children-parent, but also through sibling nodes when needed. This combination mesh/hierarchical provides a great flexibility in terms of routing and topology management.

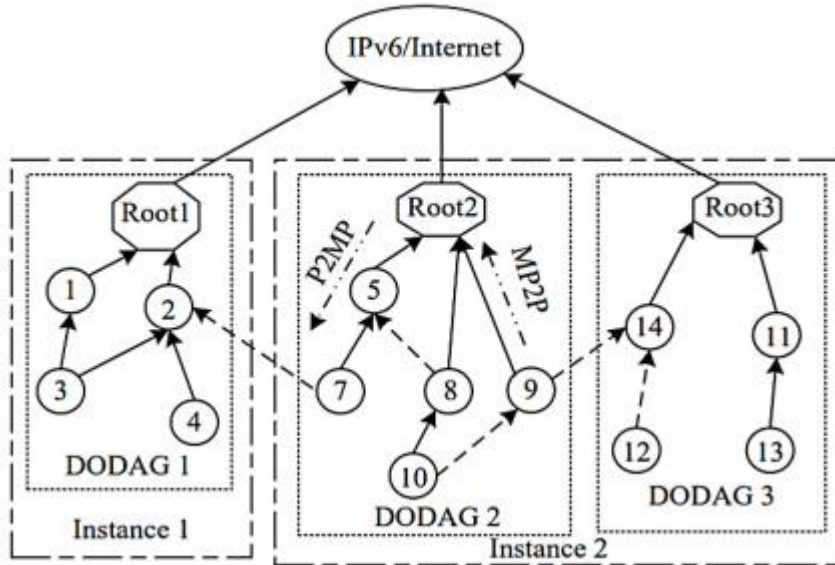


Fig 2.1: An RPL network with three DODAGs in two instances

Network model:

RPL defines three types of nodes [6]:

- Low Power and Lossy Border Routers (LBRs): It refers to the root of a DODAG that represents a collection point in the network and has the ability to construct a DAG. The LBR also acts as a gateway (or edge router) between the Internet and the LLN.
- Router: It refers to a device that can forward and generate traffic. Such a router does not have the ability to create a new DAG, but associate to an existing one.
- Host: It refers to an end-device that is capable of generating data traffic, but is not able to forward traffic.

Each node in the DODAG is assigned a rank. The rank is an integer that indicates a node's location relative to other nodes within the DODAG. The rank increases in the downstream direction, nodes in top receive smaller rank compared to those towards the bottom. The smallest rank is assigned to the DODAG root.

A node can be associated not only to its parent (with lower rank), but also to other sibling nodes (with equal ranks). The rank is used in RPL to detect/avoid routing loops, and allows nodes to

distinguish between their parents and siblings in the DODAG. RPL enables nodes to store a list of candidate parents and siblings that can be used if the currently selected parent loses its routing ability.

In the construction process of network topology, each router maintains a stable set of parents on a path towards the DODAG root in its parent list, and associates itself to a preferred parent, which is selected based on the Objective Function. The Objective Function defines how RPL nodes translate one or more metrics into ranks, and how to select and optimize routes in a DODAG. It is responsible for rank computation based on specific routing metrics (e.g. delay, link quality, connectivity, etc.) and specifies routing constraints and optimization objectives.

2.2 Objective Function

Each RPL node has its predefined objective function (OF), this function carries the metrics upon which nodes select the better parent among competing nodes. There are currently two objective functions presented by the IETF, the first one is Objective Function zero (OF0) [3] which is a simple and basic objective function that has only one metric, it uses hop count to the root as rank of the node to determine its distance from the root and selects the node with the lower (better) rank. The OF0 is designed as a general objective function used as a guide and base for other implementations. The second and the more popular one is the minimum rank with hysteresis objective function (MRHOF) [4]. This function uses the expected transmission count (ETX) as the default metric.

2.3 RPL control messages and network topology

2.3.1 RPL Control messages

RPL messages are specified as a new type of ICMPv6 control messages; its structure is depicted in Figure 2.2. [6]

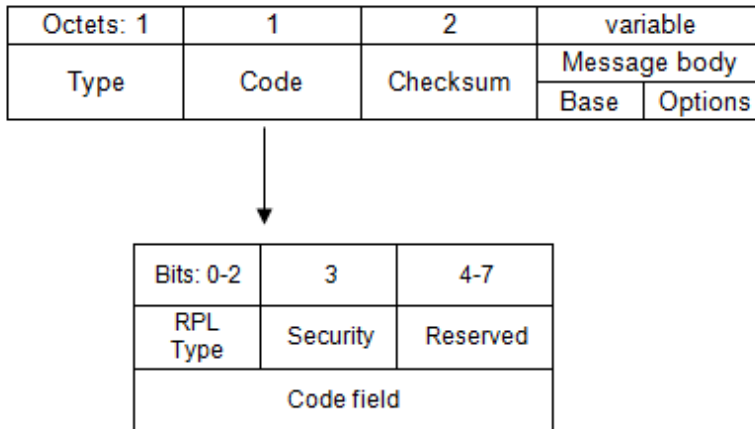


Fig 2.2: RPL control message packet format

The RPL control message is composed of (i) an ICMPv6 header, which consists of three fields: Type, Code and Checksum, (ii) a message body comprising a message base and a number of options.

The Type field specifies the type of the ICMPv6 control message; it is set to 155 in case of RPL. The Code field identifies the type of RPL control message. Four codes are currently defined:

- **DODAG Information Solicitation (DIS):** The DIS message is mapped to 0x00, and is used to solicit a DODAG Information Object (DIO) from an RPL node. The DIS may be used to probe neighbor nodes in adjacent DODAGs.
- **DODAG information object (DIO):** The DIO message is mapped to 0x01, and it is issued by the root node to construct a new DODAG [2]. The format of the DIO Base Object is presented in Figure 2.3. The main DIO Base Object fields are: (i) RPLInstanceID, an 8-bit information assigned by the DODAG root that indicates the ID of the RPL instance that the DODAG is part of (ii) Version number is used within a DODAG; it is incremented upon each network information update, this helps maintaining all nodes synchronized with new updates, (iii) Rank, a 16-bit field that specifies the rank of the node sending the DIO message, (iv) Destination Advertisement Trigger Sequence Number (DTSN) is an 8-bit flag that is used to maintain downward routes, (v) Grounded (G) is a flag indicating whether the current DODAG satisfies the application-defined objective, (vi) MOP identifies the mode of operation of the RPL instance set by the DODAG root. Four operation modes have been defined and differ in terms of whether they support downward routes maintenance and multicast or not. Upward routes are supported by default. Any node joining the DODAG must be able to cope with the MOP to participate as a router, otherwise it will be admitted as a leaf node, (vii) DODAGPreference (Prf) is a 3-bit field that specifies the preference degree of the current DODAG root as compared to other DODAG roots. It ranges from 0x00 (default value) for the least preferred degree, to 0x07 for the most preferred degree, (viii) DODAGID is a 128-bit

IPv6 address set by a DODAG root, which uniquely identifies a DODAG. Finally, DIO Base Object may also contain an Option field.

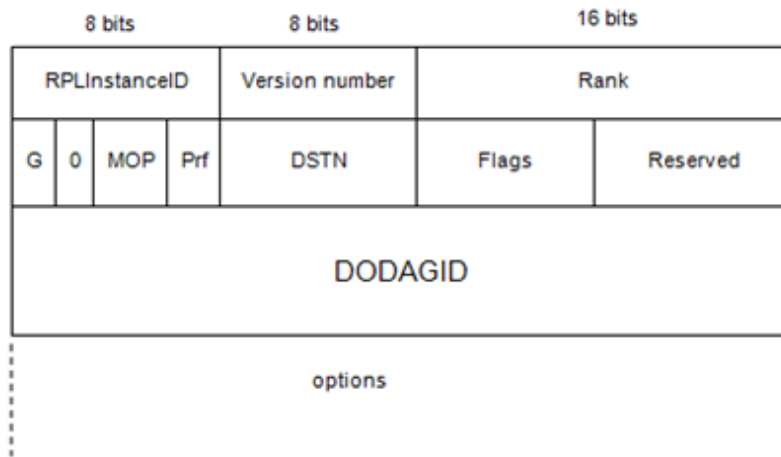


Fig 2.3: The DIO message format

The options field of DIO message may include several the DAG metric container option and configuration option.

The DAG metric container option (shown in Figure 2.4) [8] may exist in both DIO and DAO messages. It carries several routing metric / constraint objects, with flags. Flags are used to specify the nature and features of the routing object, such as whether it represents a metric or constraint, whether it is local or global, whether a metric is additive or others etc.

