ANGLE BASED BOUNDARY DETECTION

A Project Report

Submitted in Partial Fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

by

Ayush Goel **141333**
Lokesh Chaudhary **141342**

Under the Supervision of

Dr. Shailendra Shukla
(Assistant Professor)

to

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT, SOLAN – 173 234

HIMACHAL PRADESH, INDIA

May-2018

CANDIDATE'S DECLARATION

We hereby declare that the work presented in our report entitled **" Angle based boundary detection"**is 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 our own work carried out over a period from January 2018 to May 2018 under the able supervision of **Dr.Shailendra Shukla**(Dept Of CSE).

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

Student Name	Roll No	Signature
Ayush Goel	141333	
Lokesh Chaudhary	141342	

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

Supervisor Name	Supervisor Signature
Dr. Shailendra Shukla	

ACKNOWLEDGMENT

We would like to thank all the individuals whose encouragement and support has made the completion of this project possible.

We would like to express our thanks and gratitude to our supervisor **Dr. Shailendra Shukla** Sir for his valuable guidance and support throughout this semester. Thank you for all the help, valuable time and giving us an opportunity to work with you.

We would also like to thank our college administration for providing us with the best of facilities which aided us in achieving our target.

Ayush Goel 141333

Lokesh Chaudhary 141342

INDEX

FIGURES

ABBREVIATIONS

S.NO.	Abbreviation	Full Form
1.	WSN	Wireless sensor network
2.	BNs	Boundary nodes
3.	ABBD	Angle Based Boundary Detection
4.	EBDG	Energy Balanced Data Gathering
5.	BRGT	Boundary Recognition via Graph Theory
6.	SDBR	Self detection Boundary Recognition
7.	CNs	Closure Nodes
8.	CBC	Coarse Boundary Cycles
9.	LCA	Least Common Ancestor
10.	SBNS	Sequential Boundary Node Selection
11.	DBNS	Distributed Boundary Node Selection
12.	THD	Topological Hole Detection
13.	D-LPCN	Distributed Least Polar Angle Connected Node
14.	IoT	Internet of Things
15.	TCP/IP	Transmission Control Protocol/Internet Protocol

CHAPTER-1

INTRODUCTION

1.1 GENERAL

A network of devices which communicate the information gathered through wireless links from a monitored field is knowns as Wireless sensor network(WSN) . The data through set of nodes /multiple nodes is forwarded, and with the help of a gateway, the data connection to other networks like wireless Ethernet is done.

Wireless sensor network is a wireless network that comprises of severalbase stations and no. of nodes (or wireless sensors).To monitor conditions like sound, pressure, temperature and pass through data from a network to a center location ,these type of connections are used.

There have been many recent advances in the field of IOT (Internet of Things), specially with availability of cheaper hardware, lower power computing, and wireless technologies.

Internet Of Things based sensor networks are a increasingly developing field, and it is being used for survelliance and observation.

Sensor networks should have the capablity of detecting activity that occurs around the observation area, especially at its perimeter.

Boundary and nearby nodes play an important role. Firstly these nodes interact with external environment. Second, the boundary nodes helps us understand the WSN structure which might be useful for routing, guiding as well as management purposes. Third, because of connection between boudries of a WSN and its physical condition these nodes are essential to track WSN shape which shows attributes of an environment. Likewise, the inner boundaries denotes boundaries of inner holes of a WSN structure and indicate general health of WSN structure.

Therefore, boundary nodes plays a huge role and are of great applications.

1.2 BOUNDARY NODES

Nodes observing target territory are called boundary nodes. They are required to stay watchful for event identification like items entering and leaving the zone under observation. An arrangement of boundary nodes is indicated by BNs, because of dynamic interest in observation they experiences snappier faster energy loss , which results in a shorter lifetime.
Since sensor nodes are irregularly conveyed in a system, nodes failure, or a natural deterrent may happen because of which openings can be framed in the system, making sets of separated nodes and leaving revealed zones. Similarly, they can be the reason of failure of several routing protocols. When distinguishing either the nodes on the limits of gaps or on the system's limit; revealed regions will be recognized and could be repaired by an incremental expansion of new sensors, the previously mentioned location likewise enables the steering conventions to distinguish and pass these openings.

