ANALYSIS ON HOP BASED APPROACH FOR NEIGHBORS AND HOLE DETECTION IN WIRELESS SENSOR NETWORKS

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

In

Computer Science and Engineering/Information Technology

By

Akash Parmar-141343

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

Declaration by Candidate

I hereby declare that the work presented in this report entitled **" Analysis on Hop Based Approach for Neighbors and Hole Detection in Wireless Sensor Networks"** 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 May 2018 under the supervision of **Dr. Shailendra Shukla** (**Assistant Professor -Senior Grade-CSE Department)** .
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Akash Parmar-141343

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
CSE
Dated:

Acknowledgement

I would like to extend my sincere and heartfelt obligation towards all the people who have helped me in this endeavor.

Without their active guidance, help, cooperation and encouragement, I would not have made headway in the project.

I am extremely thankful and pay my gratitude to Dr. Shailendra Shukla for his valuable guidance and support on completion of this project.

I extend my gratitude to Jaypee University of Information Technology for giving me this opportunity.

I also acknowledge with a deep sense of reverence, my gratitude towards my parents and members of my family, who have always supported us morally.

Thanking You,

Akash Parmar

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

LIST OF ABBREVIATIONS

- WSN – Wireless Sensor Network
- GUI – Graphical User Interface
- GPS – Global Positioning System
- DODAG - Destination Oriented Acyclic Graphs
- RPL – Routing Protocol for Low Power and Lossy Networks
- Average-avg.
- Algorithm-algo.
- Number- no.
- temparature- temp.

ABSTRACT

In Wireless Sensor Networks (WSNs), sensors can result in failure because of less battery periods and temporal events which are spatio and external (Space and Time events). Reliable and durable routing pathways can be identified by removal of communication and coverage holes. Holes are the nodes or sensors which are not capable of incoming or outgoing data transmission whereas hopes are the subsets of distance between nodes. Our main motive is to make an efficient system in which data flow is continuous and holes are minimum. To solve this issue regarding holes, we have different approaches like Topological, Geometrical and Statistical. Best suited one is the Topological approach which uses x-hop neighbors to detect holes and neighbors. This approach can be extended to 3D networks as well as more dense networks. The GUI should be able to display boundary nodes or neighbor nodes and the path with minimum holes and maximum data efficiency. We extend our research to various transmission mechanisms like Unicast, Broadcast and Multicast mechanisms and analyze the data transfer rates and patterns.

CHAPTER-1

INTRODUCTION

1.1) INTRODUCTION

Wireless Sensor Networks:

Wireless Sensor Networks(WSNs) called wireless sensor
and actuators (WSAN), are sensors which are widely distributed and check conditions related to environment , including temp., sound and pressure, etc. .They also pass the data along the network to other servers and locations. Its main application is in Internet of Things (IoT). They are also called dust networks because of their very small sizes.

WSN is built of thousands, hundreds or even millions of nodes where connectivity of each node is to more than one nodes. A sensor node can vary in size. The cost and resources too depend upon the complexity of sensor nodes individually.

Factors contributing to size and cost constraints include the constraints on resources like energy constraints, memory constraints, computational velocity constraints and communication bandwidth constraints.

Characteristics of a WSN include:

- Constraints on power consumption
- Node failures
- Mobility
- Heterogeneity and Homogeneity of nodes
- Scalability
- Design

Human factors affect WSN and they need to be checked and maintained regularly from time to time. They need to be fed with mechanisms such as communication paths which are updated with position capabilities of new sensors. In areas which are remote, sensors are forced to run without care and thus lack required maintenance by humans. Hence, preventing or repairing defects caused by battery or destruction by other factors such as animals or spatial events are not possible. Connectivity can be lost in some parts of WSN and hence holes can occur. As it is tough to remove holes, several works are being done on limiting and minimizing effects of holes. It is stated that proper performance relies on apt finding of end boundaries and holes because it maintains data traffic and prevents loss of data . This also leads to less consumption

of energy .This occurs because of less traffic in and around the holes , which stops the holes from expanding.

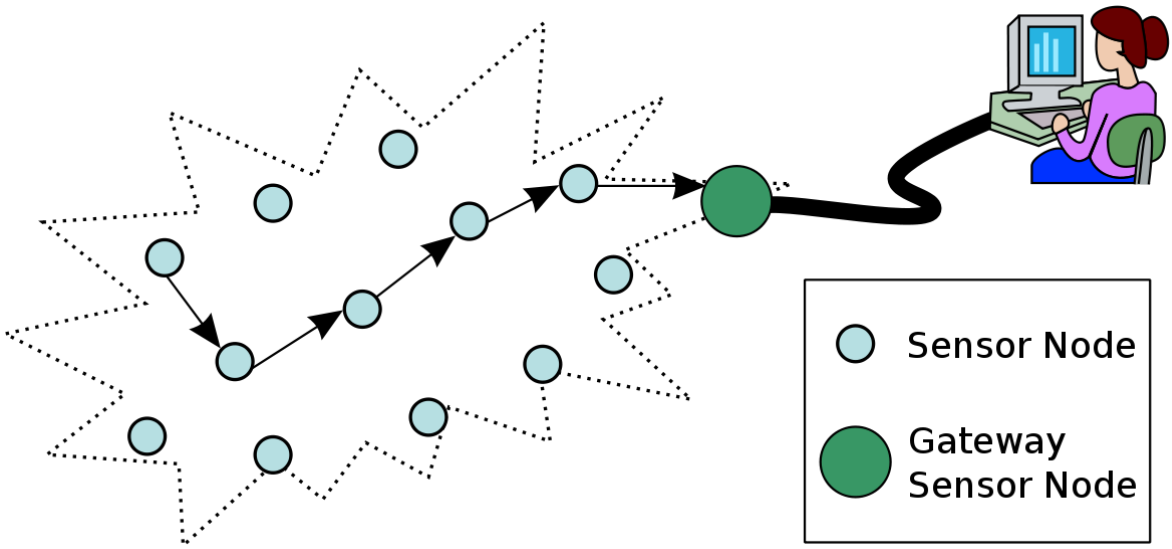

Figure 1 : Wireless Sensor Network

Source: en.wikipedia.org

Sensor Nodes:

Sensor nodes can perform processing, gather sensor information and communication with and to other nodes which are connected in the network. A mote is equivalent to a node but a node can not always be same as a mote.

Gateway Sensor Node:

A device used to connect 2 networks which are not similar is gateway. The whole pathway is connected to host network via gateway sensor node..

Holes:

In networking, black holes are regions in the network where in-coming or out-going data traffic is dropped, without any information to the source that the data had not reached its intended receiver. When the topology of network is being checked, the black holes become invisible, and only can be checked if we monitor the lost traffic; hence this name.

Black hole filtering refers to drop packets at the level of routing, usually using a routing protocol for implementing the filtering many routers at the same time, often dynamically to respond well in time to distributed DOS attacks.

Hops:

In networks, any portion or subset of the total path between source locations and destination locations is a hop. Packets of data pass along bridges, routers and also gateways while travelling between sources and destinations. Each and every time packets are sent to the next networking device, a hop happens. The hop count is the number of devices in between through which all of the data must pass between source location and the destination location.

Hop Count:

The set of devices through which the data must pass gives us the hop count of the network. It must pass through the source and destination locations instead of a single route data flow. Each router forms a hop in the way. Data transfer occurs from one network layer to another layer. Fundamentally distance measurement is done through hop count only.

A rough idea of the actual distance between two network locations can be given through hop count. Hop count of 'n' means that n number of gateways are used to separate the destination host from the source host. It is not very useful for telling the best possible as terms like speed, latency and reliability of individual hops are not taken but only the count at the end, that is, the total count. RIP is one of the protocols too use hop count.

WSN Network Topologies:

WSN Nodes are organized mostly in one of three types of topologies of network. In a topology of star, each node can be connected to gateway directly. In a tree network of tree, each node connects to a node at trees' higher level and then the gateway, data is routed from gateway and lowest tree nodes. Finally increased reliability can be achieved by using feature nodes of network than can be connected to multiple system nodes and use data through paths which are reliable. This link is called router.

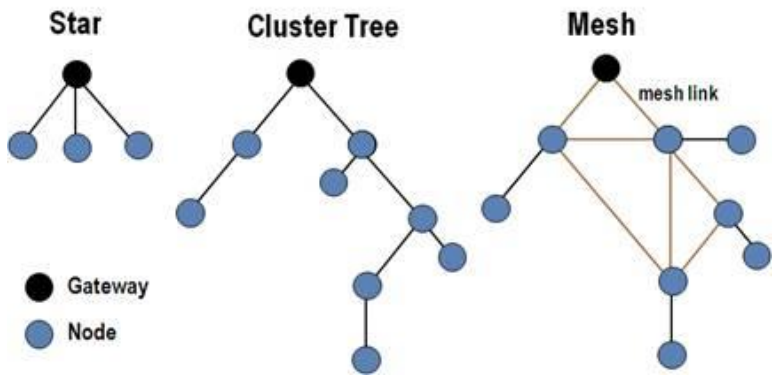

Figure 2 : Common WSN Network Topologies

Source: en.wikipedia.org

Applications of WSNs:

Area Monitoring - WSN is imposed in a region where things need to be checked. A military example can be an example of sensors' to check for the enemy movement.

Health Care Monitoring –Applications include measurement of body position, location of persons, overall treatment of patients who are ill in hospitals and at homes. Devices track the physical state of people for uninterrupted diagnosis of health, input the data from a set of depth cameras having sensor floor, or other similar devices. Area-body networks can collect info. about a person's health, fitness, and expenditure of energy. In health care applications the privacy of user's data has utmost value.

Environmental/Earth sensing – It is used to check the extent of dangerous gases around people and the environment. Instead of wired networks this can be achieved through ad hoc wireless networks .

Other applications include-

Applications related to military, Applications related to Health Applications related to Environment, Home Applications, Applications which are commercial, Area monitoring, Environmental/Earth sensors, Monitoring Of Air Pollution and
Detection of Forest Fires

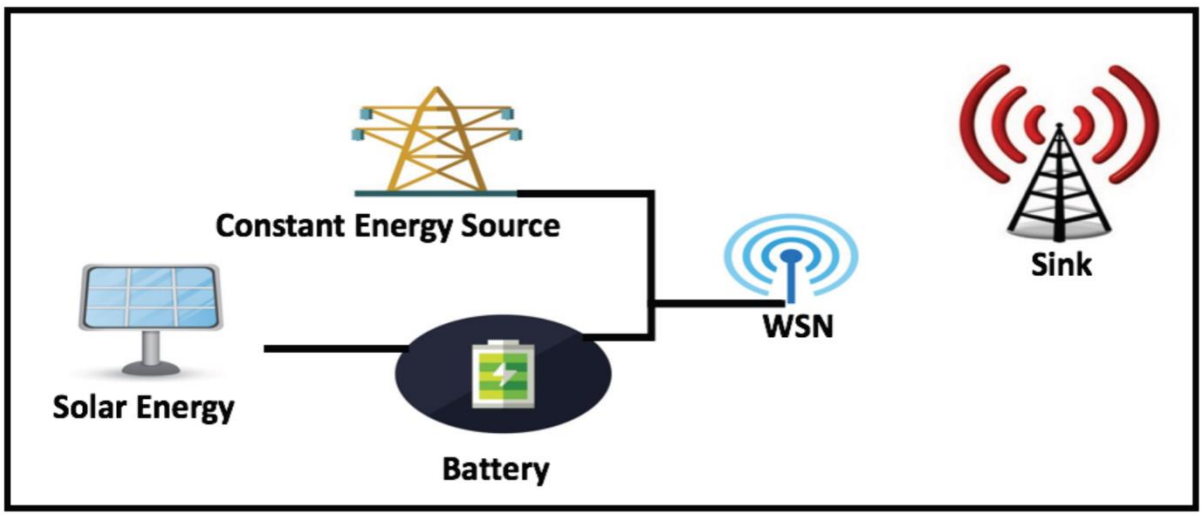

Figure -Solar energy Application of WSN:

:Source-en.wikipedia.org

1.2) PROBLEM STATEMENT

The Holes need to be identified in a network and a path should be chosen with minimum possible holes or no holes. Also, it is important to find the boundary nodes so that an optimum path can be identified and chosen. Finally, gateway node establishes the connection between the WSN and the primary network. This ensures that that the traffic flow is good and data loss is minimum. It also ensures maximum efficiency. Holes don't allow the data to pass through them(incoming or outgoing) which leads to data loss or connection interruptions. So, there is a need for paths which contain minimum possible holes and minimum loss of data. Existing algorithms find the boundary nodes but get two boundaries instead of one.

