

# **ANALYSIS OF ROUTING PROTOCOLS AND SECURITY ISSUES IN IoT**

Project report submitted in fulfillment of the requirement for the degree  
of Bachelor of Technology

in

**Information Technology**

By

Shubham Dhiman (141444)

Akarshit Sharma (141448)

Under the supervision of

Dr. Shailendra Shukla

to



Department of Computer Science & Engineering and Information  
Technology  
**Jaypee University of Information Technology Waknaghat, Solan-  
173234, Himachal Pradesh**

### **Candidate's Declaration**

I hereby declare that the work presented in this report entitled “ **Analysis of Routing Protocols and Security Issues in IoT**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2017 to December 2017 under the supervision of **Dr. Shailendra Shukla** (Assistant Professor- Senior Grade, Computer Science and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shubham Dhiman (141444)

Akarshit Sharma (141448)

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

Dr. Shailendra Shukla  
Assistant Professor- Senior Grade  
Computer Science and Information Technology  
Dated:

## **Acknowledgement**

It is our privilege to express our sincerest regards to our project coordinator, Dr. Shailendra Shukla, for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department Prof. Dr. Satya Prakash Ghrera for encouraging and allowing us to present the project on the topic “Analysis of Routing Protocols and Security Issues in IoT “at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

We take this opportunity to thank all our lecturers who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career. Last but not the least we express our thanks to our friends for their cooperation and support.

# TABLE OF CONTENTS

Certificate	i
Acknowledgement	ii
Table of Contents	iii
List of Figures	iv
List of Graphs	v
List of Tables	vi
Abstract	vii
1. Chapter-1 Project Objectives	1
1.1 Introduction	
1.2 Problem Statement	
1.3 Objectives	
1.4 Methodology	
1.5 Organization	
2. Chapter-2 Literature Survey and methodology	13
3. Chapter-3 System Development	18
4. Chapter-4 Performance Analysis	39
5. Chapter-5 Conclusions	26
References	

## **LIST OF FIGURES**

- Routing in RPL
- 4 Bit Link Estimator
- TRAIL Rank Attestation
- COOJA Simulator
- Enabling Mobility Plugin
- Figures for testing COOJA Plugin
- Addition of motes
- Flow Chart of UDP Server
- Flow Chart of UDP Client
- RPL-UDP Implementation
- RPL-UDP running simulation
- Test\_mobility.csc
- Figures demonstrating routing without mobility
- Performance Analysis (speed)
- Serial Console Details
- Figures demonstrating routing with mobility
- Motes changing positions after mobility
- Sensor Data Collect
- Sensor Data Details (mobility enabled)
- Sensor Data Collect using Sky View 1
- Temperature analysis
- Sensor Data Collect using Sky View 1 (final)
- RPL UDP modified function
- RPL UDP modified results screenshot with mote output

## **LIST OF GRAPHS**

- TRAIL Message sizes
- Sensor data collect using Collect View (Sky1) (initial)
- Average Radio duty cycle
- Instantaneous power consumption
- Received packets per node
- History power consumption
- Temperature
- Network Hops
- Sensor data collect using Collect View (Sky1) (final)

## **LIST OF TABLES**

- IoT layers and their protocols
- Analysis of various routing protocols in IoT
- Difference between various routing protocol properties

## **ABSTRACT**

### *Analysis of Routing Protocol and Security Issues in IoT:*

The Internet of things (IoT) is that the network of physical devices, vehicles, and different things embedded with software system, physical science, etc. These devices collect helpful information with the assistance of assorted existing technologies and so autonomously flow the info between different devices. IoT requires huge scalability within the network space to handle the surge of devices.

RPL (Routing Protocol for Low Power and lossy Networks) is the most fitted protocol for IoT network because it works in low power and for the lossy networks.

There are some limitations with the RPL Protocol which are in a need to be resolved. Low output when network traffic is heavy, fail to adapt to extreme dynamic traffic, invalid routes and link breakage when the motes are mobile (cannot adapt to mobility) and VeRA – Version Number and Rank Authentication Attack are some major limitations prevailing.

Our main focus here is to resolve the Mobility, analyze different attacks and the introduction of RPL-UDP protocol .

We are using COOJA – Contiki to simulate the behavior of connection establishment and acknowledged data transfer, one without the introduction of mobility, and the other with mobility enabled. There are different attacks that prevails in the RPL which still do not have appropriate solutions. Further it includes the introduction of RPL-UDP protocol to modify the existing topological network and finding the difference between the performance of the two.

### **GROUP MEMBERS:**

- 1. Shubham Dhiman (141444)**
- 2. Akarshit Sharma (141448)**



# CHAPTER 1 – INTRODUCTION

## 1.1 Introduction

IOT is combination of physical devices, items merged with electronics, software, sensors, and network connectivity which helps in exchanging and collection of data among objects. Objects are remotely controlled which creates possibility to merge the physical world with computer-based systems, and that leads to better efficient, accurate and economic benefits in addition to human interference, which will be less.

Data is collected by IOT devices with the help of various present technologies and packets move independently (containing data) between other objects. Present sample include Smart Home Devices. Examples further include smart cities, alexa, wearables like Apple watch, etc.

The Internet of things is highly scalable. IETF 6LoWPAN are used in day to day life to bring together objects with IP networks. IPv6 could engage an important part in managing the network layer scalability.

### **IOT Architecture:**

In IOT, there is an urgent requirement for a layered architecture which needs to be dynamic. Internet of Things (IOT) has the following layers:

- **Objects layer:** The first layer (perception layer) includes and represents physical sensors of IOT so that information can be first sensed, then collected and processed
- **Object abstraction layer:** The responsibility of transferring the object layer grabbed data to the service management layer using secure channels. Examples are: 3G, 4G, Wi-Fi, Bluetooth, etc.
- **Service management layer:** This layer allows IOT application programmers to work with distinct objects, without considering the dependency of the hardware platforms.
- **Application layer:** The application layer allows high-quality smart services for extracting the data as per the customer needs. It includes smart homes, transportation, smart healthcare equipment, etc.
- **Business layer:** The activities /services of an IoT system are completely managed by the business layer. It is only responsible for the development of business model, graphs and flowcharts according to the processed stats from the application layer, which facilitates as the input for the business layer to display results in a user friendly way and also fulfils clients requirements at the completion of the project.

## **IoT Protocols**

**IEEE** and **ETSI** (European Telecommunications Standards Institute) together defines the most important IOT protocols. Rather than fitting all IOT Protocols on uppermost point already present architectural models like OSI will be given a share in the layers which are related to organization levels;

### *Infrastructure*

**IPv6** - "Internet Layer protocol for inter-networking that includes packet switching and further helps in end to end transmission of datagram across numerous IP networks.

**6LoWPAN** - . . This protocol only works in the 2.4 GHz scale of frequency and have a transfer rate of 250 kbps."

**UDP** – It is an OSI transport layer protocol for network applications that includes client/server IP

**RPL** (Routing protocol for low power/Lossy networks on IPv6)

### *Protocols containing data*

#### **MQTT (Message Queuing Telemetry Transport)**

It supports the publish-subscribe recorded communication model in a very thin way. It shows convenience due to establishment of connections at distant places where only a tiny code impression is needed and the network capacity is at superiority."

#### **COAP**

COAP, an application layer protocol used in resource limited devices, like WSN nodes. COAP is such designed that it becomes easy to translate Script to HTTP to integrate with web easily, and satisfying the specific needs like multicast assistance, extremely little overhead, and simplicity. Following features for COAP are as follows

- Relaxed protocol sketch to minimize the complications with plotting it with HTTP, URI, content-type support.
- XMPP- (Extensible Messaging and Presence Protocol)  
Unsecured source technology used by real-time communication systems which includes instant messaging, voice calls & video calls, content syndication , lightweight middleware, and generalized routing of the XML data."

#### **DDS- (Data Distribution Service for Real Time Systems)**

It is unsecured source which immediately addresses publish-subscribe transmission to fix real-time operation systems"

## Communication / Transport layer

**IEEE 802.15.4** : IEEE 802.15.4 is a standard that tells the physical layer and access control of media files for personal low rate wireless area networks (LR-WPANs). It is being maintained by IEEE 802.15 group. Basis for ZigBee, ISA100.11a, Wireless HART, and MI WI specifications, which extends the standard by the development of the layers (upper layer) currently defined in IEEE 802.15.4 standard. Alternatively, can also be used with 6LoWPAN and the standard Internet protocols to develop a wireless embedded Internet.

**NFC**: It uses synthetic coupled devices at a centralized rate of 13.56 MHz. Data rate of around 424 kbps and range is within a few meters less as compared to the other wireless sensor networks.

**Bluetooth**: Bluetooth works in a 2.4 GHz ISM band. It deploys frequency tripping. The rate of data flow is 3 Mbps, range till 100m. Every application used with Bluetooth has its own profile.

## Security

**Open Trust Protocol** – It is basically used for installation, update, and deletion applications. It is used for managing the Trusted Execution Environment (TEE).w.r.t security arrangement

**X.509** – It is a used for managing digital certificates (dc) and public-key encryption. It is a key part of Transport Layer Security protocol which is used for the security of email and web communication.

Following requirements are needed to be satisfied in IOT for large applications and networks:

- Low Energy Consumption (basic).
- It can be used in Wireless Sensor Networks.
- Should supports Ad-hoc networks.
- It can support Point-to-Point (P2P) communication.
- Should Support Mobility

Analysing the comparisons, we can conclude that RPL and CTP are the most favourable protocols to be used.

There are some limitations with the RPL Protocol which need to be resolved. Low Throughput when network traffic is heavy, fail to adapt to highly dynamic traffic, invalid routes and link breakage during mobility (cannot adapt to mobility) and VeRA – Version Number and Rank Authentication Attack are some major limitations prevailing.

Mobility is required in some real life applications. Some of the examples are:

- Sensor networks in the farm fields for evaluating the moisture content in the land and various other evaluations to see whether it is suitable for the cultivation or any other specific requirement.
- Network structure in the Sea/Ocean, for recording the conditions underwater at different positions. Some examples include RADAR emission for detecting depths, any obstacle

present etc. Now the water being mobile, makes the sensors to move from one position to another (making it mobile as well).

Version Number and Rank Authentication Attack- An intruder can achieve these goals by modifying the Version Number of the DODAG. Version Number is a sequential counter, incremented (through DODAG root for the formation of a new Version of a DODAG, or/and to modify the DODAG node Rank Value. The rank value represents the location of that node, in a DODAG.

### **1.3 Objectives**

Resolving stated issues in RPL:

- 1.) Introducing mobility in RPL.
- 2.) Security Issues in RPL

Our main focus here is to introduce Mobility, analyze security issues, and studying about RPL-UDP Protocol. We are using Contiki-COOJA for simulating and understanding the behavior of motes in RPL (protocol) with and without the introduction of mobility.

**NOTE:**

Currently RPL does not support mobility and has a serious issue of VeRA. In this report we will cover the mobility concept and has gained success in introducing mobility in RPL and further improving the efficiency/throughput of the network.

## Chapter 2 – LITERATURE SURVEY AND METHODOLOGY

### RPL

Routing Protocol for Low Power and Lossy Networks (RPL) is a distance vector routing protocol, where the routing is based on (**DODAG**s).

#### RPL IN A NUTSHELL

RPL creates a routing topology in the form of DODAG e.g., a border router. By default, every node maintains multiple number of parents towards the root, the one preferred parent is used for forwarding the data packets (upwards) towards the root, while the other(s) kept as backup routes. Multipoint-to-point communication in RPL, supports the communication from devices to root (d2r) naturally, having a minimal routing state. The topology is basically created and maintained by control packets known as DODAG Information Objects (DIO). DIO is advertised or shown by each node.

DIO packets are rebroadcasted by each node according to technique which is adaptive. It ensures that the DIOs are being advertised sharply when the network which is prevailing at that time is unstable, and instead re-broadcast at a progressive slow pace while the network is stable that strikes a trade-off between reactivity to topology changes and energy efficiency.

All nodes inside a network must announce itself as a possible attainable destination to the root node by sending the (DAO) control packets. These messages are then sent upwards in DODAG topology, from the parent which will coincide with the preferred parent, hence establishing “downward” routes.

RPL defines two kinds of operation: **storing** and **non-storing**. In **storing mode**, each node is responsible for maintaining a routing table which contains mappings between all destinations which can be reached from its sub-DODAG and their next-hop nodes respectively, which are learned while receiving the DAO(s). In **non-storing mode**, the only node in network which maintains information by including it directly in the packet itself is the root. These two modes also affect the ability to support point-to-point (P2P) communication in RPL.

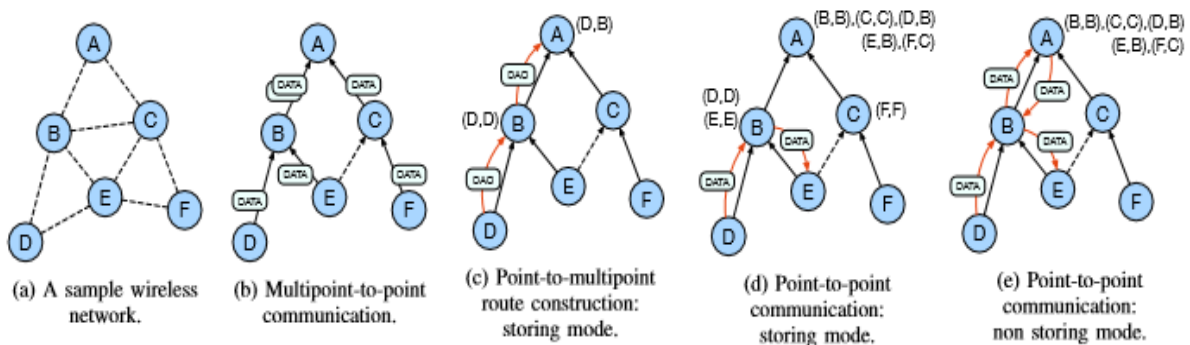


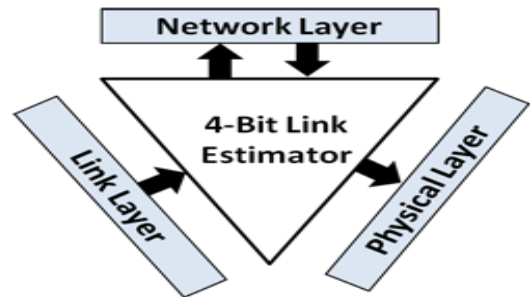
Fig. 2: Routing in RPL. Existing routes are shown next to the network nodes.

## CTP Mechanism

CTP is basically planned for the sensor networks. It tells about ways from all nodes to the root in the network.

It deploys three mechanisms to deal with problem faced by distance vector routing protocol in an extremely dynamic wireless network:

**Link Quality Estimation :** Wireless network have highly dynamic and they exhibit burst behavior over the small time scale. The four-bit link estimator used in CTP uses all the information from the physical, data link, and network layers to provide a correct link quality estimates in spite of the challenges.



**Data path Validation:** In an environment which is basically dynamic, a routing path which is considered reliable at one point of time can become unreliable and unavailable within seconds. Due to change in link qualities, there is formation of loops and that will lead to network congestion and energy drain because of these looping packets. Thus, these issues must be detected as quickly as they occur. CTP uses data path validation for investigating these problems at the time scale of the data packet transmission.

**Adaptive Beaconsing:** Routing protocols most of the time broadcast the packets at an interval which is fixed. A small interval can make the protocol better and more responsive to the changes in the network, because of more bandwidth and energy. A large interval generally uses less energy and bandwidth but it can let topological issues to stay for a longer time. CTP is using the adaptive beaconsing to overcome this issue. When there is an inconsistency in the topology, it sends beacons more rapidly. Otherwise, the beaconsing rate exponentially decreases. Thus, CTP can quickly respond to unfavorable wireless dynamics while suffering low control overhead in the long term.