Applications

- Traffic surveillance system

  Detecting the vehicles in traffic and having a detailed behaviour study.

- Image analysis & its understanding

  Discover the information about shape and reflectance in an image.

- Medical imaging

  To take out designs of different body organ & its tissue types.

- Detection of bright band in radar data

  Location and extent of bright band and estimate rainfall.

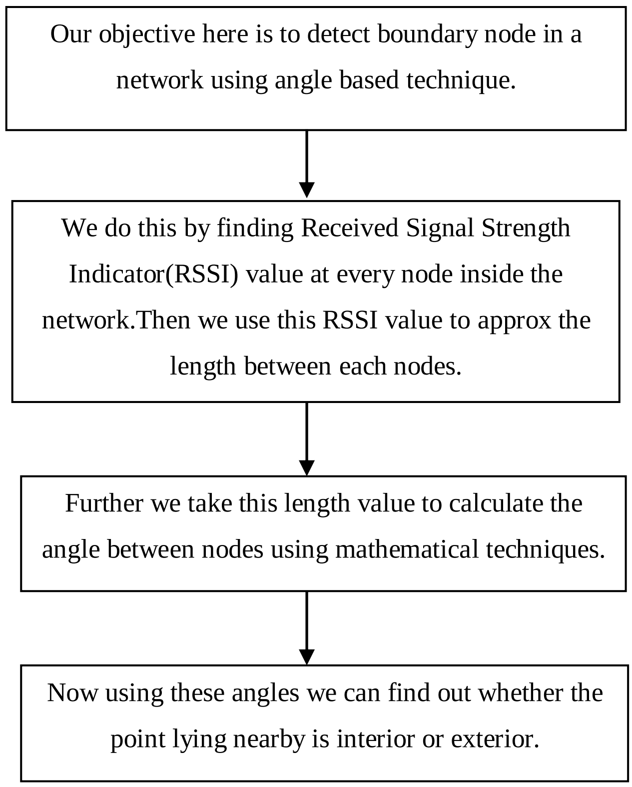
1.3 PROBLEM STATEMENT

- In our approach we calculate the distance between two nodes by scaling the RSSI value.This method may not always provide exactly accurate results.
  However it provides a very good approximation in best of times.

- Our proposed algorithm is not suitable for 3-D networks.

- Our aim is to propose an economical & efficient algorithm.
  The algorithm bypasses the burden of GPS installation on sensor nodes and the Cooja simulator in use promotes the efficiency of this algorithm.

1.4 METHODOLOGY



1.5 ALGORITHM TYPES

There are a variety of methods to detect the boundary nodes in a network.

Boundary Detection Algorithms can be divided into 3 categories:

Geometric method

Geometric methods uses geographical location information to find the boundary nodes.

The geometric technique can discover more exact boundary nodes when contrasted with different strategies, however the need of node geometric information confines its application.

Statistical method

Statistical methods often make assumptions about the probability distribution ofthe node deployment.

But there is an unrealistic expectation on node distribution and its density.

Combined stochastics, topology, and geometry,as well as estimations based on voronoi graph are used in statistical methods.

Topological method

The utilization of topological properties, for example, the connectivity data to find the boundry nodes.

Topological strategy has an expansive packet control overhead than the past strategies which is expected to collect network data from neighboring nodes; in any case, it doesn't require geographic data and has better precision rate in discovering boundry nodes than statistical technique.The main idea behind these two algo is to check whether each sensor node is covered by the polygon formed/created by its neighbor nodes.

AreaBased

If the area of the neighbour triangle equates to the sum of the main nodes of the triangle,then the node is inside,and vice versa.

Angle Based

If the sumof angle formed with the pointis less than 360, then it is non boundary, other wise it is not.

LocalizedVoronoi Diagram

This work shows a hypothetical investigation strategy for boundary node discovery on the basis of localized Voronoi polygons.

DistributedBoundaryNodeSelection(DBNS)

The node algorithm begins from the sink node, and distinguishes the boundary nodes after an iterative procedure.