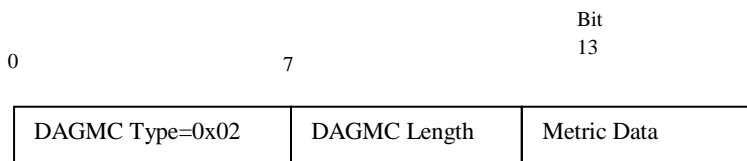


Fig 2.4: DAG Metric container

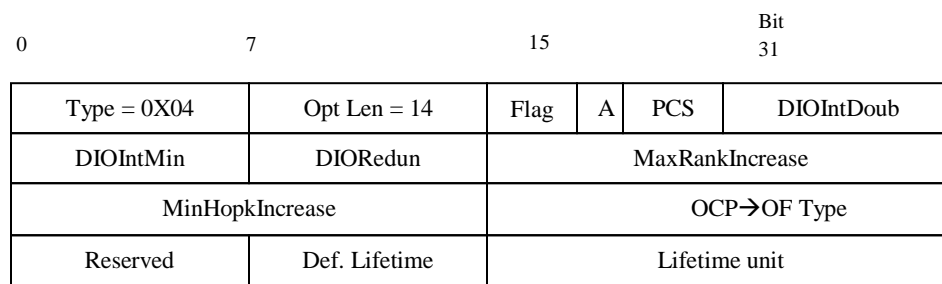


Fig 2.5: DODAG configuration option

The configuration option (shown in Figure 2.5) is used to report the configuration settings in the DODAG. It contains the rank calculation parameters, like MaxRankIncrease and MinHopIncrease. It also contains the trickle timer [5] configuration parameters: -

DIOInt Doub : DIOIntervalDoublings $\rightarrow I_{MAX}$

DIOIntMin: DIOIntervalMin $\rightarrow I_{MIN}$

DIORedun: DIORedundancyConstant \rightarrow Parameter k

- Destination Advertisement Object (DAO): The DAO message is mapped to 0x02. DAO messages are sent by each node other than the DODAG root in upward direction towards the root node. DAO messages populate the routing tables of parent nodes with prefixes of their children nodes. After passing the DAO message in the path from a particular node to the DODAG root through the default DAG routes, a complete downward path between the DODAG root and the node is established. Figure 2.6 illustrates the format of the DAO Base Object. The main DAO message fields are: (i) RPLInstanceID, is an 8-bit information indicates the ID of the RPL instance as learned from the DIO, (ii) K flag that indicates whether an acknowledgment in response to a DAO message is required or not, (iii) DAOSequence is a sequence number incremented at each DAO message, (iv) DODAGID is a 128-bit field set by a DODAG root which identifies a DODAG. This field is present only when flag D is set to 1.
- Destination Advertisement Object (DAO-ACK): The DAO-ACK message is sent as a unicast packet by a DAO recipient (a DAO parent or DODAG root) in response to a unicast DAO message. It carries information about RPLInstanceID, DAOSequence, and Status, which indicate the completion. Status codes are still not clearly defined, but codes greater than 128 mean a rejection and that a node should select an alternate parent.

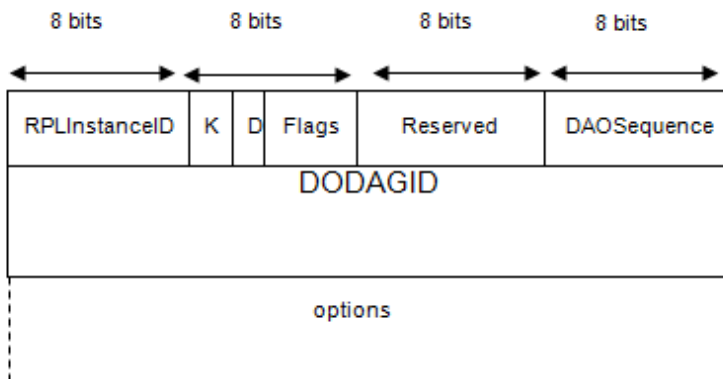


Fig 2.6: The DAO message format

2.3.2 DODAG Construction

The root node initiates the DODAG construction process, sending a DIO message to its neighbors. Each node that receives a DIO message should process it and decide whether to join the DODAG according to the used Objective Function. If it chooses to join the DODAG, it has an upward path to the root node. At this moment, the node computes its rank, refreshes its neighbor table and chooses its preferred parent, which will be used to forward messages to the DODAG root. Usually, the preferred parent is the neighbor node with the lowest rank computed according to the OF. If a node is set to be a router, it should update and forward the DIO message to its neighbors. If a node does not forward the DIO, it is admitted as a leaf node in the routing tree. Each node that receives a DIO should process it and continue the operation until all network nodes can be reached.

When a node already associated with the DODAG receives another DIO message, it can proceed in three different ways (i) discard the DIO message according to some criteria specified by RPL, or (ii) process the DIO message to either maintain its location in an existing DODAG or (iii) improve its location by getting a lower rank in the DODAG based on computing the path cost specified by the Objective Function. Whenever a node changes its rank, it must discard all nodes in its parents' list whose ranks become greater than the newly computed node's rank to avoid routing loops.

RPL permits a new node to join the network at any time. In this case, the new node uses a DIS message to request a DIO message from a node which has already joined the DODAG. On receiving the DIO message, the new node selects its preferred parent according to the OF. [1]

2.3.3 DODAG maintenance:

To maintain the DODAG, each node periodically generates DIO messages triggered by a trickle timer. The key idea of the trickle timer technique [7] is to optimize the message transmission frequency based on network conditions. In a nutshell, the frequency is increased whenever some inconsistent network management information is received for faster recovery from a potential failure, and gradually decreased when the network starts to become stable. The timer duration increases exponentially whenever the timer fires. Initially, the timer duration is set to I_{MIN} , which will be doubled $I_{DOUBLING}$ times until it reaches the maximum value $I_{MAX} = I_{MIN} * I_{DOUBLING}$. For any detected change in the DODAG (e.g. unreachable parent, new parent selection, new DODAG Sequence Number, routing loop, etc.), the trickle timer is reset to I_{MIN} .

2.3.4 Traffic flow:

RPL allows both upward (from the node to its parent) and downward (from the node to its children) routes. Upward routes are naturally created during the initial DODAG construction phase of DIO sending. However, the use of downward routes requires the handling of DAO messages. The DODAG nodes process the DAO messages according to the RPL Mode Of Operation (MOP).

In the Multipoint-to-point (MP2P) traffic pattern, the nodes send data messages to the root, creating an upward flow (Fig 2.7a). In Point-to-Multipoint (P2MP), sometimes termed as multicast, the root sends data messages to the other nodes, producing a downward flow (Fig 2.7b). In Point-to-Point (P2P), the nodes send messages to other (non-root) nodes of the DODAG; thus both upward and downward forwarding may be required (Fig 2.7c).

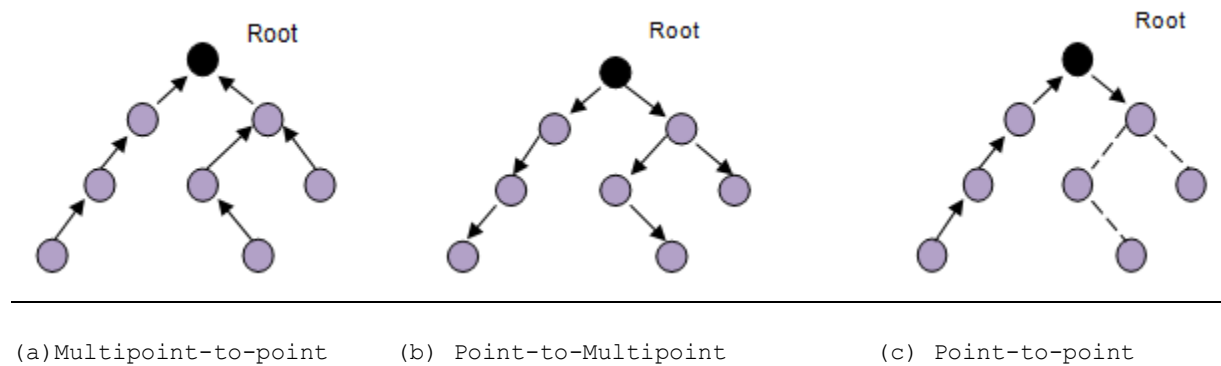


Fig 2.7: Traffic patterns supported by RPL

RPL defines four MOPs that should be used considering the traffic pattern required by the application and the computational capacity of the nodes. In the first, $MOP = 0$, RPL does not maintain downward routes; thus only MP2P traffic is enabled. In non storing MOP ($MOP = 1$) downward routes are supported, and the use of MP2P and P2P are allowed. However, all downward routed are maintained in the root node. Thus the total downward traffic should be initially sent to the DODAG root and subsequently forwarded to its destination by the root (Fig 2.8a). In storing without multicast MOP ($MOP = 2$), downward routes are supported but is different from MOP 1; the nodes maintain individually, a routing table constructed using DAO messages to provide downward traffic. Hence, downward forwarding occurs without the use of root node (Fig 2.8b). Storing with multicast MOP ($MOP = 3$) has the functioning similar to MOP 2 plus the possibility of multicast data sending. This type of transmission permits the non-root node to send messages to a group of nodes formed using multicast DAOs.

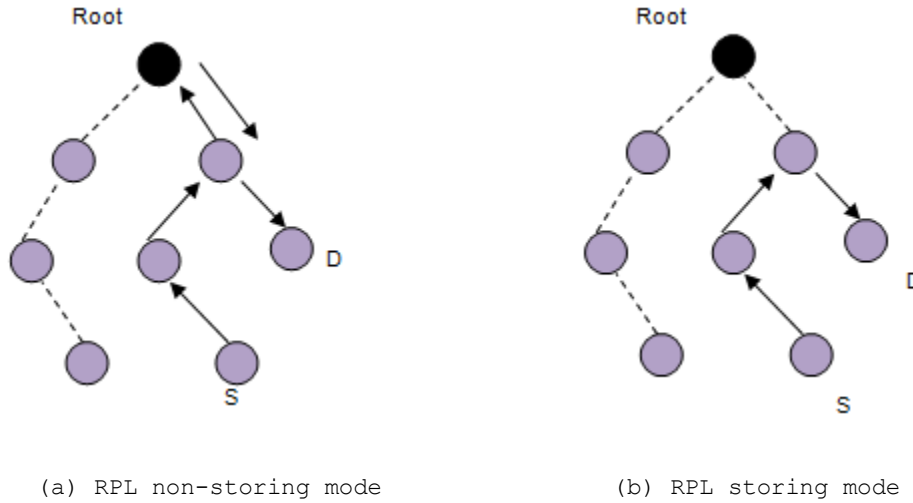


Fig 2.8: P2P message forwarding using Mode Of Operation MOP 1 and MOP 2.