This can be achieved by following an approach which requires lower network connectivity degree and also high performance at the same time. Our approach is based upon finding the neighbors of each node by checking its connectivity in the WSN. Boundary or holes can be found using each sensor node's position and by using communication pathways.

1.3) OBJECTIVES

The objective of the project is to identify the holes in the WSN and to find the boundary nodes, that is, find the boundary of the WSN. We have to find a path with minimum holes and minimum data loss. Also, we have to reduce the complexity of the algorithm if possible and extend this for 3D networks as well if possible.

Some major objectives of this project are :

- Make complex simulated networks
- Find hop neighbors and holes
- Find boundary nodes and WSN boundary
- Find path with maximum data efficiency
- Extend algorithm for dense networks
- Extend algorithm for 3D Systems, if possible

1.4) METHODOLOGY

There are three different approaches for finding the pathways:

Topological Approach: Information on topological approach is used for sensors in detection of WSN boundaries and holes. This algorithm uses randomly choosing a node set, uses a network initiating from one of the given node set and then forms isolated connection levels via each individual node. Checking is done for each node for iso-level. If broken iso-levels are found, then we can say that points at the end belong to the boundary or holes. If cycle is formed from iso-levels which is closed then there is no finding of nodes at boundary. For reasons regarding efficiency levels, the algorithm needs that at least node degree of 16 and more. The given algorithm initially attempts to check for holes in network system and then moves towards the finding of the boundary. Apart from node checking, there must be a degree of 10 and more to get highest of the performance levels.

Statistical Approach: Statistical approaches provide mechanism for finding the inner and nodes at boundary by stating that the nodes WSN's distribution follows some Statistical functions. This algorithm relies on a thought that nodes at boundaries have very less average node degrees than nodes which lie inside the network. Number of edges which connect to the node gives the degree. The algorithm uses an approximation of avg. network density and finds an approximate value. By checking the actual node degree with the value which is taken to be threshold value, each and every node could identify and send its position.. To take care of performance and reliability, this algorithm needs networks with node degree equal to 100 or exceeding 100.

Geometrical Approach: These assume that the location of nodes has been known geography through devices of awareness of location such as Global Positioning System also called GPS. There is an algorithm divided into two stages that uses the nodes' locations to find holes and later find different routes of flow of data around the holes. In the initial stage, every node checks if it is stuck or not. This only applies to cases very near to destination than all of its own1-hop neighbor nodes. Nodes which are stuck are identified

as the nodes at the boundary. In the next stage, routes are built through distributed algorithm around the hole boundaries.

Out of these mechanisms, the best approach and the one which we would use for our project is the **Topological approach** which uses the x-hop connectivity approach where x can be values belonging to integers.

The simulations and the existing paper work analyzing showed that the two-hop approach gives best performance in finding the boundaries and also consumes less energy w.r.t 3-hopapproach. This result is valid for both uniform and random networks.

CHAPTER – 2

LITERATURE SURVEY

2.1) RESEARCH PAPERS

S. NO	AUTHOR	YEAR	TITLE	REMARKS
1	I.M KHAN, N .JABEUR, S ZEADALLY	2012	Hop Based approach for Boundary and Hole Detection in WSN	Tells us about the topological approach we are about to use and we have to improve the existing algorithm from this paper. It forms the base for our every project activity.
2	EDWIN PREM KUMAR GILBERT, BASKARAN KALIAPERUMAL, ELIJAH BLESSING RAJSINGH	2012	Research Issues in WSN Applications: A Survey	Tells us about the various applications that Wireless Network Sensors can have.

Table 1:Research Papers

2.2) WEBSITES AND BOOKS

SOURCE	REMARKS
https://www.elprocus.com/architecture-of-wireless-sensor-network-and-applications/	About Wireless Sensor Networks
https://en.wikipedia.org/wiki/Wireless_sensor_network	About WSN
Book : Forouzan	Computer Networks Fundamentals

Table 2:Websites And Books

2.3) RELATED DATA

Wireless Sensor Networks(WSNs) called wireless sensorand actuators (WSAN), are sensors which are widely distributed and check conditions related to environment and physicals, such as temp., sound, pressure, etc. and to convincingly pass the data through the network to other servers and locations. Its main application is in Internet of Things (IoT). They are also called dust networks because of their very small sizes.

WSN is built of hundreds or some thousands of nodes where each node is linked to one or more sensors. A sensor node can vary in size. The cost and resources too depend upon the complexity of sensor nodes individually.

Factors contributing to size and cost constraints include the resources' constraints like energy, memory, velocity of computation and communication bandwidth

Characteristics of a WSN include:

- Constraints of Power Consumption
- Node failures
- Mobility
- Heterogeneity and Homogeneity of nodes
- Scalability which also includes-
- Distribution at large scale
- Checking capability of environmental conditions
- Simpler using it
- Cross-layer designs

Sensor Nodes:

Sensor nodes can perform processing, gather sensor information and communication with and to other nodes which are connected in the network. A mote is a node but a node can not always be same as a mote.
A device used to connect 2 networks which are not similar is gateway. The whole pathway is connected to host network via gateway sensor node..

Wireless Sensor Networks and its Advantages

The advantages of WSN include the following

- No immovable infrastructure required
- Apt for places which cannot be reached like mountains, seas or rural areas..
- Flexible
- Inexpensive
- Less wiring required
- Accommodation for new devices at a time
- Requires centralized monitoring.

WSN Architecture

Mostly commonly followed models include the OSI model and its architecture. Total of 3 cross layers and 5 layers are included in the WSN architecture. Layers which are required include Application, network, data link, physical and transport layers. Cross planes include management of

- Mobility
- Task
- Power

In order to increase efficiency of network, these layers are worked upon.

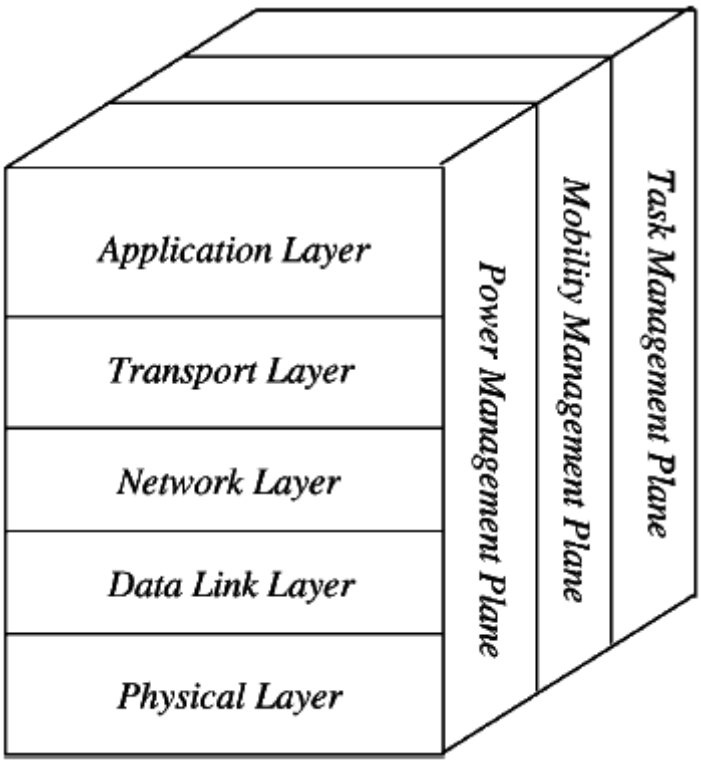

Figure 3: Architecture of a WSN

Application Layer

Management of traffic is the main role of application and can also provide software of data conversion and such applications to find information. Agricultural fields, military related fields and medical fields are their application area.

Transport Layer

It avoids congestion and improves reliability which is either practical or the upstream. Dissimilar mechanisms are used by these for recognition of loss and its recovery. The transport layer is required for connection with other networks.

Network Layer

Routing is often its main function. Application based tasks are there but the main ones include buffer management , partial memory management and conservation of power. No universal ID and self-organization is there.

Data Link Layer

Main roles include multipoint or point to point reliability checks. Its main roles include error control, MAC, checking of streams of data , detection of frames of data and its multiplexing.

Physical Layer

Physical stream is provided by transferring a bits' stream along the paths or mediums which are physical. Frequency selection, carrier frequency and its generation, signal detection, encryption of data and modulation are its key features. IEEE 802.15.4 is meant for particular areas having lower rates & WSN with cost(low), power, density, communication ranges which can also increase and improve the battery performance.

Sensor Nodes:

Sensor nodes can perform processing, gather sensor information and communication with and to other nodes which are connected in the network. A mote is always a node but a node can not always be same as a mote.

Topological Approach: Use of information on topological approach is used of sensors in detection of WSN boundaries and holes. This algorithm uses randomly choosing a node set, uses a network initiating from one of the given node set and then forms isolated connection levels via each individual node. Checking is done for each node for iso-level. If broken iso-levels are found, then we can say that points at the end belong to the boundary or holes. If cycle is formed from iso-levels which is closed then there is no finding of nodes at boundary. For reasons regarding efficiency levels, the algorithm needs that at least node degree of 16 and more. The given algorithm initially attempts to check for holes in network system and then moves towards the finding of the boundary. Apart from node checking, there must be a degree of 10 and more to get highest of the performance levels.

WSN Applications:

- Applications related to military
- Applications related to Health
- Applications related to Environment
- Home Applications
- Applications which are commercial
- Area monitoring
- Environmental/
- Earth sensors
- Monitoring Of Air Pollution
- Detection of Forest Fires

CHAPTER-3

SYSTEM DEVELOPMENT

CONTIKI: Contiki is an OS for networked, systems which have constraints on memory working on low power wireless IoT devices. Its uses include systems which are made for street lighting, monitoring of sounds in smart cities, monitoring of radiations, and alarms. This software is open source released with a BSD license.

COOJA: Cooja Simulator (OS-Contiki) is a simulator which is designed for Wireless Sensor Networks. The easiest way of running Cooja is executing it inside its very own directory.

Running Cooja Simulator:

- Create a new simulation
- Create a new mote type
- Adding motes and running the simulation
- Saving simulation file

RPL: Routing Protocol for Low Power and Lossy Networks (RPL)
Falls under category of distance vector route protocol, where the routing relies on Destination Oriented Acyclic Graphs (DODAGs) . It ensures avoidance of loop and its detection.

DODAG:A Directed graph which doesn't have cycles, centered towards a root node, e.g, a border route.
DAG Information Options messages are often broadcastedfor tree building; includes a node's rank.

RSSI - RSSI is the relative received signal strength in anenvironment of wireless sensors, in arbitrary units. RSSI is the levels of powerwhich are received by the receiver radio after the antenna and possible cable loss. Therefore, Signal is as strong as the RSSI Signal. RSSI, or "Received Signal Strength Indicator", is a measure of how strongly your device is capable of hearing a signal from a router. Its value determines whether you can have a good signal connection or not.

RSSI Vs dbM- dBm and RSSI are both different units of measurement that represent the same thing which is none other than strength of signal. The difference rises here that RSSI is a index which is relative, while dBm is an absolute number which represents levels of power in mW (milliwatts).

LQI - LQI (Link Quality Indicator)indicates received signal's quality. The LQI gives an estimate of the ease of received signal and how it can be demodulated by storing the level of the error regarding difference in ideal constellations and the received signal.