Algorithm	Message Complexity	Accuracy	Dimension (2d/3d)	Year	Type	Centralized/ Distributed
ABBD	O(n/k + nb/k) = number of rounds $O(kd^3)$ = number of messages per round	≅ 88%	2d	2014	Topological	Distributed
LVP	O(n d)	≅ 92%	2d	2006	Topological	Distributed
PDiscovery	O(n d h)	≅ 95%	2d	2009	Geometric	Distributed
DBNS	O(n-k) = number of rounds O(k d) = number of messages per round	≅ 100%	2d	2013	Topological	Distributed
D-LPCN	O(d nb + n)	≅ 100%	2d	2016	Geometric	Distributed

1.5 PRELIMINARY

Our primary approach is to use angle based technique to detect boundary node in a network.The angle based boundary detection(ABBD) algorithm is of topological type.In topological network the devices used for communication are referred as nodes and the connection between these devices is called links between the nodes.The network type used here is distributed in nature.A distributed network is a variety of computer network which is spread over multiple networks. It gives a single data communication network, which we can be jointly or separately managed by each network.

Basic definitions:

- Interior nodes: A node is interior only if it is surrounded by boundary nodes.We characterize inside node criteria as being encased in a triangle shaped by three nodes.
- Boundary node: A node can be said as boundary node in the event that it's anything but an interior node.
- Network boundary: A line that join all the boundary nodes.

The algorithm proceeds as follows:

- It is expected that sensor nodes are equally spread in a geographical area with the end goal that it form a connected graph.We accept that nodes are ignorant of their local data.
- The thought is to test the coverage information at every individual node. A specific node will be called as inside node in the event that it is surrounded by atleast three nodes.Our algo experiences location ignorance i.e no node knows its info.
- That's why we determined an inside node criteria on the basis distance.

Working of angle based boundary detection:

Consider a subgraph of four nodes {A,B,C,P}.All the nodes are exposed to each other without loss of generality.Assume node P as reference hub and we need to check whether node is an inside node or outside node.

Using law of cosines we can deduce the angles stated below

∠CAB=Cos inverse$(B^2+C^2-A^2)/(2BC)$

∠ABC=Cos inverse$(A^2+C^2-B^2)/(2AC)$

∠ACB=Cos inverse$(A^2+B^2-C^2)/(2AB)$

∠PAB=Cos inverse$(X^2+C^2-Y^2)/(2XY)$

∠PBC=Cos inverse$(Y^2+A^2-Z^2)/(2AY)$

∠PAC=Cos inverse$(X^2+B^2-Z^2)/(2BX)$

∠PCA=Cos inverse$(Z^2+B^2-X^2)/(2BZ)$

∠PBA=Cos inverse$(Y^2+C^2-X^2)/(2CY)$

Condition 1: A node is an interior node if it satisfies any of below equations:

∠PAB <∠CAB

∠PCA <∠ACB

∠PBC <∠ABC

∠PCB <∠ACB

∠PBA <∠ABC

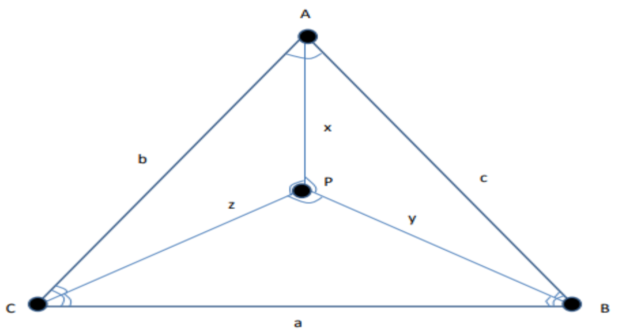


Fig.1.Int_Node

Condition 2: A node is an boundary node if it satisfies any of below equations:

∠PAB >∠CAB

∠PAC >∠CAB

∠PCA >∠ACB

∠PBA >∠ABC

∠PBC >∠ABC

∠PCB >∠ACB

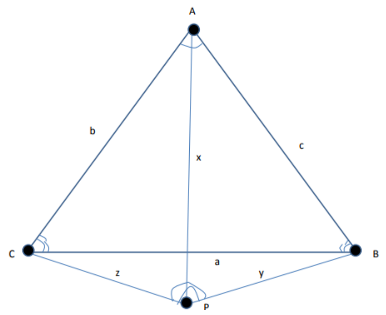


Fig.2.Bound_n

CHAPTER 2

LITERATURE SURVEY

1.Recognization of boundary nodes in WSN based on local connectivity information

The author has proposed an algorithm which is distributed & supports connectivity based boundary detection for Wireless Sensor Networks. This algo identifies boundaries by checking whether a closed route is present in the 2- hop of every node & encloses this node. Then an erosion operation is performed to limit the suspected boundary nodes. The precise boundaries corresponding inner/outer boundaries are obtained & provide valuable knowledge abour the boundaries of Wireless Sensor Networks. For this proposed algorithm to work well in lower density wireless sensor network, the original graph of a WSN is replaced by its power graph in the initial step of the algo. Also, the algorihm is expanded from the UDG model to the QUDG model by initiating the dilation operation.A complexity computation analysis has observed this algorithm as energy efficient. Lastly, a thorough evaluation is carried out by checking the accuracy of this algo and its applicability in dense & sparse deployments of wireless sensor network.

Some restrictions are also notable in applying the algo. For every boundary, some "thickness" of nodes are found as suspect boundary nodes. If two boundary nodes are close to one another, their relevant suspected boundary nodes are often connected, with the problem that this proposed algo cant separate them to correctly find them.Any two boundaries should be present at very minimum five hops away from one another. The proposed algo is not suitable for recognizing smaller perimeter holes.

2. Detection of Coverage Boundary Nodes in WSNs

This papers objective is to give a distributed arrangment which allows individual sensor nodes to recognize themselves as being present on and around the coverage boundary, which is required in a no. of functionalities at the network as well as application levels.A deterministic method is devoloped for detection of boundary node on basis of localised Voronoi polygons,

This origination of this technique is from computational geometry. This method is advantageous in different ways: it is of deterministic type and can be put to any arbitrarily deployed sensor structure, it is localised, & requires 1-hop neighbours information,and it guarantees the energy efficiency and scalability of the detection algo and requires only limited sample of easy local computations.Some mathematical & experimental proof is provided to determine the correctness & efficiency of this procedure.

3. Detection of location free boundary in mobile WSN through a distributed approach

In wireless sensor networks detection of location free boundary is an important concern. In network services locating & Detecting boundaries have a great importance. For computations , such as coverage verification and routing protocol. Designs, which accept approaches based on topology for understandin obstacles/network boundaries, they dont consider the mobile sensing nodes environmen. Whenever there is a change in network topology ,an approach that is topology based will have to reform all its boundary nodes. The study develops a boundary detection algo which is distributed in finding the boundary nodes of obstacles as well as networks. The 3 hop info of each node is required. Added informations like node locations are not needed. Nodes with DBD can examine through a distributed manner whether they are a boundary node . The Distributed boundary detection further recognises the networks outer boundary.The performance study shows that Distributed boundary detection can correctly detect boundaries in both mobile & static surroundings. The study also show applicability of Distributed boundary detection is in a real sensor world.

4. Discovery of perimeter in WSN

The study is done with the eyes on the problem of perimeter detection using WSN, there are a wide range of benefits of perimeter detection. A decentralised localised algo is presented in which sensor nodes check whether they are present around the perimeter of a WSN. The proposed algo has used the info of neighbourhoods location in concurrence with the

Barrycentric technique to find whether the sensor node are circled by neighbouring nodes, and simultaneously, if it is present inside the interior of the WSN. The performance metrics is defined to understand the performance of this method and the simulation defines the algorithm which increases the accuracy of outputs.

A decentralized localised algo is put forward for finding perimeter sensing nodes in a Wireless sensor network.The algo uses neighbourhood info as well as the Barycentric technique to see whether the sensor node is surrounded by neighboring nodes, and if it is inside the WSN interior. The density of deployment & communication radius are chosen in the sensor nodes as the parameters and performance of this algo is calculated uses different performance standards. The results are an indication of the large dependance of performance on the parameters R and rho. With the increase in density, there is a large increase in accuracy. The algorithm shows characteristics which are highly in line to wireless sensor networks requirements. Also, the vastly distributed & localized nature of this algo minimizes the amount of locally handled data and the amount of data that is relayed among different sensing nodes. This is critical in Wireless sensor networks due to its acquired energy limitations.