2.4 RPL network management

2.4.1 Fault Tolerance:

→DODAG repair: Repair mechanisms are very important for a routing protocol to dynamically update routing decisions and adapt the network topology to potential node/link failure.

RPL supports two complementary repair mechanisms: (i) global repair and (ii) local repair. [6]

When a node detects a network inconsistency (such as a link between two nodes fails or a local loop is detected), it triggers a local repair operation. It consists of urgently finding a backup path without trying to repair the whole DODAG. This alternate recovery path may not be an optimal path.

If local repairs are not efficient for network recovery due to several inconsistencies, the DODAG root may trigger a global repair operation and then it increments the DODAG version number and initiates a new DODAG version. The global repair operation leads to a reconstruction of the network topology. Nodes in the new DODAG version can choose new parent nodes and calculate their rank accordingly; it is independent of the old DODAG version.

→Loop avoidance and detection: Loops are a common undesirable problem in distance-vector routing protocols. To overcome it, RPL relies on rank-based and path-validation mechanisms for loop avoidance and loop detection.

Loop avoidance: A loop may occur when a node that loses all its parents, for some reasons, and attaches to another node in its own sub-DODAG. This may happen in particular when DIO messages are lost, and referred to as DIO loops. Several rules have been defined to avoid loops. The `max_depth` rule states that a node is not allowed to select another node as a parent whose rank higher than the node's rank+`DAGMaxRankIncrease`. This rule does not prevent loops from occurring but avoids count-to-infinity when a loop is formed.

Another way to avoid loops is that a node may poison its sub-DAG by advertising a rank of `INFINITE_RANK` without having to use `DAGMaxRankIncrease`.

Loop detection: Detection mechanisms rely on the concept of data path validation, which is piggybacking routing control data in data packets by setting flag in the packet header to ensure that a packet is moving in forward direction and not getting trapped in any loop. For instance, the current RPL version allows one-hop sibling path, which means that a packet can be forwarded to a sibling (when the parent node can't be reached directly)only once along its path to the destination. The packet is then dropped in the second attempt to forward the packet to a sibling of another node. This can be encoded with a flag in packet header. This avoids a potential sibling loop.

2.4.2 QoS-Aware Routing:

RPL is a QoS-aware and constrained-based routing protocol. As discussed previously, DIO options field contains the routing metric /constraints objects in DAG Metric container. The main routing metric and constraint objects are:

- **Node State and Attribute (NSA) Object:** This metric reports information on node state and attributes such as CPU overload, available memory. The ROLL working group decided to set a 1-bit flag when a node faces a congestion situation, assessed according to a local policy. If the flag is set, traffic will be re-routed to avoid passing through the congested node.
- **Node Energy Object:** This object is used as a constraint to avoid using nodes with low power level. The Node Energy Object can be a combination of (a) the node's power mode, encoded by 2-bit flag indicating the type of the node's power sources: main-powered, battery powered, or scavenger, (b) the estimated remaining lifetime, which provides an estimation of the remaining power-level for nodes operating with batteries and scavengers
- **Hop Count Object:** This metric simply reports the number of hops crossed along the path between a node and the destination.

- Link Throughput Object: This metric reports the range of throughput that the link can handle, in addition to the current link throughput.
- Link latency Object: This metric is used to report the path latency. It can also be used as a constraint (e.g. pruning all links with latency higher than a certain threshold).
- Link Reliability Object: This metric specifies the link reliability level, which can be expressed in several ways such as packet reception ratio, bit error rate, mean time between failures, and others.

2.4.3 Security:

RPL is expected to support message confidentiality and integrity. It has three basic security modes:

- Unsecured: In this mode, RPL control messages are sent without any additional security mechanisms; it could use other present security primitives to meet the application requirements.
- Pre-installed: in this mode, nodes joining a RPL instance have pre-installed keys that enable them to process and generate secured RPL messages.
- Authenticated: In authenticated mode, nodes have preinstalled keys as in pre-installed mode, but these preinstalled keys may only be used to join a RPL instance as a leaf.

The RPL security services proposed do not address all possible attacks and remain exposed to some threats that may compromise RPL security, such as broadcasting fake messages by a compromised internal node. Therefore, security in RPL still needs further investigation.

Chapter 3

Problem with existing approach

In RPL, The Objective Function calculates the rank of the nodes based on only one routing metric, and then selects and optimizes the routes. In OF0, hop count is used as a single routing metric; and in case of MRHOF, expected transmission count (ETX) is used as a single routing metric. Because of the single metric, both OF0 and MRHOF suffer from long hops when selecting the routes to the sink. In case of OF0, the Objective Function tries to select a path with the minimum number of hops to the sink node. This may choose a poor path which regardless of having few hops, the quality of the links may be bad. In case of MRHOF, the sum of ETX values between each node of a route is used to choose the best route. Since the summation of ETX for a relatively poor-quality route having a few hops might appear to be better than summation of ETX of a good route having more number of hops. But in reality, the long hop route will suffer from more packet loss. This is the long-hops problem.

The other problem with RPL is the unbalanced choice of parents that make bottleneck nodes (i.e., nodes having a lot of children nodes) to cause more network delay and high packet loss ratio, especially the nodes located near the sink node. This problem becomes more severe in case of dense networks (i.e. large number of nodes) with heavy traffic. This is because the Objective Function of RPL (either MRHOF or OF0) does not take into account factors such as the remaining energy level of the nodes or the congestion level of the nodes while choosing next hops or preferred parents for forwarding data packets to the sink. This in turn creates problems such as more packet drops, network delays and eventual break of the network due to battery exhaustion of the overloaded nodes.

This might be illustrated by an example shown in figure 3.1. Node D and E both may choose either B or C as the next hop. However, both D and E choose C as the preferred parent since it offers a better link (low ETX). Eventually nodes D, E and F transmit their own data or forward their child node's data to node C. This will cause more traffic at node C; and consequently the energy of node C will deplete more quickly. Eventually the communications between nodes D/E/F and node C will be disconnected because of the depletion of node C's energy. Instead of choosing node C, node B should have been chosen as the preferred parent by node D to balance the energy consumption as well as traffic.

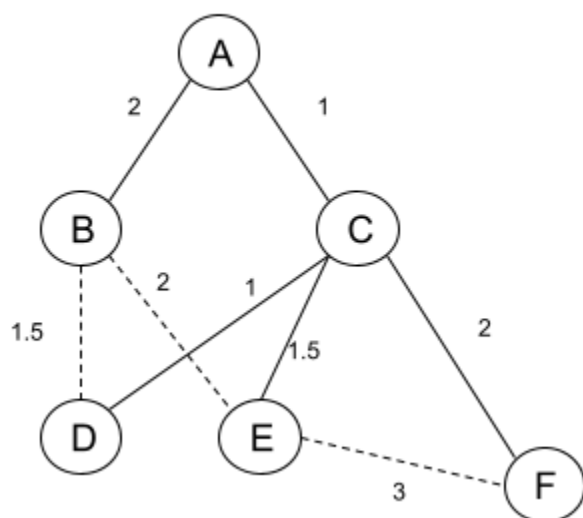


Fig 3.1: An Example of DODAG construction and parent node selection. The numbers beside the edges denote ETX values of the links.

Chapter 4

Literature Survey

Although RPL greatly satisfies the requirements for LLN for IoT applications, some issues remain open for improvement such as energy aware routing and effect of congestion on QoS. In recent years, many solutions have been proposed to address these issues.

Energy conservation is an important issue in RPL. The use of transmission delay and residual energy as a routing metric is done in paper [19]. Selection and rank calculation adopts an Ant Colony Optimization (ACO) method for route metric calculation.

Sausa et al. [20] proposed an Energy Efficient and Path Reliability Aware Objective Function (ERAOF) for IoT applications for reliable data transmission and energy-efficient communication.

Kamgueu et al [21] and Xu et al [22] proposed to use residual energy as a routing metric. However, they do not consider the radio link quality.

Iova et al [23] proposed a mechanism to anticipate the remaining energy in bottleneck nodes and proposed a new metric called Expected Life Time (ELT). Accordingly, the authors proposed multiple parents in RPL to boost the network lifetime, which can be achieved through mitigating the number of DODAG reconstructions.

Gaddour and Kouba [24] proposed QoS-aware fuzzy logic based Objective Function (OF-FL). It combines several routing metrics such as hop count, link quality, residual energy and latency as the metrics in the process of path optimization. However, it is not clear which metric should be optimized for the specific purpose.

Several studies have been carried out for the solution of load-balancing issue in RPL.

Kim et al proposed Queue Utilization RPL (QU-RPL) [25] for load balancing under heavy traffic. They observed the problem of unbalanced load in TinyRPL. TinyRPL implements the RPL standard with OF0 ; it uses hop count for rank calculation and ETX for parent selection. QU-RPL takes the queue utilization of the neighboring nodes into consideration when selecting

parent nodes. The authors described Queue Utilization (QU) as the number of packets in the queue divided by the total queue size.

In [26] an adaptive parent selection mechanism in RPL for advanced metering infrastructure network named energy and congestion aware routing metric (ECRM) has been proposed. ECRM considers the residual energy and queue utilization of neighboring nodes as a metric for parent selection criteria. It minimizes the energy consumption and improves the packet delivery ratio of the network.