Extreme cases to illustrate how RSSI and LQI work:

- low RSSI and high LQI means a weak signal and noise is there
- low RSSI and low LQI means no noise at all
- high RSSI and high LQI means that the noise is very strong
- high RSSI and low LQI means that the signal is strong
- High RSSI and high LQI means that the signal is very strong

Steps to create a Simulation -

- Install VMWare Workstation 12 or a lower version
- Download InstantContiki and copy it to a folder
- Run VMWare Workstation and run .vmx file from InstantContiki folder
- Run the Virtual Machine named Instant Contiki 2.7
- Ubuntu loads and opens. Open Cooja from desktop
- **CREATING A SIMULATION –Click on File > New Simulation . Choose Simulation name and Radio Medium. Click on Create**
- **Click on Motes > Add Motes> Create New Mode Type > Sky Mote. Choose Firmware(neighbors.c-our source file). Click on Compile. Create the motes**
- **Choose number of Motes (say, 10) . Click on Add Motes. Also , add multihop motes. Also add unicast, broadcast and blink nodes.**
- **Click on Start in Simulation Window.**
- **See Mote Output for Data transfer and neighbor data.**

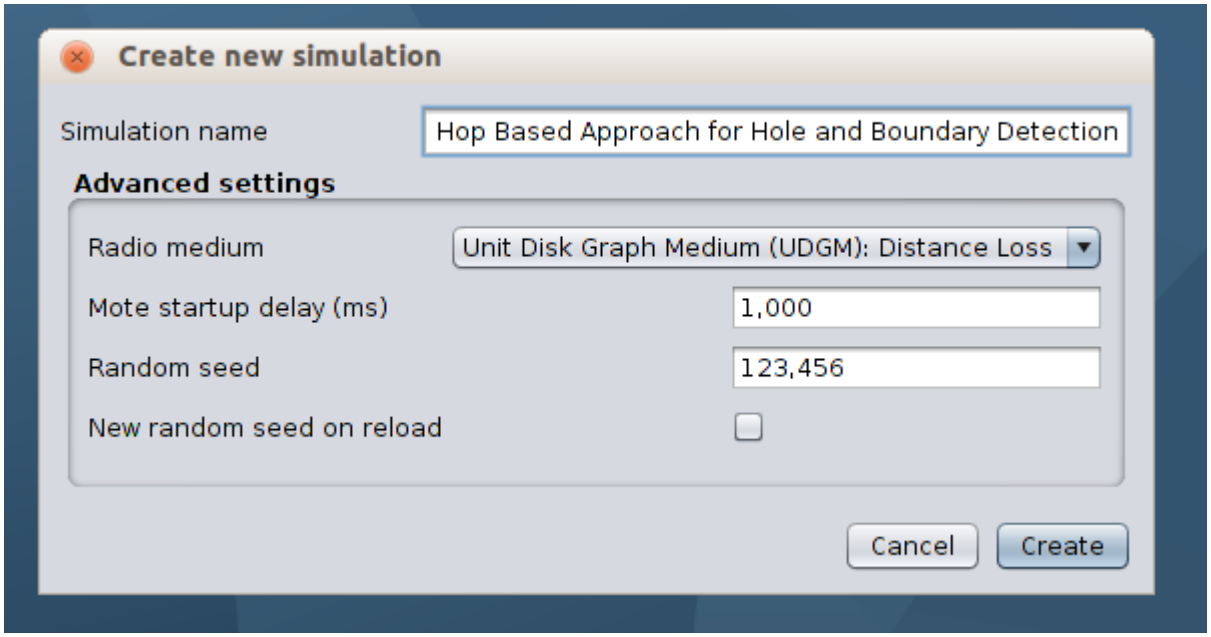

Figure 4 : Create New Simulation

CREATE A NEW SIMULATION AND CHOOSE A SIMULATION NAME AND MEDIUM IS SET TO BE UDGM TO KEEP THE DISTANCE CONSTANT. CLICK ON CREATE TO CREATE THE SIMULATION.

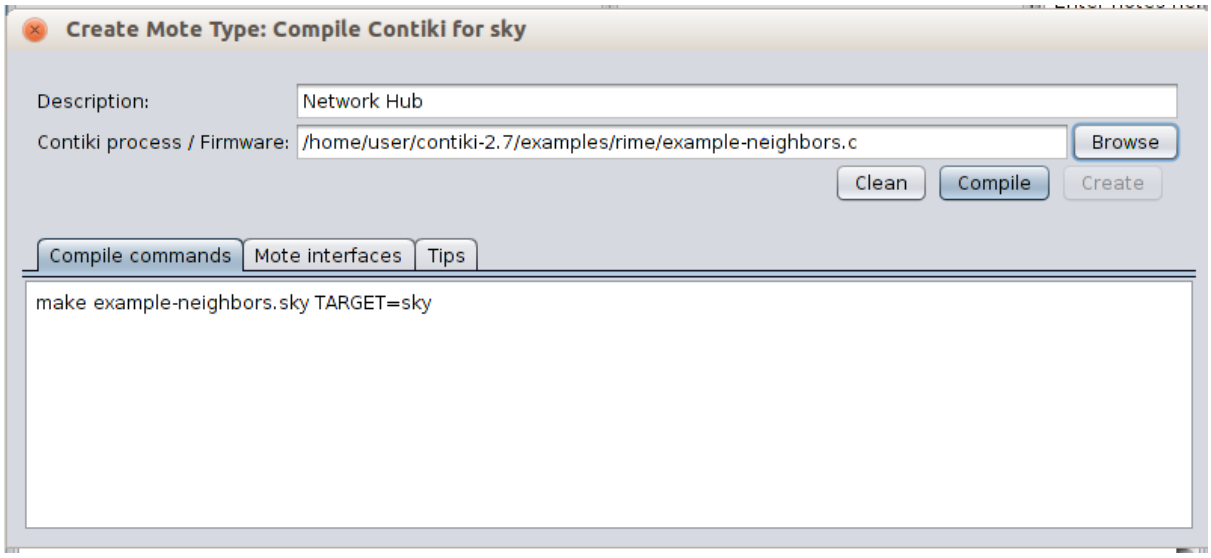

Figure 5: Creating Motes

ADD MOTES TO THE SIMULATION USING THE FILE NEIGHBORS.C AND CHOOSE THE NO. OF MOTES

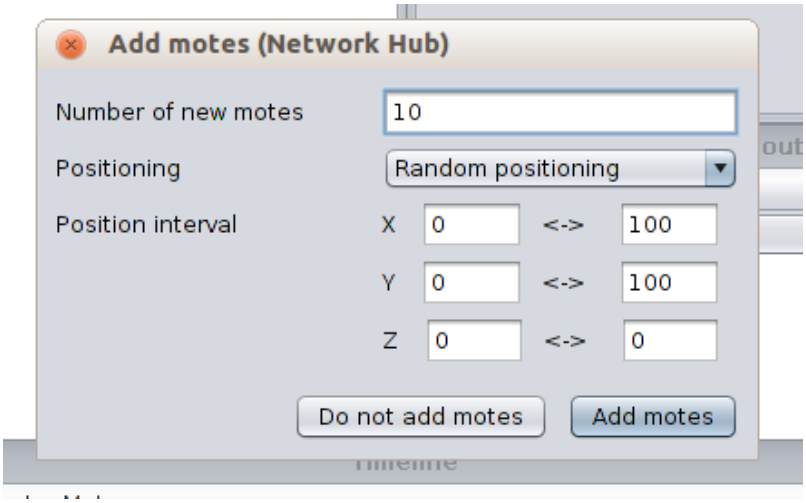

Figure 6: Adding Motes

ADD NO. OF MOTES USING THIS FILE. WE HAVE ADDED 10 MOTES HERE.