## RPL v/s CTP

- 1.) CTP - a distance-vector routing algorithm developed as a solution to routing in WSNs. It builds a topology (tree-based), with the root node present at the sink of the network, similar to the structure of RPL's DODAG. An adaptive beaconing mechanism is basically used to broadcast the control messages.
- 2.) While CTP relied on a specific link-layer technology for topology formation, RPL uses the OF that's more customizable for specific applications. CTP was best known for its efficiency in high Packet Reception Ratio (PRR) and energy consumption.
- 3.) It was found that in smaller networks, CTP showed a much better PRR, but as the network size is increased to 49 nodes, RPL showed a better PRR along with less energy consumption. On increasing the data traffic, CTP's PRR is decreased further. The churn (number of times parents were switched) was also significantly less in RPL as compared to CTP. Hence RPL improved Packet Reception Ratio (PRR) and maintained low levels of energy consumption.
- 4.) Because of the strict rules of DODAG formation and cooperation between the different control messages (DIO, DIS and DAO), RPL was found to perform better when the network was scaled in terms of growing data traffic and size of the network.
- 5.) Upon testing the PRR, it was found that RPL and CTP competes on the basis of performance and both protocols achieved a PRR higher than 99.8%. Contrary to results obtained, RPL have a higher churn than CTP.
- 6.) RPL had a slightly higher per-hop ETX value and selected longer routes than CTP. However, RPL offers major benefit in terms of catering to a variety of traffic patterns such as P2P and P2MP, that the CTP was incapable of doing.
- 7.) RPL can directly connect to the Internet nodes by the exchange of packets with global IPv6 addresses.

## RPL SECURITY FEATURES

A RPL message code bit identifies whether the RPL message is secure or not. RPL includes secure versions of the control messages DIO, DIS, DAO and DAO-ACK. It includes many messages that are relevant only in networks being enabled with security. Given the resource constraints in LLNs along with the size and complexity of RPL, security features defined in the RFC are optional and need not to be implemented if not necessary.

RPL consists of 3 basic security modes:

- **Unsecured Security Mode**

The messages are used with basic version and the security could be implemented using mechanisms like LLS or ALS

- **Pre-installed Security Mode**

RPL uses messages that are secure. Hosts as well as routers require a key to affix RPL instances. These keys provide confidentiality, integrity, and credibility related to messages

- **Authenticated Mode of Security**

RPL uses messages that are secured. Keys that are pre-installed won't be considered in a network as a leaf to supply confidentiality, integrity, and credibility associated with messages. Second key should be obtained from a key authority so that network can be affixed as a router.

Since RPL doesn't support asymmetric algorithms, authenticated security mode can't be implemented based on the current advancement.

## RPL ISSUES

RPL security services will not be able to protect against an internal node, that is compromised

- **Version Number Attack:**  
The best way is the modification of Version Number of the DODAG, DODAG with a newer version is made by a sequential counter, incremented with the help of DODAG root or to change DODAG node's rank value, which is representing the location of the node in a DODAG.
- **Rank Authentication Attack:**  
An internal attacker publishes a decreased Rank value which is illegal in this context.



## Existing Security Protocols in RPL

### **TRAIL:**

Although basic security modes are defined by RPL, it remains susceptible to topological attacks that facilitate black holing, interception, and resource exhaustion. TRAIL, is a generic scheme for topology authentication in RPL to deal with the above vulnerabilities.

TRAIL defines a test procedure which inquire the actual properties of the path of routing system.

Using round trip message, TRAIL validates the upward paths towards the root. Without depending on the encryption chains like in the case of VeRA, a node will conclude its rank integrity from an upward path being recursively intact.

TRAIL is meant for node resource consumption and to attenuate network message exchanges. The transmissions of bits needed by path stay possible for typical challenged environments, which a workplace of typical form will operate path with restricted further effort..

### **VeRA (Version Number and Rank Authentication)**

The Version Number and Rank Authentication (VeRA) security scheme proposed for RPL prevents:

1. Intruder's nodes from pretending to be a DODAG root and sending a DIO message with an illegal increased Version Number.
2. It prevents the intruder's nodes from displaying an illegal decreased rank. Using one-way hash chain with a fixed length it prevents Version Number and Rank attacks.

Building blocks of VeRA:

1. Hash - MD5, SHA and others.
2. MAC - Keypad-Hashing for Message Authentication (HMAC).
3. Digital Signature – ECC, RSA, DSA.

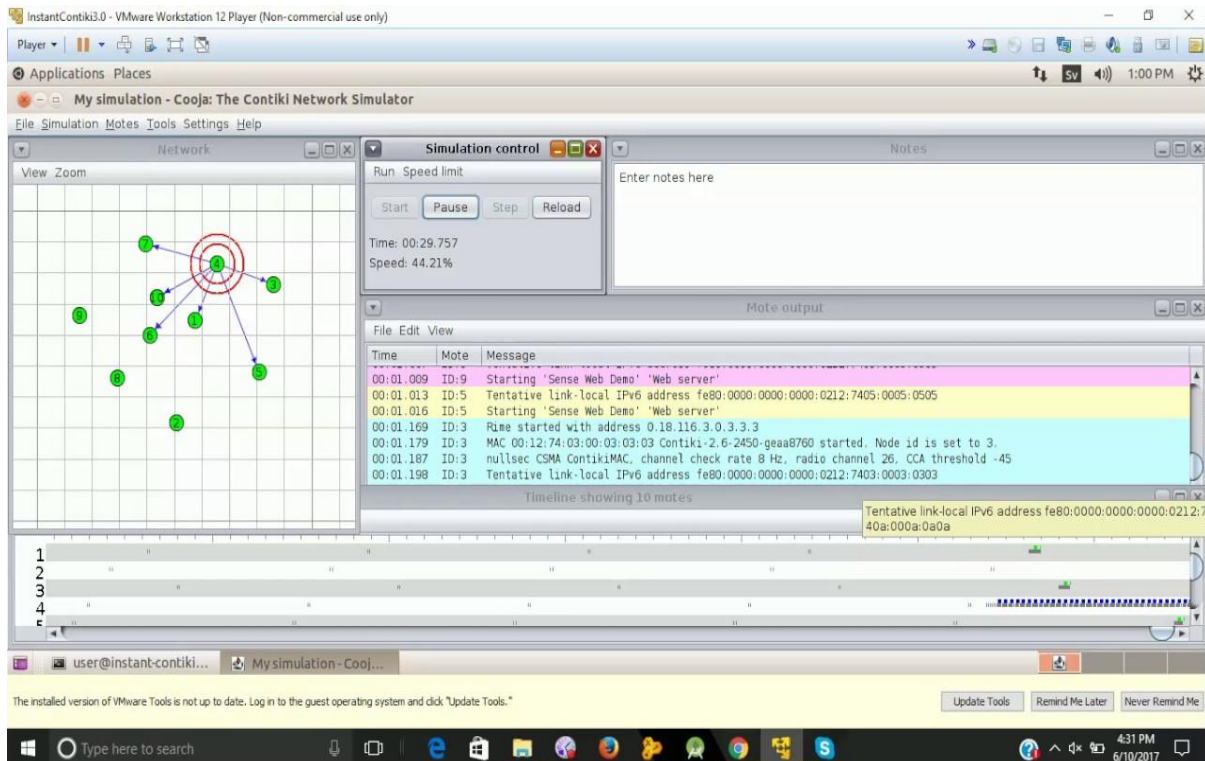
## Chapter 3 – SYSTEM DEVELOPMENT

### COOJA – Contiki

Cooja is a simulator which was initially developed by Contiki-OS developers to simulate applications related to Contiki. It is a flexible and light-weight operating system for forced networked devices. Cooja Simulator is specially designed for WSN. The easiest way to run Cooja is executing it in its own directory.

#### How to run Cooja Simulator:

- A new simulation has to be created
- A new mote type is created
- Motes are added and run the simulation
- The simulation file is saved



## Enabling Mobility in Contiki-2.7 Cooja Simulator

**Step 1:** Download the plugins from the given link below

[https://github.com/vaibhav90/Mobilty\\_Interference\\_Plugin\\_Patch\\_Contiki2.7/tree/master/mobility](https://github.com/vaibhav90/Mobilty_Interference_Plugin_Patch_Contiki2.7/tree/master/mobility)

A new directory is created at

```
cd contiki
```

```
cd tools
```

```
cd cooja
```

```
cd apps
```

- Name the new directory as mobility.
- Download the files for the URL and move them to the “mobility” folder.

**Step 2 :** Building the mobility plugin

Go to the mobility directory using:

```
cd contiki/tools/cooja/apps/mobility
```

Use sudo ant jar command so that plugin can be built

### How to enable Mobility

Start the Simulator

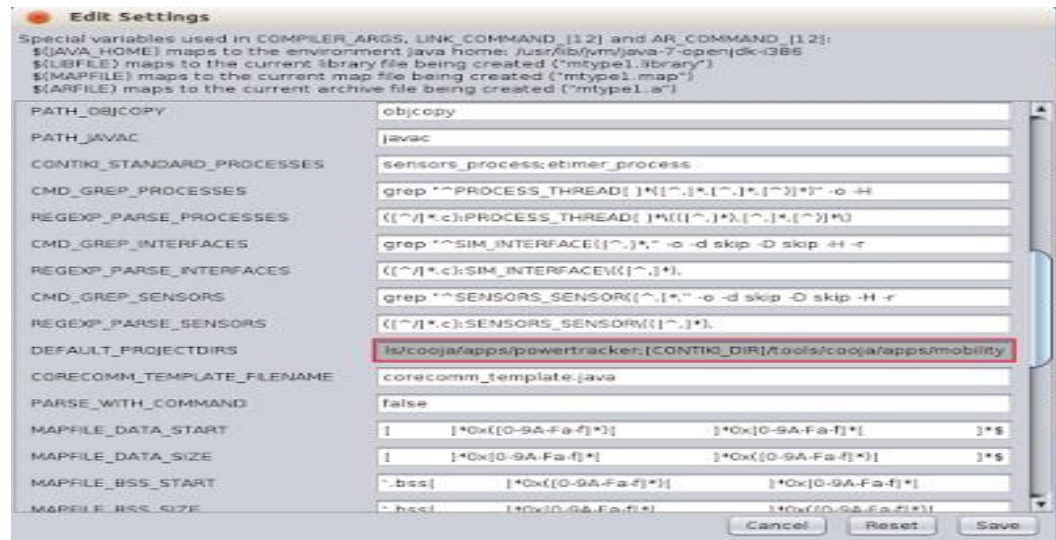
```
cd contiki/tools/cooja
```

```
sudo ant run
```

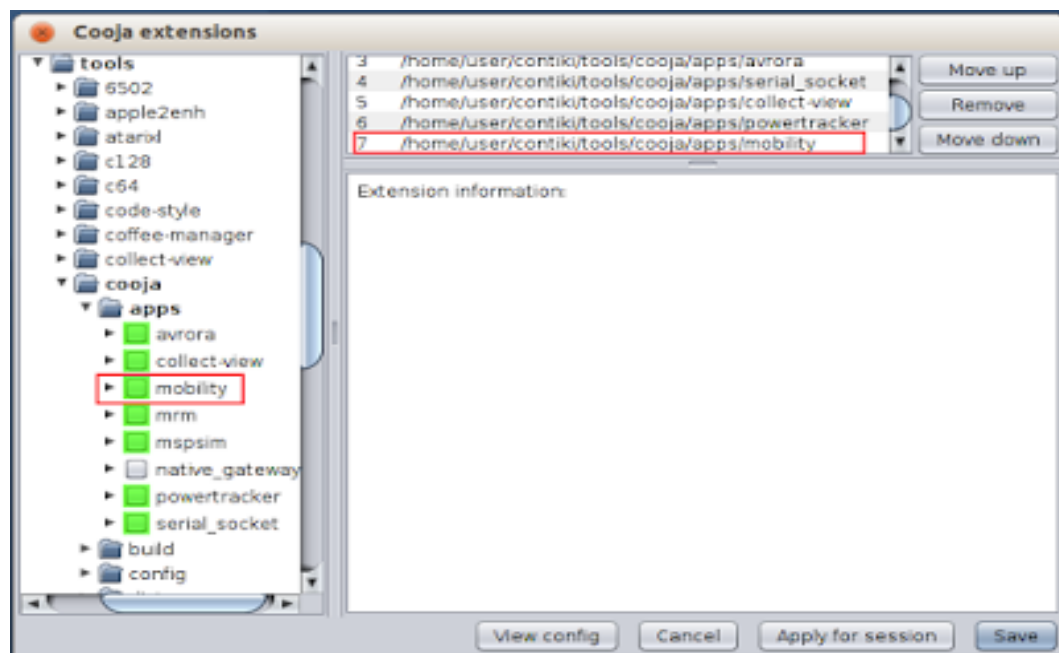
In cooja, first step is to move to Settings then move to External Tool Path then further move to DEFAULT\_PROJECTDIRS

It is important to add complete paths of all the plugins in Cooja Simulator. For this, add mobility plugin's path to the existing paths by inserting ';' symbol.

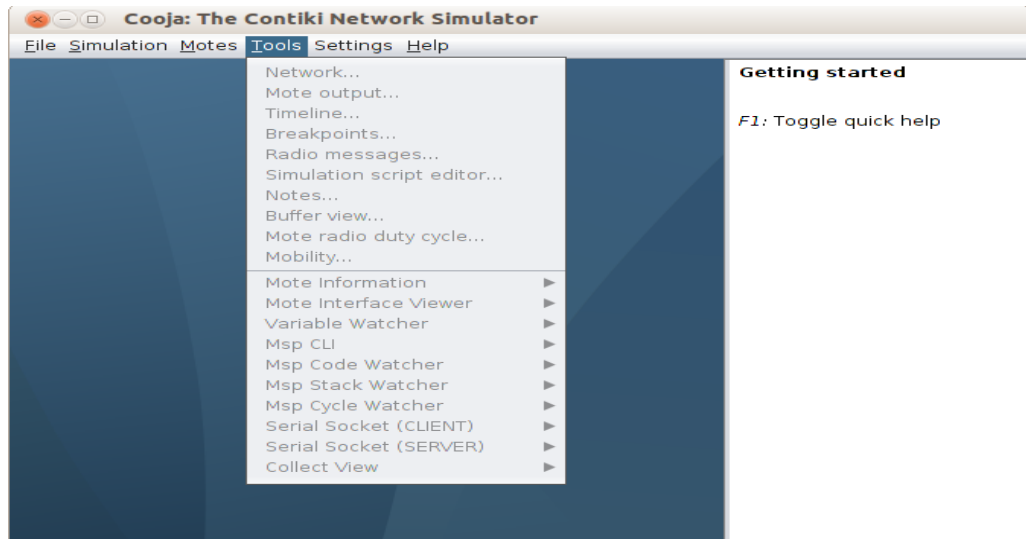
- This path needs to be added - [CONTIKI\_DIR]/tools/cooja/apps/mobility
- The changes are saved
- After the first two steps, restart the COOJA simulator



- Move to Settings the further select Cooja Extension
- Cooja Extensions Window will be shown or gets popped up
- Next select mobility further selects Apply for the session
- This would show mobility plugins for the tools drop down list.



