Design Algorithm for WSNs with Mixed Traffic Using RPL

Submitted in partial fulfillment of the requirements

of the degree, of

(Bachelor of Engineering)

By

RESHMA DASTAGEER USTAD	**14DET66**
SARIM INTEZAR QAZI	**14DET100**
MD UMAR MOHD OBAID	**14DET93**

GUIDED BY:

(Asst.Prof. Chaya Ravindra)

(Electronics & Telecommunication)

AIKTC/Mumbai University

(2016-2017)

CHAPTER – 1
INTRODUCTION

Nowadays, Internet of Things (IoT) becomes a potential future scenario of the applicability and impact of technology in human life. The benefits of connecting both WSN and other IoT elements go beyond remote access, as heterogeneous information systems can be able to collaborate and provide common services Our work is a combination of Scilab and its implementation on a Conkiti OS with Cooja simulator and using IOT which can provide us with number of parameters and applications. The specification of the Ipv6 Routing Protocol for Low-power and Lossy Networks was specified inside RFC 6550 [1]. Scilab is used for the design the RPL algorithm. To create the topology of WSN RPL algorithm we used one sink node and many remaining node. The main Aim of RPL is to send packets between sink node and sensor node and monitor the parameters like temperature, light, power cost etc. for each sensor node predecessor is already defined we are focusing on two parameters temperature and light. The routing is optimized for all communication between sink nodes and sensor nodes when RPL is initiated from sink node the Peer-To-peer communication is consider but the path is selected which is longer then the corresponding shortest path belongs to the constructed DODAG tree using physical link packets are directly transmitted in this project focusing on the random topology generation using IPv6 routing protocol. Next, we bring in section how to construct the tree in DODAG. Afterward in another section we understand the concept of how to create topology in Scilab using NARVAL Toolbox. In further section, we describe the Implementation of RPL in Contiki OS using Cooja simulator and IOT (Internet of things) with the help of IOT how to measure the parameters like Temperature and light.

1.1 - WIRELESS SENSOR NETWORKS

Wireless sensor networks (WSN), sometimes called wireless sensor and actuator are spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting.

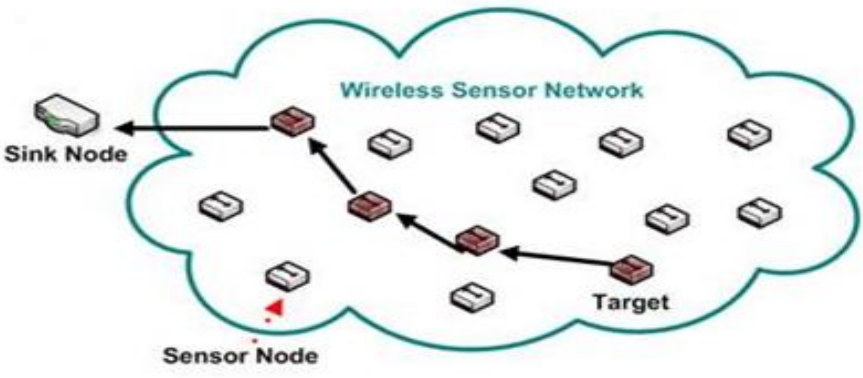

Fig 1.1 Wireless Sensor Network

1.2 - TRAFFIC PATTERNS

WSNs supports three basic traffic flows:

- Point-to-Point (P2P),
- Multipoint-to-Point (MP2P) and
- Point-to-Multipoint (P2MP).

P2P describes the pattern of communication between a designated sender and receiver. In WSNs, this traffic pattern can occur in two ways: Firstly, a sensor node might be requesting measurements from another node somewhere in the network. In this case, which is, it is likely that this request and the response have to pass via intermediate sensor nodes due to the size of the WSN. Secondly, the P2P traffic pattern could be used to prompt measurements from specific nodes. Gathering data measured within the WSN requires a collection protocol which draws information from many nodes and forwards it to one or more sinks in a MP2P fashion.

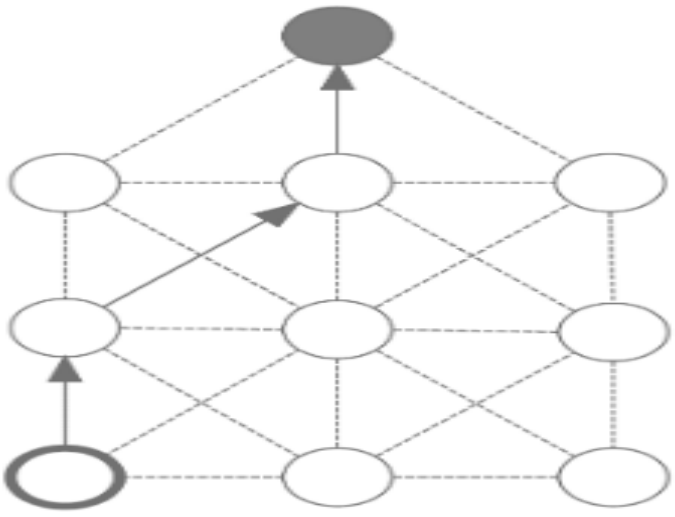

Figure 1.2 Point-to-Point Traffic

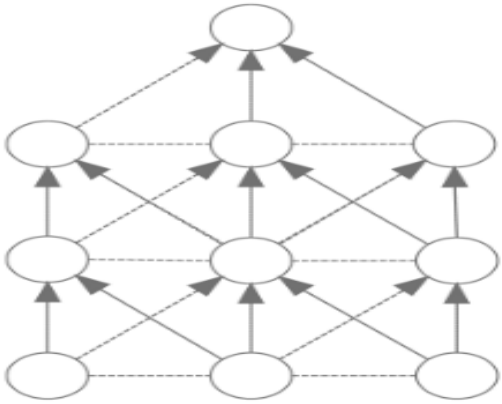

Figure 1.3 Multipoint-to-Point Traffic

Gathering data measured within the WSN requires a collection protocol which draws information from many nodes and forwards it to one or more sinks in a MP2P fashion shows this traffic pattern. Data collection protocols does not necessarily require reliability This is due to the fact that a lot of the measurements tend to be threshold based, so a single node's result being lost do not severely impact the results (using MP2P). The level of reliability required of data collection protocols differs on the application space.

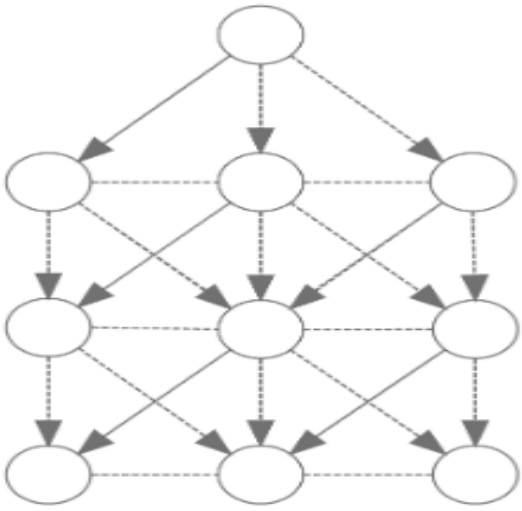

Figure 1.4: Point-to-Multipoint Traffic

1.3 ROUTING IN WSNs

Routing is a process of selecting paths in a network along which to send network traffic. Once the paths have been selected, data traffic is forwarded from one endpoint of the transmission via intermediate nodes to the other endpoint. Routing algorithms are used to determine the "best" paths towards the destination according to one or more metrics, depending on the application requirements. For example, one widely used metric selects the route with the least number of hops trough intermediate nodes towards the destination. Alternative metrics could be to use the best link quality, or least energy consumption. More on routing metrics for WSNs can be found in. The restrictions imposed by WSNs, add further requirements to suitable routing algorithms. The algorithms have to efficiently deal with an ever changing topology, whilst imposing as little control traffic overhead as necessary on the network, as the transmission of messages is very costly in terms of energy The routing protocol needs to compute routes between nodes in the network in order to actually be able to send data to each node. Proactive protocols periodically re-compute these routes, whereas reactive protocols do so only on demand, i.e. when a data packet need to be transmitted. The following section describes the difference between proactive and reactive protocols and give examples of routing protocols for each category.

- **Reactive Protocols**

In reactive routing protocols, no path to the destination is currently known when a packet needs to be forwarded. Routes are acquired by nodes on demand by triggering a route discovery process, e.g. by diffusing a route request packet through the network and then wait for a response for the destination node. This response might take time to arrive, causing the packet delivery to be delayed. The overhead of control packets in a reactive protocol is depending on the amount of

data traffic in the network. Reactive protocols do not require each node to store routes for the entire network, rather computed only for destinations to which data traffic is to be forwarded. The Ad-hoc On-Demand Distance Vector (AODV) is an example of a reactive protocol.

- **Proactive Protocols**

In proactive routing protocols, nodes regularly compute routing tables of the complete network, thus pre-provisioning all possible paths for the entire network topology. Hence, there is no delay imposed by route acquisition before sending the data traffic to its destination. However, a certain amount of control traffic is needed to maintenance the routing tables, and keep them consistent over the whole network. The Optimized Link State Routing protocol (OLSR) is a prominent example of a proactive protocol

- **Flooding Algorithm**

A variety of routing protocols exist for WSNs, which uses different strategies to address the restrictions introduced in WSNs. The most straightforward way to diff use information in a WSN is to use a flooding algorithm. The flooding algorithm transmits broadcast data which are consecutively retransmitted in order to make them arrive at the intended destination. To prevent broadcast storms, several mechanisms are available: nodes check for duplicates, i.e. messages they already received, and packets may contain information about how many times they are allowed to be retransmitted. The drawbacks of the flooding algorithm are that nodes redundantly receive multiple copies of the same data messages. Inconveniences are highlighted when the number of nodes in the network increases.

CHAPTER NO: 02
LITERATURE SURVEY

2.1 RPL: IPv6 Routing Protocol for Low Power and Lossy Network.

This document specifies the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), which provides a mechanism whereby multipoint-to-point traffic from devices inside the LLN towards a central control point as well as point-to-multipoint traffic from the central control point to the devices inside the LLN are supported. Support for point-to-point traffic is also available. Low-Power and Lossy Networks (LLNs) are a class of network in which both the routers and their interconnect are constrained. LLN routers typically operate with constraints on processing power, memory, and energy (battery power). Their interconnects are characterized by high loss rates, low data rates, and instability.

2.2 IEEE PAPERS

1) Mobility Enhanced RPL for Wireless Sensor Networks

In this paper, they investigate the problem of supporting mobility over RPL (IPv6 Routing Protocol for Low power and Lossy Networks) when applied to route traffic in Wireless Sensor Networks (WSNs). RPL is a routing protocol adapted for information routing with low power, low storage and processing sensor devices, in static topologies commonly found in WSNs, but which is not directly designed for mobile scenarios. Specifically, RPL actively decreases control traffic, at the price of lower reactivity to topology changes. They introduce some new mechanisms to the native RPL that reconcile decrease in control traffic and reactivity. They are based on an identification of mobile nodes, and furthermore they enhance RPL behavior in case of node mobility.