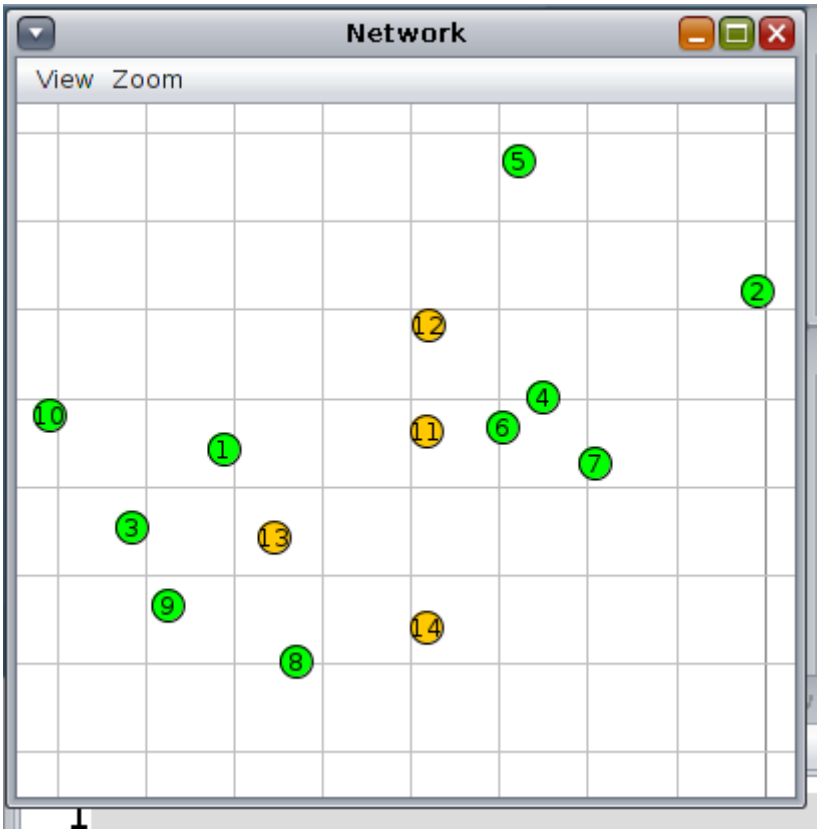

Figure 7:Created Motes

WINDOW SHOWING CREATED MOTES AND THEIR TYPES

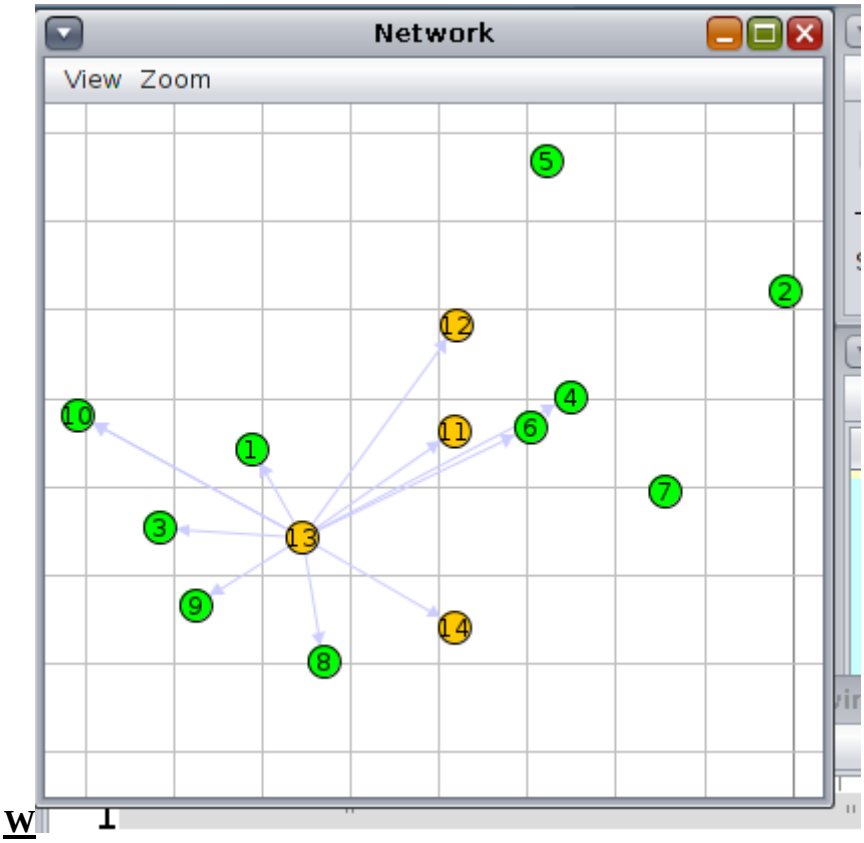

Figure 8:Network Simulation

START OF THE NETWORK SIMULATION AND DEPICTION OF NEIGHBORS

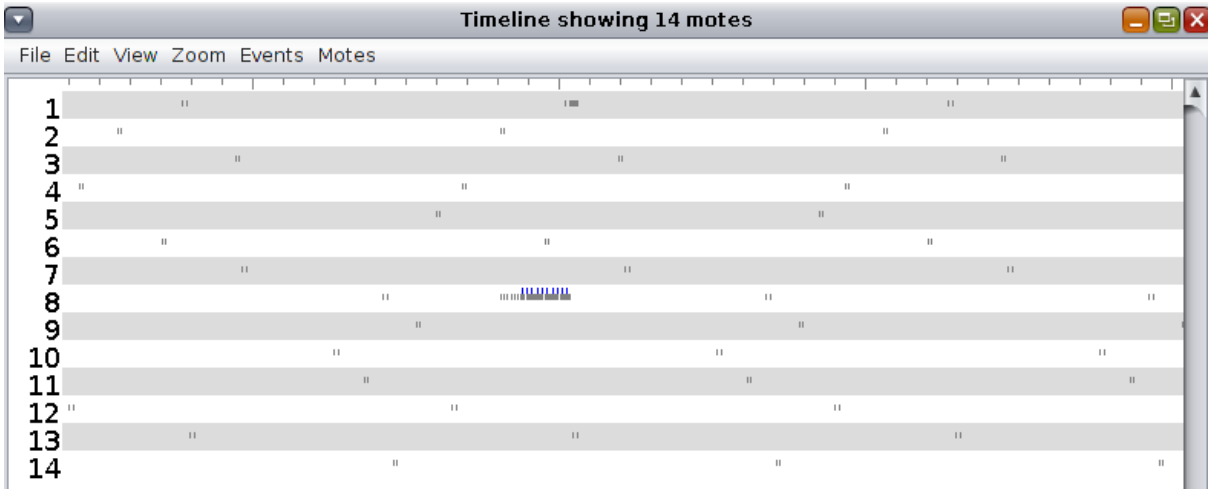

Figure 9 : Timeline

TIMELINE SHOWING TEMPORARY DATA IN 14 MOTES

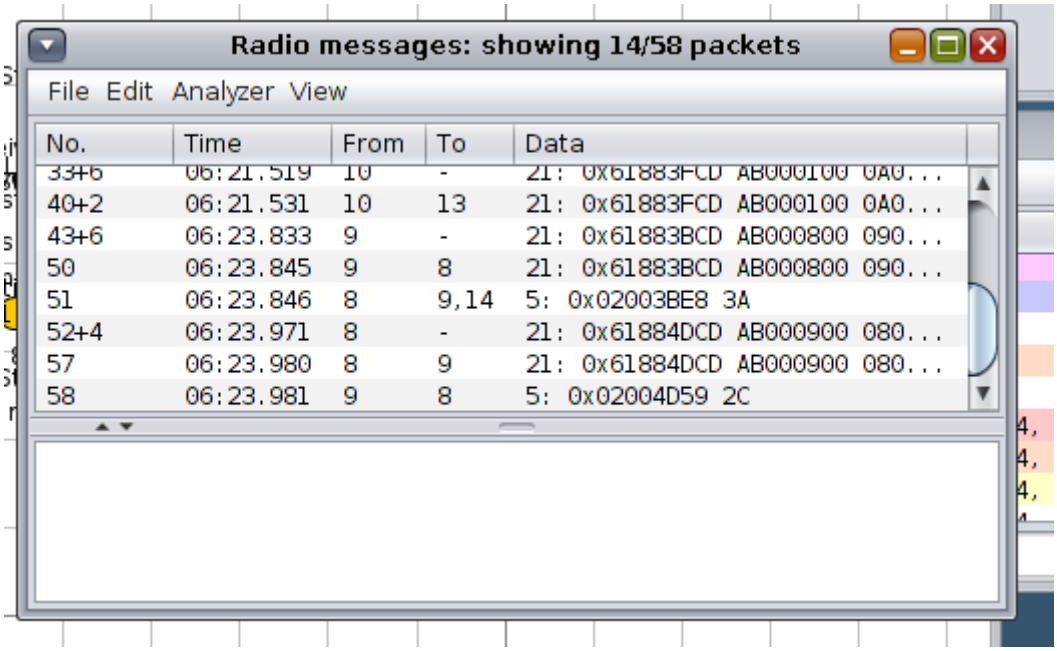

Figure 10: Packets

PACKET INFORMATION

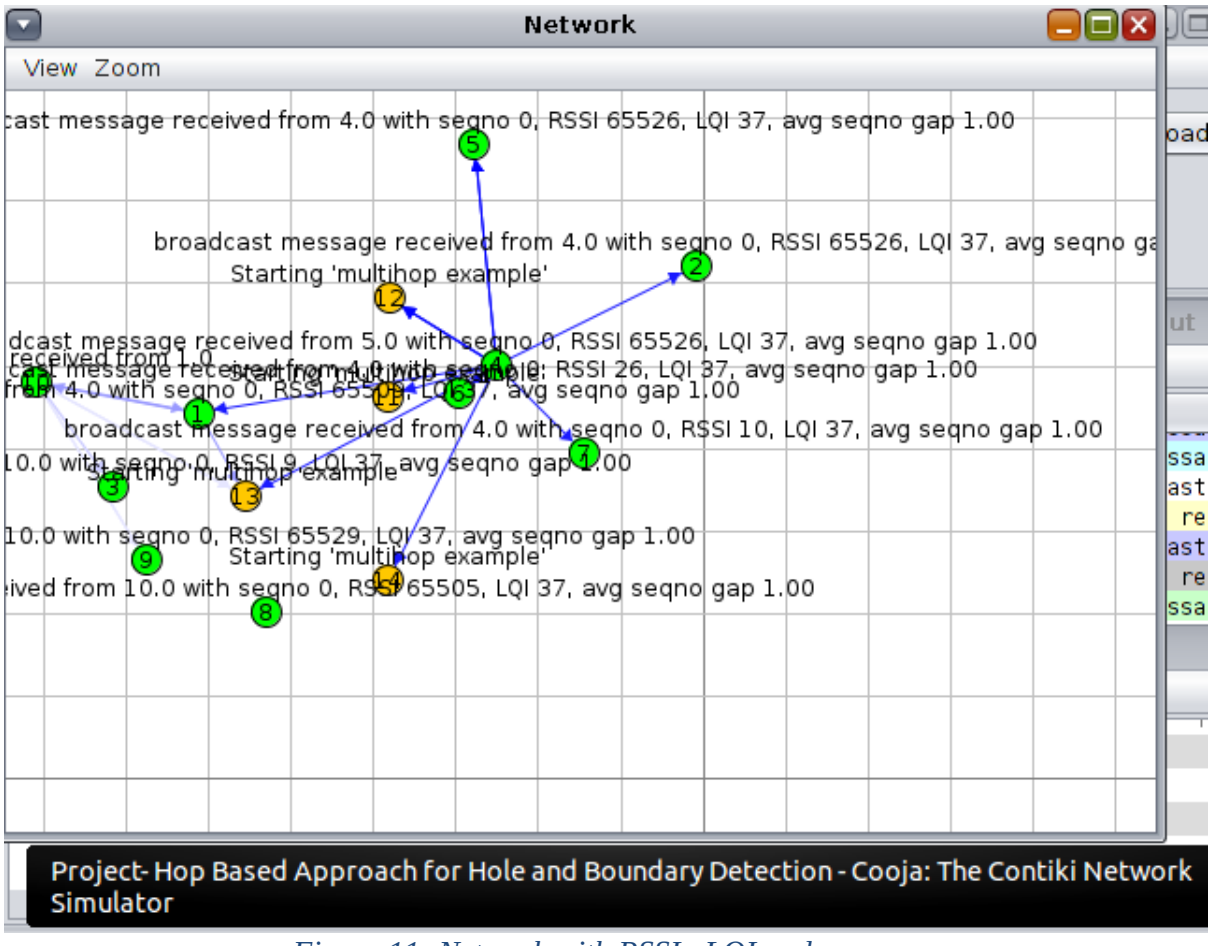

Figure 11: Network with RSSI , LQI and seq. no.