In [27], the authors have proposed an energy-efficient and reliable composite metric called lifetime and latency aggregatable metric(L2AM). A node running L2AM first computes the relation of transmitting power and residual power using an exponential equation, which is called the primary metric. To obtain the total metric cost, the ETX metric is multiplied by the primary metric, and this cost needs to be minimized when selecting the parent node.

In [28], the authors used three routing metrics , viz ETX, delay and energy. These were combined using a two-stage fuzzy-system. In the first stage, the delay and ETX are mixed to get the QoS. In second stage, energy and the computed QoS values are combined. The suggested fuzzy-based system was tested using a testbed of 28 sensor nodes; and compared with the standard RPL. It shows that the proposed solution outperforms ETX by upto 20% in terms of packet-loss ratio. Also they shows the proposed solution produces more stable route topology denoted by less parent switching per hour.

In [29], the authors implemented a RPL OF called CA-OF. Along with ETX, this OF chooses a new routing metric called buffer occupancy (BO). It applies adaptive weight assignment to ETX and BO values, depending on the traffic congestion situation around a node. Thus it forms an additive metric which depends on ETX metric when selecting a candidate parent with low-traffic; and as the candidate parent node appears to be congested, the ETX metric is ignored and only the BO value of the candidate parent considered for parent selection. According to the authors, CA-OF achieves better network performance in terms of power-consumption, reliability and throughput.

Chapter 5

Proposed Solution

In RPL, the routing parameters, constraints and local policies can be freely chosen to formulate the objective function for parent selection and route establishment process. In this project, we propose a simple yet effective objective function design wherein we design the objective function based on combination of two routing metrics: expected transmission count (ETX) and battery depletion index of the nodes. This significantly improves the network lifetime by preserving the energy consumption of the nodes in the network. Also it improve the end-to-end packet delivery ratio (PDR) compared to the standard RPL.

5.1 The metrics of interest

1. Expected Transmission Count (ETX): ETX is link reliability metric. It is the expected number of transmissions required for a data packet to be delivered successfully. This metric helps in selecting a path with relatively good communication links. Lower value of ETX means better link quality. The best possible ETX value = 1, and the value increases as the link quality deteriorates. ETX is an additive metric that adds the ETX of each link to get the path ETX.

$$ETX (N_i) = \frac{1}{PDR_{s \rightarrow d} * PDR_{d \rightarrow s}}$$

where PDR $s \rightarrow d$ is the packet delivery ratio from source to destination node; and PDR $d \rightarrow s$ is the packet delivery ratio (acknowledgement) from destination to source.

2. Battery Depletion Index (BDI): This metric indicates the percentage of energy depleted from the node's battery. It is calculated from the energy consumed (EC) by the node from the beginning of its operation.

Based on the different operating states, the energy consumed by each node is calculated. A node can be in four different states at any time: Listen mode (which includes listen, receive (RX) and idle modes), transmission mode (TX), processing mode (CPU mode) and low-power mode (sleep mode).

Energy consumption by the node i , $EC(i)$ can be calculated based on the power consumed by the node in each of the four possible states in which the node can operate. The energy estimation module of Contiki, called Energest will be used to obtain information about how long the system spends in different states

$$\text{Energy Consumption, } EC(i) = \frac{\text{Energest Value} * \text{Current} * \text{Voltage}}{RTIMER_SECOND}$$

Energest Values or the time spent in each of the states is the number of ticks, which CPU is running, and it is obtained from Energest module. The current values in each of the four states are obtained from the Tmote Sky datasheet [30]. The initial battery voltage level is 3V and RTIMER_SECOND is the constant number of ticks/second in platform (32768/s).

Current energy of each node is calculated with the following equation:

$$E_{\text{Current}} = E_{\text{Initial}} - E_{\text{consumed}}$$

Based on this, the residual energy ratio (RER) of the node is calculated as:

$$RER_{Ni} = \frac{E_{\text{Initial}}}{E_{\text{current}}}$$

The Battery Depletion Index (BDI) of the node N_i is calculated as:

$$BDI(N_i) = \frac{\text{Energy consumed}(EC_i)}{\text{Initial Node energy}}$$

The BDI of the path P_x is obtained as product of $BDI(N_i)$ values of all the nodes in the path.

$$BDI(P_x) = \prod_{i=1}^N BDI(N_i)$$

.

5.2 Objective Function design using ETX and BDI as composite routing metric

This objective function using both ETX and battery depletion index (BDI) as routing metric can be used to select routing paths which have better links for packet delivery ratio as well as avoiding nodes which are low on energy.

5.2.1 Rank Calculation

The parent selection is based on DODAG rank of the candidate parent nodes. The DODAG rank is calculated from parent rank and rank increase value. The rank increase is calculated from the objective function step value and MinHopRankIncrease. (Default value of MinHopRankIncrease = 256).

$$\text{Step} = w1 * \text{ETX} + w2 * \text{BDI}$$

Here $w1$ and $w2$ are weights given to the metrics. We can give equal weights to both the metrics by taking $w1=w2=1/2$

$$\text{Rank_Increase} = \text{Step} + \text{MinHopRankIncrease}$$

$$\text{Rank}(N) = \text{Rank}(\text{Parent node}) + \text{Rank_Increase}$$

$$\text{Finally, DAGRank}(N) = \text{floor}\left(\frac{\text{Rank}(N)}{\text{MinHopRankIncrease}}\right)$$

5.2.2. Preferred parent selection:

During the DODAG construction phase, the root node sends DIO message to its neighbors. After those nodes calculate their rank and join the DODAG, they send their DIO message further down the network. This process continues until the entire DODAG has been constructed. Later, when a participant node wants to join the DODAG, it sends DODAG Information Solicitation (DIS) message to DODAG. In response the DODAG nodes in close proximity sends DIO message to it. The DIO message transmission is based on the trickle timer algorithm. Whenever there is any link inconsistency, node failure due to battery exhaustion or if there is any loop detected, trickle

timer will reset the interval to the minimum interval I_{min} and increase the frequency of DIO control message. At this point, a node can switch from its preferred parent to other candidate parent in case there is a better parent available in its parent list.

Parent Selection Algorithm using ETX and BDI

```

1  : Input
    Node(M), Parent_Node_ID, SenderNode_ParentID, Candidate_Set(S),
    BestParent_Rank=INF;
2  : Output
    Preferred parent (PP) selection
3  : Start
4  :   for all candidate parent node  $\in S$  do
5  :       update the ETX value and BDI values for each node in S
6  :       Calculate  $Rank(M) \leftarrow Rank(Parent\_Node) + Rank\_Increase$ 
7  :        $Rank\_Increase \leftarrow Step + MinHopRankIncrease$ 
8  :        $Step \leftarrow w1*ETX + w2*BDI$ 
9  :       If Best_Parent_Rank  $\geq$  PP_Rank then
10 :         Best_Parent_Rank  $\leftarrow$  PP_Rank
11 :       End If
12 :   End for
13 :   While PP_Rank = Best_Parent_Rank do
14 :       Participant_Node_Parent_ID PP_Node_ID;
15 :   End

```

If the new rank of the node is less than rank of existing parent node, then the sender node switches to this new node as preferred parent.

Chapter 6

Simulation Results Evaluation

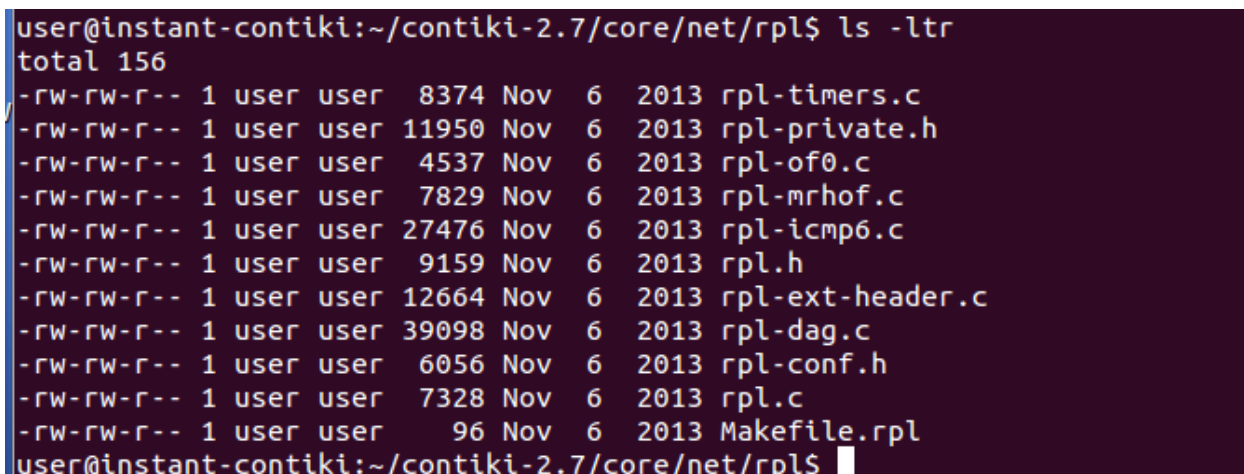
Different tools and simulations platforms are available for the evaluation of IoT network protocols.

6.1 Overview of simulation environment

The proposed protocol will be implemented using the Contiki OS; and its associated Cooja simulator. Contiki is an open source operating system for wireless sensor networks, dedicated to the Internet of Things. Being able to run on microcontrollers with 64 to 256 KB flash and 16 to 32 KB of RAM, the software loads a network stack IPv6/ 6LoWPAN and RPL routing protocol. The operating system has the ContikiRPL protocol which is a reliable implementation of RPL designed by the Swedish Institute of Computer Science (SICS), which has also developed and implemented the ContikiMAC protocol.