2) M-RPL: A Design Algorithm for WSNs with Mixed traffic

This paper proposes a design algorithm called M-RPL. It is IPv6 based routing protocol for low power, lossy Networks (LLNs) that concern routing of both types of traffic. However, since multicast traffic model could be employed in many situations and could be managed by various kinds of multicast routing protocols. The efficiency of M-RPL is evaluated for various traffic demands and networks of 100 nodes and the total unicast traffic in/out per node of 64, 128, 256, 512 and 1024 kbps, compared with RPL. The experimental results show that, in almost all cases, M-RPL give better performance in term of installation cost.

3) Energy-efficient communication protocol for wireless microsensor networks

In this paper, they look at communication protocols, which can have significant impact on the overall energy dissipation of these networks. Based on our findings that the conventional protocols of direct transmission, minimum-transmission-energy, multi-hop routing, and static clustering may not be optimal for sensor networks, we propose LEACH (Low-Energy Adaptive Clustering Hierarchy), a clustering-based protocol that utilizes randomized rotation of local cluster based station (cluster-heads) to evenly distribute the energy load among the sensors in the network.

4) PEGASIS-E: Power Efficient Gathering in Sensor Information System Extended

In this paper, an improved energy efficient PEGASIS based protocol (PEGASIS-E) has been proposed. PEGASIS-E uses average distance among the sensor nodes as the criteria for chaining, thereby providing better performance in terms of energy dissipation and amount of information sent to BS. The simulation results obtained show that PEGASIS-E gives an increase in the network lifetime as compared to PEGASIS.

2.3 OBJECTIVE

The objective of this project is to evaluate and simulate RPL (Routing protocol for low power and lossy network) for WSN (Wireless Sensor Networks) and measure it performance in terms of temperature and light. This evaluation should be done theoretically and through simulation. Project is simulated by the Scilab, Contiki OS with Cooja simulator. The project also included the goal to generate a simulation environment that could be used as a platform for further studies within the area WSNs.

2.4 PROBLEM DESCRIPTION

- Performance of routing protocols is an important issue in WSNs (wireless sensor networks). Wireless sensor networks are the collection of wireless sensor nodes that can exchange information dynamically among them without pre-existing fixed infrastructure and they're highly dynamic in nature.

- This evaluation should be done theoretically and through simulation. The project also included the goal to generate a simulation environment that could be used as a platform for further studies within the area of wireless sensor networks and its protocols.

- The goal of this project is to:
    1. Get a general understanding of WSNs (Wireless Sensor Networks) and RPL protocol.
    2. Design of RPL routing protocol for wireless sensor networks in Scilab using NARVAL toolbox.
    3. Implementation of RPL protocol in Contiki os using Cooja simulator.
    4. Compute temperature and light from the simulation from a particular single wireless sensor node.

3.1 Algorithm

This algorithm for designing RPL tree in Scilab using NARVAL tool box

Step1: Deciding network size.

Step2: Decide a network squared area side and Locality radius.

Step3: Generate a topology with respect to the Locality method.

Step4: Selection of the source node

Step5: Displaying a parameter in window index like node diameter, node border, node color

Step6: Graphical visualization of a WSN network

Step7: Creating shortest path with the help of Dijkstra algorithm from a source on a topology.

Step8: Generate a vector of integer values from the range [0,N-1].

Step9: Find the closest node from a geographic location and create possible routes.

Step10: Build a RPL tree from a source node on a graph.

Step11: Highlight a DODAG tree generated by the RPL algorithm.

Step12: Display the index of each node within a graph.

Step13: STOP

3.2 Flowchart:

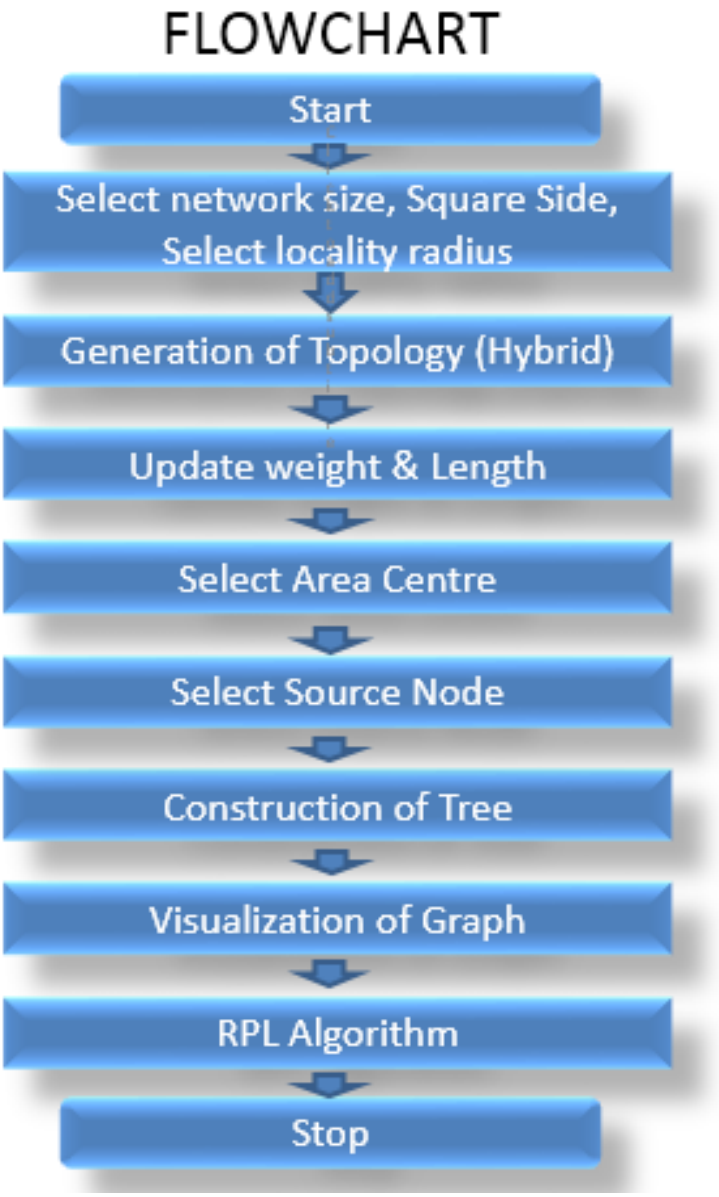

Figure 3.1 Algorithm of RPL design in Scilab

CHAPTER NO:04
ROUTING PROTOCOL

4.1 RPL ROUTING PROTOCOL

The Internet Engineering Task Force (IETF) has a Routing Over Low power and Lossy networks (ROLL) working group currently specifying an IPv6-based unicast routing protocol for WSNs, denoted RPL ("IPv6 Routing Protocol for Low power and Lossy Networks"). IETF ROLL working group have extensively evaluated existing routing protocols such as OSPF, AODV, IS-IS and OLSR and concluded that they are not suitable for the routing requirements specified in, and RPL is a proactive routing protocol, constructing its routes in periodic intervals. RPL may run one or more RPL instances. Each of the instances has its own topology built with its own unique appropriate metric. Nodes can join multiple RPL instances but only belong to one DODAG within each instance. Starting from one of more root nodes, each Instance builds up a tree-like routing structure in the network, resulting in a Destination-Oriented Directed Acyclic Graph (DODAG). For the rest of this chapter, the protocol is explained for one RPL Instance with one DODAG.

4.2- Topology Formation

Topology formation in RPL starts with designating one node as root node. The configuration parameters of the network are determined by the root node, and disseminated to the network using a DODAG Information Object (DIO) message. The mandatory information contained in a DIO comprises amongst others: RPL Instance ID for which the DIO is sent, the DODAGID of the RPL Instance of which the sending node is part of, the current DODAG version number, and the node's rank within the DODAG. The RPL Instance ID is a unique identifier of an RPL Instance in a network. The DODAGID serves the same purpose: to uniquely identify a DODAG in an RPL Instance. The rank represents the nodes individual position relative to the root node. The rank increases in the downward direction

from the root towards the leaf nodes. The node's rank gets calculated by an Object Function (OF) which uses a metric to determine the node's desirability (in terms of application goals, which might e.g., be load balance for energy preservation) as a next hop on a route to the root node. When forming the DODAG, each node is required to select a parent from its neighbors. Accordingly, the node has to calculate its own rank so that it is larger than any of its parents. In this way, the formation of loops in the routing structure is prevented.

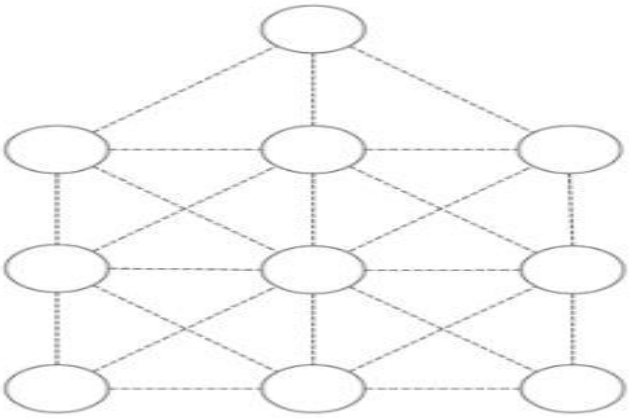

Figure 4.1 Example Network: circles illustrate wireless sensor nodes, connected with 802.15.4 links depicted by dashed lines.

The OF can be used to tailor RPL closely to serve a specific application. To give an example, a node's energy level or its power resource could be used in he OF to calculate the rank. When a node then selects a parent, it will choose the neighbor with the lowest rank, hence the preferable node energy or power resource. For the sake of simplicity, the hop count metric is used as the OF in the following example. The root node starts the DODAG formation by broadcasting a DIO message to its neighbor as illustrated in Figure 4.2 (round 1). The root node of a DODAG is the only node allowed to initiate the diffusion of DIOs. Throughout the whole topology formation, RPL Instance ID and the DODAGID remain unchanged. The only field updated whilst the DIO message are traversing the network, is the rank. The root node has a rank equal to 0, since the distance from itself is zero. When the neighbors receive the broadcast DIO message, they

calculate their rank according to the OF by computing its hop count distance to the root node and sets its rank to From the DIO message received, each node retains a candidate neighbor set, in which it keeps track of the neighbors with lower or equal rank then itself. The candidate neighbor set is used to select parent nodes, which have to have a lower rank than itself. If there are more than one selected parent, the node elects a so-called preferred parent, which serves as the node's next hop when routing a data packet towards the root. This choice is determined by the OF. In round 1, there is only one candidate parent, so they pick the root as their preferred parent. this relationship is represented by the bold plain arrows. After calculating its rank, each node broadcast the updated DIO message to its neighbors. The root node will discard the DIO messages received since they originate from nodes with higher rank than itself. The other neighbors will repeat the process of calculating its own rank according to the OF, and update the DIO message before broadcasting it to their neighbors. there are several nodes in the node's parent set that might fulfill the conditions imposed by the OF, making them qualified as preferred parent. In this case, as described later the preferred parent will be topology formation: all nodes of the network have received DIO messages and joined the DODAG by calculating their rank, whilst the nodes have selected proffered parents represented by the bold plain arrows in Figure4.2