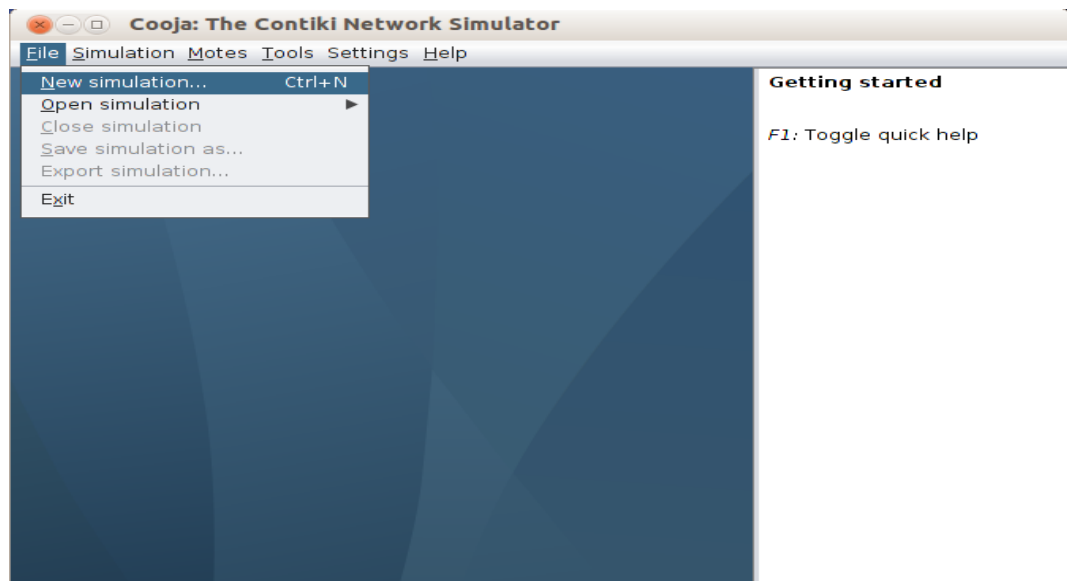
Under the Tools tab we can now an option named Mobility



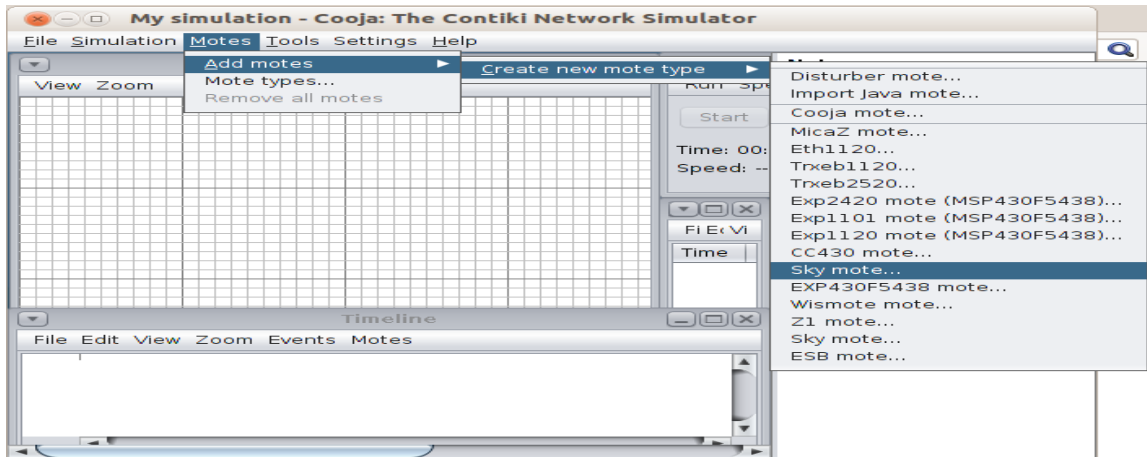
## Plugin gets tested

- Start a simulation(new)

*Select file and click on new simulation*

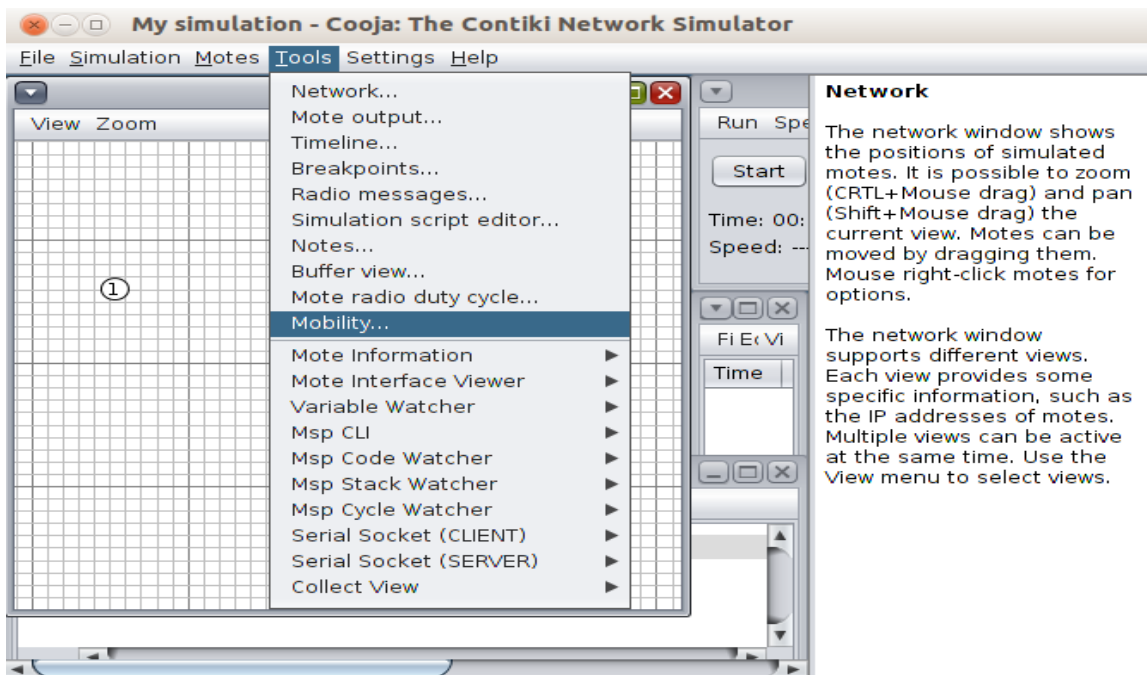


- A c language code which includes hello world is compiled on this mote. This leads to creation of mote

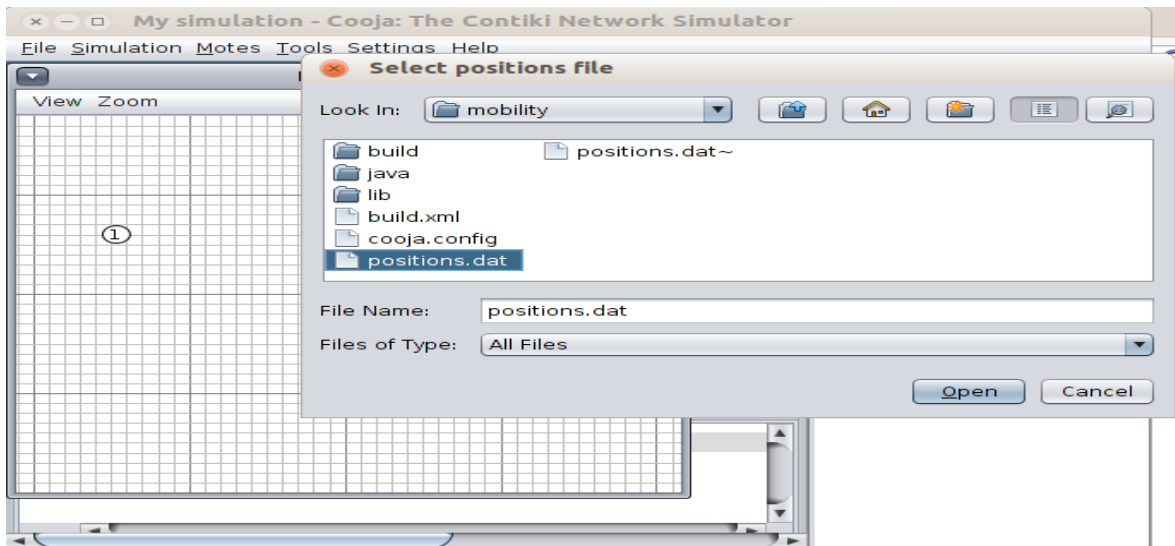


### Testing plugin on a particular mote

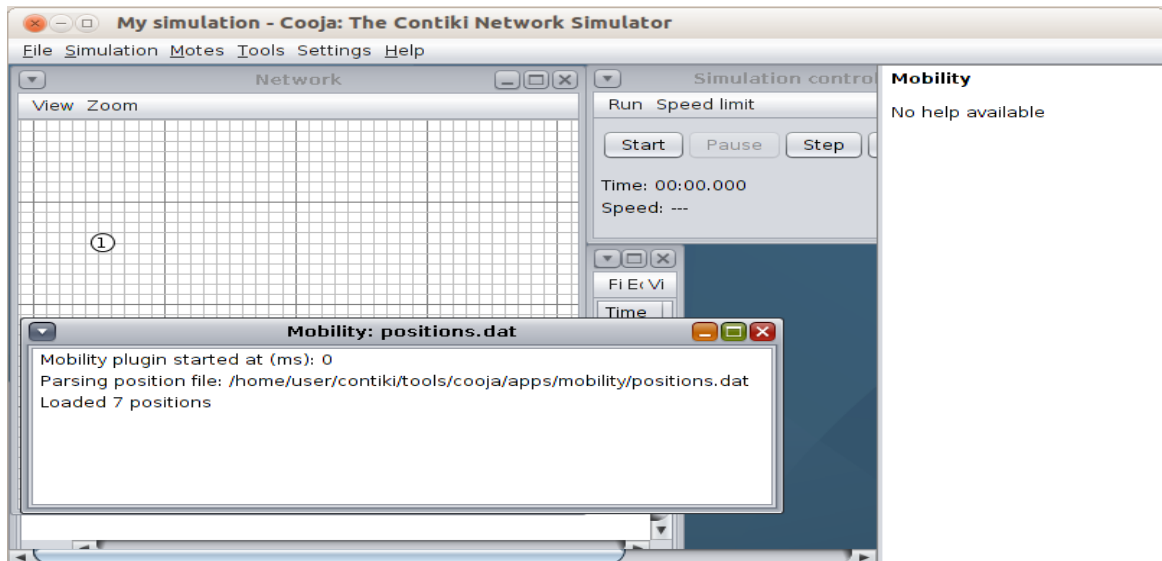
- Click on Tools tab then further select mobility
- Tools -> Mobility*



- Further browse and go to *contiki/tools/cooja/apps/mobility/positions.dat*

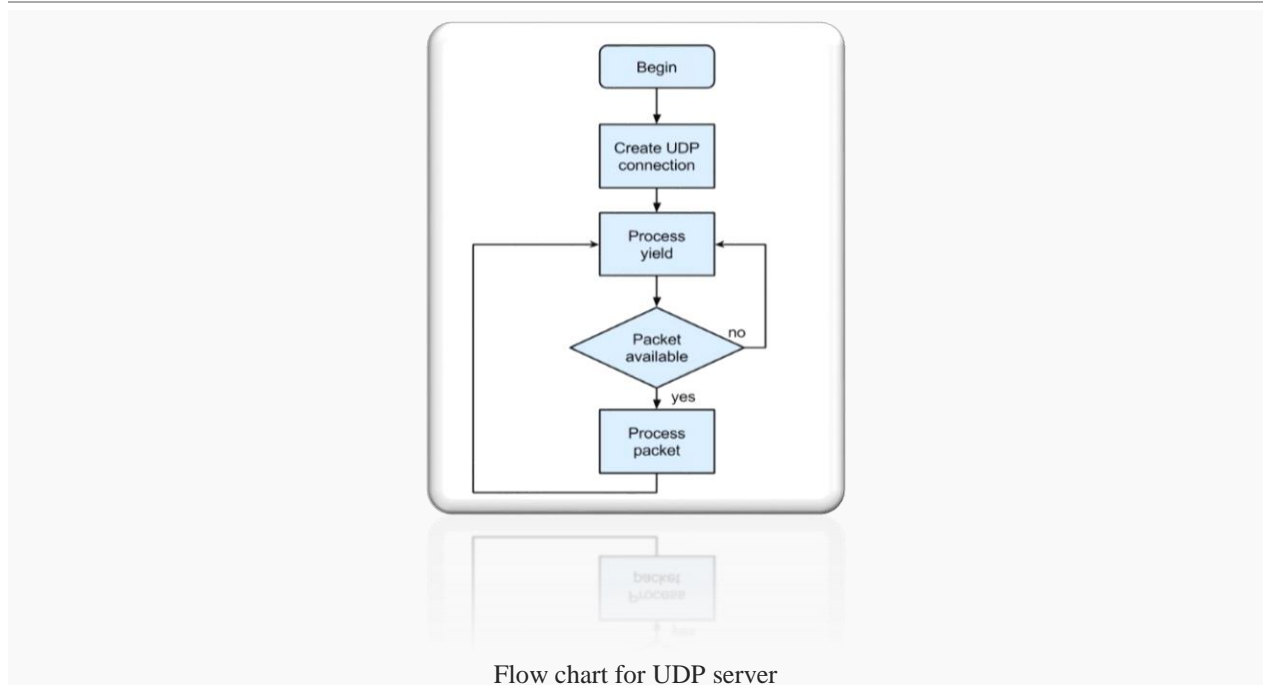


- We have used our mobility plugin to get the co-ordinate information from positions.dat file and further *Click Open*. A secondary window will appear as shown below



- Now, click on the *“Start Simulation”*. The notes will start moving according to the specified positions.dat file

## UDP SERVER



The server will perform primarily the following three tasks.

1. Initialization of the RPL DA Graph.
2. Set up an UPD connection.
3. Wait for the packets from consumer, receives it and prints them on stdout.

### Initialize RPL DAG

```
// check if the variable ADDR_MANNUAL was successfully set or not(1 or 0)
uip_ds6_addr_add(&ipadr, 0, ADDR_MANNUAL);
if_root= uip_d6_addresslookup(&ipaddr);
if(if_root!= NULL) {
    rpl_dag_t *graph;
    graph = rpl_set-root(RPL_DEFAULT_INSTANCE,(uip_ip6adr_t *)&ipadr);
    uip_ip6adr(&ipadr, 0xaaaa, 0, 0, 0, 0, 0, 0);
    rpl_set_prefix(DAG, &ipadr, 64);
    printf("New RPL DAG created.\n");
} else {
    printf("Creation of new RPL DAGraph failed\n");
}
```



## Setting up of a UDP connection

```
//creates a new connection using UDP to the client's port
server_con = new_udp(NULL, UIP_HTON(UDP_CLIENT_PORT), NULL);
if(server_con == NULL) {
    printf("UDP connection not available, EXIT!\n");
    PROCESS_EXIT();
}
//ping the connection to the local port of the server'
udpbind(server_con, UIP_HTON(UDP_SERVER_PORT));

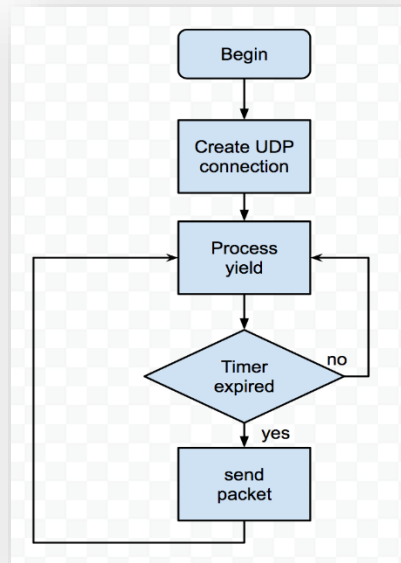
printf("Server connection created with the remote address ");
print6addr(&server_con->ripaddr);
printf(" (local) remote port %u/%u\n", UIP_HTON(server_con->lport),
UiP_HTON(server_con->rport));
```

## Receives and Processes incoming packet

```
while(1) {
    PROCESS_YIELD();
    if(env==tcp_ip_event) //if packet is available
        tcp_ip_handler();
    else if (env==sensors_event && data==&button_sensor) {
        printf("global repair initiation\n");
        rpl_repairroot(RPL_DEFAULT_INSTANCE);
    } }
static void tcp_ip_handler(void) { //call function (if the packet is available)
    char *app_data;
    if(uip_newdata()) {
        app_data = (char *)uip_app_data;
        app_data[uip_datalen()] = 0;

        //print the data of packet
        printf("DATA recieved '%s' from ", app_data);
        printf("%d", UIP_IP_BUF-> srcipaddr.u8[ sizeof(UIP_IP_BUF->srcipaddr.u8) - 1 ]);
        printf("\n");
    }
}
```

## UDP Client



Flow chart for UDP client

Here, UDP server will primarily perform two tasks.