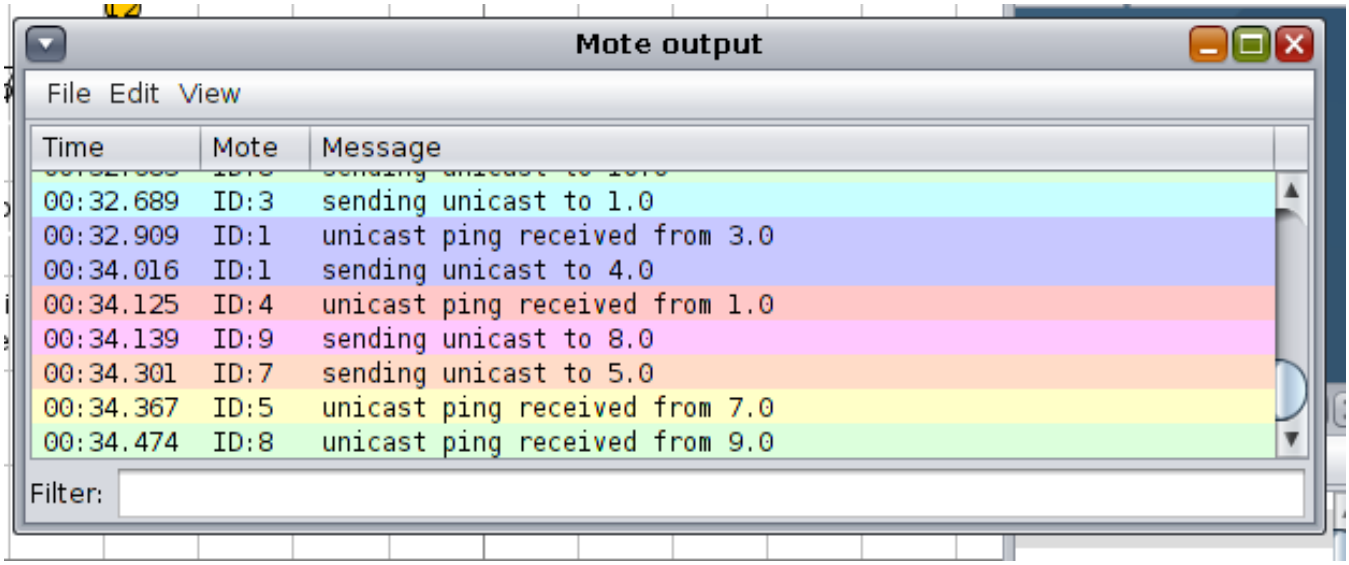

Figure 12: Mote Output

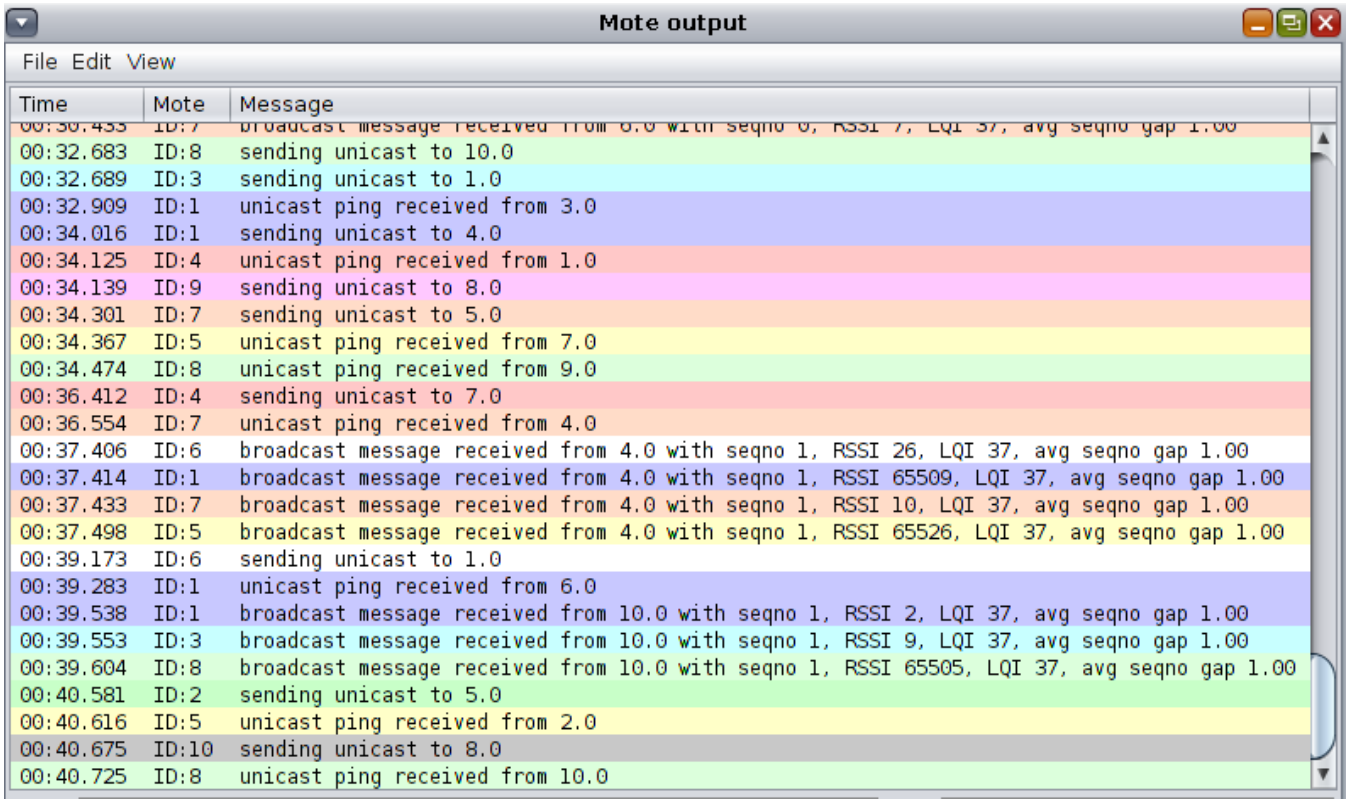

Figure 13: Maximized Mote Output

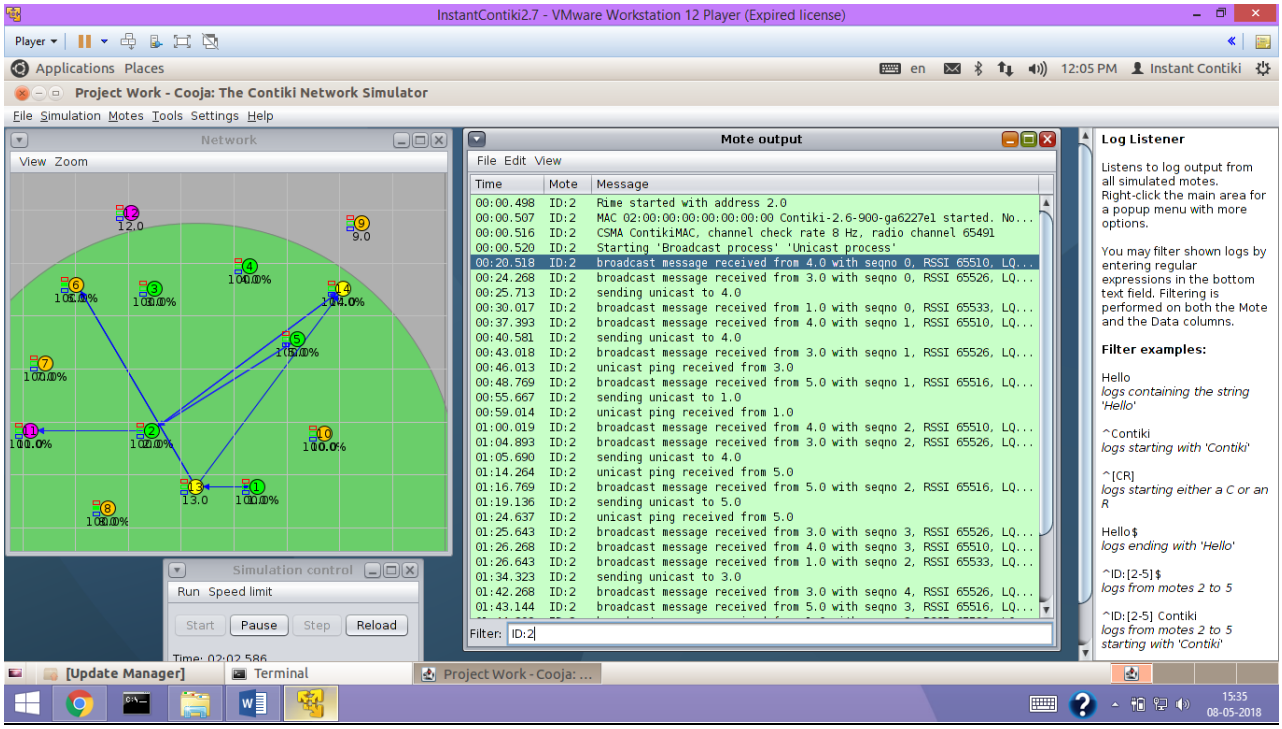

Figure 14 :ID-2 MoteOutput

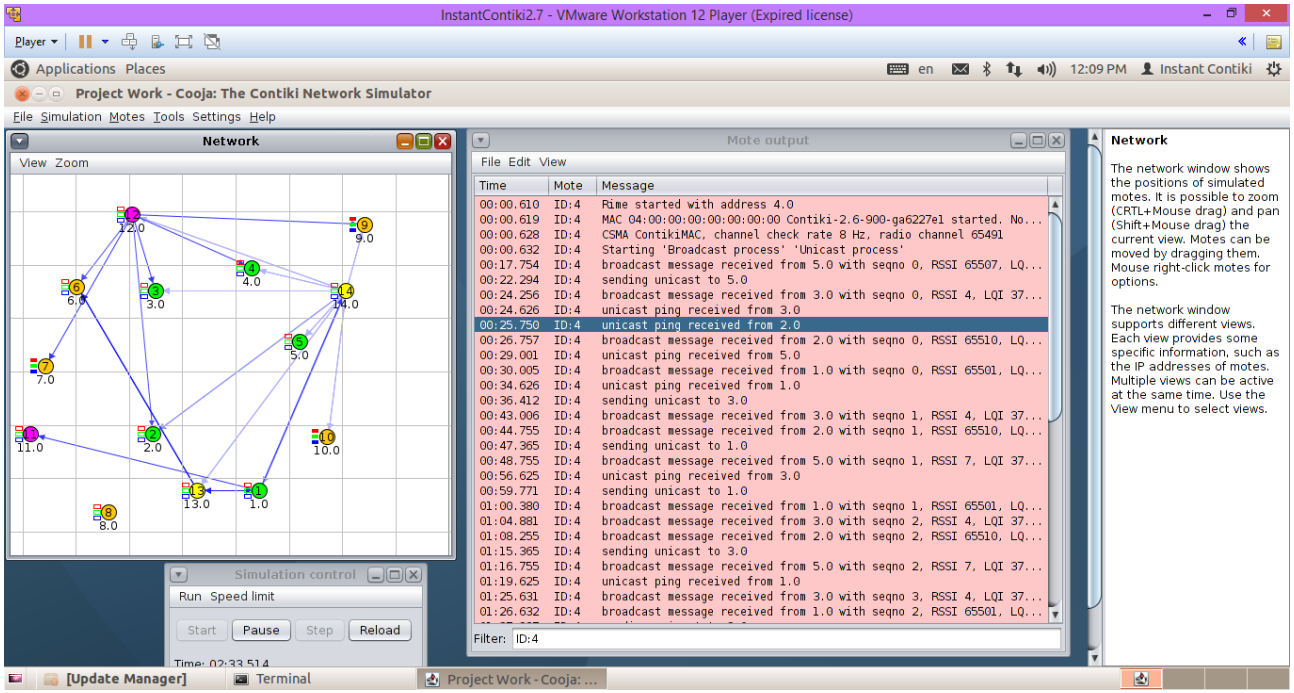

Figure 16 :ID-4 With Blink function

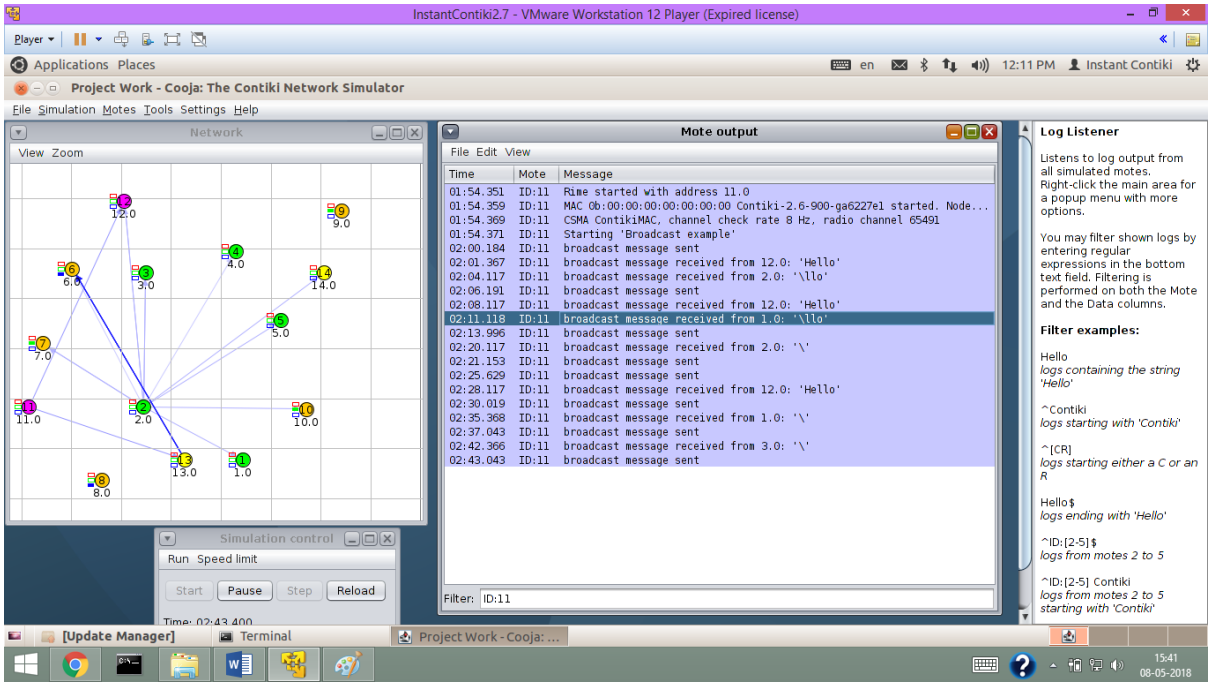

Figure 15 :ID-11 MoteOutput

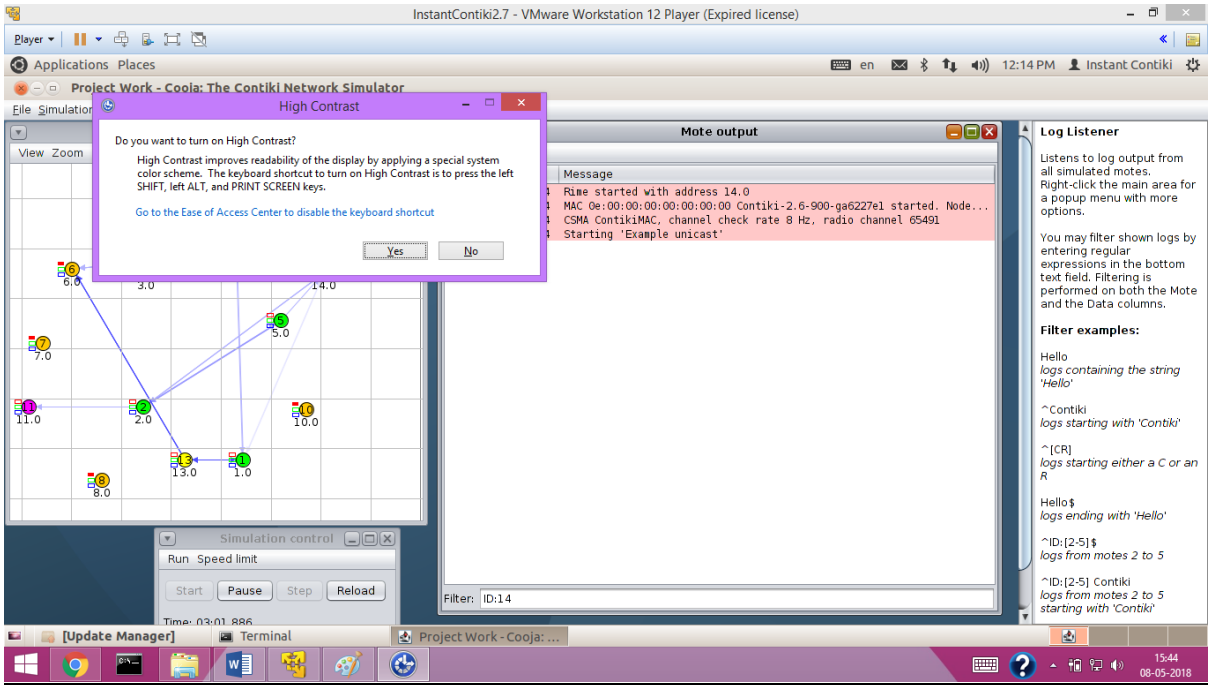

Figure 17: ID-14 MoteOutput

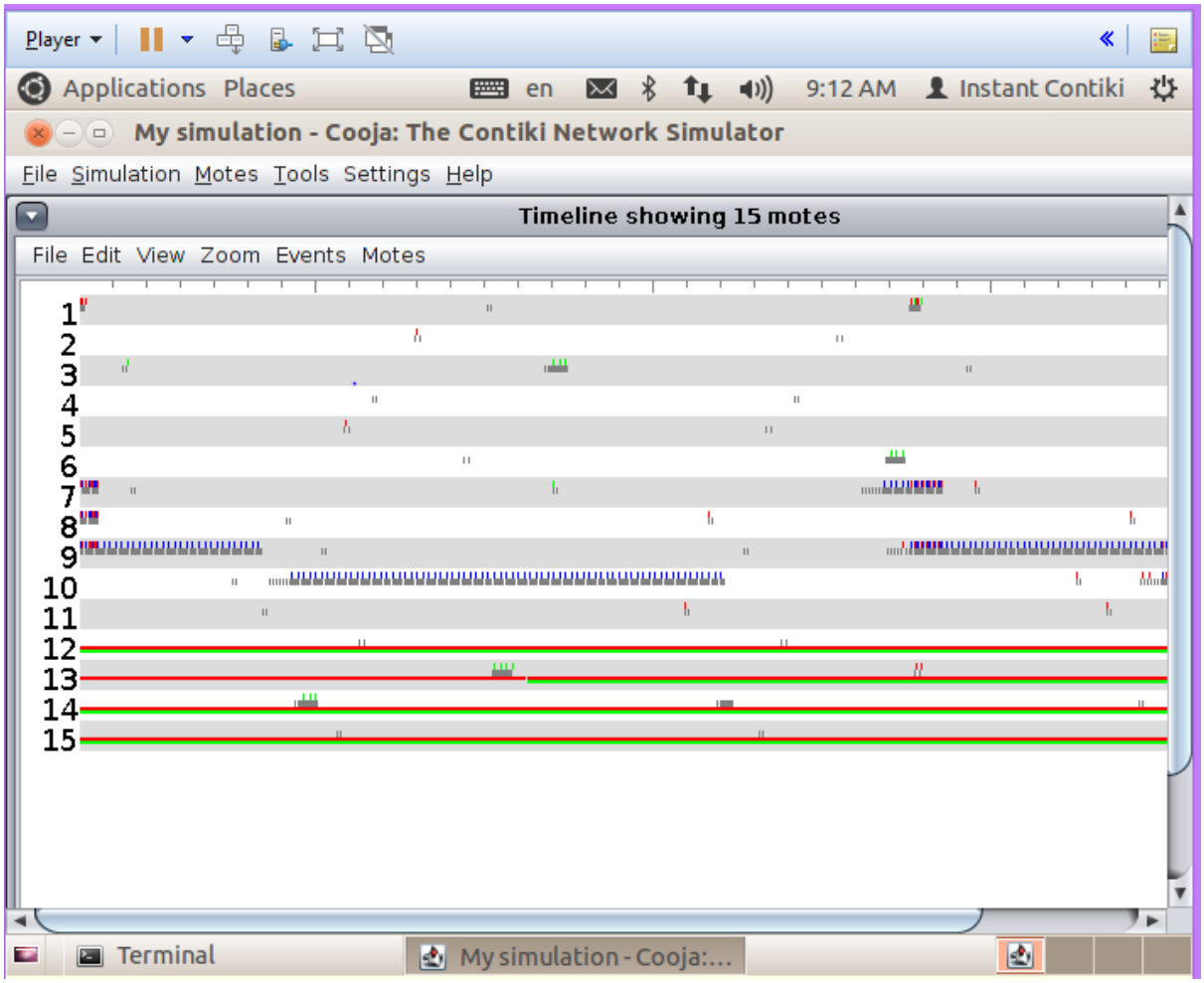

Figure 18 :Timeline

CHAPTER- 4

PERFORMANCE ANALYSIS

Implementation-
The Holes need to be identified in a network and a path should be chosen with minimum possible holes or no holes. Also, it is important to find the boundary nodes so that an optimum path can be identified and chosen.. Holes don't allow the data to pass through them(incoming or outgoing) which leads to data loss or connection interruptions. So, there is a need for paths which contain minimum possible holes and data loss is minimum. Existing algorithms find the boundary nodes but get two boundaries instead of oneAfter running the simulation, we find the unicast and broadcast details from the Mote output. Broadcast uses one sender to many receiver connection whereas Unicast is one to one connection. Multi hop nodes use many hops to reach the destination from the source . A timeline of motes is also maintained. We can also view the Mote types and Id's and the IP Addresses. In this way neighbor list is maintained through each node in network. Packets can also be viewed through Radio Messages.

Best X-Hop Approach according to already proposed Algorithms

Based on the analysis work and our own results-
2-hop approach gives much better results in correct identification of boundaries and holes than 1- hop approach and 3- hop approaches both for randomly organized and uniformly marked WSNs

++

Codes used from Libraries–

neighbors.c

```c
#include "contiki.h"
#include "lib/list.h"
#include "lib/memb.h"
#include "lib/random.h"
#include "net/rime/rime.h"

#include <stdio.h>

/* This is the structure of broadcast messages. */
struct broadcast_message {
  uint8_t seqno;
};

/* This is the structure of unicast ping messages. */
struct unicast_message {
  uint8_t type;
};

/* These are the types of unicast messages that we can send. */
enum {
  UNICAST_TYPE_PING,
  UNICAST_TYPE_PONG
};

/* This structure holds information about neighbors. */
struct neighbor {
  /* The ->next pointer is needed since we are placing these on a
     Contiki list. */
  struct neighbor *next;

  /* The ->addr field holds the Rime address of the neighbor. */
  linkaddr_t addr;

  /* The ->last_rssi and ->last_lqi fields hold the Received Signal
     Strength Indicator (RSSI) and CC2420 Link Quality Indicator (LQI)
     values that are received for the incoming broadcast packets. */
  uint16_t last_rssi, last_lqi;

  /* Each broadcast packet contains a sequence number (seqno). The
     ->last_seqno field holds the last sequenuce number we saw from
```

```
broadcast_recv(struct broadcast_conn *c, const linkaddr_t *from)
{
  struct neighbor *n;
  struct broadcast_message *m;
  uint8_t seqno_gap;

  /* The packetbuf_dataptr() returns a pointer to the first data byte
     in the received packet. */
  m = packetbuf_dataptr();

  /* Check if we already know this neighbor. */
  for(n = list_head(neighbors_list); n != NULL; n = list_item_next(n)) {

    /* We break out of the loop if the address of the neighbor matches
       the address of the neighbor from which we received this
       broadcast message. */
    if(linkaddr_cmp(&n->addr, from)) {
      break;
    }
  }

  /* If n is NULL, this neighbor was not found in our list, and we
     allocate a new struct neighbor from the neighbors_memb memory
     pool. */
  if(n == NULL) {
    n = memb_alloc(&neighbors_memb);

    /* If we could not allocate a new neighbor entry, we give up. We
       could have reused an old neighbor entry, but we do not do this
       for now. */
    if(n == NULL) {
      return;
    }

    /* Initialize the fields. */
    linkaddr_copy(&n->addr, from);
    n->last_seqno = m->seqno - 1;
    n->avg_seqno_gap = SEQNO_EWMA_UNITY;

    /* Place the neighbor on the neighbor list. */
    list_add(neighbors_list, n);
```

```
     this neighbor. */