5. Boundary node selection and target tracking algorithms of WSN for various internet services

WSN is a importat part of IoT, in which to keep track of eye catching targets under observation sensor nodes can be used. The study of target tracking is becoming an important area of research.In target tracking sensors has a large number of applications like campus security,battle field observing,habitat and surveillance. In an ad hoc multihop fashion info can be forwarded through internet to understand some specific area and it can create an ubiquitous network for various internet services. Here,SBNS & DBNS algo to locate the boundary nodes of the WSN are proposed .In addition, a protocol to keep track of target is proposed here to check the entry/exit of the targets using the boundaries. The results of simulation reflects that the choice of boundaries in our protocol is very close to the optimal choice & the time of boundary nodes selection will not increase at a fast rate, with increase in the volume of already deployed node.

Different algorithms for boundary node selection for Wireless Sensor Networks are proposed here to know the boundaries of a monitored region. A protocol to track target is put forward to examine the entry/exit of the targets through the monitored region. The simulation analysis show that the Distributed Boundary Node Selection algorithm has very mirroring performance with the Sequential Boundary Node Selection for selecting the boundaries.The communication overhead of Distributed Boundary Node Selection is smaller than the Sequential Boundary Node Selection because the information exchange is b/w lesser neighbor nodes.Moreover, as the network sizes are increasing rapidly, the selection procedure of boundary node in Distributed Boundary Node Selection is swifter than the Sequential Boundary Node Selection.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Simulation:

The WSN algorithmwill betested andsimulated in the CoojaSimulator onContiki OS.

Instant Contiki

Instant Contiki: An environment for development of Contiki. It is an Ubuntu-Linux VM which is run in VMWare player.It has Contiki and all the required development tools, necessary compilers, and needed simulators which are required in Contiki development pre-installed.

About Cooja

Cooja is the network simulator for Contiki. It allows to simulate large and smaller networks of Contiki motes. Motes are imitated at a slower speed at the hardware level however permits exact examination of the system behaviour, or at a less point by point level, which is quicker in correlation and permits simulation of some huge networks.

How to get started with Contiki?