Cooja is a java-based interface to be used for the network simulation which comes with Contiki OS. It provides real hardware platforms for simulation wherein a sensor mote in Cooja is a compiled version of Contiki system. We use the Tmode sky as the sensor nodes in the IoT network.

The implementation of ContikiRPL consists of several sub-modules, as shown in figure 6.1.



```
user@instant-contiki:~/contiki-2.7/core/net/rpl$ ls -ltr
total 156
-rw-rw-r-- 1 user user 8374 Nov 6 2013 rpl-timers.c
-rw-rw-r-- 1 user user 11950 Nov 6 2013 rpl-private.h
-rw-rw-r-- 1 user user 4537 Nov 6 2013 rpl-of0.c
-rw-rw-r-- 1 user user 7829 Nov 6 2013 rpl-mrhof.c
-rw-rw-r-- 1 user user 27476 Nov 6 2013 rpl-icmp6.c
-rw-rw-r-- 1 user user 9159 Nov 6 2013 rpl.h
-rw-rw-r-- 1 user user 12664 Nov 6 2013 rpl-ext-header.c
-rw-rw-r-- 1 user user 39098 Nov 6 2013 rpl-dag.c
-rw-rw-r-- 1 user user 6056 Nov 6 2013 rpl-conf.h
-rw-rw-r-- 1 user user 7328 Nov 6 2013 rpl.c
-rw-rw-r-- 1 user user 96 Nov 6 2013 Makefile.rpl
user@instant-contiki:~/contiki-2.7/core/net/rpl$
```

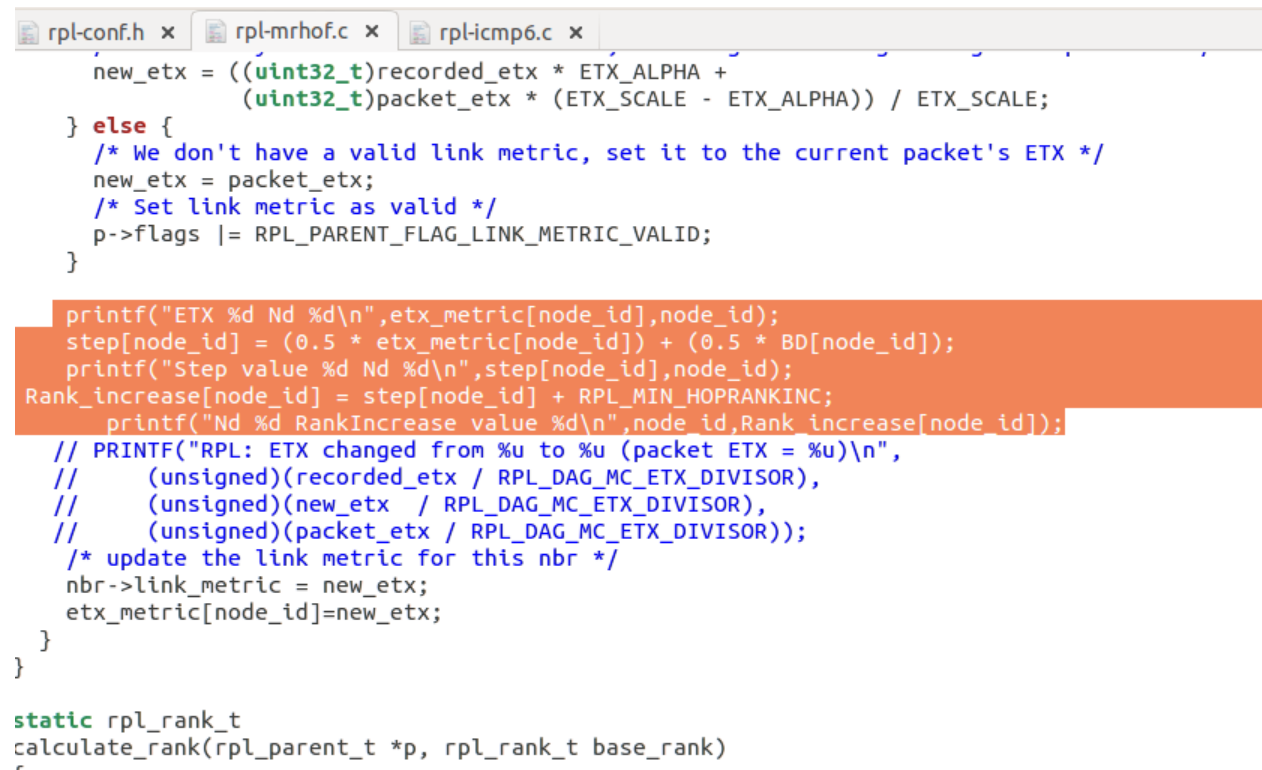
Fig 6.1: The different files of RPL module implemented in Contiki

The RPL directory contains several files implementing different RPL functions:

- a) rpl-times.c is responsible for sending periodic updates using DIO messages.
- b) rpl-dag.c contains the necessary functions for DAG manipulations.
- c) rpl.c is the main RPL function for modifying the IPv6 routing entries.
- d) rpl-icmp6.c is the ICMP message functions for RPL protocol
- e) rpl-of0.c is the implementation of Objective Function zero (OF0)
- f) rpl-mrhof.c contains the implementation of MRHOF OF.
- g) rpl-ext-header.c contains management of extension headers for ContikiRPL.

Additionally, for the implementation of our proposed objective function, the path of some important files is mentioned below:

The ETX, step value and rank increase value is calculated from rpl-mrhof.c file and the DAG rank is calculated from udp-client.c file



```
rpl-conf.h x rpl-mrhof.c x rpl-icmp6.c x
new_etx = ((uint32_t)recorded_etx * ETX_ALPHA +
           (uint32_t)packet_etx * (ETX_SCALE - ETX_ALPHA)) / ETX_SCALE;
} else {
    /* We don't have a valid link metric, set it to the current packet's ETX */
    new_etx = packet_etx;
    /* Set link metric as valid */
    p->flags |= RPL_PARENT_FLAG_LINK_METRIC_VALID;
}

printf("ETX %d Nd %d\n", etx_metric[node_id], node_id);
step[node_id] = (0.5 * etx_metric[node_id]) + (0.5 * BD[node_id]);
printf("Step value %d Nd %d\n", step[node_id], node_id);
Rank_increase[node_id] = step[node_id] + RPL_MIN_HOPRANKINC;
printf("Nd %d RankIncrease value %d\n", node_id, Rank_increase[node_id]);
// PRINTF("RPL: ETX changed from %u to %u (packet ETX = %u)\n",
//         (unsigned)(recorded_etx / RPL_DAG_MC_ETX_DIVISOR),
//         (unsigned)(new_etx / RPL_DAG_MC_ETX_DIVISOR),
//         (unsigned)(packet_etx / RPL_DAG_MC_ETX_DIVISOR));
/* update the link metric for this nbr */
nbr->link_metric = new_etx;
etx_metric[node_id] = new_etx;
}
}

static rpl_rank_t
calculate_rank(rpl_parent_t *p, rpl_rank_t base_rank)
{
```

Fig 6.2: Code snippets from rpl-mrhof.c file

```

}
/*-----*/
static void
send_packet(void *ptr)
{
    static int seq_id;
    char buf[MAX_PAYLOAD_LEN];

    seq_id++;

    PRINTF("SENSOR node send 'Sensed_data %d' to %d\n", node_id, server_ipaddr.u8[sizeof(server_ipaddr.u8) - 1]);

    dag_rank[node_id] = floor(Rank[node_id] / RPL_MIN_HOPRANKINC);
    printf("Node %d dag_Rank %d\n", node_id, dag_rank[node_id]);

    uip_udp_packet_sendto(client_conn, buf, strlen(buf),
                          &server_ipaddr, UIP_HTONS(UDP_SERVER_PORT));
}
/*-----*/
static void
print_local_addresses(void)
{
    int i;
    u_int8_t state;

    PRINTF("Client IPv6 addresses: ");

```

Fig 6.3: Code snippets from udp-client.c file

The energy consumption is defined in tcpip.c file and the BDI calculation is done by rpl-icmp6.c file:

```

unsigned long cpu_time, lpm_time, rx_time, tx_time;
unsigned long energy_consumed, t1, t2, t3, t4, t11, t22, t33, t44;

lpm_time = energest_type_time(ENERGEST_TYPE_LPM);
cpu_time = energest_type_time(ENERGEST_TYPE_CPU);
rx_time = energest_type_time(ENERGEST_TYPE_LISTEN);
tx_time = energest_type_time(ENERGEST_TYPE_TRANSMIT);

t1 = 1.8 * cpu_time;
t2 = 0.0545 * lpm_time;
t3 = 20.0 * rx_time;
t4 = 17.7 * tx_time;

t11 = (unsigned long) (((20.0 * rx_time + 17.7 * tx_time) * 3) / RTIMER_SECOND);
t22 = (unsigned long) (((1.8 * cpu_time + 0.0545 * lpm_time + 20.0 * rx_time + 17.7 * tx_time) * 3) / RTIMER_SECOND);
t33 = (unsigned long) (((t22 * (1.8 * cpu_time + 0.0545 * lpm_time)) * 3) / RTIMER_SECOND);
t44 = t33/1000;

energy_consume[node_id] = t44;
current_energy[node_id] = RPL_INITIAL_ENERGY - energy_consume[node_id];