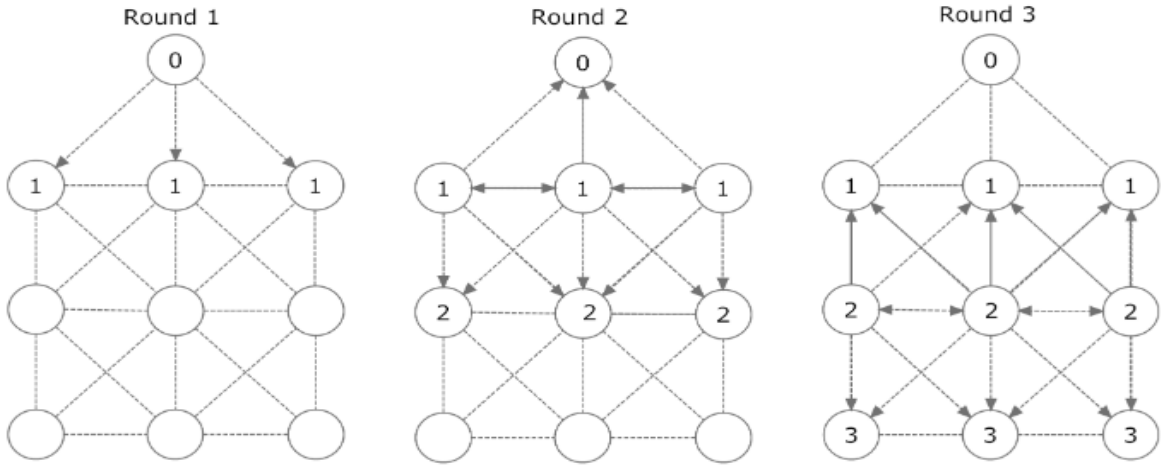

Figure 4.2 DIO messages broadcasted (indicated by arrows) to their neighbors towards leaf nodes. The numbers indicate the node's respective rank, i.e., their logical distance to the root.

4.3 Traffic Flows Supported by RPL

By default, RPL provides a mechanism for multipoint-to-point (MP2P) data traffic from nods within the network to the root node. This traffic flow is called "upward" and is enabled by the DIO mechanism. RPL also provides a mechanism to support "downward" traffic flow, which is needed to enable point-to-multipoint (P2MP) or point-to-point (P2P) traffic patterns. Downward routes are established using Destination Advertisement Object (DAO) messages. P2P traffic is routed "upwards" until it reaches a common ancestor which knows a route "down" the DODAG to its destination. The specification distinguishes between storing and non-storing traffic mode. In storing mode, all nodes keep state of routes to

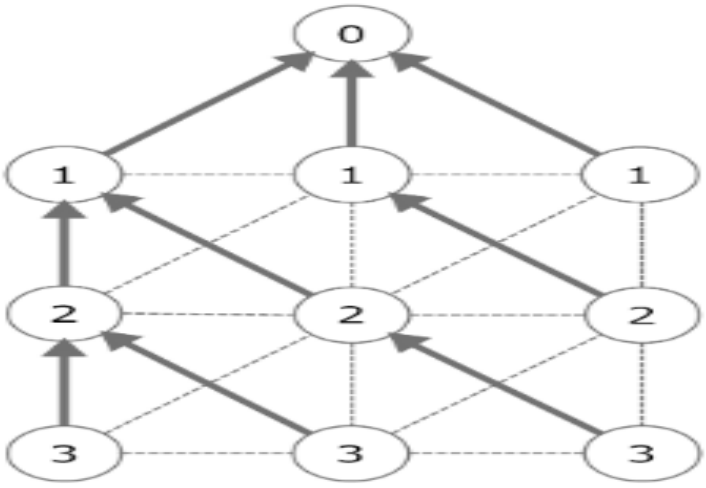

Figure 4.3 The parents are selected (indicated by the bold plain arrows) before the DIO message are updated and rebroadcasted illustrated in Figure1.7 This Figure shows all parents selected when the DIO message has propagated to the leaf nodes. The rules deciding the selection of parents are described

other nodes. In non-storing mode, intermediate nodes do not know any downward routes, so packets are always routed to the root node of its DODAG and then source-routed to its destination. In respect to Figure 4.3, the direction from the leaf nodes towards the DAG roots is referred to as the "up" direction. Hence, the direction from the DAG roots towards the leaf nodes are referred to

as the down direction. A node's rank defines the node's individual position in the DODAG, relative to other nodes with respect to the DODAG root. Rank strictly increases in the "down" direction and strictly decreases in the "up" direction. How the rank is calculated depends on the DAG's Objective Function (OF). Witch OF the DODAG uses are identified by the Objective Code Point (OCP). The rank is also used when selecting a parent, where traffic is sent on the path towards the DODAG root (which has the lowest rank in the DODAG)

4.4 RPL Control Messages

This section will describe the three different messages used by RPL: DIO, DAO and DIS messages, and fields included. Transmission scheduling rules for the different RPL messages are also presented

4.4.1 DODAG Information Object (DIO)

Transmission scheduling rules for the different RPL messages are also presented. DODAG Information Object (DIO)The DIO Base Object shows the DIO message format. The DIO message format consist of nine different fields of witch four was necessary for the measurements preformed in this paper's implementation. The fields that are excluded are the once required when there is multiple DODAGS. The implementation consists of a single DODAG with a single RPL instance. Mode of Operation (MOP) - The Mode of Operation (MOP) is always set to storing mode. Rank - 16-bit unsigned integer indicating the DODAG rank of the node sending the DIO message. Destination Advertisement Trigger Sequence Number (DTSN): 8-bit unsigned integer set by the node issuing the DIO message. The Destination Advertisement Trigger Sequence Number (DTSN) flag is used as part of the procedure to maintain downward routes. The details of this process are described in Section Flags: 8-bit unused field reserved

for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

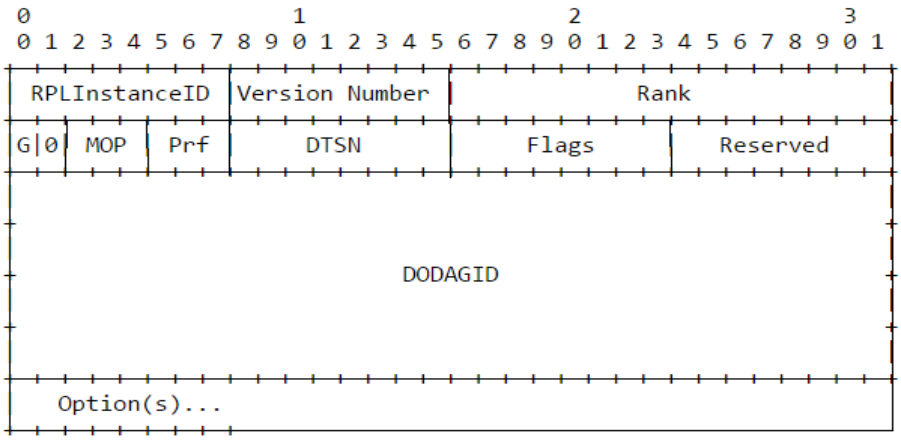

Figure 4.4 The DIO Base Object

4.4.2 Destination Advertisement Object (DAO)

When establishing downward routes, DAO messages are used to propagate destination information up-wards along the DODAG. In storing mode, the DAO-messages are unicast by the child to the selected parent(s). In non-storing mode, the DAO message is unicasted to the DODAG root. A Destination Advertisement Acknowledgement (DAO-ACK) message may be used. The DAO-ACK is sent as a unicast packet by DAO recipient (a DAO parent or DODAG root) in response to a unicast DAO message

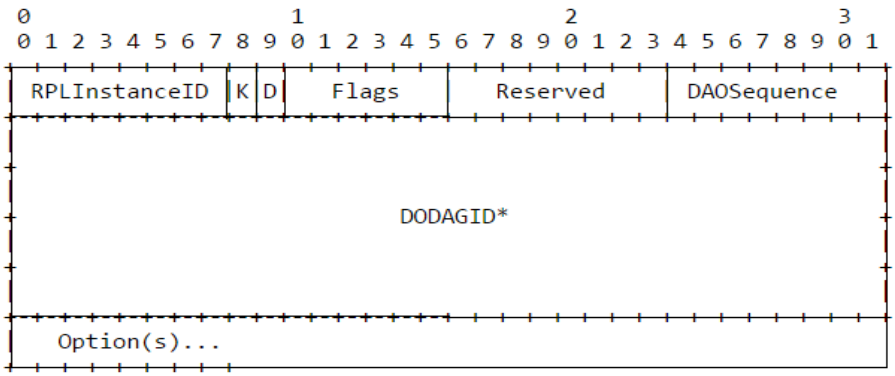

Figure 4.5 The DAO Base Object

Format of the DAO Base Object

DAO message format. The DAO message format consists of seven different fields of which four were essential for measurements done in this thesis. The functionalities implemented are presented below as they are in the 'K' flag indicates that the recipient is expected to send a DAO-ACK back. Flags: The 6-bits remaining unused in the Flags field are reserved for flags. The field MUST be initialized to zero by the sender DAO Sequence: Incremented at each unique DAO message from a node and echoed in the DAO-ACK message.

4.4.3 Destination Advertisement Object Acknowledgement (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.

Format of the DAO-ACK Base Object:

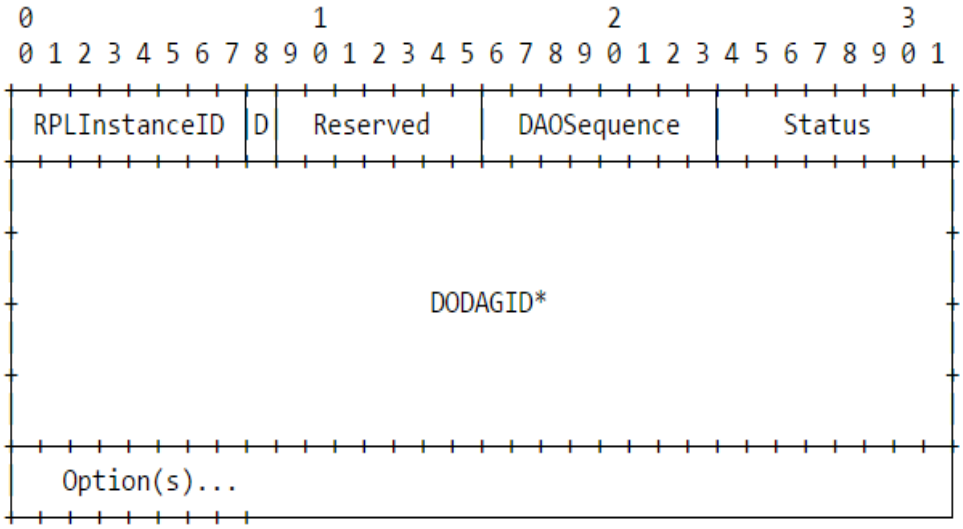

Figure 4.6 The DAO ACK Base Object

4.4.4 DODAG Information Solicitation (DIS)

The DODAG Information Solicitation (DIS) message may be used to solicit a DODAG Information Object from a RPL node. Its use is analogous to that of a Router Solicitation as specified in IPv6 Neighbor Discovery; a node may use DIS to probe its neighborhood for nearby DODAGs.