1. Setting up of an UPD connection;
2. Send the packets to UDP server occasionally.

### Sets up UPD connection

```
/* establishment of a new connection with host */
client_con = new_udp(NULL, UIP_HTON(UDP_SERVER_PORT), NULL);

if(client_con== NULL) {
    printf("No UDP connection available, EXIT!\n");
    PROCESS_EXIT();
}
bind_udp(client_conn, UIP_HTON(UDP_CLIENT_PORT));
printf("Created a connection with the server ");
print6addr(&client_con->ripaddr);
printf(" (LOCAL) Port %u/%u\n", UIP_HTON(client_con->lport), UIP_HTON(client_con->rport));
```

## Sends packet

```
//set the time interval with SEND_INTERVAL
etimer_set(&periodic, SEND_INTERVAL);
while(1) {
    PROCESS_YIELD();

    if(ev==tcp_ip_event)
        tcp_ip_handler();
    if(etimer_expired( &periodic )) { //sends packet at every interval
        etimer_reset( &periodic );
        ctimer_set( &backoff_timer, SEND-TIME, send-packet, NULL);
    }
    static void send_packet(void *ptr)
    {
        static int sequence_id;
        char buff[MAX_PAYLOAD_LEN];
        sequence_id++;
        printf("The data is send to %d 'Hello %d'\n", server_ipadr.u8[sizeof(server_ipadr.u8) - 1], sequence_id);
        sprintf(buff, "Hello %d (-client)", sequence_id);
        //to send the packet using client_conto the server
        uip-udp_packet_send_to( client_conn, buf, strlen(buf), &server_ip_adr, UIP_HTON(UDP_SERVER_PORT));
    }
}
```

---

## RPL-UDP Function Modification

---

Following are some important files that defines the basic RPL parameters and implementation:

- rpl-config.h
- rpl-of0.c
- rpl-mhrof.c

Here we have mentioned only the important and modified functions of different files.

**rpl-config.h:** Following changes have been done to the file.

### RPL\_CON\_STATS

```
/* RPL_CON_STATS */  
#ifndef RPL_CON_STAT  
#define RPL_CON_STAT 1 // 1 to enable : 0 to disable  
#endif
```

Here we have enabled RPL Configuration Statistics.

### RPL\_DAG\_MC

```
/*FileName: RPL_CONF_DAG_MC */  
#ifdef RPL_CON_MCDAG  
#define RPL_DAG_MC RPL_CON_MCDAG  
#else  
#define RPL_DAG_MC RPL_MCDAG_NONE  
#endif
```

Here the RPL configuration metric using Energy and ETX is supported. We have added this two functionality further more can be added based on the requirement.

## **RPL\_CONFOF**

```
/*FileName: RPL_CONFOF */  
#ifdef RPL_CONFOF  
#define RPL-Of RPL_CONFOF  
#else  
#define RPL_Of rpl_mrhof  
#endif
```

It configures the RPL Objective Function. Default Function - ETX. It is defined as the name of an rpl\_of object linked in the image of the system pertaining, e.g., rpl\_of0.

## **RPL\_LEAF\_ONLY**

```
#ifdef CON_RPL_LEAF_ONLY  
#define RPL_ONLY_LEAF CON_RPL_LEAF_ONLY  
#else  
#define RPL_ONLY_LEAF 0  
#endif
```

It decides whether the node is the leaf or not.

## **RPLINIT\_LINKMETRIC**

```
#ifndef RPL_CONF_INIT_LINKMETRIC  
#define RPLINIT_LINKMETRIC 1  
#else  
#define RPLINIT_LINKMETRIC RPL_CONF_INIT_LINKMETRIC  
#endif
```

This is the initial metric assigned to the link when the ETX value is unknown. It can be changed to any desirable value.

## rpl-of0.c

An implementation of RPL's objective function 0 is in this file.

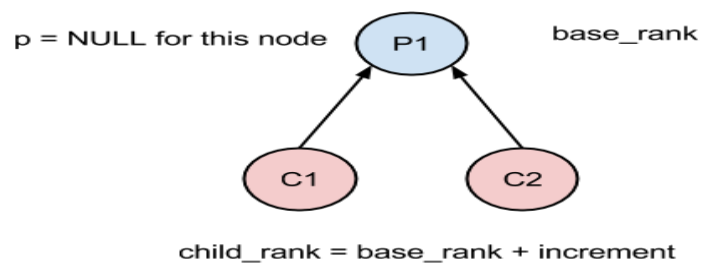
*calculate\_rnk(rpl\_parent\_t \*p, rpl\_rank\_t base\_rank)*

```
static rpl_rank_t calculate_rank(rpl_parent_t *parent, rpl_rank_t base_rnk)
{
    rpl_rank_t inc;
    if(base_rnk == 0) {
        if(parent == NULL) {
            return INF_RANK;
        }
        base_rnk = parent->rank;
    }
    if(inc=parent!=NULL)
        return parent->dag->instance->mini_hop_rank_in_c;
    else
        return DEFAULT_RANK_INC;

    if((rpl_rank_t)(base_rnk+increment)<base_rnk) {

        printf("RPL: Of_0 rank %d is inc++ to infinite rank because of wrapping\n",base_rank);

        return INF_RANK;
    }
    return base_rnk+increment;
}
```



In this segment we calculate the rank of the node. Node rank depends on the rank of its parent along with the base rank.

- If base rank computes to zero and the node doesn't have any parent, then the rank is set to Infinite.

- If base rank is equal to zero and the node has a parent, the base rank is set to that of the parent.
- If base rank is not zero, and also if it has a parent, then the increment equates equal to `min_hop_rank_in_c` of the parent in the instance of the DA Graph, or else if the parent is not present then to the `DEFAULT_RANK_INCREMENT`.

Summarizing: If the `parent==NULL` then the base rank is `inc++` by a default increment or else it applies information about the parent so as to increment the `base_rank`. At last, if the (new calculated rank)  $<$  (base rank), then due to wrapping around, the new rank sets to infinite. The equated rank is equal to (increment + base rank).

*best\_dag(rpl\_dag\_t \*d1, rpl\_dag\_t \*d2)*

```
static rpl_dag_t *best_dag(rpl_dag_t *dag1, rpl_dag_t *dag2)
{
    if(dag1->grounded) {
        !dag2->grounded ? return dag1 : return dag2;

        if(dag1->preference<dag2->preference)
            return d2;
        else {
            if(dag1->preference>dag2->preference)
                return dag1;
        }
        dag2->rank<dag1->rank ? return dag2: return dag1;
    }
}
```

Here we compare two Directed Acyclic Graphs (DAGs) created and then returns the best from the two taken into consideration. DAGs (they are passed input `dag1` & `dag2`) according to the Objective function.

There are three criteria's for finding the best DAG here:

- If the DAG is grounded or not.
- Details of the preference metrics for each DAG.
- Each DA Graph's rank.

*best\_parent(rpl\_parent\_t \*p1, rpl\_parent\_t \*p2)*

```
static rpl_parent_t *best_parent( rpl_parent_t *parent1, rpl_parent_t *parent2)
{
    rpl_rank_t rank1, rank2;
    rpl_dag_t *d;
    printf("Comparing the parent (the one present) ");
    print6addr(rpl_get_parent_ipadr( parent1));
    printf("(Conf: %d : Rank %d) along with parent", parent1->LINKMETRIC, parent1->rank);
    /*Displays confidence and rank of the node in accordance to the parent.*/
    print6addr(rpl_get_parent_ipaddr(parent2));
    printf(" (Confidence %d : Rank %d)\n", p2->LINKMETRIC, p2->rank);
    rank1 = DAG_RANK(p1->rank, parent1->d->instance) * RPL_MIN_HOP_RANK_IN_C +
        parent1->LINKMETRIC;
    rank2 = DAG_RANK(parent2->rank, parent1->d->instance) * RPL_MIN_HOP_RANK_IN_C +
        parent2->LINKMETRIC;
    d = (rpl_dag_t *) parent1; /*The parents should be in the same DAG. */

    if(rank1 < rank2 + MIN_DIFERENCE && rank1 > rank2 - MIN_DIFERENCE)
        return d->preferred_parent;
    else if(rank1 < rank2)
        return parent1;
    else
        return parent2;
}
```

Here we compare the parents of the node and then returns the best in accordance to the defined objective function. We compare the parents on the basis of their rank and their ETX. It is important as the formation on root is dependent on it after choosing the best Directed Acyclic Graph (DAGs).



## rpl-mhrof.c

This file shows the Minimum Rank with Hysteresis Objective Function (MRHOF). The objective function (OF) use ETX as a routing metric and also have stubs for energy metric.

*calculate\_path\_metric(rpl\_parent\_t \*p)*

```
static rpl_path_metric_t calculate_path_metric(rpl_parent_t *parent)
{
    if(parent == NULL) {
        return (MAX_COST_OF_PATH) * (RPL_MC_DAG_ETX_DIVISOR);
    }
    #if RPL_DAGMC==RPL_DAGMC_NONE
        return parent->rank+(uint16_t)parent->LINKMETRIC;
    #elif RPL_DAGMC==RPL_DAGMC_ETX
        return parent->mc.obj.etx +(uint16_t)parent->LINKMETRIC;
    #elif RPL_DAGMC == RPL_DAGMC_ENERGY
        return parent->mc.obj.energy.energy_est+(uint16_t)parent->LINKMETRIC;
    #else
        #error "RPL_MCDAG configuration not supported. Goto rpl.h."
    #endif
}
```

Here we are calculating path metric with respect to the Objective Function (OF). If there is a situation when no parent is there then Maximum Path Cost is used to calculate the default metric. If OF is not been mentioned, then the path metric is calculated with the rank sum. If the OF is using energy we add energy value to the link metric. The best parent calls the function to compare the path metrics of the parents under consideration.

*callback\_neighbor\_link(rpl\_parent\_t \*p, int status, int numtx)*

```
static void callback_neighbor_link(rpl_parent_t *parent, int status, int numtx)
{
    uint16_t rec_ETX= parent->LINKMETRIC;
    uint16_t packetETX = numtx * RPL_MC_DAG_ETX_DIVISOR;
    uint16_t new_ETX;

    /* No penalty on ETX in case of any transmission or collision error. */
    if(status == MAC_TX_OK || status == MAC_TX_NOACK) {
        if(status == MAC_TX_NOACK) {
            packetETX = MAX_LINK_METRIC * RPL_MC_DAG_ETX_DIVISOR;
        }
    }
}
```

```

new_ETX = ((uint32_t)rec_ETX* ETX_ALPHA + (uint32_t)packetETX * (ETX_SCALE - ETX_ALPHA)) /
ETX_SCALE;

printf("Changed from %u to %u (Packet ETX Value = %u)\n",
      (unsigned) (recorded_etx/RPL_MC_DAG_ETX_DIVISOR),
      (unsigned) (new_ETX/RPL_MC_DAG_ETX_DIVISOR),
      (unsigned) (packet_ETX/RPL_MC_DAG_ETX_DIVISOR));
parent->LINKMETRIC=new_ETX;
}
}

```

Link-layer's neighbor information is received by the function. We set the parameter value to 0. Current ETX(estimated transmissions) is specified by the numetx for the neighbor. The recorded ETX value is also provided. We then calculate new\_ETX by using the formula both for the recorded and packet ETX values. We update the link metric with a new defined ETX value.

*calc\_rank(rpl\_parent\_t \*p, rpl\_rank\_t base\_rank)*

```

static rpl_rank_t calc_rank(rpl_parent_t *parent, rpl_rank_t b_rank)
{
    rpl_rank_t nrank;
    rpl_rank_t rank_inc;

    if(parent==NULL) {
        if(b_rank==0) {
            return INF_RANK;
        }
        rank_inc = RPL_INIT_LINK_METRIC*RPL_MC_DAG_ETX_DIVISOR;
    } else {
        rank_inc=parent->LINKMETRIC;
        if(b_rank==0) {
            b_rank=parent->rank;
        }
    }

    if(rank_inc > INF_RANK-b_rank) {
        /* Max rank achieved */
        nrank=INF_RANK;
    } else {

```

```

/* Rank is calculated based on the information of new rank fro DIO. */
nrank=b_rank+rank_inc;
}
return nrank;
}

```

Here in the OF, the rank\_inc is the inc++. Parent = NULL, the rank\_inc equates to RPL\_INIT\_LINK\_METRIC. Else, it is equal to the LINKMETRIC. The new rank is added to the base rank.

*best\_parent(rpl\_parent\_t \*p1, rpl\_parent\_t \*p2)*

```

static rpl_parent_t * best_parent (rpl_parent_t *parent1,rpl_parent_t *parent2)
{
    rpl_dag_t *d;
    rpl_path_metric_t min_dif;
    rpl_path_metric_t parent1_metric;
    rpl_path_metric_t parent2_metric;

    d=parent1->d;
    /* Both parents are present in same DAG as required for rank calculation */
    min_dif=(RPL_MC_DAG_ETX_DIVISOR) / (PARENT_SWITCH_THRESHOLD_DIV);

    parent1_metric=calculate_path_metric(parent1);
    parent2_metric=calculate_path_metric(parent2);
    /* In case of similar ranks, it sustain the stability of preferred parent */
    if(parent1==d->preferred_parent || p2==d->preferred_parent) {
        if(parent1_metric<parent2_metric+min_dif &&parent1_metric>parent2_metric-min_dif) {
            printf("RPL: MRHOF hysteresis: %u <= %u <= %u\n",parent2_metric - min_dif, parent1_metric,
                parent2_metric+min_dif);
            return d->preferred_parent;
        }
    }
    return parent1_metric < parent2_metric ? parent1 : parent2;
}

```

Here we calculates the path metric for each parent. It is the basis for comparison.

At the beginning, we check if any parent is set as the preferred parent. If yes, then we choose it as the best parent else the one with the lower metric is defined as the best parent. Here, we only change the value of best parent to the new parent iff its metrics are less than that of the previous best parent. Here we have defined the PARENT\_SWITCH\_THRESHOLD\_DIV to 2.

Following shows the cases of the Parent Selection process:

- at the initial formation of network /or
- when the path cost to the neighbor(ing) nodes changes /or
- New node created/added appears in the neighbor of the node being focused on.

Here the DGRM model is employed.  
The following are the steps to create a replacement simulation:

### Start Cooja

To start the simulation, click on the Contiki folder and navigate to /tools/cooja directory and run "ant" to begin the COOJA simulation.