1. Download Instant Contiki from web.
2. Now Install VMWare Player.
3. Then start Cooja and Open a terminal window

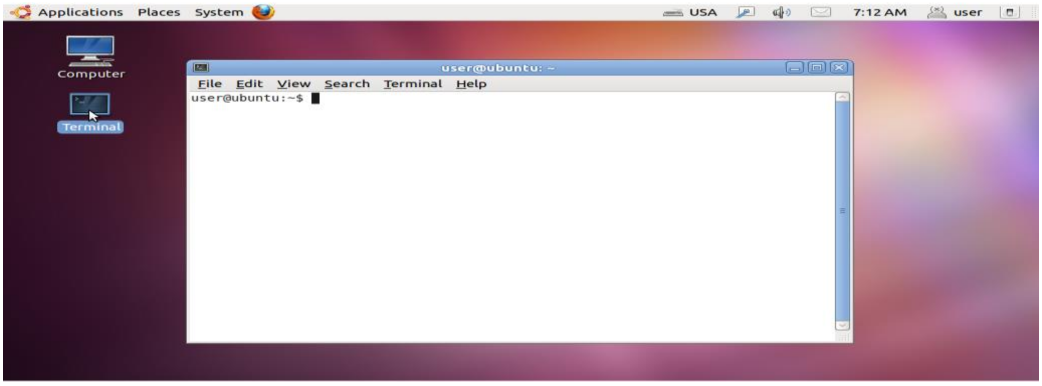


Fig.3 Term_Window

4.Create a new simulation

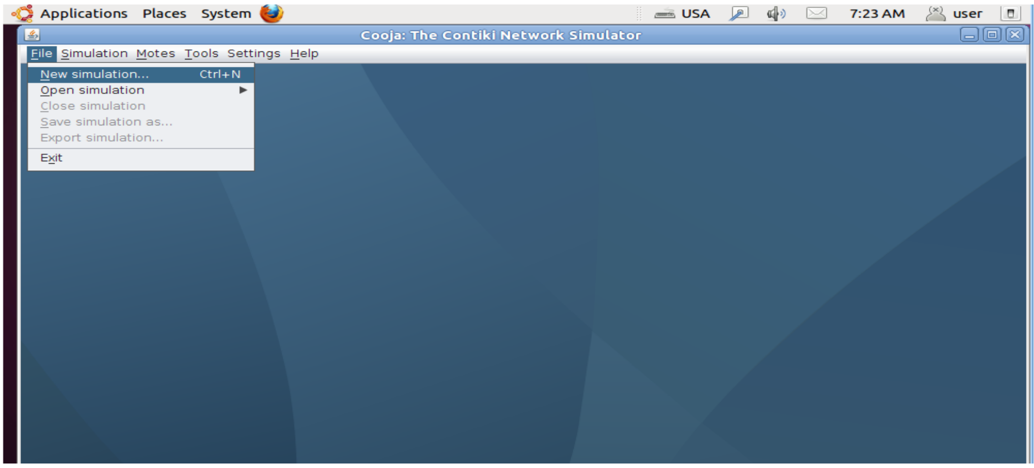


Fig.4 Sim_New

5. Set simulation options

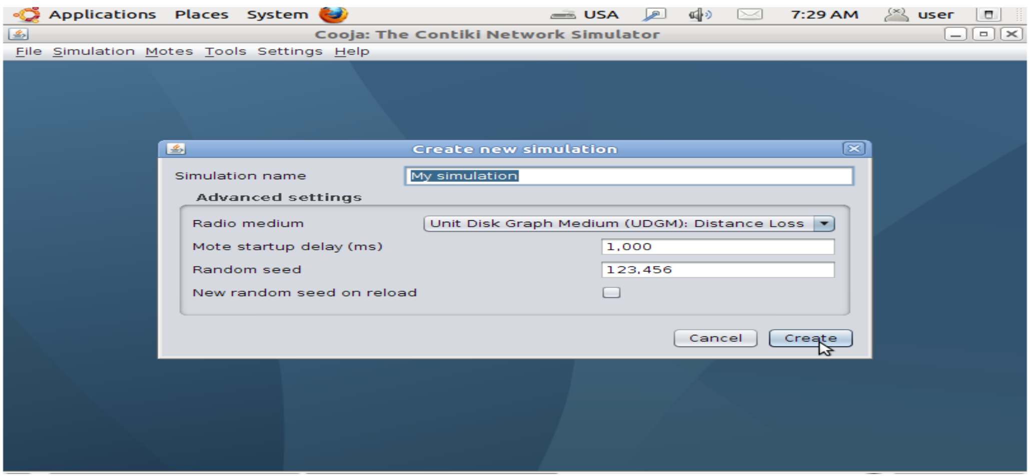


Fig 5.Sim_Options

6.Simulation windows

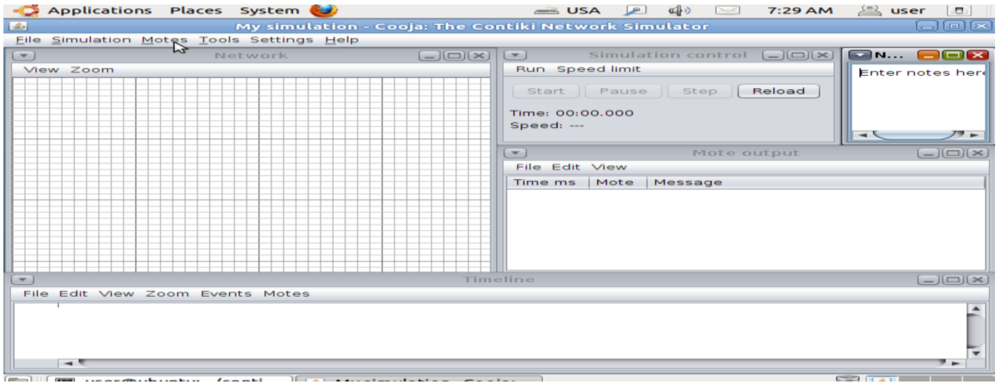


Fig 6.Sim_Window

8.Add motes to simulation

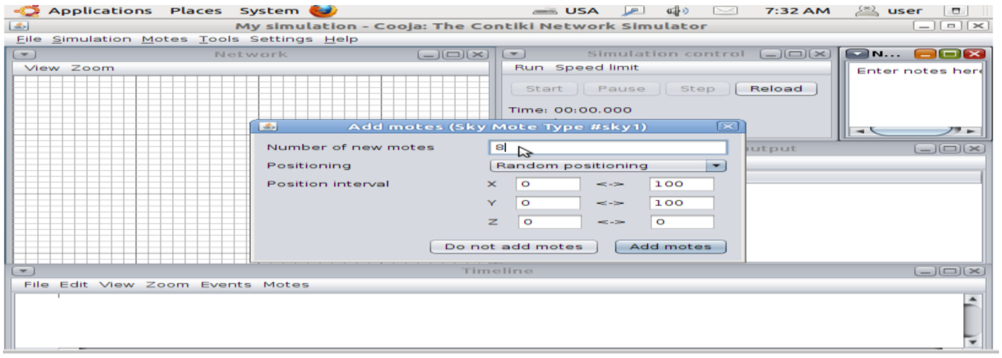


Fig.7 Motes

9.Start the simulation

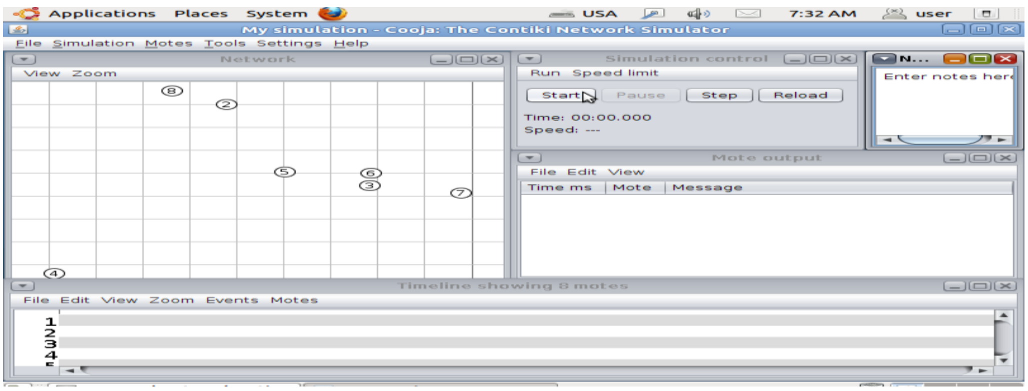