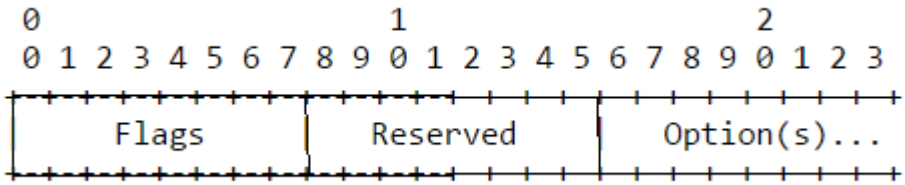

Figure:4.7 The DIS base object

Flags: 8-bit unused field reserved for flags. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver. Reserved: 8-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver. Unassigned bits of the DIS Base are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

4.5 IPV6(Routing Protocol Low-Power and Lossy Networks (LLNs):

T. Winter and P. Thubert.in [1] analyze the IPv6 Routing Protocol Low-Power and Lossy Networks (LLNs) are a class of network in which both the routers and their interconnect are constrained. LLN routers typically operate with constraints on processing power memory, and energy (battery power). Their interconnects are characterized by high loss rates, low data rates, and instability. LLNs are comprised of anything from a few dozen to thousands of routers. Supported traffic flows include point-to-point (between devices inside the LLN), point-to-multipoint (from a central control point to a subset of devices inside the LLN), and multipoint-to-point (from devices inside the LLN towards a central control point). This document specifies the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), which provides a mechanism whereby multipoint-to-point traffic from devices inside the LLN towards a central control point as well as point-to-multipoint traffic from the central control point to the devices inside the LLN are supported. Support for point-to-point traffic is also available. On the other hand, RPL builds upon prior research on WSNs and focuses on practical issues such as IP compatibility; it is a new IPv6 routing protocol designed for LLNs. It uses the 6LoWPAN (IPv6 Low Power Wireless Personal Area Networks) [7] adaptation layer for sending IPv6 packets over LLNs data link layer using encapsulation and header compression mechanisms. 6LoWPAN is designed for the IEEE 802.15.4 medium access layer [8]. In the following, we present the RPL protocol basics. B. The RPL protocol presentation RPL is an IPv6 distance vector routing protocol for LLNs. It is designed to operate with low memory devices, and low

CHAPTER NO:05
DESIGNING AND SIMULATION

5.1 THE RPL DESIGNING IN SCILAB USING NARVAL TOOLBOX

5.1.1 SCILAB: Scilab is an open source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, numerical optimization, and modeling, simulation of explicit and implicit dynamical systems and (if the corresponding toolbox is installed) symbolic manipulations.

5.1.2 NARVAL TOOLBOX

It is focusing on the analysis of network protocols and algorithms. In fact, each network of communicating devices such as computers, phones or sensors, needs to follows specific rules in order to organize and control the data exchange between source and destination nodes. Communication protocols enable to perform the network topology, and to propagate the data traffic between network entities. The main goal of our toolbox is to provide a complete software environment enabling the understanding of available communication algorithms, but also the design of new schemes in order to improve the traffic behaviour of any connection between two network entities.

NARVAL permits to generate random topologies according to various algorithms such as Locality, Waxman, and Barabasi-Albert and hierarchical models. The user can also design his own topology by providing nodes' coordinates, visualization parameters, and also links' information. The combination of these functions enables to build a large range of topologies with distinct routing properties. Thus, the NARVAL module permits to study the impact of routing algorithms on the effectiveness of transmission protocols used by data

communications on a network topology. We provide a set of basic functions to create network graphs, compute routing algorithms (AODV, BFS, DFS, Bellman-Ford, Dijkstra, Flood, Floyd-Warshall, Multiple Paths, RPL, ARC, etc.) on them and finally make statistical analysis on the data exchange. The mobility of nodes (Mobile/Vehicular Ad hoc NETwork MANET/ VANET) is also supported according to models such as Random Direction, Random Walk, Random Way Point, etc.

5.1.3 Designing of RPL tree

RPL allows redundancy in the tree (several parents) and therefore RPL actually constructs a Destination Oriented Direct Acyclic Graph (DODAG), used to route traffic from multipoint-to-point devices inside the network towards one or several central control points (DODAG root(s)). Further options allow point-to multipoint traffic from the central control point(s) to the devices as well. Building the DODAG requires the computation of an objective function-that operates on a combination of metrics and constraints to compute the 'best' path- and the usage of new ICMPv6 (Internet Control Messages Protocol) messages adapted to the RPL context.1) RPL messages: RPL introduces four control messages required for DODAG construction and maintenance:

1)DIO (DODAG Information Object): broadcast message sent by the DODAG root(s) to initially trigger the DODAG construction, and later by router nodes in the DODAG. This message contains general information required to build the DODAG, for instance the DODAG ID, the RPL Instance ID, the DODAG Version Number, the emitter node rank, the objective function with corresponding metrics/constraints.

2)DIS (DODAG Information Solicitation): message designed to be sent by a new node to join the DODAG.

3)DAO (Destination Advertisement Object): message sent by the non-root devices to permit parent nodes to record reverse paths to the multipoint devices.4)DAO-ACK (DAO Acknowledgment): message sent to acknowledge the reception of a DAO message

5.1.4 PATH EXTENSION ANALYSIS ON A SELECTED TOPOLOGY

The path extension analysis has been done in respect the network simulation environment NARVAL. Here is our approach. We first generate random network topologies with variable size. We used the function NARVAL_T _LocalityConnex that creates a random connex network topology in respect with the locality method. Other methods are available, but this function generates relevant connectivity topologies for wireless sensor networks. The locality method is based on a radial communication range. In that model, n nodes are randomly placed inside a square of side L The probability to create a link between two nodes depends on their distance di and the locality radius R. If di ≤R, a link is created. The largest connex component of the generated graph is extracted. [n, L, R] are input parameters of the function NARVAL_T_ LocalityConnex. In our simulations, belonged to the range [100,250]. An example is shown in Fig. 2. It is composed by 115 nodes connected with 289 links.

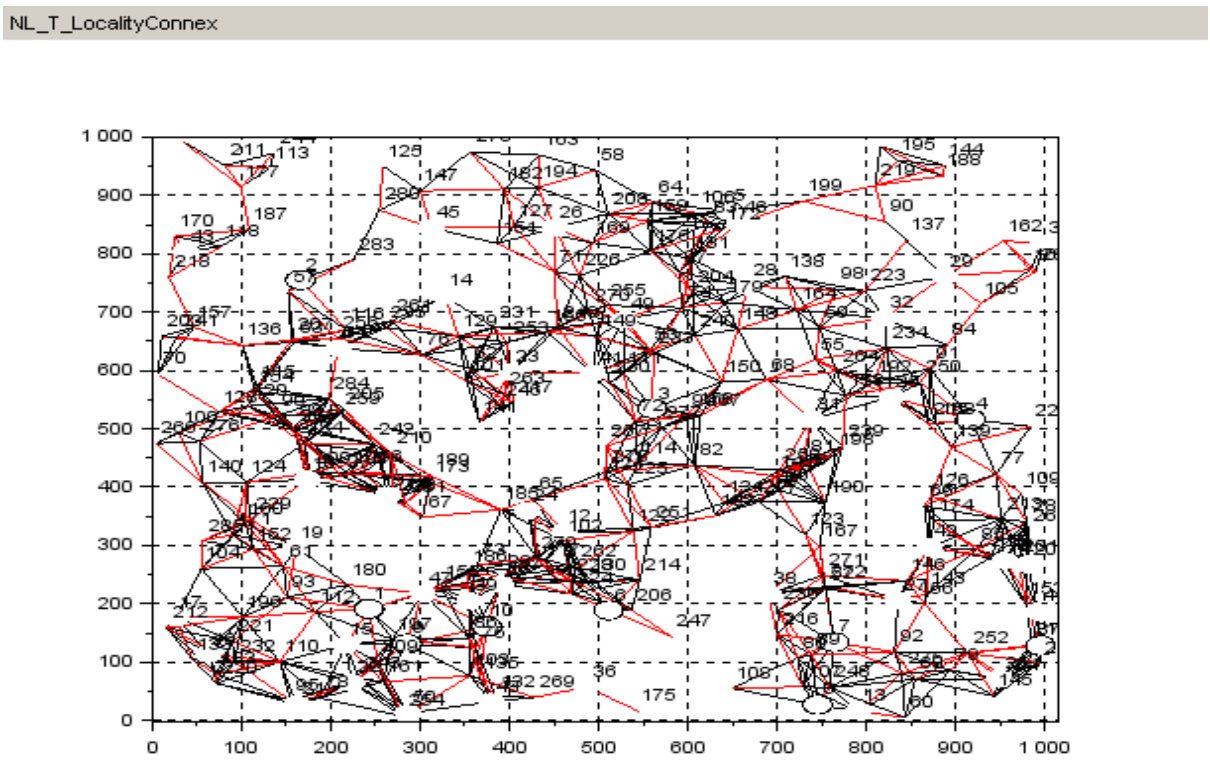

Figure. 5.1 Random topology generated with NARVAL_T_LocalityConnex.

We first provide a complete analysis of this topology. Thereafter we will reproduce the same analysis on many distinct topologies with different size. The node degree distribution represents the proportion of nodes of the topology that have x direct neighbours inside their communication range. In this case, it follows a Gaussian distribution centered in 5.026. Then a node randomly selected will present a high probability to have 5 direct links towards other nodes. We now build the DODAG tree from the selected root S110 according to the RPL algorithm.