```

Fig 6.4: Code snippets from tcp-ip.c file

```

RER[node_id]=((RPL_INITIAL_ENERGY*100.0)/current_energy[node_id]);
if(RER[node_id] < 1) {
RER[node_id]=1;
}
BD[node_id]=(((RPL_INITIAL_ENERGY+current_energy[node_id])*100.0)/current_energy[node_id]);
int bd;
bd=BD[node_id];
int count=0;
do{
count++;
bd/=10;
}while(bd != 0);

```

Fig 6.5: Code snippets from rpl-icmp6.c file

The data aggregation method is defined in uip6.c file

```

}

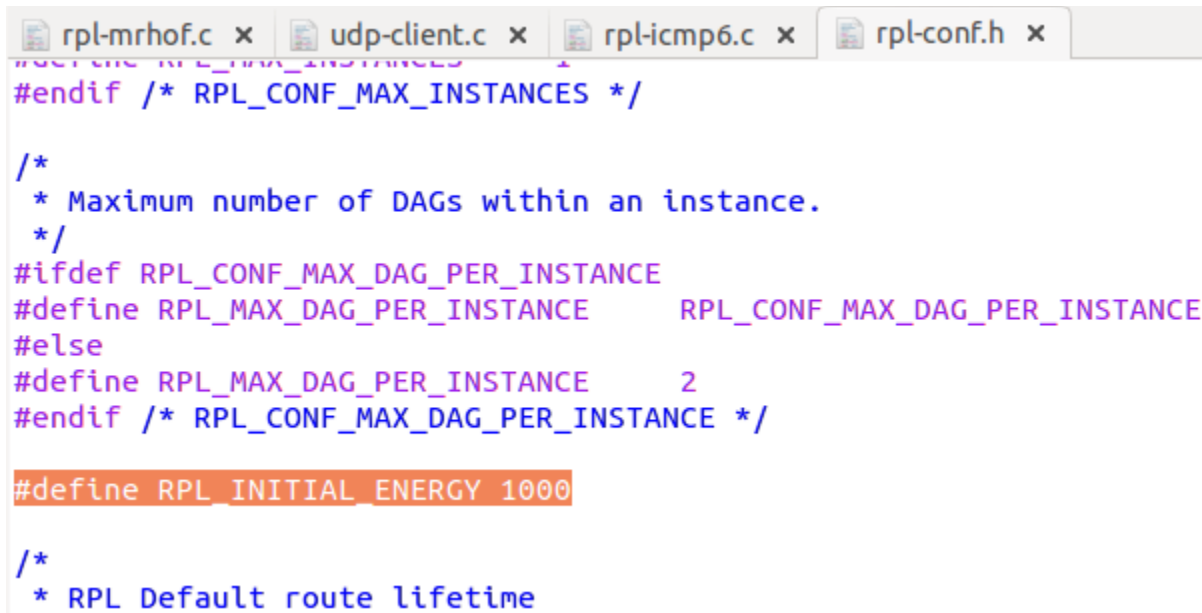
////////data aggregation //////////
uip_ipaddr_t from;
uint8_t hops;
hops = uip_ds6_if.cur_hop_limit - UIP_IP_BUF->tll + 1; //hops count estimation
if(hops > 10 ) {
hops=1;
}
int srcadd;
uip_ipaddr_copy(&from, &UIP_IP_BUF->srcipaddr);
srcadd=from.u8[sizeof(from.u8)-1];
n=random_rand() % 10; //estimate random number
if(n<0) {
n=n*(-1);
}
n=n+node_id+srcadd; //aggregate the all received data from the downward node
n=n/hops; // average the aggregate data
sprintf(ss, \"%d\", n);
app = ss;
uip_appdata = app; //attach the aggregate data to uip buffer
dat=(char *)uip_appdata;
printf(\"note %d send data ' %s ' through the %d hop path\\n\", node_id, dat, hops);

////////

```

Fig 6.6: Code snippets from uip6.c file

The RPL initial energy value is filed in rpl-conf.h file



The screenshot shows a code editor with four tabs: rpl-mrhof.c, udp-client.c, rpl-icmp6.c, and rpl-conf.h. The rpl-conf.h tab is active, displaying the following code snippets:

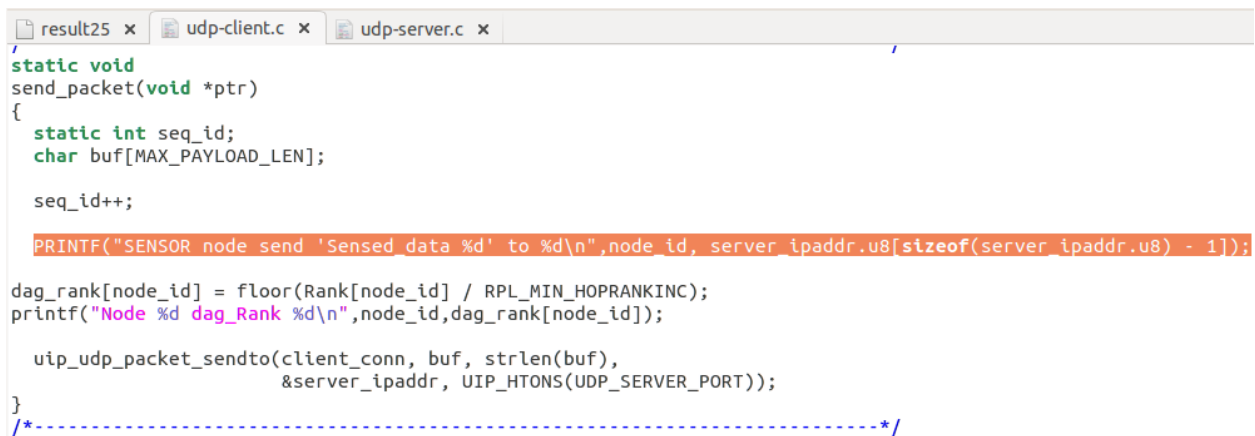
```
#endif /* RPL_CONF_MAX_INSTANCES */

/*
 * Maximum number of DAGs within an instance.
 */
#ifdef RPL_CONF_MAX_DAG_PER_INSTANCE
#define RPL_MAX_DAG_PER_INSTANCE    RPL_CONF_MAX_DAG_PER_INSTANCE
#else
#define RPL_MAX_DAG_PER_INSTANCE    2
#endif /* RPL_CONF_MAX_DAG_PER_INSTANCE */

#define RPL_INITIAL_ENERGY 1000

/*
 * RPL Default route lifetime
```

Fig 6.7: Code snippets from rpl-conf.h file



The screenshot shows a code editor with three tabs: result25, udp-client.c, and udp-server.c. The udp-client.c tab is active, displaying the following code snippets:

```
static void
send_packet(void *ptr)
{
    static int seq_id;
    char buf[MAX_PAYLOAD_LEN];

    seq_id++;

    PRINTF("SENSOR node send 'Sensed data %d' to %d\n", node_id, server_ipaddr.u8[sizeof(server_ipaddr.u8) - 1]);

    dag_rank[node_id] = floor(Rank[node_id] / RPL_MIN_HOPRANKINC);
    printf("Node %d dag_Rank %d\n", node_id, dag_rank[node_id]);

    uip_udp_packet_sendto(client_conn, buf, strlen(buf),
                          &server_ipaddr, UIP_HTONS(UDP_SERVER_PORT));
}

/*-----*/
```

Fig 6.8: Code snippets from udp-client.c file

```

result25 x  udp-client.c x  udp-server.c x
static void
tcpip_handler(void)
{
    char *appdata;

    if(uiplib_newdata()) {
        appdata = (char *)uiplib_appdata;
        appdata[uiplib_datalen()] = 0;
        uint8_t hops;

        hops = uiplib_ds6_if.cur_hop_limit - UIP_IP_BUF->tll + 1;

        PRINTF("SENSOR node recv 'aggregate data %s' from %d through %d hops\n", appdata, UIP_IP_BUF->srcipaddr.u8[sizeof(UIP_IP_BUF->srcipaddr.u8) -
1], hops);
        PRINTF("\n");
        #if SERVER_REPLY
        PRINTF("DATA sending reply\n");
        uiplib_ipaddr_copy(&server_conn->ripaddr, &UIP_IP_BUF->srcipaddr);
        uiplib_udp_packet_send(server_conn, "Reply", sizeof("Reply"));
        uiplib_create_unspecified(&server_conn->ripaddr);
        #endif
    }
}
/*-----*/

```

Fig 6.9: Code snippets from udp-server.c file

RPL parameter MinHopRankIncrease value is maintained in rpl-private.h file

```

rpl-conf.h x  rpl-mrhof.c x  rpl-icmp6.c x  tcpip.c x  uip6.c x  udp-clier
#define RPL_ZERO_LIFETIME 0

#define RPL_LIFETIME(instance, lifetime) \
    ((unsigned long)(instance)->lifetime_unit * (lifetime))

#ifndef RPL_CONF_MIN_HOPRANKINC
#define RPL_MIN_HOPRANKINC 256
#else
#define RPL_MIN_HOPRANKINC RPL_CONF_MIN_HOPRANKINC
#endif
#define RPL_MAX_RANKINC (7 * RPL_MIN_HOPRANKINC)

#define DAG_RANK(fixpt_rank, instance) \
    ((fixpt_rank) / (instance)->min_hoprankinc)

/* Rank of a virtual root node that coordinates DAG root nodes. */
#define BASE_RANK 0

/* Rank of a root node. */
#define ROOT_RANK(instance) (instance)->min_hoprankinc

#define INFINITE_RANK 0xffff

```

Fig 6.10: Code snippets from rpl-private.h file

6.2 Simulation and Network Setup

We conduct the simulation of our proposed approach using Cooja Simulator; and evaluate the its network performance against the standard RPL with MRHOF OF. We compare the performance of both the protocols on the following parameters:

- Packet delivery ratio with varying number of node density
- Energy consumption of the nodes
- Latency delay

We use medium size of network (25 – 50 nodes) using sky motes, and in random topology of Cooja.