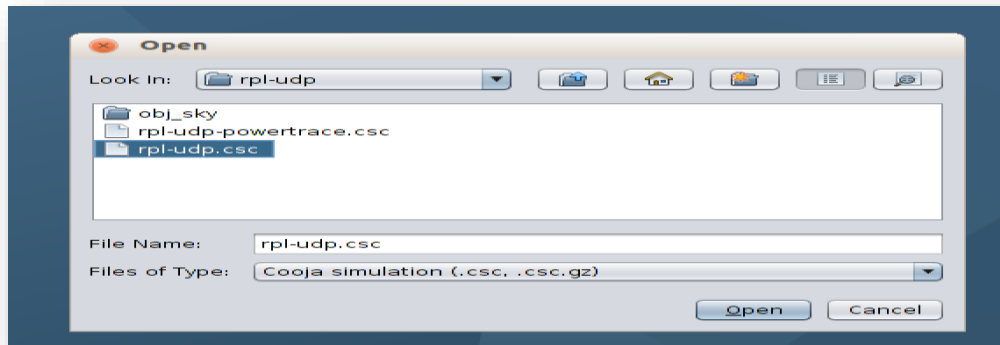
```
sudo ant run
```

### Open the simulation file

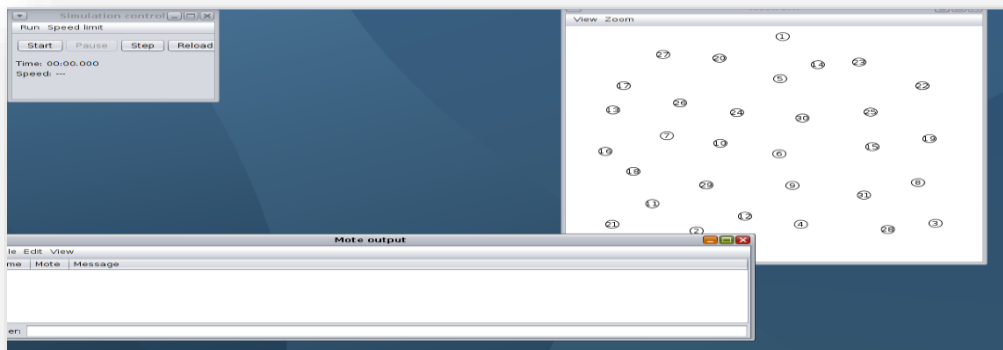
In the application, click on File->Open simulation->Browse

The dialogue box shows up, Open home/contiki-2.7/examples/ipv6/rpl-udp/rpl-udp.csc

**Note:** Clone the directory from github.com.



Simulation file shows up like this.



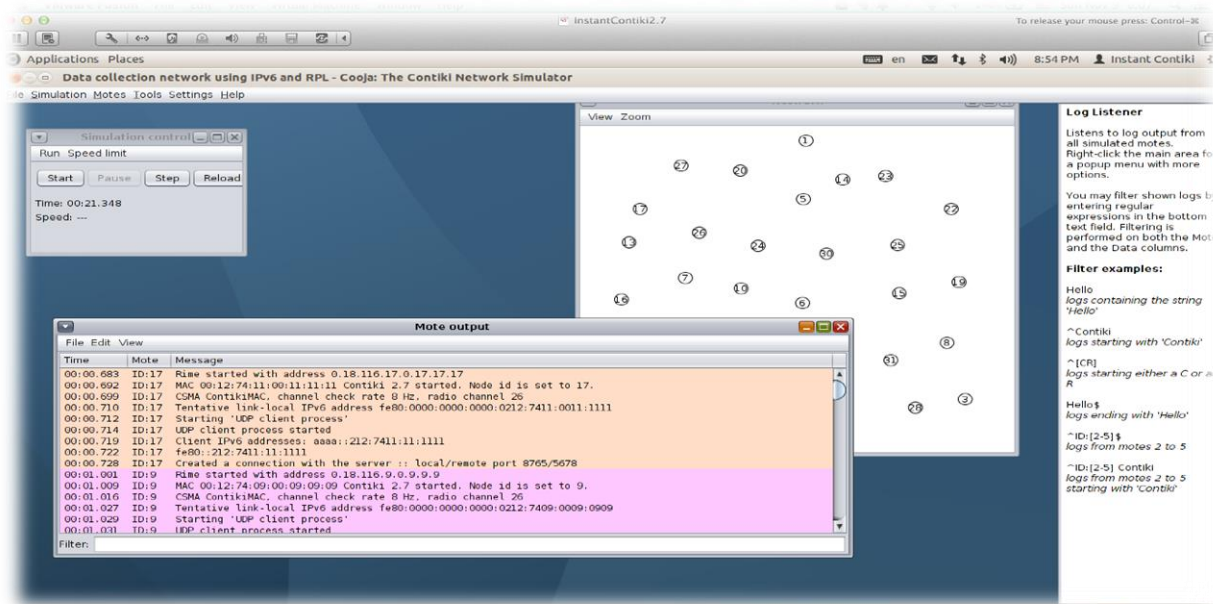
### Run Simulation

Press the *Start* button in the *SC- Simulation Control* window. This initializes the motes and allocate all the motes with a new Rime address and remaining initial processes.

## Output

The output and debug messages of the mote that are generated in the complete process can be viewed from the *Motes Output* window. In the output window, we can filter out the output on the basis of node ID to observe the specified node/mote. *Mote Output* has three main functions, i.e. *File*, *Edit* and *View*. The *File* option provides a functionality for saving the output to a file. There are options in *Edit* to copy the output (i.e. either full or a specific selected messages). Clear all messages option is used to clear the complete application window (i.e. bringing the workspace to its initial empty space).

The output messages that are saved in the file can be used later to make the observations, then plot graphs in accordance to the objective of the project (simulation).



## CHAPTER 4 – PERFORMANCE ANALYSIS

### Introduction

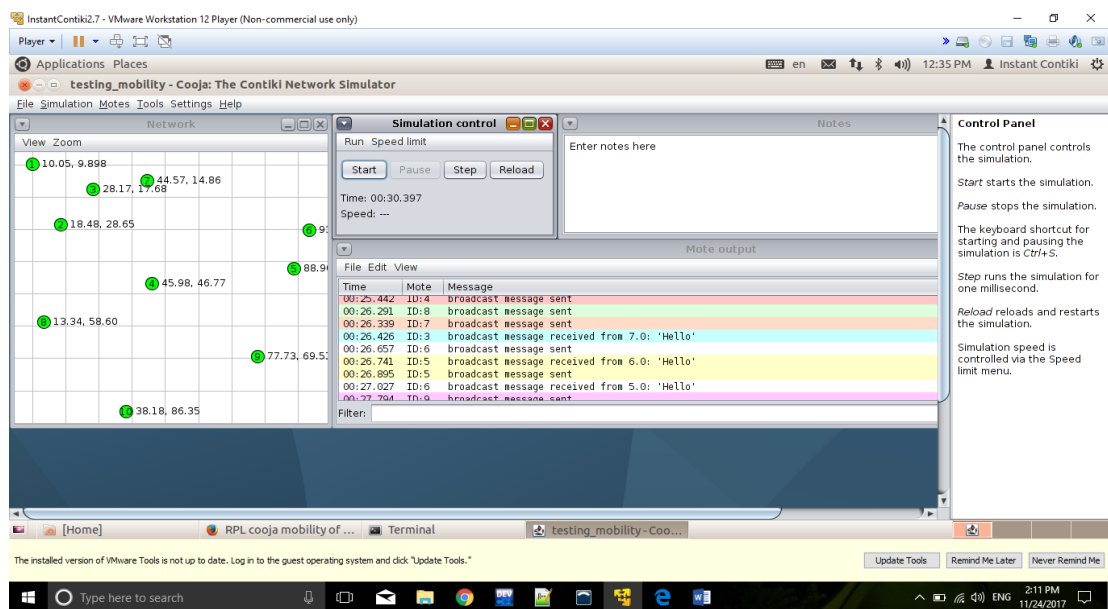
In this chapter we are going to focus on the performance analysis of the situations both when mobility is not enabled and with mobility enabled. Here we have considered two files.

One **testing\_mobility** – in this we have not enabled mobility. It is used to see the behavior of the motes without them being mobile i.e they are fixed in this case.

The other file is **mobility\_project** – here we have enabled the mobility plugin and defined a positions.dat file which shows different Cartesian locations of motes on the network graph. We have to modify this positions.dat accordingly as per where the respective mote goes after being on a given location, so as to increase the transmission rate and further, the throughput of the complete network being mobile.

### *mobility.csc*

1. As discussed in the previous chapter, create a new simulation, with 10 sky motes, positioned randomly.



2. Click on Start, in simulation control box. In view tab of Network, enable Radio Traffic, Mote Type, Positions and Radio Environment to analyze the situation.

## Without Mobility enabled:

Following screenshots shows the behavior and analyze various result parameters without mobility.

The screenshot displays the 'testing\_mobility - Cooja: The Contiki Network Simulator' window. It features a network map on the left with nodes at various coordinates. The central 'Simulation control' panel shows 'Run Speed limit' with buttons for 'Start', 'Pause', 'Step', and 'Reload'. The 'Time' is 01:06.260 and the 'Speed' is 52.49%. The 'Mote output' window shows a log of broadcast messages between nodes, including times, IDs, and messages. The 'Log Listener' panel on the right provides instructions on how to filter logs using regular expressions.

Time	Mote	Message
01:03.138	ID:2	broadcast message received from 3.0: 'Hello'
01:03.939	ID:10	broadcast message sent
01:04.215	ID:4	broadcast message sent
01:05.681	ID:2	broadcast message sent
01:05.801	ID:3	broadcast message received from 2.0: 'Hello'
01:05.971	ID:8	broadcast message sent
01:06.163	ID:1	broadcast message sent
01:06.239	ID:9	broadcast message sent

Speed: 52.48% - (this is the transfer speed efficiency)

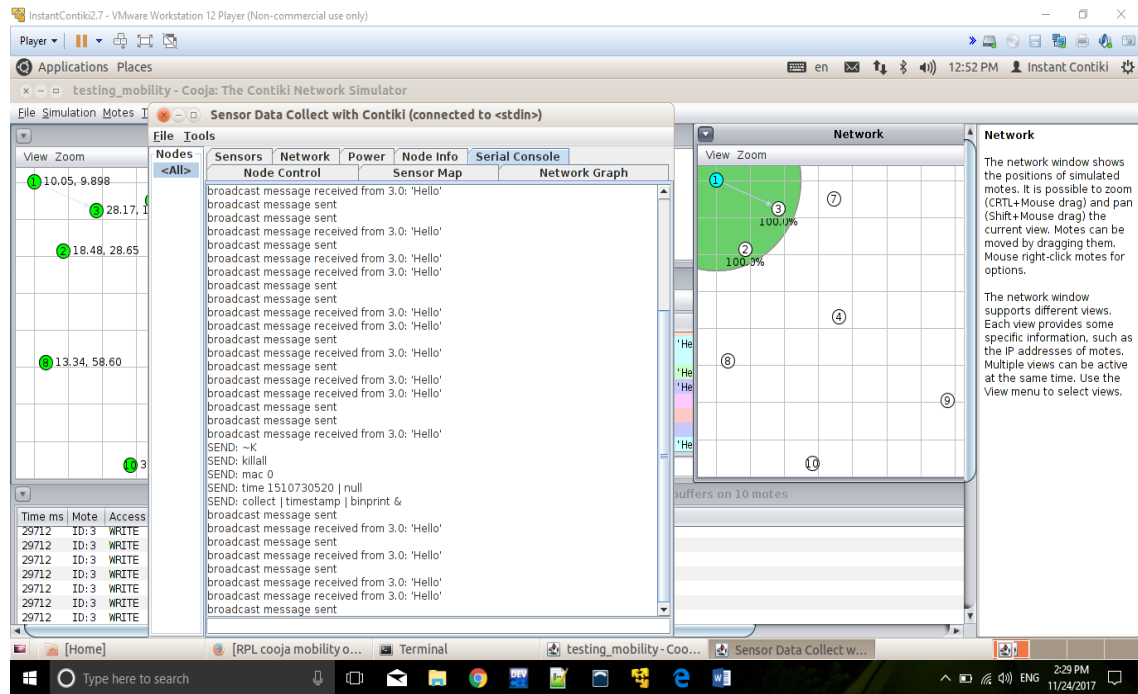
Mote Output- It shows the connection establishment between different motes and their respective messages transmission between them, along with the behavior of parameters like speed for different time durations during the transmission of messages.

This screenshot shows a more detailed view of the simulator's log output. The 'Mote output' window is filled with a list of broadcast messages, each with a timestamp, node ID, and the message content. A 'Speed: 40.56%' indicator is visible. The 'Control Panel' on the right explains the functions of the simulation controls: 'Start' starts the simulation, 'Pause' stops it, 'Step' runs it for one millisecond, and 'Reload' restarts it. It also notes that simulation speed is controlled by the 'Speed limit menu'.

Time	Mote	Message
03:11.072	ID:2	broadcast message sent
03:11.075	ID:4	broadcast message sent
03:11.175	ID:3	broadcast message received from 2.0: 'Hello'
03:11.827	ID:1	broadcast message sent
03:11.926	ID:3	broadcast message received from 1.0: 'Hello'
03:13.223	ID:5	broadcast message sent
03:13.276	ID:6	broadcast message received from 5.0: 'Hello'
03:14.917	ID:7	broadcast message sent
03:15.040	ID:3	broadcast message sent
03:15.051	ID:3	broadcast message received from 7.0: 'Hello'
03:15.243	ID:6	broadcast message sent
03:15.263	ID:2	broadcast message received from 3.0: 'Hello'
03:15.282	ID:1	broadcast message received from 3.0: 'Hello'
03:15.956	ID:4	broadcast message sent
03:16.054	ID:1	broadcast message sent
03:16.091	ID:9	broadcast message sent
03:16.176	ID:3	broadcast message received from 1.0: 'Hello'
03:16.229	ID:8	broadcast message sent
03:16.728	ID:10	broadcast message sent
03:16.970	ID:2	broadcast message sent
03:17.051	ID:3	broadcast message received from 2.0: 'Hello'
03:17.980	ID:5	broadcast message sent
03:19.133	ID:3	broadcast message sent
03:19.160	ID:1	broadcast message received from 3.0: 'Hello'
03:21.194	ID:1	broadcast message sent
03:21.302	ID:3	broadcast message received from 1.0: 'Hello'
03:21.380	ID:9	broadcast message sent
03:21.457	ID:4	broadcast message sent
03:22.073	ID:7	broadcast message sent
03:22.177	ID:3	broadcast message received from 7.0: 'Hello'
03:23.034	ID:8	broadcast message sent
03:23.133	ID:6	broadcast message sent



## Serial Console Details:



The screenshot displays the Cooja network simulator interface. The main window is titled "Sensor Data Collect with Contiki (connected to <stdin>)" and is divided into several panes. The "Serial Console" pane shows a series of broadcast messages: "broadcast message received from 3.0: 'Hello'", "broadcast message sent", and "broadcast message received from 3.0: 'Hello'". Below this, there is a "Time ms | Mote | Access" table with the following data:

Time ms	Mote	Access
29712	ID:3	WRITE
29712	ID:3	WRITE
29712	ID:3	WRITE
29712	ID:3	WRITE
29712	ID:3	WRITE
29712	ID:3	WRITE

The "Network" pane shows a grid with 10 nodes (motes) numbered 1 through 10. A green shaded region is centered around node 1, indicating its transmission range. The "Nodes" pane on the left shows the coordinates of the motes: (10.05, 9.898), (28.17, 18.48), (18.48, 28.65), (13.34, 58.60), and (1, 3).