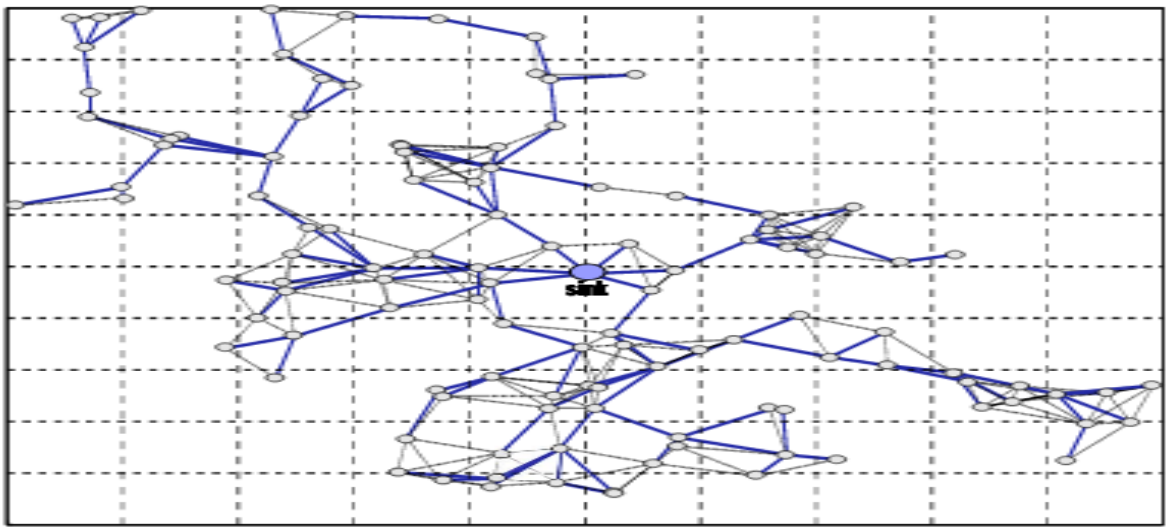

Fig. 5.2 DODAG tree visualization for the root node S110

Each node selects the best parent to reach the sink node S110 in respect with a defined objective function. In our case, we aim to optimize the propagation delay or the path hop length between each node and the sink. As soon as the routing table in each node is updated by the RPL algorithm. the path from each node towards the sink can be easily retrieved. The path hop length for any communication between two peers is varying from 2 to 15 nodes, approximately following a Gaussian distribution centered in 8.56. The main part of our analysis is based on the performance of the path hop length for any peer-to-peer communication [Si, Sj] inside the defined topology. On the first hand, we calculated this value for the path generated according to the routing tables updated by the RPL algorithm. On the other hand, we computed the same parameter for the path generated with the Dijkstra's algorithm initiated from the node Si toward the node Sj Our goal is to analyze the path extension for all peer-to-peer communications for RPL in comparison to a shortest path algorithm. Thus, we compute the path extension for all couple [Si, Sj] E [1:115] with Si not equal to Sj We represents in Fig. 4 the path extension between RPL and the Dijkstra's algorithm for each couple of Si and Sj

The x-coordinate provides the first node of the connection. The y-coordinate corresponds to the second one. The pixel value of coordinates (Si, Sj) gives the path extension in hops (color map) for the connection between the nodes Si and Sj. This is the difference between the path generated by the RPL algorithm and the one computed by the Dijkstra's algorithm.

5.1.5 SIMULATION IN NARVAL

We presented in this work a statistical analysis of the path extension generated by the RPL algorithm for peer-to-peer communications within a small wireless sensor network. Our goal was to study the efficiency of the specific IPv6 Routing Protocol for Low-power and Lossy Networks that intrinsically supports point-to-point traffic. We have shown that RPL often provides non-optimal paths for peer-to-peer communications. In fact, data packets are following longer routes with larger metrics than shortest paths. In this paper, we studied the effectiveness of RPL compared to the Dijkstra's algorithm. We analyzed peer-to-peer communications inside random wireless sensor network topologies. We have built a particular simulation environment named Network Analysis and Routing eVALuation (NARVAL). This toolbox permits to generate random topologies in order to study the impact of routing algorithms on the effectiveness of communication protocols. In our work, we first generated many random network topologies where we selected the location of the sink node. We built the Destination Oriented Directed Acyclic Graph (DODAG) from the chosen sink in respect with the RPL algorithm. We finally performed all paths between each couple of two distinct sensor nodes and compared them to the corresponding shortest paths obtained by the Dijkstra's algorithm. Our approach permitted to retrieve some statistics on the path extension between RPL and the Dijkstra's algorithm. We also analyzed the impact of the sink position and the network size on this path extension. Future works will consist of studying and designing new

specific P2P protocols adapted to 6LoWPAN networks. This initial work is based on random network topologies of reasonable size for a sensor cluster, e.g. less than 250 nodes. We plan to analyze if the results will scale for larger networks, composed by more than 1000 nodes. However, it becomes necessary to design clusters and to take care of the communications between clusters in a large sensor network. We also plan to test other objective functions such as interference from the obstacles or other radio sources on the channel, remaining energy in each node, security, etc. A more effective peer-to-peer communication protocol needs also to be taken into account in our future research.

5.2 PROGRAM TO BUILD RPL TREE IN SCILAB:

```
n=289;//network size
L=1000;//network squared area side
dmax=100;//Locality radius
[g]=NL_T_LocalityConnex(n,L,dmax);//generation of a topology
in//respect with the Locality method
i=NL_F_RandInt1n(length(g.node_x));//selection of the source node
dw=2;//display parameter
ind=1;//window index
g.node_diam(i)=50;//node diameter
g.node_border(i)=10;//node border
g.node_color(i)=5;//node color
[f]=NL_G_ShowGraphN(g,ind);//graph visualization
[dist,pred]=NL_R_Dijkstra(g,i);//application of NL_R_Dijkstra
ETX=5;
[v]=NL_F_RandVector0nminus1(length (g. head), ETX);//update of weigth
v=v+1;
g.edge_weight=g.edge_length;
g.edge_length=v;
xc=l/2;//area center
yc=l/2;
[s]=NL_G_NodeClose2XY(g,xc,yc);//root node
c=5;//5 possible routes
[pred,dist,ra,DAG,DIO]=NL_R_RPL(g,s,c);//application of NL_R_RPL
[go]=NL_R_RPLPlot(g,pred);//highlight RPL tree
ind=1;//window index
f=NL_G_ShowGraphN (go, ind);//graph visualization
```

5.3 IMPLEMENTATION OF RPL IN CONTIKI OS USING COOJA SIMULATOR:

CONTIKI OS: Contiki is an operating system for networked, memory-constrained systems with a focus on low-power wireless Internet of Things devices. Extant uses for Contiki include systems for street lighting, sound monitoring for smart cities, radiation monitoring, and alarms.[1] It is open-source software released under a BSD license.

Contiki provides three network mechanisms: the uIP TCP/IP stack,[5] which provides IPv4 networking, the uIPv6 stack,[6] which provides IPv6 networking, and the Rime stack, which is a set of custom lightweight networking protocols designed for low-power wireless networks. The IPv6 stack was contributed by Cisco and was, when released, the smallest IPv6 stack to receive the IPv6 Ready certification.[7] The IPv6 stack also contains the Routing Protocol for Low power and Lossy Networks (RPL) routing protocol for low-power lossy IPv6 networks and the 6LoWPAN header compression and adaptation layer for IEEE 802.15.4 links.

COOJA SIMULATOR: Cooja has been proven to be an ideal tool for the simulation of RPL in WSNs, there are challenges involved in its use. This is particularly pertinent in regard to the lack of documentation available. The Contiki website may be a first port of call in regard to Cooja, and provides an image of Instant Contiki which can then be used with the virtualization tool VMware. However, once Instant Contiki is successfully started, the Contiki website can then be referred to for nothing more than brief instructions regarding simple network setup on Cooja.

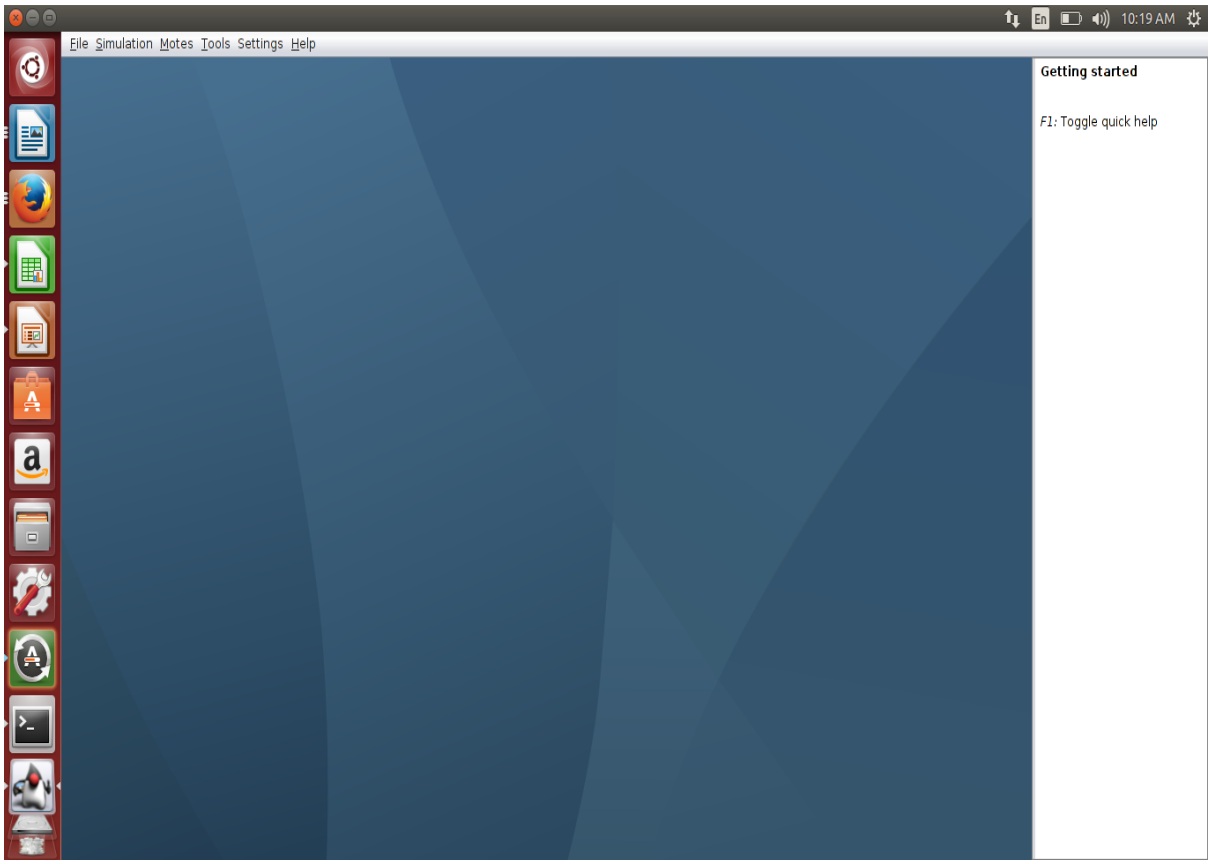

Fig5.3 Initial Cooja Simulator Screen

With this the sum total of any official documentation regarding Cooja, with the majority of support being provided within internet discussion board. Cooja to be able to create network layouts, compile motes, examine output using the Sensor Data Collect plugin and also utilize scripts to produce more fine-grained results. From this starting point, we move onto more complex tasks including the manipulation of the Cooja code and the use of Cooja in physical nodes. First create the simulation environment in cooja simulator, add some motes in Add motes, create new mote type and then Sky mote from the resulting drop-down menu The Sky mote is the simplest of motes for use within a WSN and ideal for initial configurations within a Cooja simulation. After creating the cooja simulation initialized the simulation and Add motes and Locate Mote Firmware [10]