  uint8_t last_seqno;

  /* The ->avg_gap contains the average seqno gap that we have seen
     from this neighbor. */
  uint32_t avg_seqno_gap;

};

/* This #define defines the maximum amount of neighbors we can remember. */
#define MAX_NEIGHBORS 16

/* This MEMB() definition defines a memory pool from which we allocate
   neighbor entries. */
MEMB(neighbors_memb, struct neighbor, MAX_NEIGHBORS);

/* The neighbors_list is a Contiki list that holds the neighbors we
   have seen thus far. */
LIST(neighbors_list);

/* These hold the broadcast and unicast structures, respectively. */
static struct broadcast_conn broadcast;
static struct unicast_conn unicast;

/* These two defines are used for computing the moving average for the
   broadcast sequence number gaps. */
#define SEQNO_EWMA_UNITY 0x100
#define SEQNO_EWMA_ALPHA 0x040

/*---------------------------------------------------------------------------*/
/* We first declare our two processes. */
PROCESS(broadcast_process, "Broadcast process");
PROCESS(unicast_process, "Unicast process");

/* The AUTOSTART_PROCESSES() definition specifices what processes to
   start when this module is loaded. We put both our processes
   there. */
AUTOSTART_PROCESSES(&broadcast_process, &unicast_process);
/*---------------------------------------------------------------------------*/
/* This function is called whenever a broadcast message is received. */
static void
```

```
  /* We can now fill in the fields in our neighbor entry. */
  n->last_rssi = packetbuf_attr(PACKETBUF_ATTR_RSSI);
  n->last_lqi = packetbuf_attr(PACKETBUF_ATTR_LINK_QUALITY);

  /* Compute the average sequence number gap we have seen from this neighbor. */
  seqno_gap = m->seqno - n->last_seqno;
  n->avg_seqno_gap = (((uint32_t)seqno_gap * SEQNO_EWMA_UNITY) *
                      SEQNO_EWMA_ALPHA) / SEQNO_EWMA_UNITY +
                      ((uint32_t)n->avg_seqno_gap * (SEQNO_EWMA_UNITY -
                                                     SEQNO_EWMA_ALPHA)) /
    SEQNO_EWMA_UNITY;

  /* Remember last seqno we heard. */
  n->last_seqno = m->seqno;

  /* Print out a message. */
  printf("broadcast message received from %d.%d with seqno %d, RSSI %u, LQI %u, avg seqno gap %d.%02d\n",
         from->u8[0], from->u8[1],
         m->seqno,
         packetbuf_attr(PACKETBUF_ATTR_RSSI),
         packetbuf_attr(PACKETBUF_ATTR_LINK_QUALITY),
         (int)(n->avg_seqno_gap / SEQNO_EWMA_UNITY),
         (int)(((100UL * n->avg_seqno_gap) / SEQNO_EWMA_UNITY) % 100));
}
/* This is where we define what function to be called when a broadcast
   is received. We pass a pointer to this structure in the
   broadcast_open() call below. */
static const struct broadcast_callbacks broadcast_call = {broadcast_recv};
/*---------------------------------------------------------------------------*/
/* This function is called for every incoming unicast packet. */
static void
recv_uc(struct unicast_conn *c, const linkaddr_t *from)
{
  struct unicast_message *msg;

  /* Grab the pointer to the incoming data. */
  msg = packetbuf_dataptr();

  /* We have two message types, UNICAST_TYPE_PING and
     UNICAST_TYPE_PONG. If we receive a UNICAST_TYPE_PING message, we
```

```
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(unicast_process, ev, data)
{
  PROCESS_EXITHANDLER(unicast_close(&unicast);)

  PROCESS_BEGIN();

  unicast_open(&unicast, 146, &unicast_callbacks);

  while(1) {
    static struct etimer et;
    struct unicast_message msg;
    struct neighbor *n;
    int randneighbor, i;

    etimer_set(&et, CLOCK_SECOND * 8 + random_rand() % (CLOCK_SECOND * 8));

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

    /* Pick a random neighbor from our list and send a unicast message to it. */
    if(list_length(neighbors_list) > 0) {
      randneighbor = random_rand() % list_length(neighbors_list);
      n = list_head(neighbors_list);
      for(i = 0; i < randneighbor; i++) {
        n = list_item_next(n);
      }
      printf("sending unicast to %d.%d\n", n->addr.u8[0], n->addr.u8[1]);

      msg.type = UNICAST_TYPE_PING;
      packetbuf_copyfrom(&msg, sizeof(msg));
      unicast_send(&unicast, &n->addr);
    }
  }

  PROCESS_END();
}
/*---------------------------------------------------------------------------*/
```