Table 6.1: Simulation Parameters.

Parameter	Value
Operating System	Contiki 3.0
Simulator	Cooja
Number of nodes	25-50 nodes
Routing protocol	RPL protocol
Transmission Range	50 m
Area	300m × 300m
Simulation Time	300 sec
Objective Function	MRHOF, Composite RPL
Radio medium model	UDGM – Distance loss
Mote type	Tmote sky
Topology	Random topology
Full Battery	1000 mJ
RPL parameter	MinHopRankIncrease = 256

6.3 Simulation results

6.3.1 First scenario: Network with 25 nodes:

Figure 6.11 shows the random distribution of 25 nodes. The standard RPL protocol (MRHOF) and the proposed protocol are implemented on this network.

The evaluated data is shown in the below table 6.2:

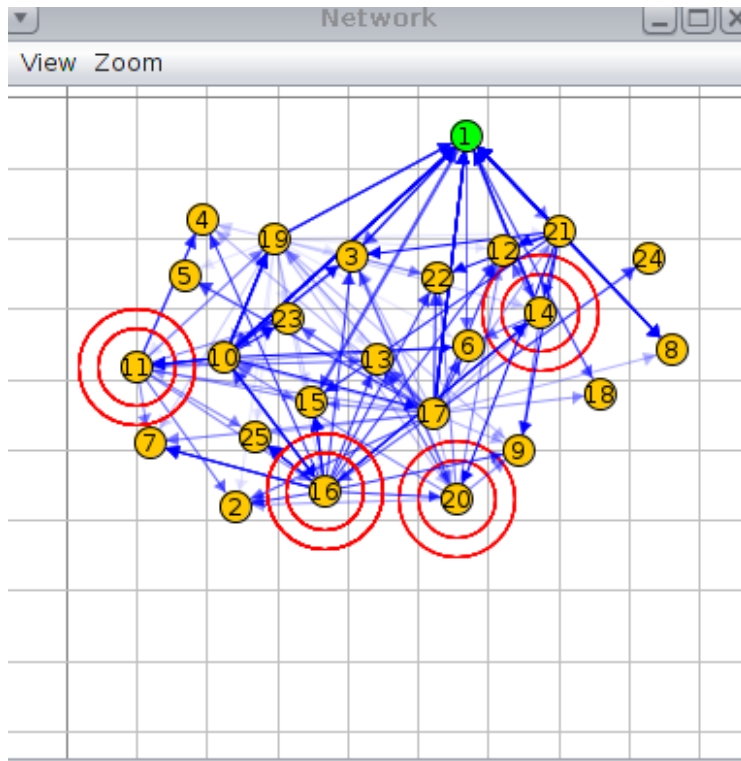


Fig 6.11: RPL random topology with 25 nodes

Table 6.2: Results of simulation for network with 25 nodes.

Performance metric	Test case 1		Test case 2	
	MRHOF	Proposed	MRHOF	Proposed
Packet Delivery Ratio (PDR)	98.18867	98.26964	98.18814	98.26589
Energy consumption (joules)	0.0019819	0.0012928	0.0019814	0.0012929
Delay	29.6252	29.7600	29.5428	29.6548

6.3.2 Second scenario: Network with 50 nodes

In the second scenario, we have taken randomly distributed topology with 50 nodes. Both the standard MRHOF RPL as well as our proposed composite metric RPL have been tested with this network as shown in the figure 6.12.

The evaluated data is shown in the below table 6.3:

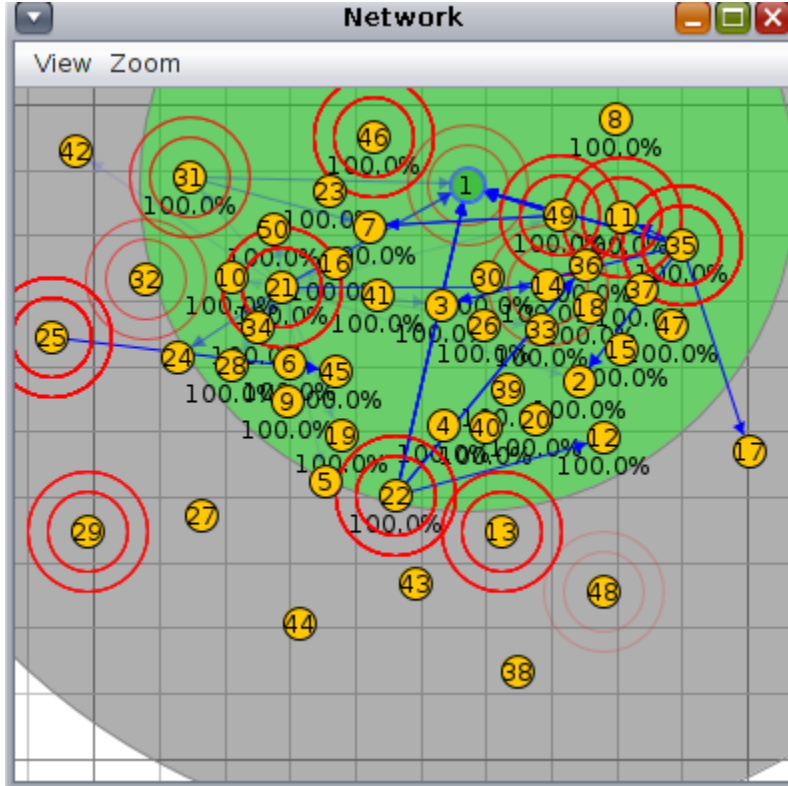


Fig 6.12: RPL random topology with 50 nodes

Table 6.3: Results of simulation for network with 50 nodes.

Performance metric	Test case 1		Test case 2	
	MRHOF	Proposed	MRHOF	Proposed
Packet Delivery Ratio (PDR)	98.31150	98.52335	98.31132	98.32583
Energy consumption (joules)	0.0016190	0.0013051	0.0016191	0.0013056
Delay	29.7439	29.9342	29.4328	29.7516

6.3.3 Performance comparison

Figure 6.13 below shows the PDR with varying number of nodes in both the protocols. It can be seen that the proposed composite RPL succeeds in achieving better PDR as compared to the normal MRHOF RPL.

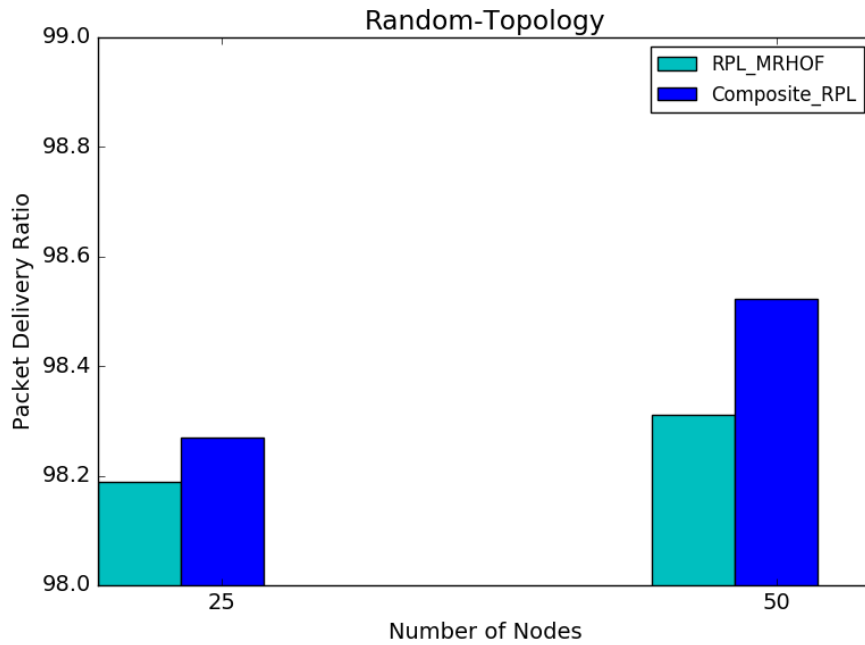


Fig 6.13: Packet Delivery Ratio as function of varying number of nodes

Figure 6.14 below shows the comparison of average energy consumption of the nodes in the network with 25 and 50 nodes, for both protocols. The energy consumption results for the proposed scheme shows good results for both the scenarios; as less energy consumed by the nodes directly contributes to the lifetime of the IoT network.