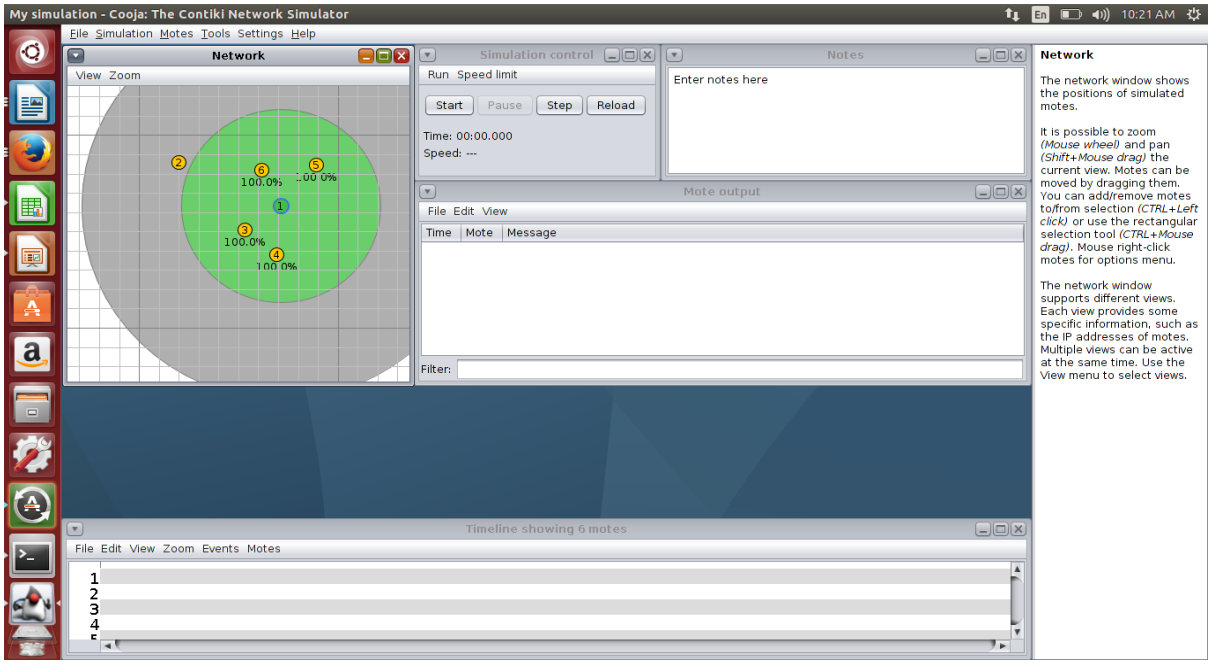

Figure 5.4 Initial cooja simulation Screen with added motes and border router

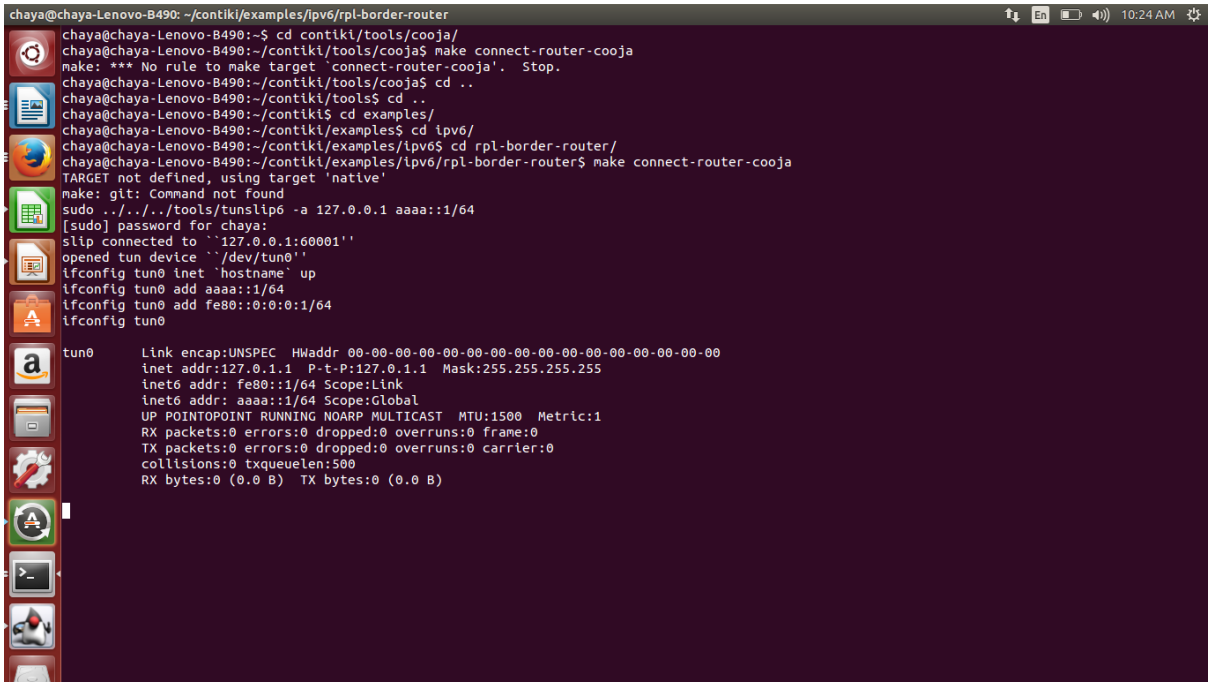

Figure 5.5 Connection of IP of Border router

In the above Figure 5.5, we are making the connection of border router with other nodes in cooja simulator and set the IP address. A great advantage of Cooja is that the motes used in a Cooja simulation use the same firmware as actual physical devices. At this point the firmware to be used to create the simulated mote must be located in order to create and compile it. It should be noted that the location of the firmware is extremely important. Compile the cooja simulation here we are adding 5 motes and initialize the network. and set the IP address of each node We are using Simulation Script Editor to simulate the program which is java script

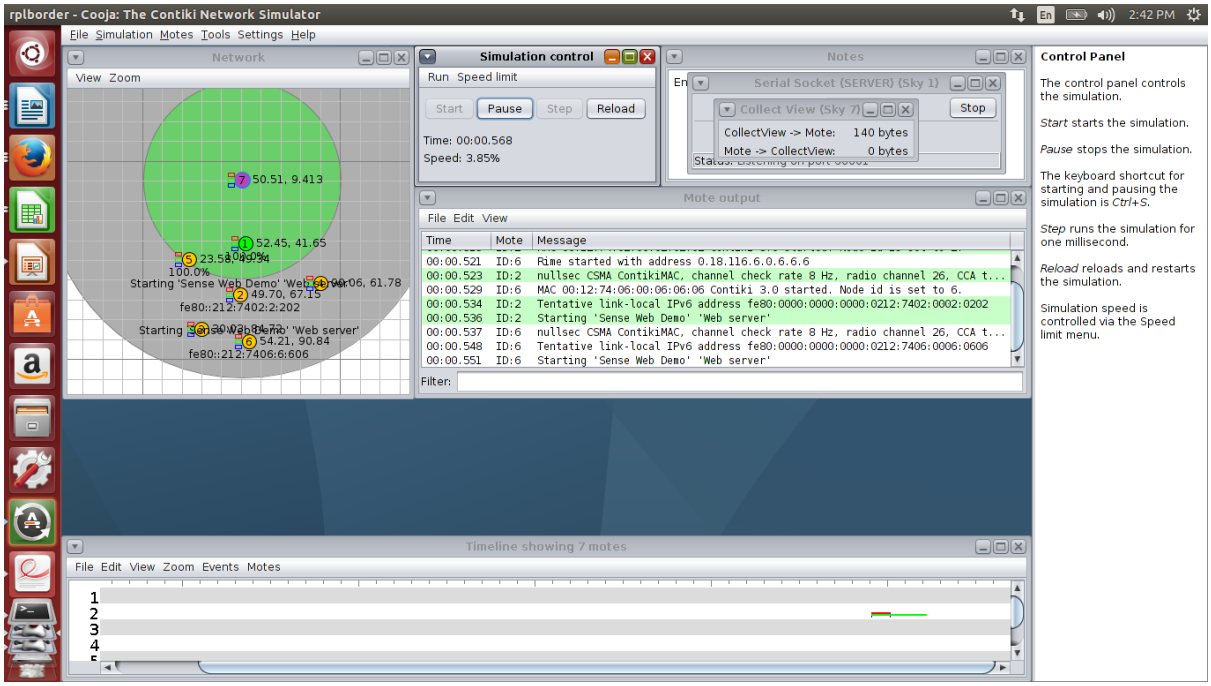

Figure 5.6. Cooja compilation screen

Cooja has sophisticated tools for collecting data from motes, however, it is not immediately obvious how to enable this in a simulation. For data collection in a network with a sink and several senders the collection should be performed from the viewpoint of the sink. This will then display data for all the senders in the network. To enable the Sensor Data, collect view there are two options. Firstly, right-click the sink mote, select ‘Mote tools for Sky 1’and then ‘Collect View ‘Once the simulation is complete a great amount of data will be available from the Sensor Collect View. Some of this information is displayed graphically which

displays the average power consumption of the motes in the network and see Sensor Collect Node Info

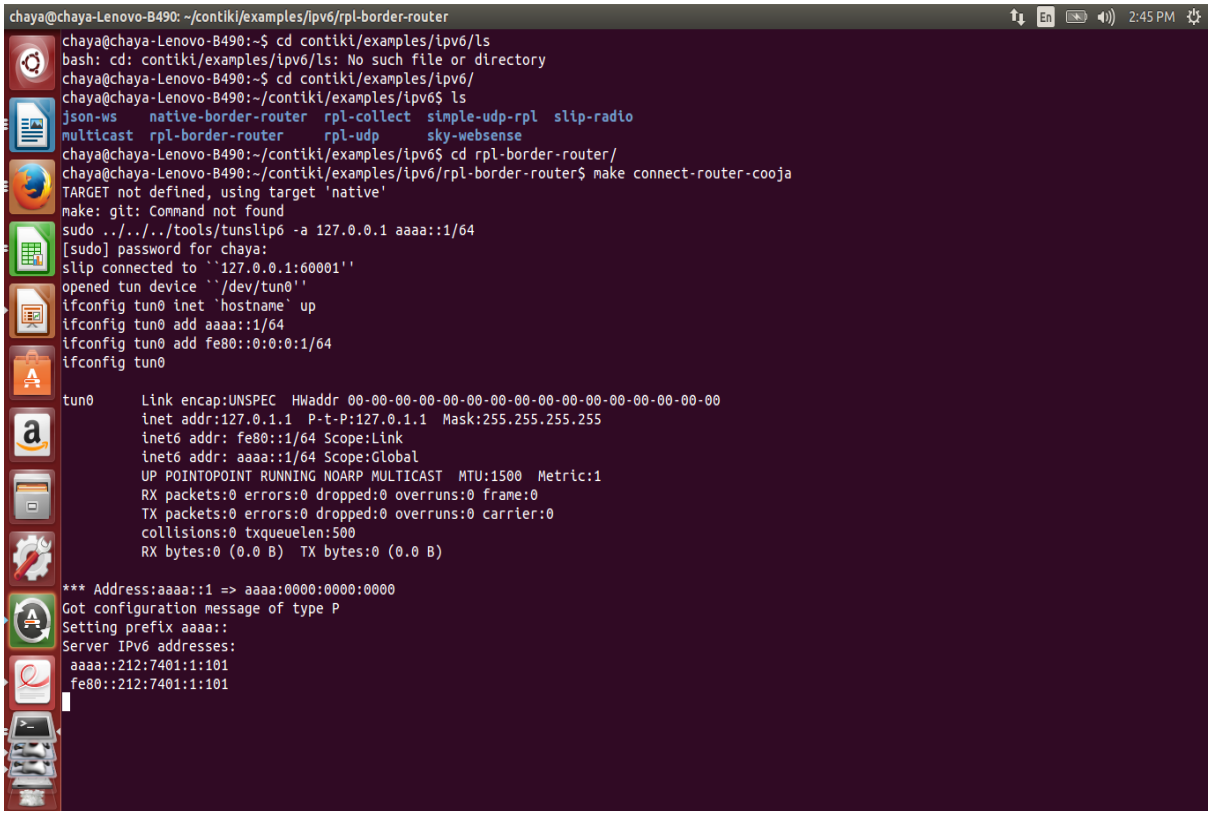

Figure 5.7 After simulation screen

After simulation, the IP address is present with the help os IPV6 It is also possible to display the Border Router's routing table by opening up a web page and entering the IPv6 address of the Border Router as in Figure 5.8