The green region here shows the range of the mote present in the network, this can be changed as per the requirement by following the steps:

- Right Click on the respective mote.
- Go-to the Change Transmission Ranges.
- The box will appear. Change the ranges as per the requirement of the network.

**In the above screenshot, we can see the serial output representing some messages not been able to be transmitted, and hence got killed as the distance between the source and the destination are comparatively more than the ranges of the motes specified. Not being in range of each other, the motes are not able to contact each other.**

**To resolve this problem we are about to introduce Mobility in this scenario and further see the simulation results that to what extent the situation improves.**

## **With Mobility Enabled – mobility\_project**

Mobility allows the motes to alter their positions with respect to the positions.dat file which is needed to be changed to define the respective positions of motes after each interval so as to bring all the motes together in a specified arrangement so that they become capable of communicating with each other when its required.

### **Position.dat File Structure**

Position.dat file contains this information

**#node time(s) x y**

**0 0.0 0 20**

**0 1.0 10 20**

**0 2.0 12 0**

**0 3.0 0 10**

**0 4.0 0 0**

1st column - **node number**;                      second column - **time-stamp**,  
third column - **x-coordinate**;                      fourth column - **y-coordinate**

**#node time(s) x y**

**0 0 0 2 0**

It tells node 0 (i.e. mote 1) at 0.0sec will be positioned with position co-ordinates(0,20)

**Note:** Here node 0 will be the 'mote 1' on the cooja. If we specify node 1 in the first column it will be for mote 2

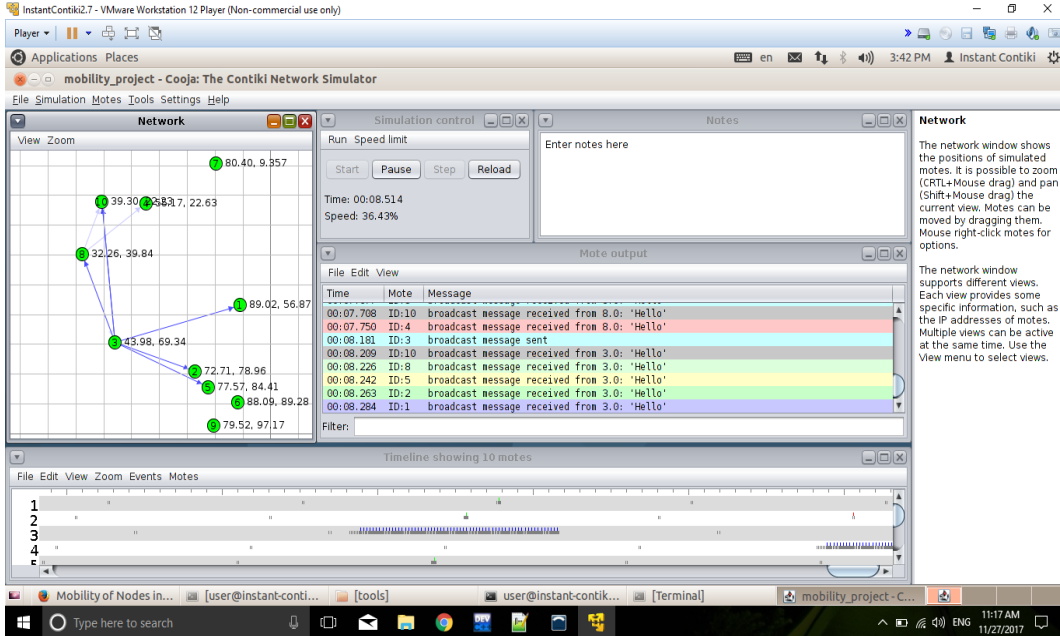
In general, positions.dat node (n) will be mote (n+1) on Cooja.

The position.dat file needs to be changed. Replace file with the new file under **Tools>Mobility**.

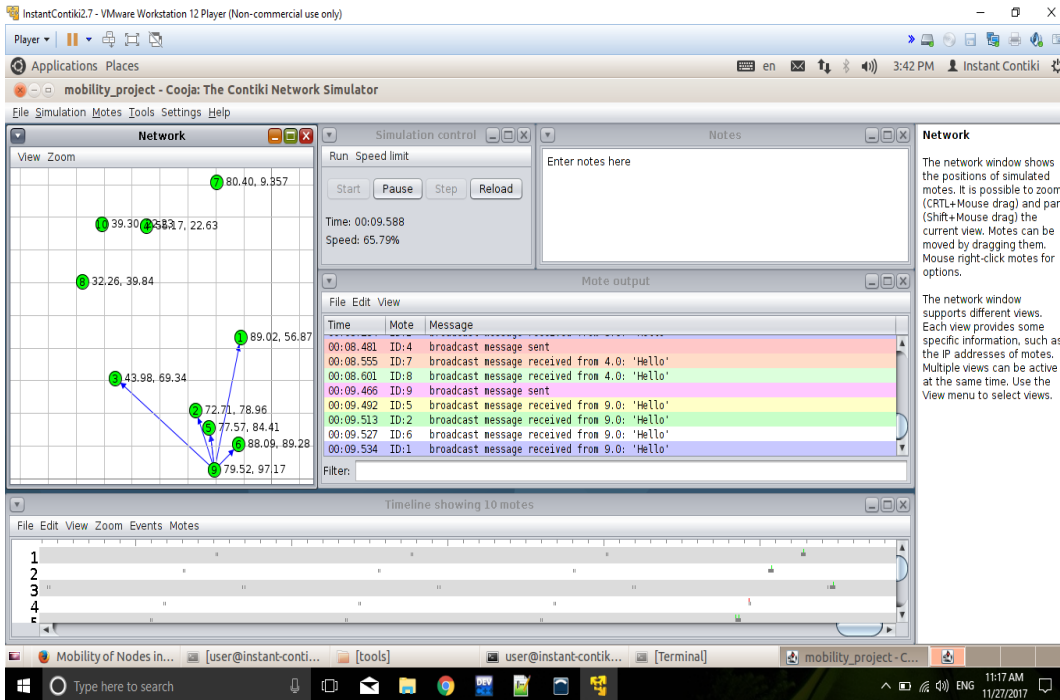
Start the simulation by adding the number of motes required in the simulation process.

The new positions defined will get loaded and the nodes will move in accordance with the new specified positions.

Further screenshots show the mobility in COOJA with 10 motes positioned randomly at first:

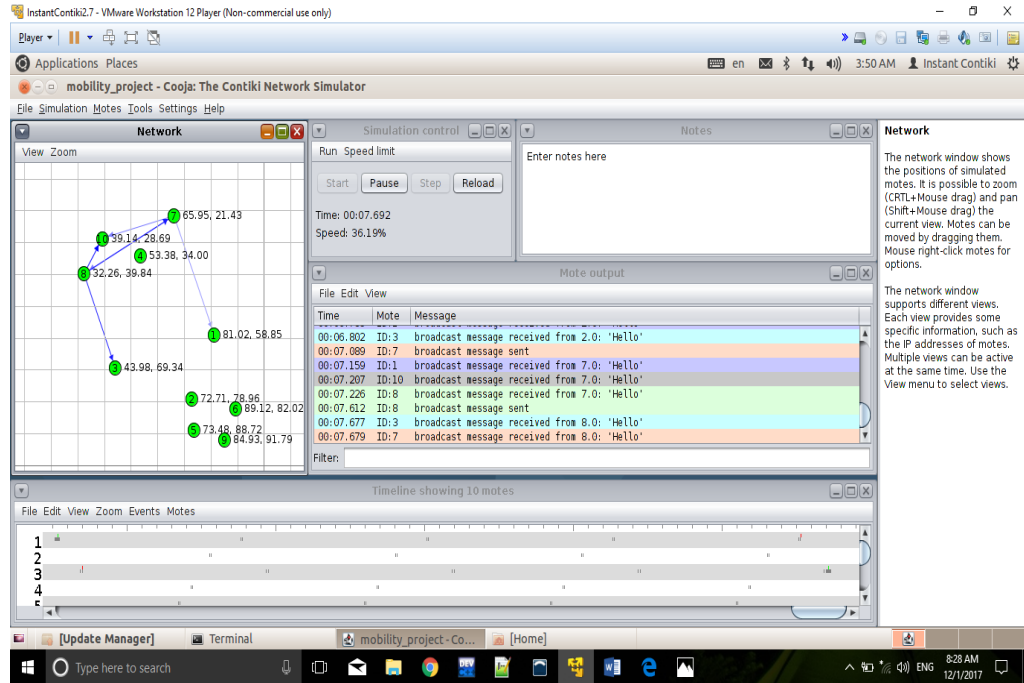


➤ First is the initial position interactions between the motes.

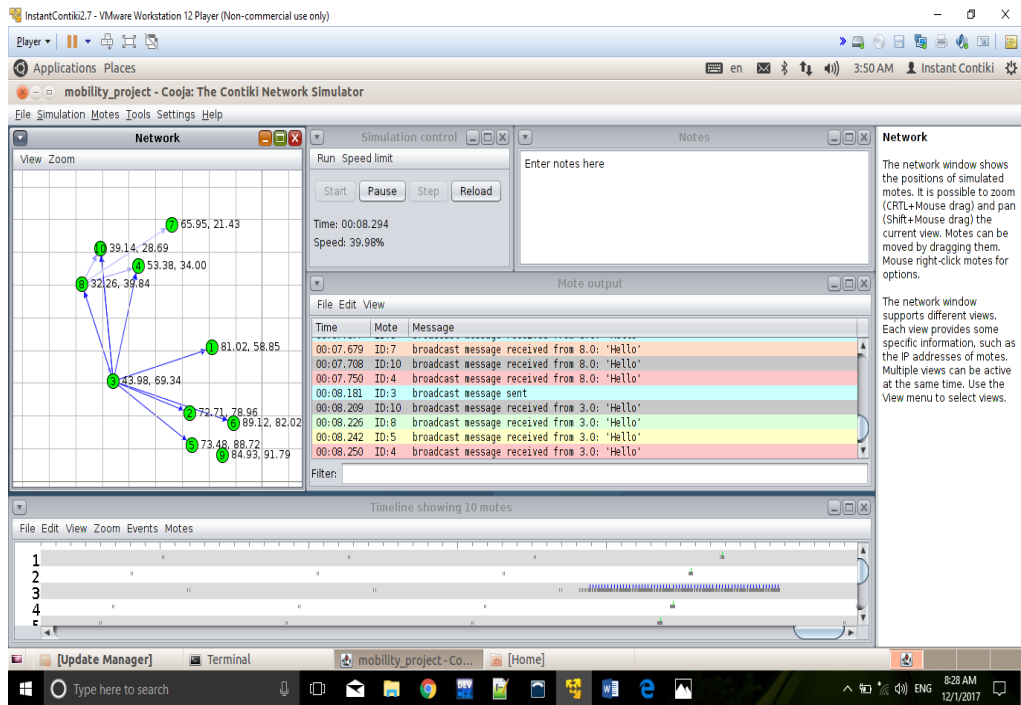


We can see above that some of the motes are not able to contact each other due to restriction of the mote network range.

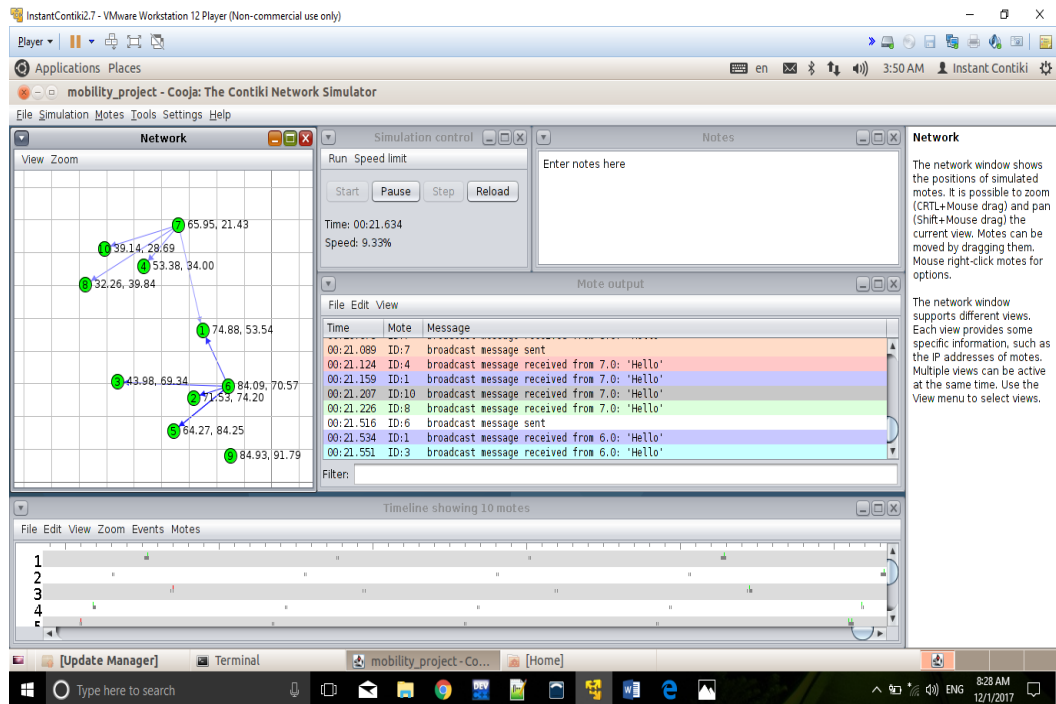
- Further Screenshots shows change in positions as the mobility is enabled. This shows the initial phase when motes are at their initial positions.



This is the scenario when motes have changed their positions for the first time so as to be in connection with each other.

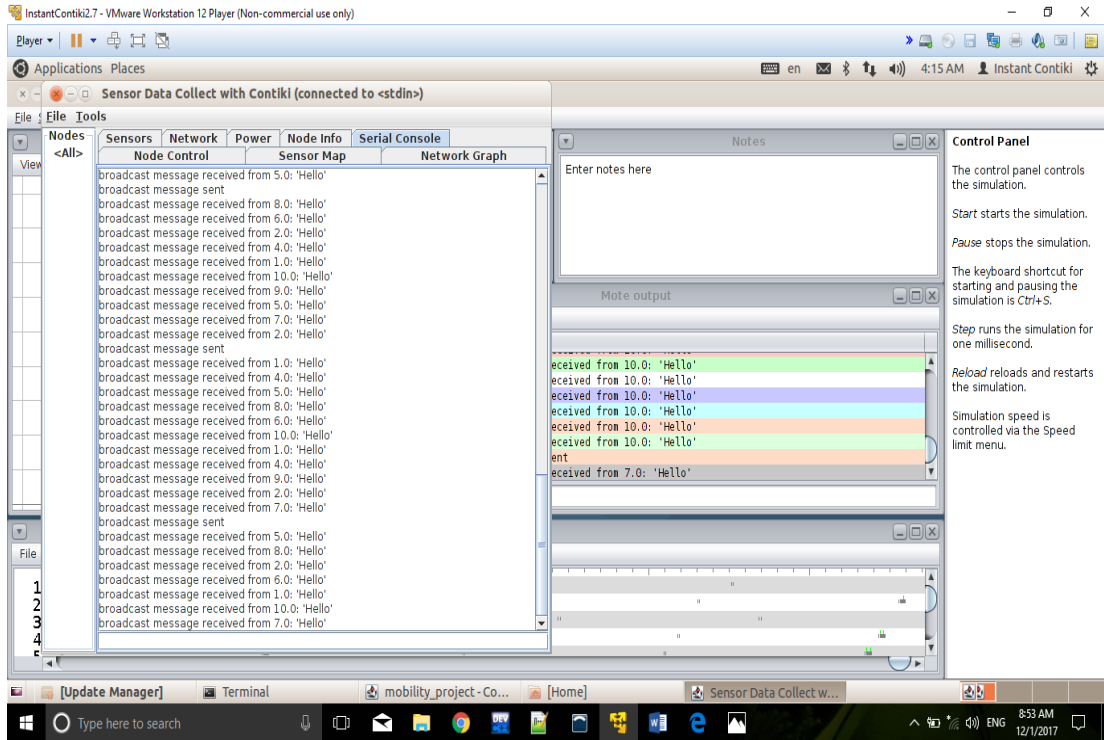


Motes changing their positions further to come in the range of the other motes.