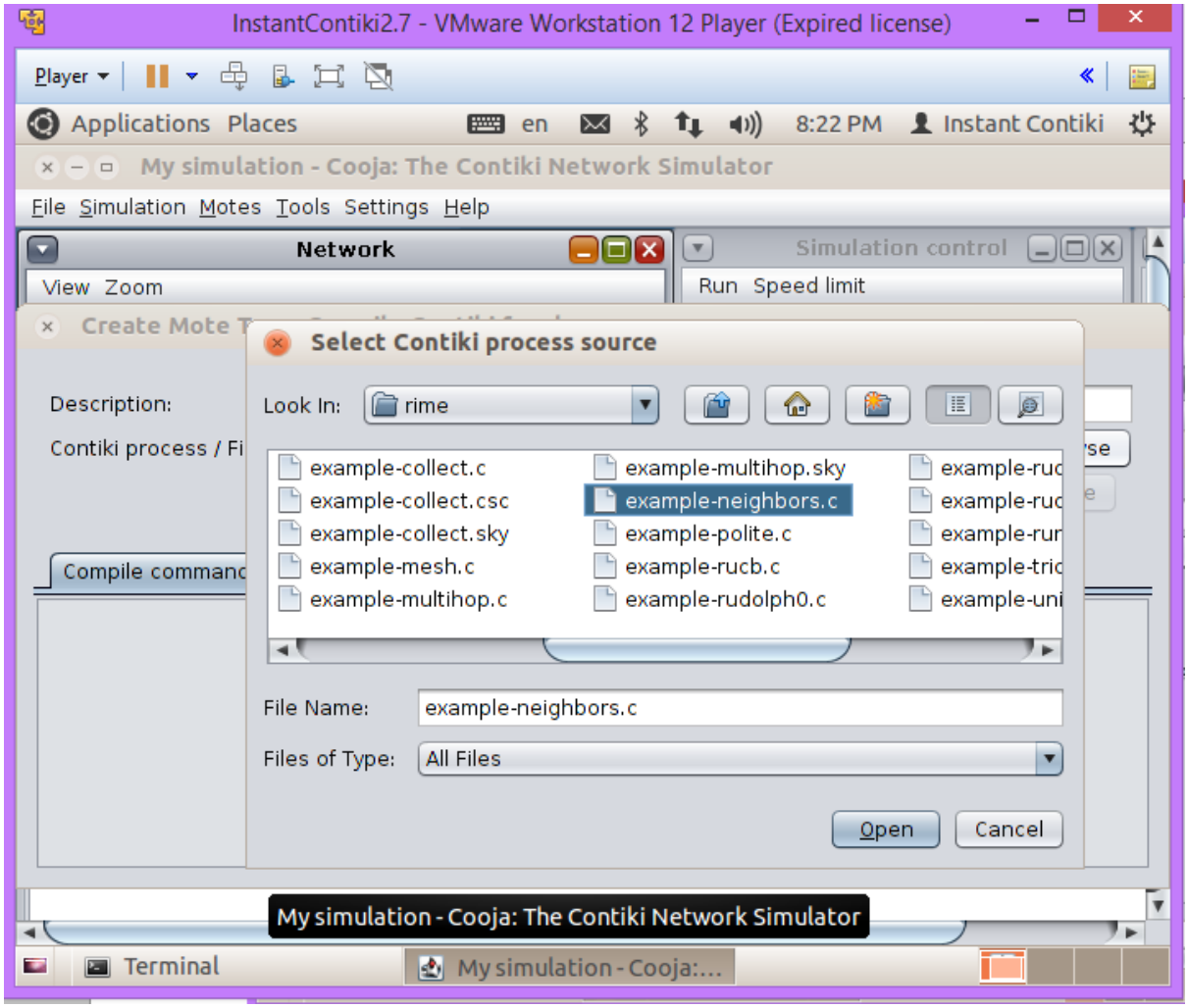

Figure 19 : neighbors.c Location

Blink-hello.c

```
#include "contiki.h"
#include "dev/leds.h"

#include <stdio.h> /* For printf() */
/*---------------------------------------------------------------------------*/
static struct etimer et_hello;
static struct etimer et_blink;
static uint16_t count;
static uint8_t blinks;
/*---------------------------------------------------------------------------*/
PROCESS(hello_world_process, "Hello world process");
PROCESS(blink_process, "LED blink process");
AUTOSTART_PROCESSES(&hello_world_process, &blink_process);
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data)
{
  PROCESS_BEGIN();

  etimer_set(&et_hello, CLOCK_SECOND * 4);
  count = 0;

  while(1) {
    PROCESS_WAIT_EVENT();

    if(ev == PROCESS_EVENT_TIMER) {
      printf("Sensor says #%u\n", count);
      count++;

      etimer_reset(&et_hello);
    }
  }

  PROCESS_END();
}
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(blink_process, ev, data)
{
  PROCESS_BEGIN();

  blinks = 0;

  while(1) {
    etimer_set(&et_blink, CLOCK_SECOND);

    PROCESS_WAIT_EVENT_UNTIL(ev == PROCESS_EVENT_TIMER);

    leds_off(LEDS_ALL);
    leds_on(blinks & LEDS_ALL);
    blinks++;
    printf("Blink... (state %0.2X)\n", leds_get());
  }

  PROCESS_END();
}
/*---------------------------------------------------------------------------*/
```

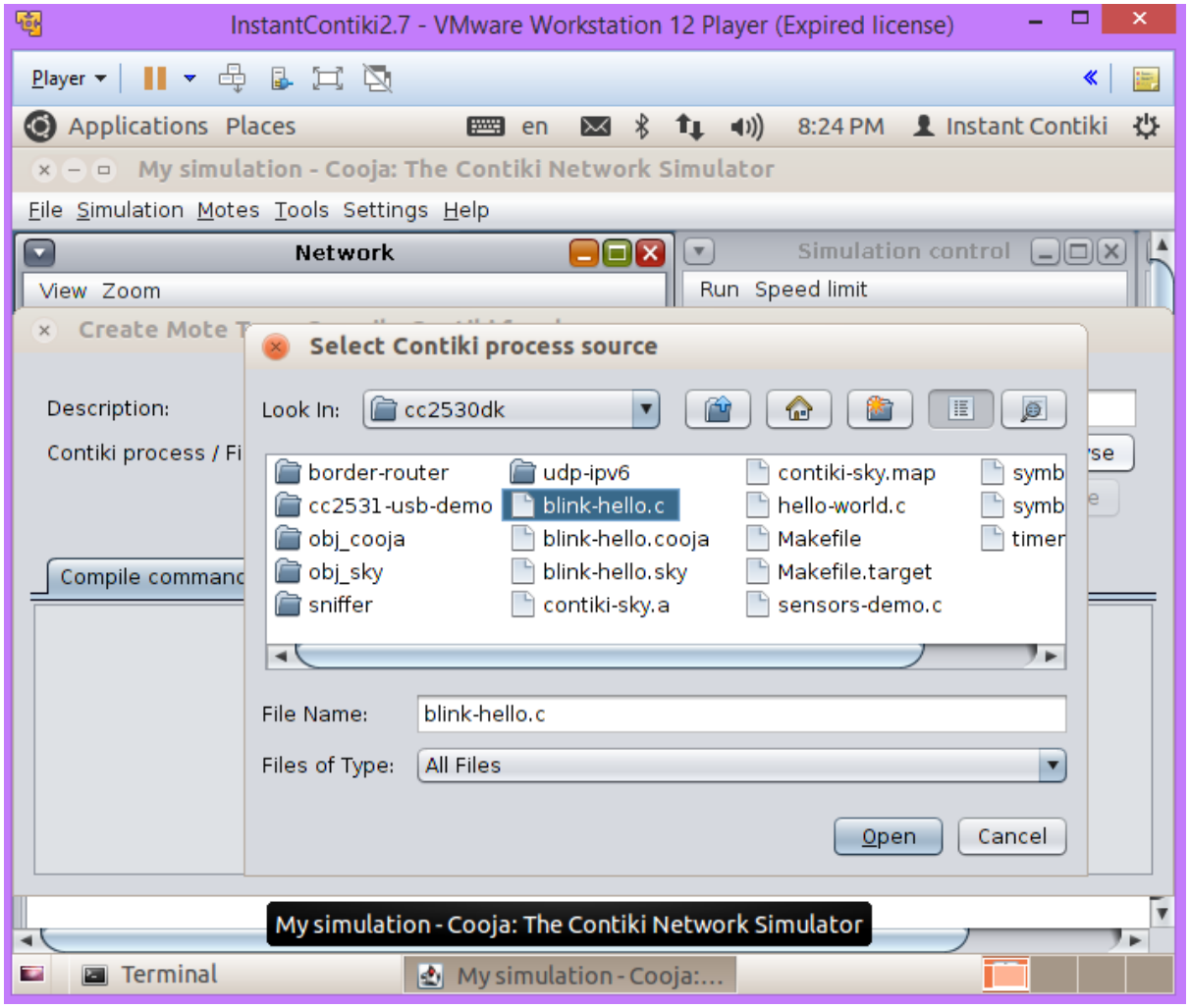

Figure 20 : blink.c Location

Broadcast.c

```c
#include "contiki.h"
#include "net/rime.h"
#include "random.h"

#include "dev/button-sensor.h"

#include "dev/leds.h"

#include <stdio.h>
/*---------------------------------------------------------------------------*/
PROCESS(example_broadcast_process, "Broadcast example");
AUTOSTART_PROCESSES(&example_broadcast_process);
/*---------------------------------------------------------------------------*/
static void
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)
{
  printf("broadcast message received from %d.%d: '%s'\n",
         from->u8[0], from->u8[1], (char *)packetbuf_dataptr());
}
static const struct broadcast_callbacks broadcast_call = {broadcast_recv};
static struct broadcast_conn broadcast;
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  static struct etimer et;

  PROCESS_EXITHANDLER(broadcast_close(&broadcast);)

  PROCESS_BEGIN();

  broadcast_open(&broadcast, 129, &broadcast_call);

  while(1) {

    /* Delay 2-4 seconds */
    etimer_set(&et, CLOCK_SECOND * 4 + random_rand() % (CLOCK_SECOND * 4));

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

    packetbuf_copyfrom("Hello", 6);
    broadcast_send(&broadcast);
    printf("broadcast message sent\n");
  }

  PROCESS_END();
}
```

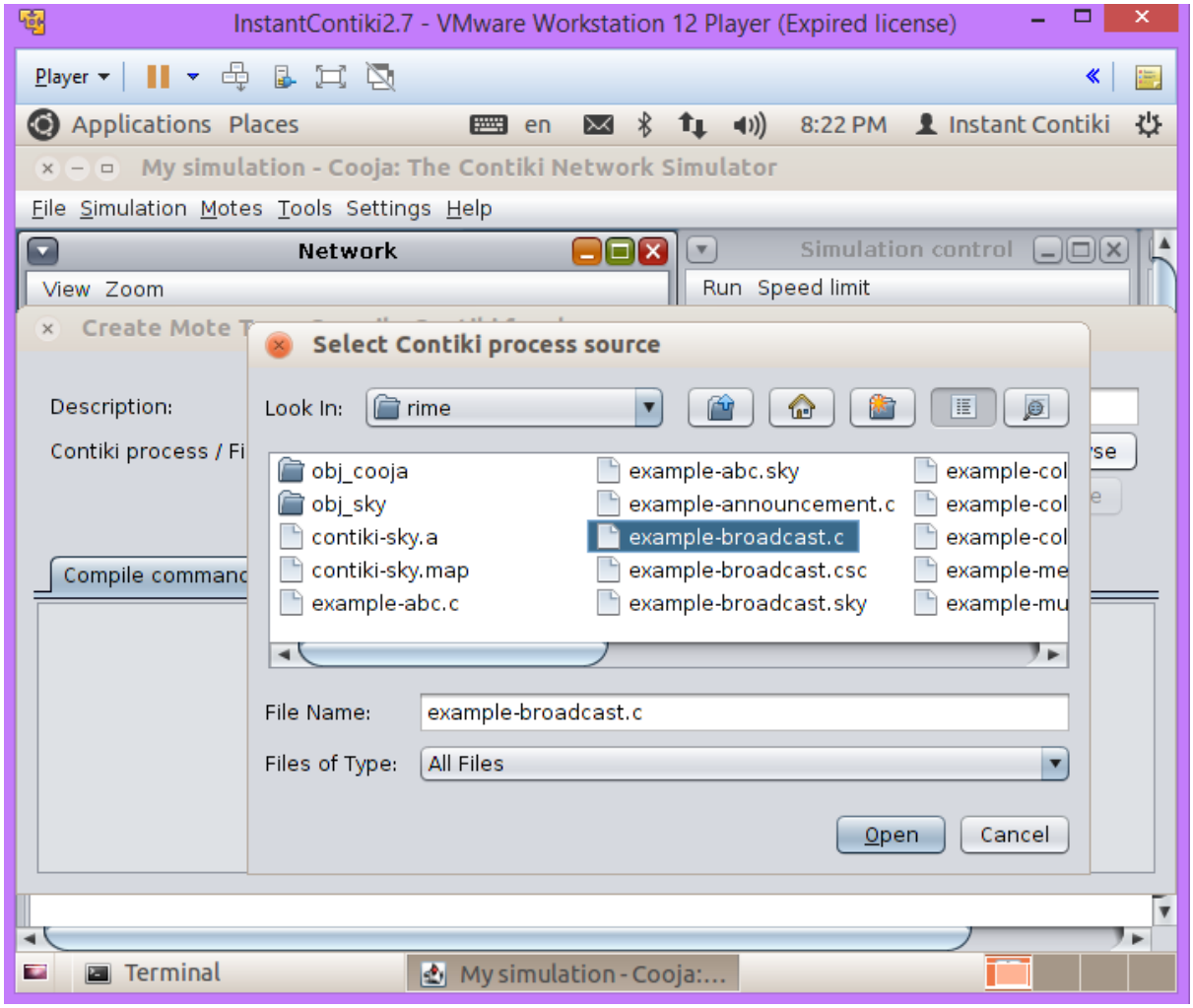

Figure 21 :broadcast.c Location

Unicast.c

```c
#include "contiki.h"
#include "net/rime.h"

#include "dev/button-sensor.h"

#include "dev/leds.h"

#include <stdio.h>

/*---------------------------------------------------------------------------*/
PROCESS(example_unicast_process, "Example unicast");
AUTOSTART_PROCESSES(&example_unicast_process);
/*---------------------------------------------------------------------------*/
static void
recv_uc(struct unicast_conn *c, const rimeaddr_t *from)
{
  printf("unicast message received from %d.%d\n",
         from->u8[0], from->u8[1]);
}
static const struct unicast_callbacks unicast_callbacks = {recv_uc};
static struct unicast_conn uc;
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(example_unicast_process, ev, data)
{
  PROCESS_EXITHANDLER(unicast_close(&uc);)

  PROCESS_BEGIN();

  unicast_open(&uc, 146, &unicast_callbacks);

  while(1) {
    static struct etimer et;
    rimeaddr_t addr;

    etimer_set(&et, CLOCK_SECOND);

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

    packetbuf_copyfrom("Hello", 5);
    addr.u8[0] = 1;
    addr.u8[1] = 0;
    if(!rimeaddr_cmp(&addr, &rimeaddr_node_addr)) {
      unicast_send(&uc, &addr);
    }

  }
```