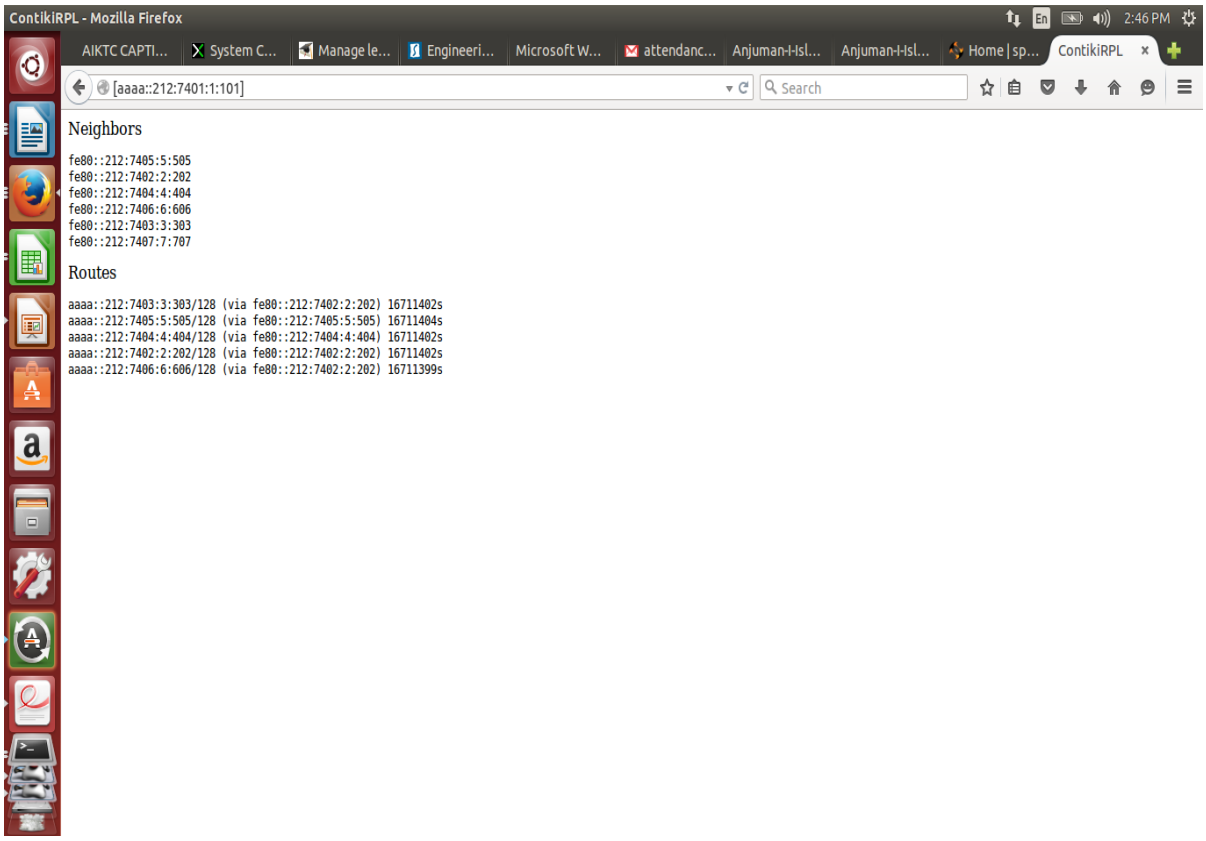

Figure 5.8 IPv6 address of the Border Router

This shows a snapshot of routes established, however, these may be in a constant state of flux, especially with regard to motes on the fringes of the network. The output is present at web browser

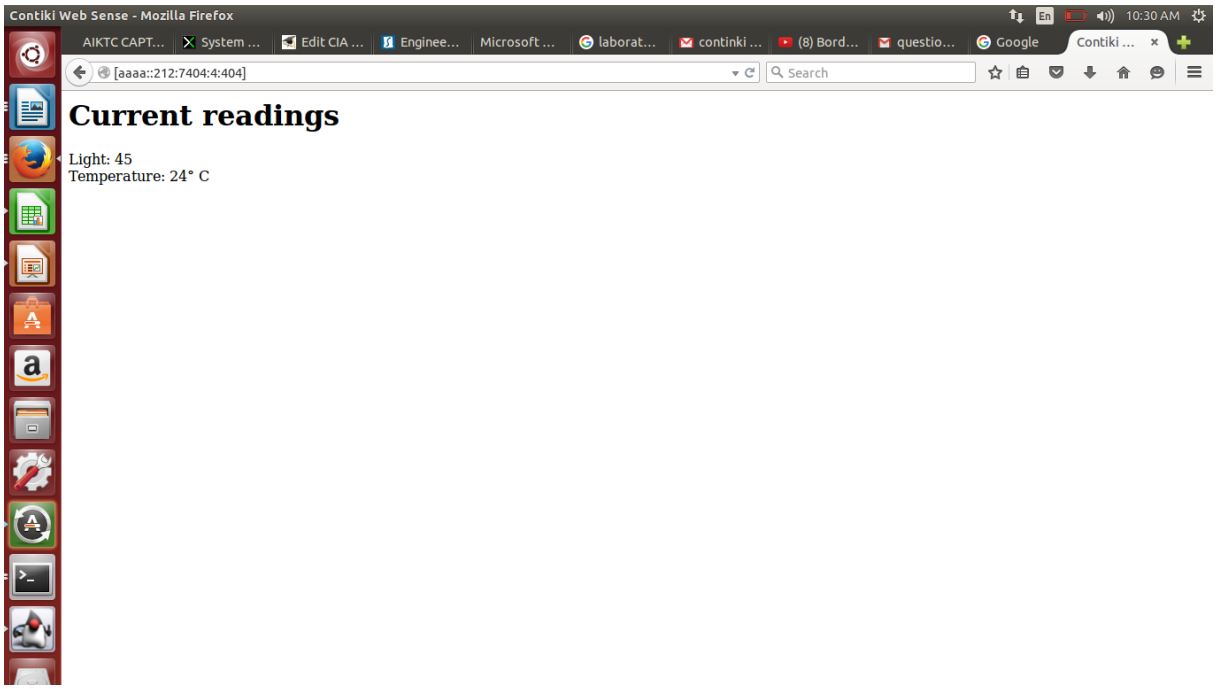

Figure 5.9 readings of particular single node through browser/explorer

It shows the temperature how much amount of power consume during transmission and reception with neibouring motes and readings of aparticular motes in terms of temperature and light which is shown in above figure 5.9

IP address and details of packet loss and how much time is required is shown below in figure 5.10

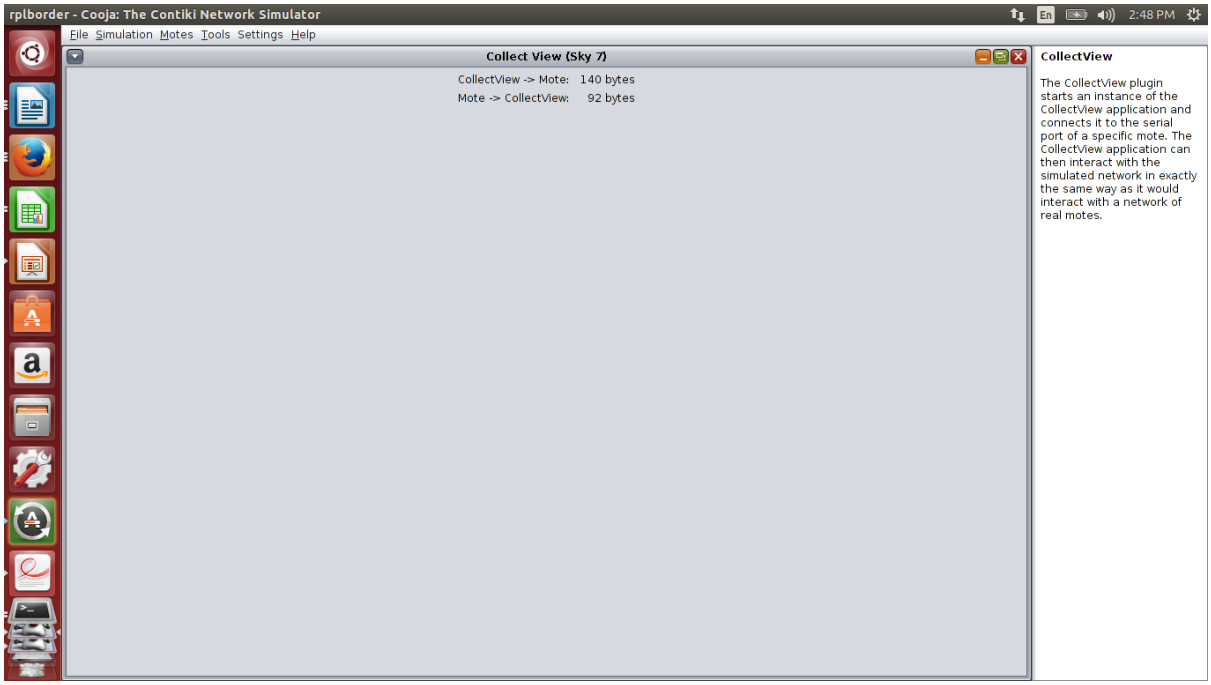

Figure 5.10 Collect view data

CHAPTER 6
SIMULATION RESULTS AND DISCUSSIONS

In the simulation, we create the RPL tree in scilab environment and Implemented RPL routing protocol in contiki os using cooja simulator to analyzed the temperature of WSN, Power consumption during communication with neighboring nodes border router and sensor nodes and packet transmission and reception and loss of packet during transmission