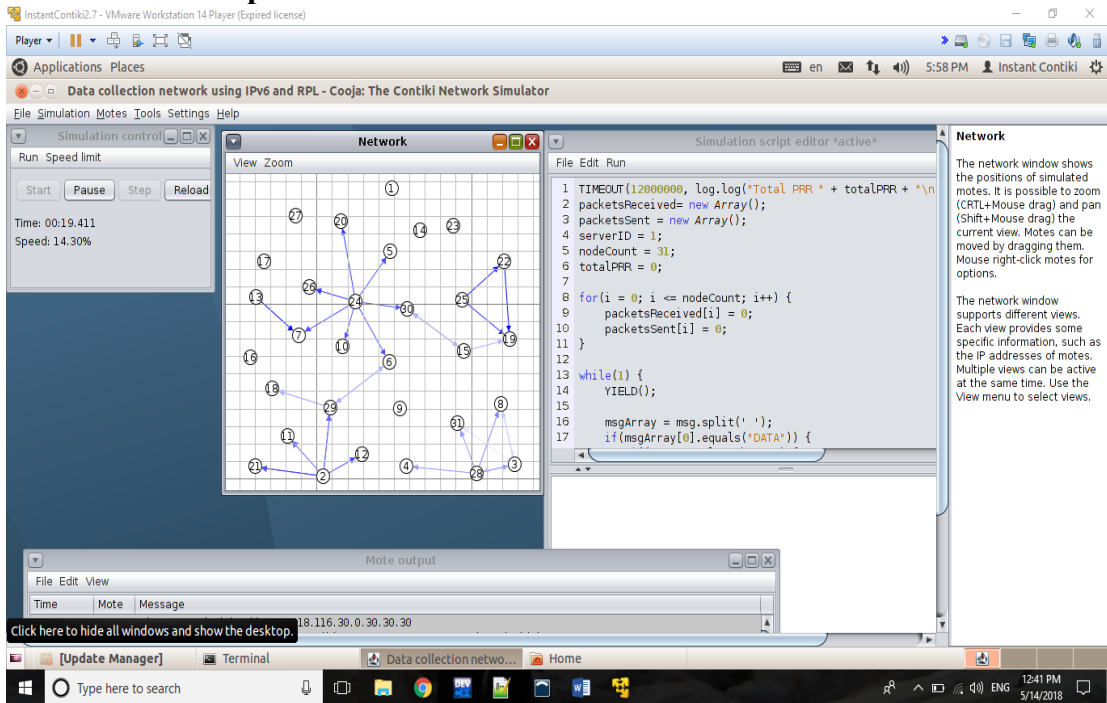


This is the scenario when motes have changed their positions for the first time so as to be in connection with each other.





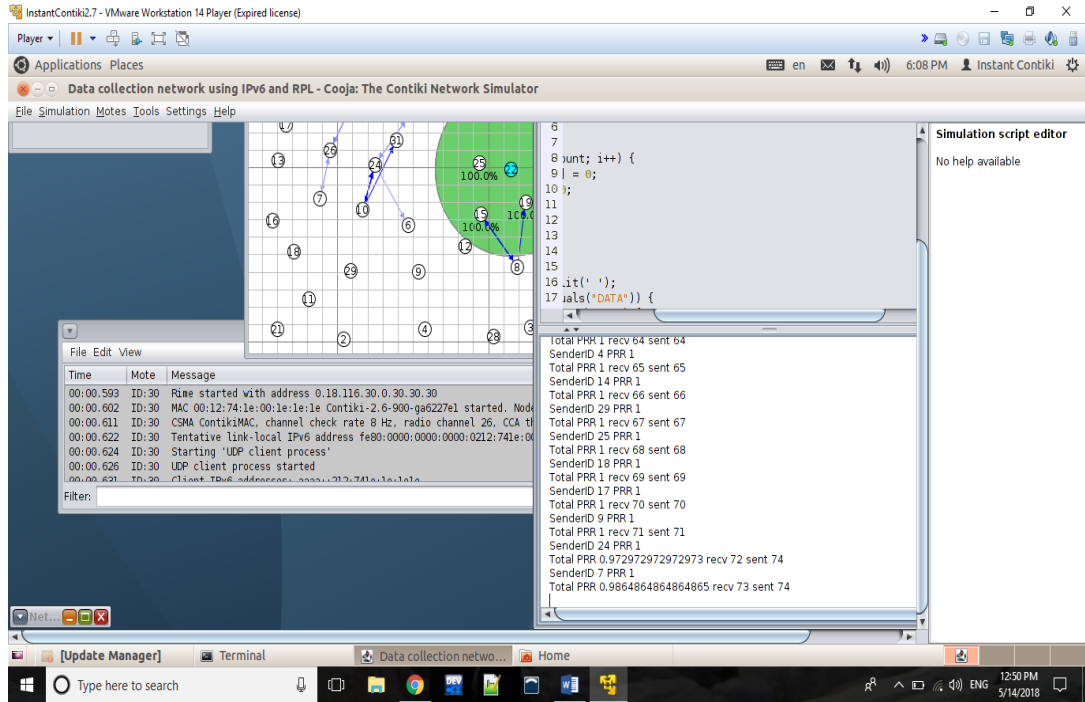
## RPL-UDP Implementation Screenshots:



This shows the modification of the objective function of RPL-UDP sink/source implementation





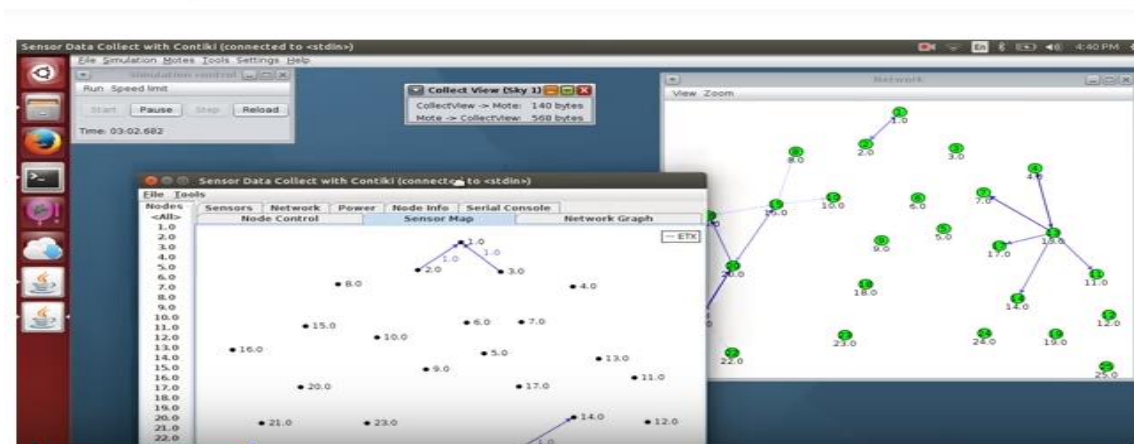


This describes the packets sent/receive after function modification.

## Chapter 5 – CONCLUSIONS

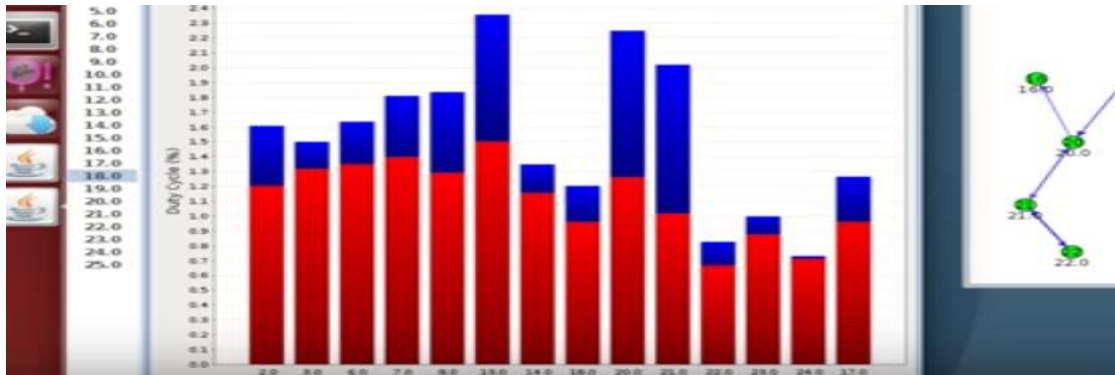
- Sensor data collect using Collect View (Sky1) (initial)

The Sensor Map consists of all the nodes ranging from 1.0 to 25.0  
Clearly the arrows from 2.0 and 3.0 are pointing towards 1.0 and in this way nodes are starting to connect with each other which is initial step in the map.



- Average Radio duty cycle

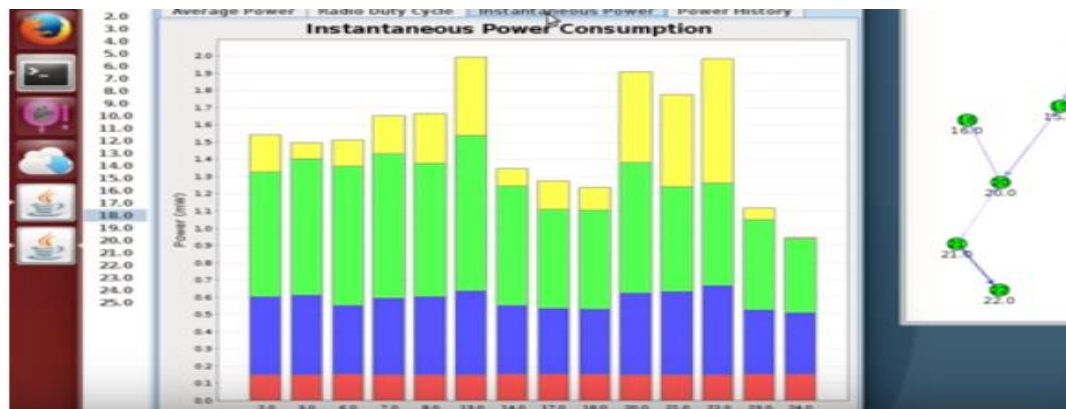
Y axis consists of duty cycle and X axis consists of nodes  
There is a constant fluctuation in the readings of the duty cycle. Firstly, it moves up to 1.6 and then lowers it down to 1.5 and then again increases itself to 1.6 and then keeps on increasing up to 2.4 and then lowers itself again to 1.5



- Instantaneous power consumption

Y axis consists of Power and X axis consists of Nodes

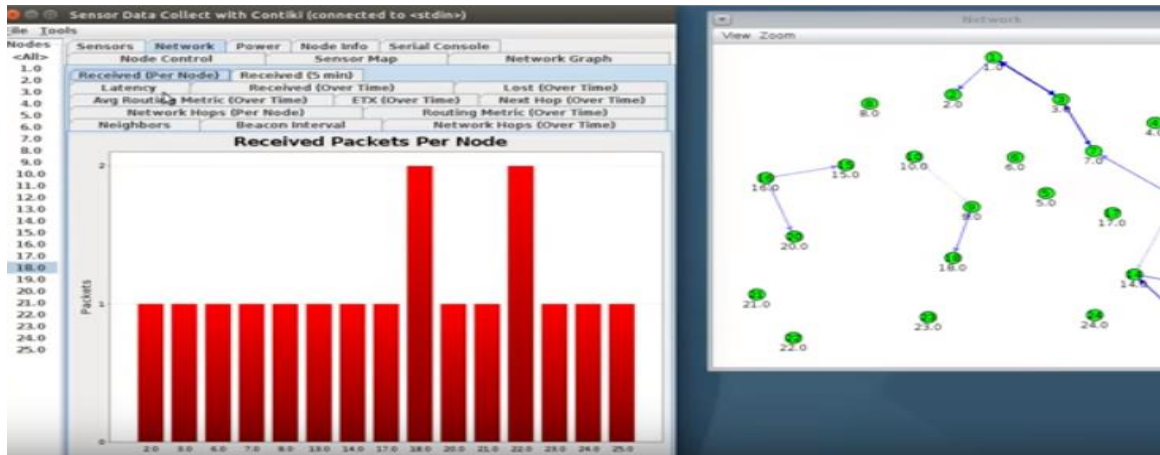
Fluctuations can be noticed again in this graph, Power consumption is around 1.5 and a little above up to 6.0 then it increases to 1.6. Clearly there is no specific pattern and more of a fluctuation can be noticed



- Received packets per node

Y axis consists of Power and X axis consists of Nodes

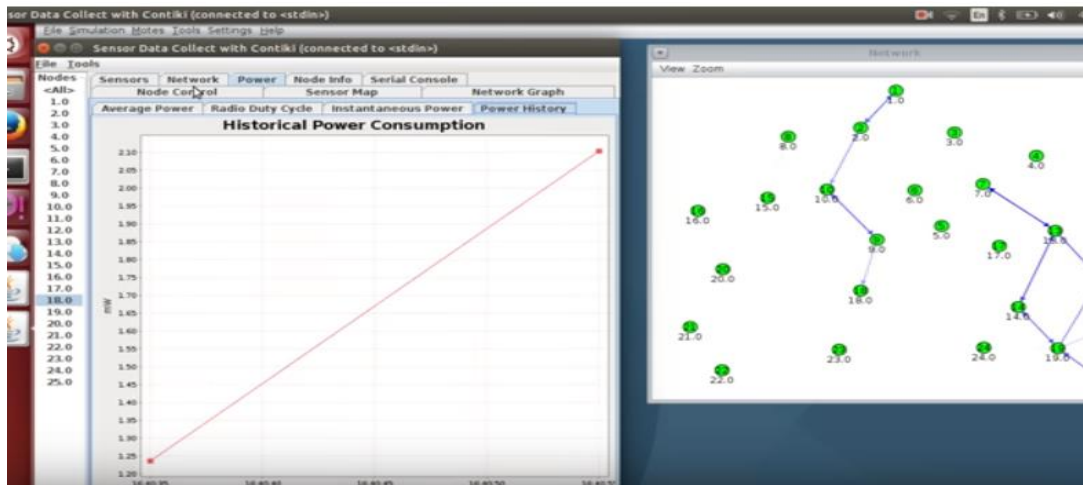
Most of the graph looks constant with the packets value being 1 for all the nodes except 18.0 and 22.0 which changes packet value to 2 and this brings a little variation in consistency of the graph.