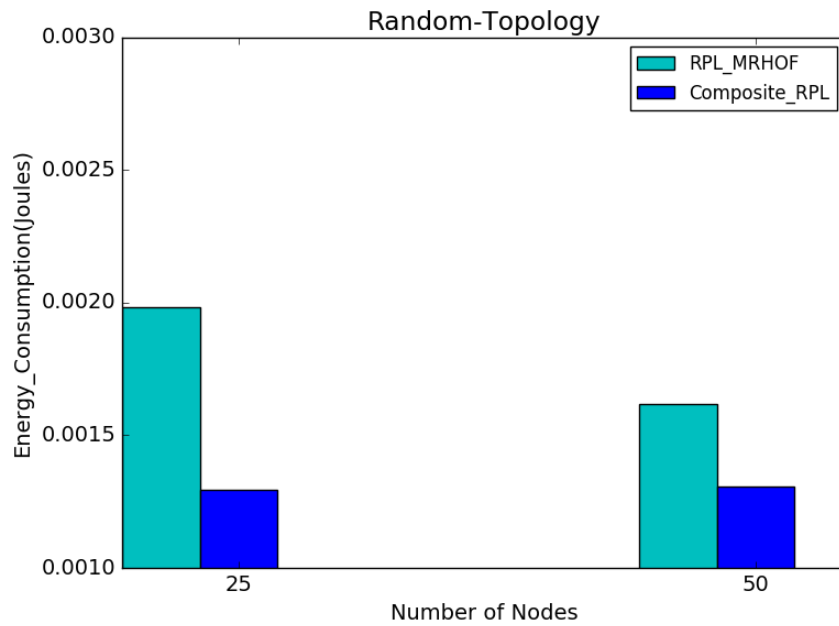


Fig 6.14: Average Energy consumption of the nodes as a function of varying number of nodes

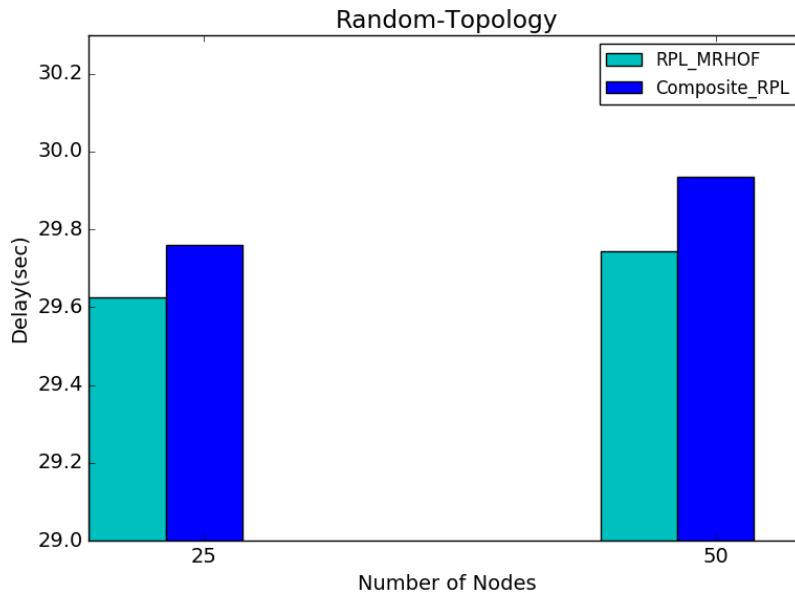


Fig 6.15: End-to-end delay as a function of varying number of nodes

Figure 6.15 above shows the end-to-end delay. Here the delay is slightly more in case of the proposed design as compared to the MRHOF RPL. This is understandable because in our proposed scheme, the preferred parent for routing is selected based on both link quality as well as the nodes battery depletion index. As it avoids the nodes which are on low energy, even though that route may have offered a shorter route to the sink node; hence the end-to-end delay may be seen as more delay in different network scenarios.

Chapter 7

Conclusion and Future Scope

In this project, we proposed a new energy and link-quality aware routing protocol for RPL. The objective function of this protocol uses the BDI and ETX metric to compute the DODAG rank among the competing parent nodes as the next hop. The selected path will have nodes with better energy reserves and good quality of the links so that packet drops are minimal. Using the Cooja simulator, we have compared the performance of the proposed protocol with MRHOF RPL. The simulation results show that proposed protocol provides better performance in terms of packet delivery ratio and network lifetime.

Future scope: This work can be extended to include other metrics in rank computation such as considering the congestion of the nodes. Also the weights assigned to each of the metric can be made adaptive, depending on the link quality and traffic congestion around a particular node in real time.

References

- [1] Winter, T.; Thubert, P.; Brandt, A.; Hui, J.W.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.P.; Alexander, R.K. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Available online: <https://tools.ietf.org/html/rfc6550>
- [2] Vasseur, J.P.; Kim, M.; Pister, K.; Dejean, N.; Barthel, D. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. Available online: <https://tools.ietf.org/html/rfc6551>
- [3] Thubert, P. Objective Function Zero (OF0) for the Routing Protocol for Low-Power and Lossy Networks. Available online: <https://tools.ietf.org/html/rfc6552>
- [4] Gnawali, O.; Levis, P. The Minimum Rank with Hysteresis Objective Function (MRHOF). Available online: <https://tools.ietf.org/html/rfc6719>
- [5] Levis, P.; Clausen, T.H.; Hui, J.; Gnawali, O.; Ko, J. The Trickle Algorithm. Available online: <https://tools.ietf.org/html/rfc6206>
- [6] Olfa Gaddour, Anis Koubâa - RPL in a nutshell: A survey
- [7] Harith Kharrufa, Hayder A. A. Al-Kashoash , and Andrew H. Kemp , *Senior Member, IEEE* - RPL-Based Routing Protocols in IoT Applications: A Review
- [8] Xiyuan Liu, Zhengguo Sheng , Changchuan Yin, *Senior Member, IEEE*, Falah Ali, *Senior Member, IEEE*, and Daniel Roggen - Performance Analysis of Routing Protocol for Low Power and Lossy Networks (RPL) in Large Scale Networks
- [9] Anna Triantafyllou ,Panagiotis Sarigiannidis ,and Thomas D. Lagkas - Network Protocols, Schemes, and Mechanisms for Internet of Things (IoT): Features, Open Challenges, and Trends
- [10] Mohammed Riyadh ABDMEZIEM, Imed Romdhani, D. Tandjaoui - Architecting the Internet of Things: State of the Art
- [11] Md Anam Mahmud, Ahmed Abdelgawad, Kumar Yelamarthi - Improved RPL for IoT Applications
- [12] <https://www.ti.com/lit/pdf/swry013>
- [13] Ala Al-Fuqaha, Senior Member, IEEE, Mohsen Guizani, Fellow, IEEE, Mehdi Mohammadi, Student Member, IEEE, Mohammed Aledhari, Student Member, IEEE, and

Moussa Ayyash, Senior Member, IEEE- Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications

[14] Carles Gomez, Josep Paradells, Carsten Bormann, Jon Crowcroft, Universitat Politècnica de Catalunya (UPC), Universitat Politècnica de Catalunya (UPC)/i2CAT Foundation, Universität Bremen TZI, University of Cambridge-

From 6LoWPAN to 6Lo: Expanding the Universe of IPv6-Supported Technologies for the Internet of Things

[15] José V. V. Sobral, Joel J. P. C. Rodrigues, Ricardo A. L. Rabêlo, Jalal Al-Muhtadi and Valery Korotaev - Routing Protocols for Low Power and Lossy Networks in Internet of Things Applications

[16] Sharwari S. Solapure, Harish H. Kenchannavar: Design and analysis of RPL objective functions using variant routing metrics for IoT applications

[17] Feng Wang, Eduard Babulak - Liberty University: SL-RPL: Stability-Aware Load Balancing for RPL

[18] S.Sankar, P.Srinivasan: Energy and Load Aware Routing Protocol for Internet of Things

[19] Mohamed, B., & Mohamed, F. (2015). QoS routing RPL for low power and lossy networks. Hindawi Publishing Corporation, International Journal of Distributed Sensor Networks, Article ID 971545

[20] Sousa, N., & et al. (2017). ERAOF: A new RPL protocol objective function for Internet of Things applications. In 2nd international multidisciplinary conference on computer and energy science (SpliTech). IEEE.

[21] P. Kamgueu et al. Energy-Based Routing Metric for RPL. Research Report RR-8208, INRIA, 2013.

[22] Gen Xu and Gang Lu. Multipath routing protocol for DAG-based WSNs with mobile sinks. In ICCSEE. Atlantis Press, 2013.

[23] Iova, Oana, Fabrice Theoleyre, and Thomas Noel. "Improving the network lifetime with energy-balancing routing: Application to RPL." Wireless and Mobile Networking Conference (WMNC), 7th IFIP. IEEE, 2014.

[24] O. Gaddour, A. Koubaa, N. Baccour, and M. Abid, "OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol", in Proc. 12th Int. Symp. Modeling Optim. Mobile Ad-Hoc Wireless Netw., May 2014, pp. 365-372.

- [25] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks" *IEEE Trans. Mobile Comput.*, vol. 16, no. 4, pp. 964-979, Apr. 2016.
- [26] Ullah, R.; Faheem, Y.; Kim, B.S. Energy and congestion-aware routing metric for smart grid AMI networks in smart city. *IEEE Access*. 2017, 5, 13799–13810.
- [27] Capone S, Brama R, Accettura N, Striccoli D, Boggia G. An energy efficient and reliable composite metric for RPL organized networks. In: *The 12th IEEE international conference on embedded and ubiquitous computing*. 2014. pp. 178–184.
- [28] Kamgueu PO, Nataf E, Ndie Djotio T. On design and deployment of fuzzy-based metric for routing in low-power and lossy networks. In: *The IEEE 40th local computer networks conference workshops (LCN Workshops)*. 2015. pp. 789–795.
- [29] Al-Kashoash HAA, Al-Nidawi Y, Kemp AH. Congestion-aware RPL for 6LOWPAN networks. In: *Wireless Telecommunications Symposium (WTS)*. 2016
- [30] Tmote Sky: Ultra Low Power IEEE 802.15.4 Compliant Wireless Sensor Module. Moteiv Corporation, San Francisco, CA, USA, 2006.