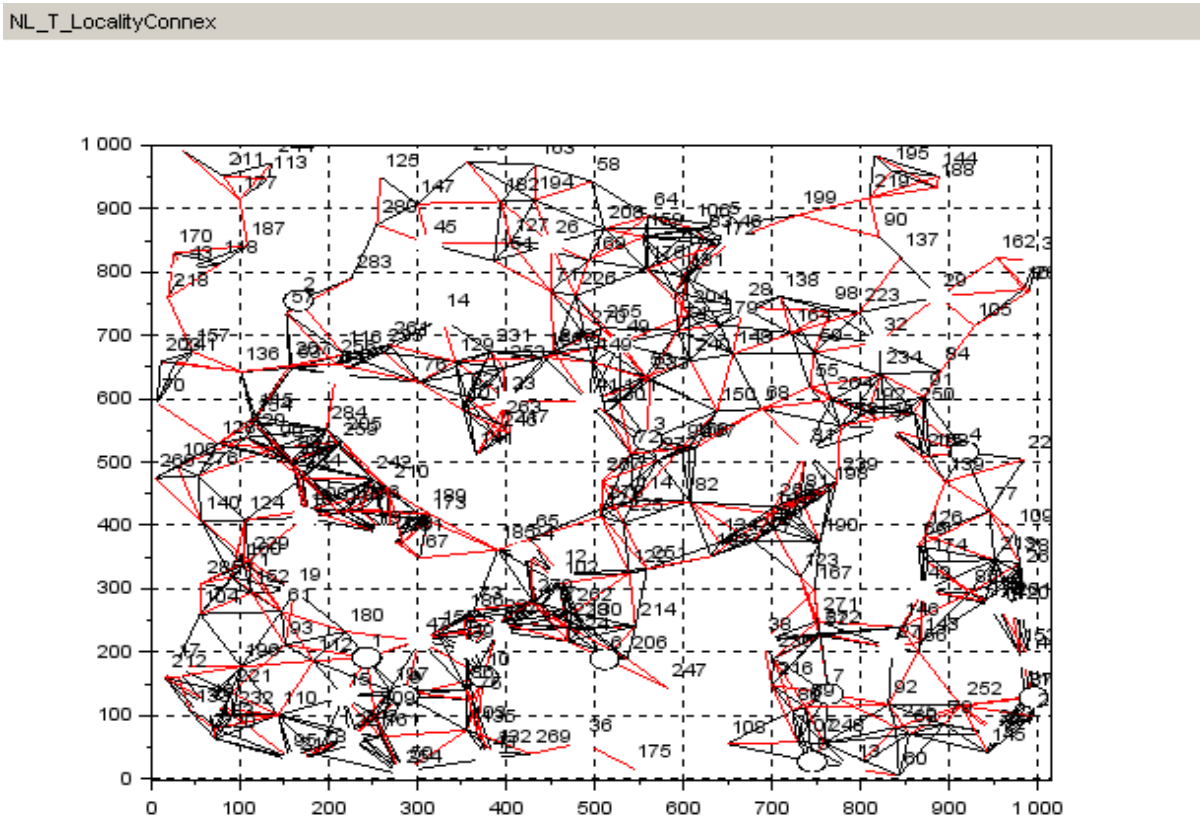

Figure 6.1 Design of RPL tree in scilab

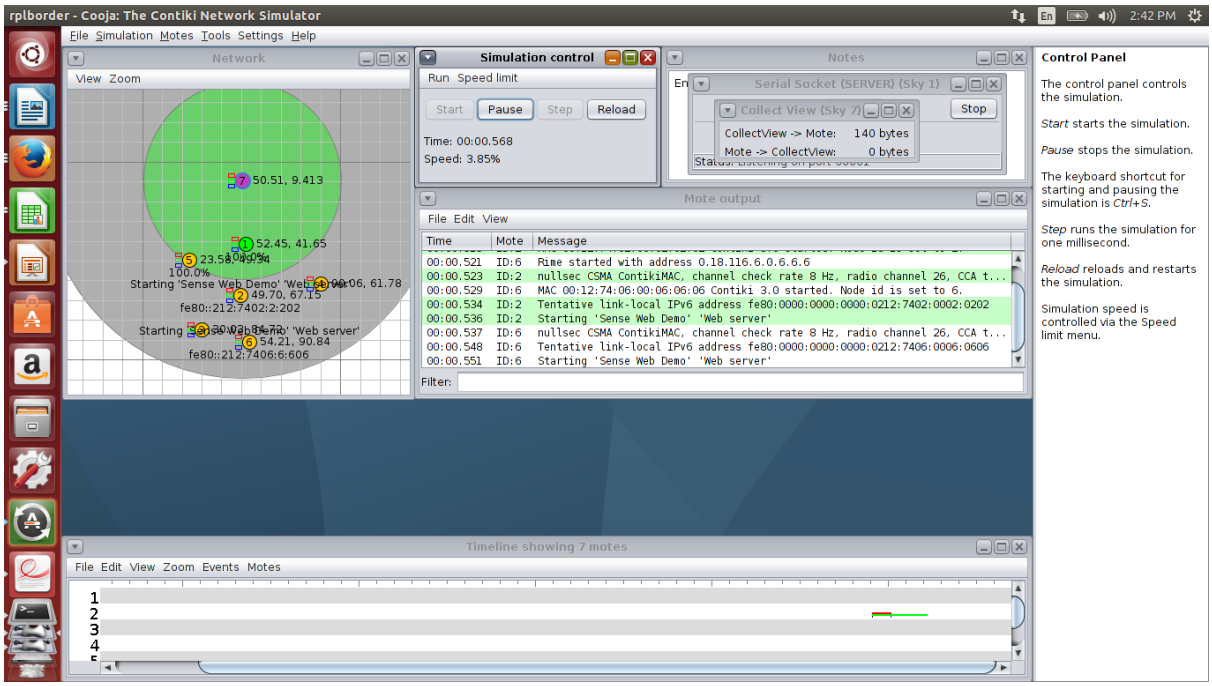

Figure 6.2 Implementation result of RPL tree in Contiki os – cooja simulator

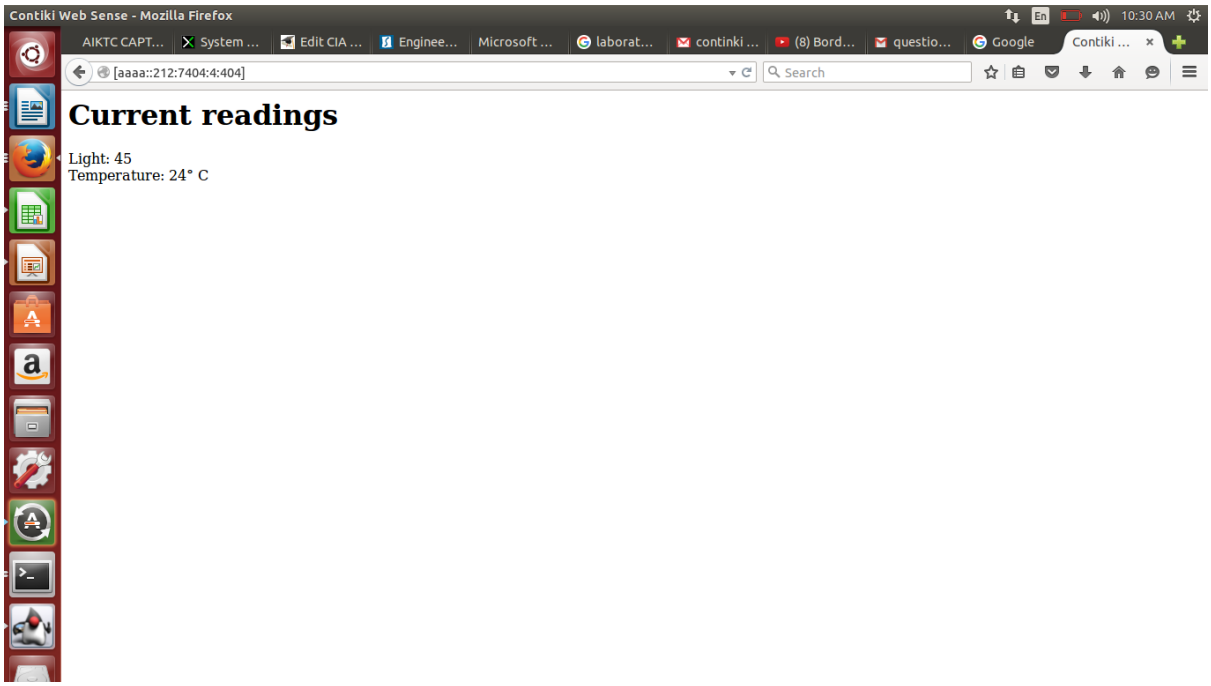

Figure 6.3 Measuring parameter using RPL Algorithm

- Environmental/Habitat monitoring
- Acoustic detection
- Seismic Detection
- Military surveillance
- Inventory tracking
- Medical monitoring
- Smart spaces
- Process Monitoring

CHAPTER 8
FUTURE SCOPE

- Real time implementation of WSN and RPL protocol
- Measuring more physical as well as environmental parameter.
- In further real time we measure as well as control the parameters like temperature, light, humidity, etc.

CHAPTER 9
CONCLUSION

This study proposes design routing algorithm for WSNs called RPL that support both unicast and multicast traffic simultaneously. However, since multicast traffic model could be employed in many situations and could be managed by various kinds of multicast routing protocols' uses Prim-Dijkstra Algorithm, to find shortest path. Designing RPL in scilab used the NARVAL toolbox and Designing RPL in scilab gives the shortest path between two nodes and creating the netwok topology and contiki is operating system which is design by IOT in contiki os we are using cooja simulator for monitoring different parameters like temperature and light of WSN and how many packet are transmitted and receive during transmission and how many packets are receive in terms of byte and how many packet are lost

REFRENCES

[1] T. Winter and P. Thubert. “RPL: IPv6 Routing Protocol for Low Power and Lossy Network.” Internet Draft, draft-ietf-roll-rpl-07, Mar. 2010

[2] T. Winter, P. Thubert et al. “RPL: IPv6 Routing Protocol forLow power and Lossy Networks,” IETF Internet Draft draft-ietf-roll-rpl-19 (work in progress), March 2011.

[3] Ines El Korbi∗, Mohamed Ben Brahim∗, Cedric Adjih†and Leila Azouz Saidane “Mobility Enhanced RPL for Wireless Sensor Networks” ∗National School of Computer Science, University of Manouba, 2010 Tunisia Inria Paris Rocquencourt, Domaine de Voluceau, 78153 Le Chesnay Cedex, France

[4] Annop Monsakul” M-RPL: A Design Algorithm for WSNs with Mixed traffic” Journal of Advances in Computer Network Vol. 4, No.2, June 2016

[5] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, ”Energy-Efficient Communication Protocol for Wireless Microsensor Networks”, In proc. of the 33rd Hawaii International Conference on System Sciences (HICSS ’00), January 2000.

[6]S. Lindsey, C. Raghavendra, “PEGASIS: Power-Efficient Gathering in Sensor Information Systems”, In proc. of the IEEE Aerospace Conference, 2002, Vol. 3, pp. 1125-1130.

[7]Z. Shelby and C. Bormann, ”6LoWPAN: The Wireless Embedded Internet”, John Wiley & Sons, year 2009.[8]IEEE Standard for Information technology 802.15.4 - 2006, ”Part 15.4:Wireless Medium Access Control (MAC) and Physical Layer (PHY)specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)”, 2006.

[9]Y. Yao and J. Gehrke, ”The cougar approach to in-network query processing in sensor networks”, in ACM SIGMOD Record, Volume 31 Issue 3, September 2002

10]Contiki, “Contiki: The Open Source Operating System for the Internet of Things,”2015. [Online]. Available: http://www.contiki os.org/. [Accessed: 09-Nov-2015].