- History power consumption

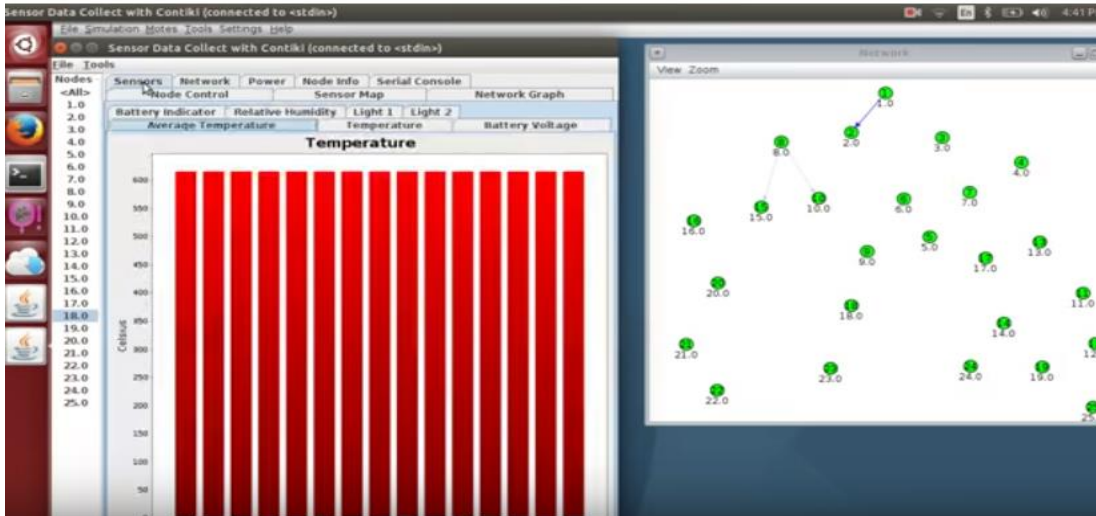
Y axis consists of mW and X axis consists of Time

Clearly it can be noticed that the graph formed between X axis and Y axis is a linear graph  
Historical power consumption varies linearly with time.



- Temperature

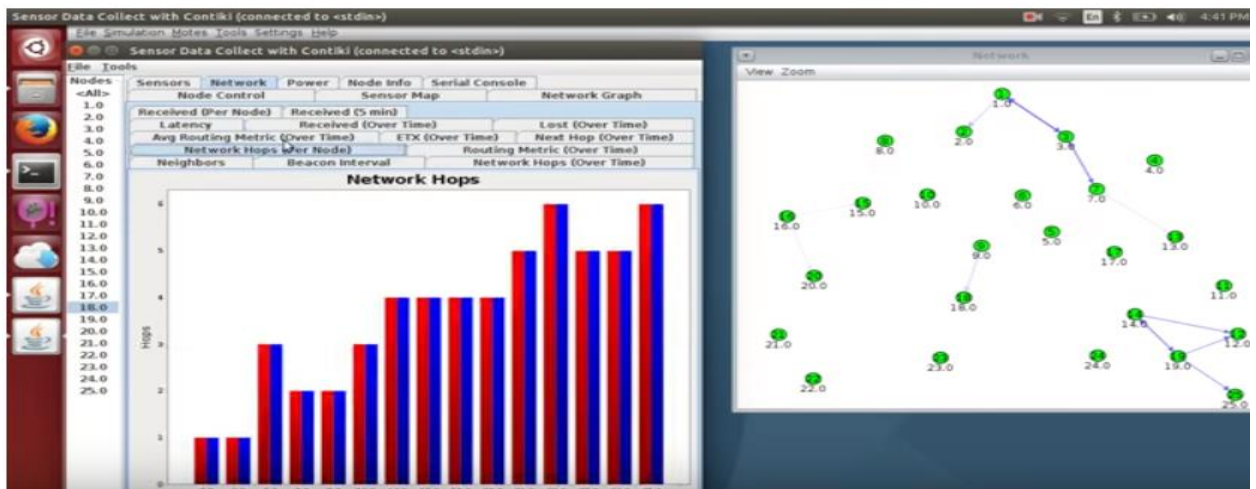
Y axis consists of Celsius and X axis consists of nodes. The graph clearly shows consistency and the temperature for all the nodes starting from 3.0 to 25.0 consists of the same value starting from 0 and ending up at 600 in Celsius.



- Network Hops

Y axis consists of Hops and X axis consists of Nodes

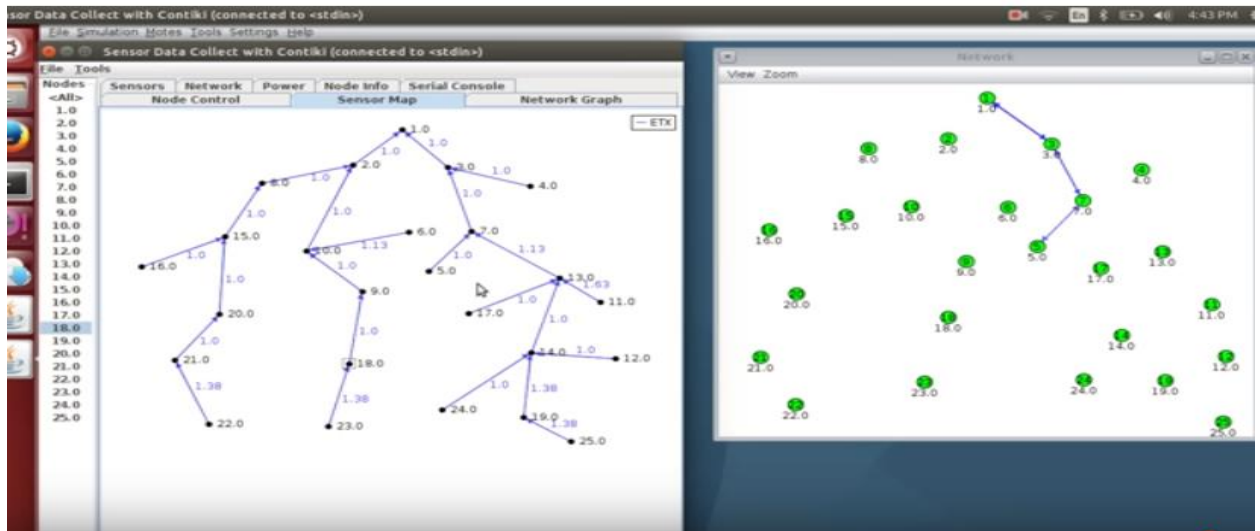
This graph again shows fluctuation and the hop values keep on varying for the different nodes. For nodes 2.0 and 3.0 the hop value remains 1 and for 6.0 it varies to 3 and this variation can be seen further in the graph



- Sensor data collect using Collect View (Sky1) (final)

The Sensor Map consists of all the nodes ranging from 1.0 to 25.0

It can clearly be seen that all the nodes are getting connected in one way or the other Incoming Arrows can clearly be seen joining or pointing the nodes



## Applications:

Mobility is one of the most important feature to be enabled in RPL as it will expand the ideas and application domain. There are many real world applications

- Ocean sensors to check various details like depth, types of minerals found, habitat research etc.
- Sensors in grain fields to check the humidity content and soil quality check are a few application to name.
- In Big Data and Cloud Techniques

With the introduction of mobility the scope of IoT has further increased to a great extent making the angle of vision and ideas more vivid. Now a days a lot of applications are in requirement which can be accomplished after the mobility has been achieved in RPL.

## Chapter 6 –REFERENCES

- Olfa Gaddour, Anis Koubaa / “**RPL in a nutshell a survey**” /CES Research Unit, National School of Engineers of Sfax, University of Sfax, Tunisia /Article history: 11 Received 23 December 2011 12 Received in revised form 2 June 2012 13 Accepted 16 June 2012
- Jorge Granjal, Edmundo Monteiro, George Silva /” **Security for the Internet of Things**”/ University of Coimbra /” IEEE Communications Surveys & Tutorials · July 2015
- Muneer Bani Yassein Department of Computer Science. Jordan University of Science and Technology Irbid, Jordan, Mohammed Q. Shatnawi Department of Computer Information Systems Jordan University of Science and Technology Irbid, Jordan, Dua’ Al-zoubi Department of Computer Science Jordan University of Science and Technology Irbid, Jordan /” **Application Layer Protocols for the Internet of Things: A survey**”
- Qusai Q. Abuein, Muneer Bani Yassein, Mohammed Q. Shatnawi, Laith Bani-Yaseen, Omar Al-Omari, Moutaz Mehdawi and Hussien Altawssi /” **Performance Evaluation of Routing Protocol (RPL) for Internet of Things**”/ International Journal of Advanced Computer Science and Applications, Vol. 7, No. 7, 2016
- **Contiki Development at ANRG, University of Southern California** /” [http://anrg.usc.edu/contiki/index.php/Mobility\\_of\\_Nodes\\_in\\_Cooja](http://anrg.usc.edu/contiki/index.php/Mobility_of_Nodes_in_Cooja)”/ last modified on 8 November 2014
- <https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiHicrimOnXAhWFtY8KHZ3PBNYQFggoMAA&url=https%3A%2F%2Fsourceforge.net%2Fp%2Fcontiki%2Fmailman%2Fmessage%2F29788769%2F&usg=AOvVaw0patY-2s3J6Ix7O-sTARRQ>
- <https://github.com/contiki-os/mspsim>
- Yu C., Cui Y., Zhang L., & Yang S., Zigbee wireless sensor network in environmental monitoring applications, In 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1-5, IEEE, 2009.
- Fu H. L., Chen H. C., & Lin P., APS: Distributed air pollution sensing system on Wireless Sensor and Robot Networks, Computer Communications, 35(9), pp. 1141-1150, 2012.
- Biswas K., Muthukkumarasamy V. and Singh K., An Encryption Scheme Using Chaotic Map and Genetic Operations for Wireless Sensor Networks, Sensors Journal, IEEE, 15(5), pp. 2801-2809, 2015.
- Bussi K., Dey D., Kumar M. and Dass B.K., A Lightweight Hash Function,

- International Association for Cryptologic Research, 2016.
- • Carl Endorf, Eugene Schultz and Jim Mellander, Intrusion Detection & Prevention, McGraw-Hill, 2004.
- • Chen C.L., Chen C.C. and Li D.K., Mobile Device Based Dynamic Key Management Protocols for Wireless Sensor Networks, Journal of Sensors, 2015.
- • Cochavy, Baruch, Method of efficiently sending packets onto a network by eliminating an interrupt, US Patent, 2004.
- • He D. & Zeadally S., An Analysis of RFID Authentication Schemes for Internet of Things in Healthcare Environment Using Elliptic Curve Cryptography, IEEE Internet Things J. 2, pp. 72–83, 2015.
- • Mallikarjunaswamy N. J., Latha Yadav T. R., and Dr. Keshava Prasanna, Message Authentication Protocol for Lifetime Proficient Hash Based Algorithm in Wireless Sensor Networks, Int. J. Advanced Networking and Applications, vol. 07, pp. 2963-2966, 2016.
- • Biswas, K., Muthukumarasamy, V. and Singh, K.. An Encryption Scheme Using Chaotic Map and Genetic Operations for Wireless Sensor Networks. Sensors Journal, IEEE, 15(5), pp.2801-2809.(2015)
- • Bussi, K., Dey, D., Kumar, M. and Dass, B.K. Neeva: A Lightweight Hash Function.(2016)
- • Carl Endorf, Eugene Schultz and Jim Mellander, Intrusion Detection & Prevention, McGraw-Hill, (2004)
- <https://www.youtube.com/watch?v=98z-erQjbPw>
- [http://anrg.usc.edu/contiki/index.php/Mobility\\_of\\_Nodes\\_in\\_Cooja](http://anrg.usc.edu/contiki/index.php/Mobility_of_Nodes_in_Cooja)
- <http://vrajesh2188.blogspot.in/2016/04/mobility-in-contiki-2.html>
- [https://github.com/vaibhav90/Mobility\\_Interference\\_Plugin\\_Patch\\_Contiki2.7/tree/master/mobility](https://github.com/vaibhav90/Mobility_Interference_Plugin_Patch_Contiki2.7/tree/master/mobility)
- <https://www.hindawi.com/archive/2012/904308/>
- <https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwjG8ay68tLXAhUITI8KHffVDRYQFghDMAM&url=https%3A%2F%2Fblog.imaginea.com%2Fiot-network-testbed-simulation-with-cooja%2F&usg=AOvVaw1EjXdgdvnmk2B7KyOH3QfdF>
- <https://github.com/cetic/6lbr/tree/master/tools/cooja/apps/mobility>
- <https://github.com/contiki-os/contiki/blob/master/tools/cooja/java/org/contikios/cooja/dialogs/ExternalToolsDialog.java>
- <https://sourceforge.net/projects/contiki/files/Instant%20Contiki/>
- <https://github.com/msloth/RWMMSim/tree/master/mobility>
- <https://sourceforge.net/p/contiki/projects/code/HEAD/tree/sics.se/mobility/CONTIBUTORS.txt>



- [http://anrg.usc.edu/contiki/index.php/Mobility of Nodes in Cooja](http://anrg.usc.edu/contiki/index.php/Mobility%20of%20Nodes%20in%20Cooja)
- <https://www.semanticscholar.org/paper/VeRA---Version-Number-and-Rank-Authentication-in-R-Dvir-Holczer/205751bd5c26a06e63495dd344baa664b1605482/figure/2>
- <http://ieeexplore.ieee.org/document/6934253/references>
- <http://ieeexplore.ieee.org/document/6934253/>
- <http://www.sagepub.com/sites/default/fil>
- <http://www.sapub.org/global/showpaperpdf>
- [Nguyen Thanh Long, , Niccolo De Caro, Walter Colitti, Abdellah Touhafi, and Kris Steenhaut. "Comparative performance study of RPL in Wireless Sensor Networks", 2012 19th IEEE Symposium on Communications and Vehicular Technology in the Benelux \(SCVT\), 2012.](#)
- <https://www.scribd.com/document/35224471>
- [Pongle, Pavan, and Gurunath Chavan. "A survey: Attacks on RPL and 6LoWPAN in IoT", 2015 International Conference on Pervasive Computing \(ICPC\), 2015.](#)  
<https://en.wikipedia.org/wiki/Talk:Briti>
- <https://www.mindtools.com/pages/article/>
- [http://anrg.usc.edu/contiki/index.php/Mobility Plugin](http://anrg.usc.edu/contiki/index.php/Mobility%20Plugin)
- [Mohammad Nikravan, Ali Movaghar, Mehdi Hosseinzadeh. "A Lightweight Defense Approach to Mitigate Version Number and Rank Attacks in Low-Power and Lossy Networks", Wireless Personal Communications, 2018](#)  
<http://anrg.usc.edu/contiki/index.php/Mobility>
- [Gaddour, Olfar, and Anis KoubÃ©a. "RPL in a nutshell: A survey", Computer Networks, 2012](#)
- [Zhao, Ming, Arun Kumar, Peter Han Joo Chong, and Rongxing Lu. "A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities", Peer-to-Peer Networking and Applications, 2016.](#)
- [Landsmann, Martin, Matthias Wahlisch, and Thomas Schmidt. "Topology Authentication in RPL", 2013 IEEE Conference on Computer Communications Workshops \(INFOCOM WKSHPS\), 2013](#)  
<http://www.physics.umd.edu/courses/Phys1>
- [Rashmi Sahay, G. Geethakumari, Koushik Modugu. "Attack graph — Based vulnerability assessment of rank property in RPL-6LoWPAN in IoT", 2018 IEEE 4th World Forum on Internet of Things \(WF-IoT\), 2018](#)
- <https://github.com/contiki-os/contiki/pull/1934>
- [http://anrg.usc.edu/contiki/index.php/RPL\\_UDP](http://anrg.usc.edu/contiki/index.php/RPL_UDP)
- <https://github.com/contiki-os/contiki/tree/master/examples/ipv6/rpl-collect>
- [http://anrg.usc.edu/contiki/index.php/RPL objective function modification and simulation in cooja](http://anrg.usc.edu/contiki/index.php/RPL_objective_function_modification_and_simulation_in_cooja)