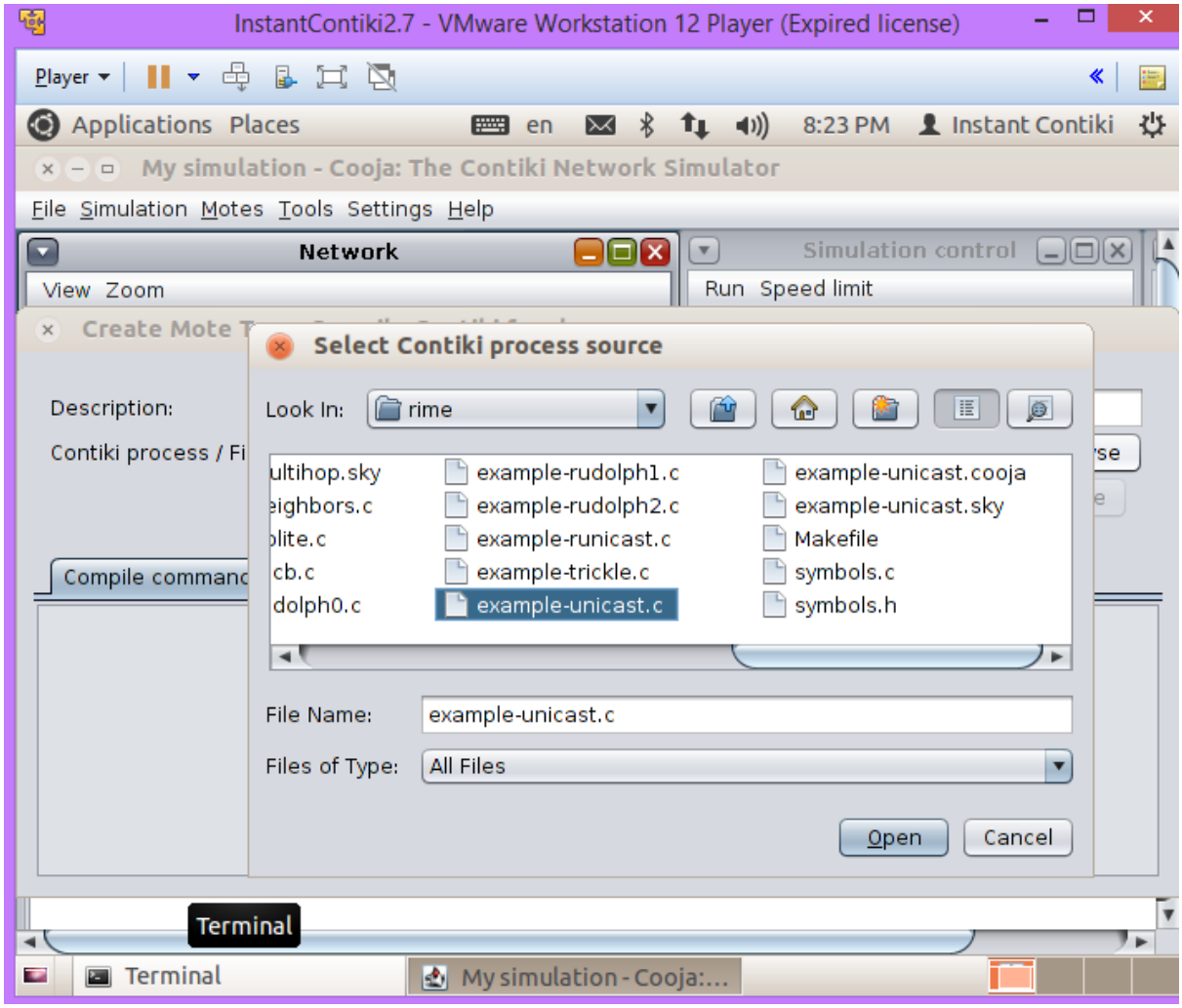

Figure 22 : unicast.c Location

- Using the Broadcast and unicast codes for making nodes and sending data across sensors
- Using the neighbors code to find neighbors and holes through analyzing.
- Then using the blink function to indicate connection status.

CHAPTER – 5

CONCLUSIONS

5.1) CONCLUSIONS

In the project , Topological approach is presented for finding the holes and neighbors.Themethod uses checking the connection of the hop-x neighbors of any of the sensor nodes which is given. 2-hop approach gives a lot better results in correct identification of boundaries and holes than 1- hop approach and 3- hop approaches both for randomly organized and uniformly marked WSNs.

In the simulation, we find the unicast and broadcast details from the Mote output.

- Broadcast uses one sender to many receiver connection
- Unicast is one to one connection.
- Multi hop nodes use many hops to reach the destination from the source .

A timeline of motes is also maintained. We can also view the Mote types and Id's and the IP Addresses. In this way neighbor list is maintained through each node in network. Packets can also be viewed through Radio Messages.

We hence state that the hop-2 approach is an aptapproach in findingthe boundaries and holes of the network sensors. We have successfully found the neighbors in the unicast and broadcast networks of the sensor nodes and we have also linked the multi hop nodes with the simples ones.

This neighbor information or neighbor table helps us identifying the holes for a particular network. Blink Function is used for some nodes to indicate the data transfer states and various message transfer mechanisms are also compared. Broadcasting results in a slow network and may lead to duplication of data but at the same time provides us with option of sending data to multiple sensors. Unicasting is fast and data is uniquely sent but can take a lot of space and memory. A combination of unicast and broadcast should be preferred depending upon the type of data transfer required by the individual nodes.

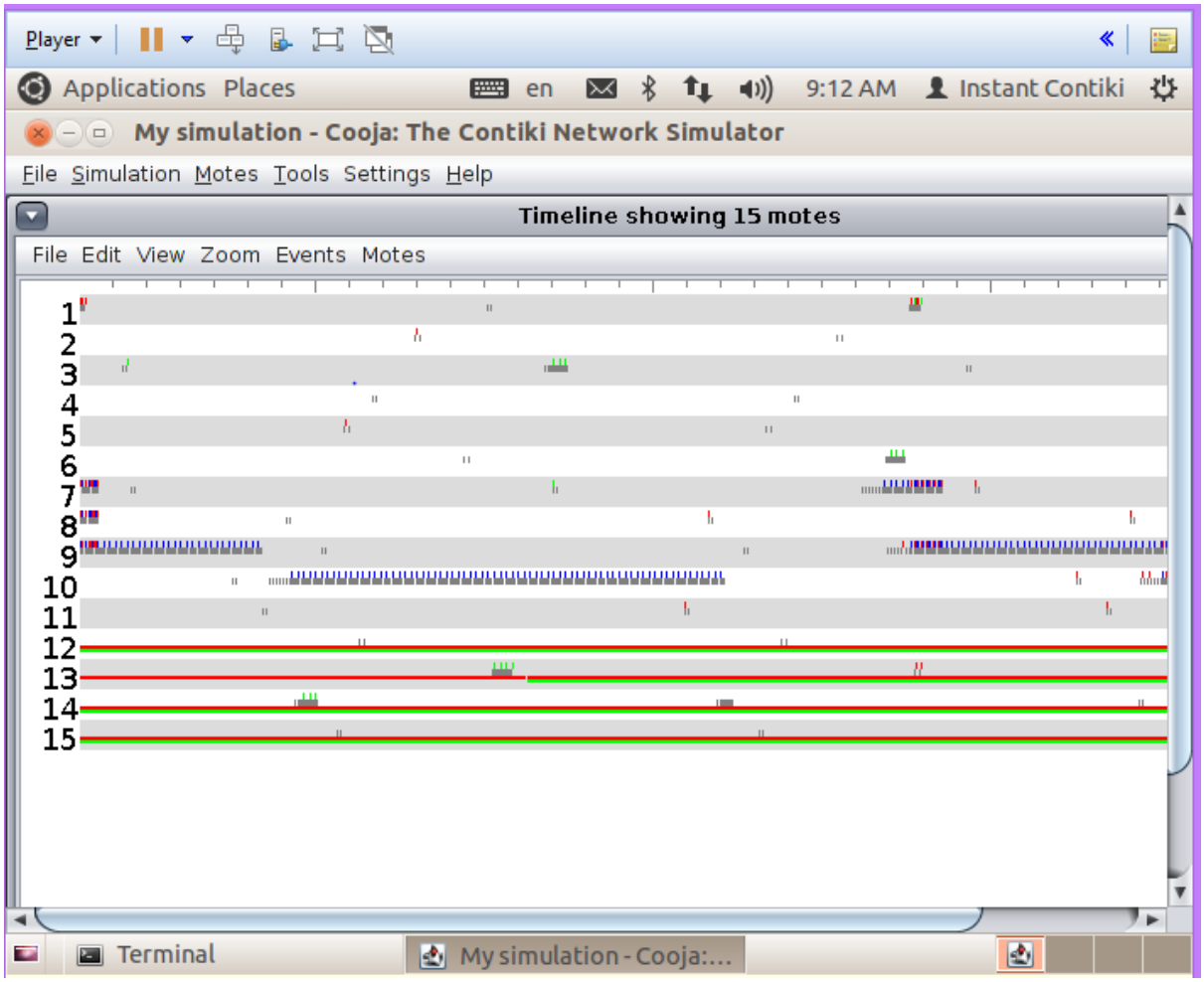

Figure 18 :Timeline

REFERENCES

- Khan, I.M., Mokhtar, H., Merabti, M.: 'An overview of holes in wirelesssensor networks'. Proc. 11th Annual Post Graduate Symp. on theConvergence of Telecommunications, Networking & Broadcasting(PGNET'10), Liverpool, United Kingdom, June 2010

- Shih, K.P., Chen, H.C., Tsai, J.K., Li, C.C.: 'PALMS: A partitionavoidance lazy movement protocol for mobile sensor networks'. IEEE Wireless Communications and Networking Conf. (WCNC'07), 2007,pp. 2484–2489

- P. Bose, P. Morin, I. Stojmenovic and J, Urrutia, Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM '99)* (1999) pp. 48–55.

- S. Bruck, J. Gao and A. Jiang, Localization and routing in sensor networks by local angle information. In *Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, (May 2005)

- D. Ganesan and D. Estrin, DIMENSIONS: Why do we need a new data handling architecture for sensor networks? In *Proceedings of the ACM Workshop on Hot Topics in Networks* (ACM Princeton, NJ, USA, Oct. 2002), pp. 143–148.

- J. Gao, L.J. Guibas, J. Hershberger, L. Zhang and A. Zhu, Geometric spanner for routing in mobile networks. In *Proc. 2nd ACM Symposium in Mobile Ad Hoc Networking and Computing*, (2001) pp. 45–50

- B. Karp and H. Kung, GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, (2000) pp. 243–254.

- F. Kuhn, R. Wattenhofer and A. Zollinger, Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, (2002) pp. 24–33.

- F. Kuhn, R. Wattenhofer and A. Zollinger, Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proc. 4th ACM Symposium on Mobile Ad Hoc Networking and Computing*, (2003) pp. 267–278.

- J. Li, J. Jannotti, D.S.J.D. Couto, D. R. Karger and R. Morris, A scalable location service for geographic ad-hoc rorting. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, (2003) pp. 120–130.