Fig.8.Start_Sim

3.2 Implementation

- In our approach to implement our angle based boundary detection,we start by creating a network of three nodes in shape of a triangle.These three nodes act as a reference node to us.

- The first step in our simulation is to start broadcasting packets from one node to another. This packet transfer helps us in calculating the RSSI value of nodes.

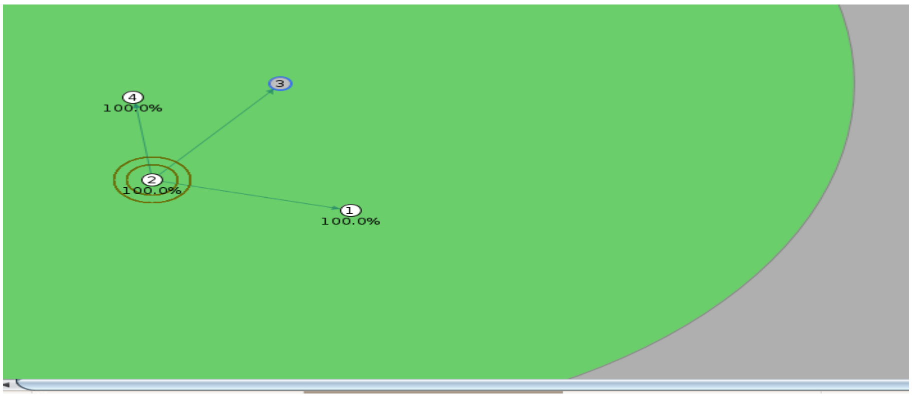


Fig.9 Broadcasting

- To calculate distance between nodes, we are assuming RSSI value of -30db to be equal to a length of 1m.

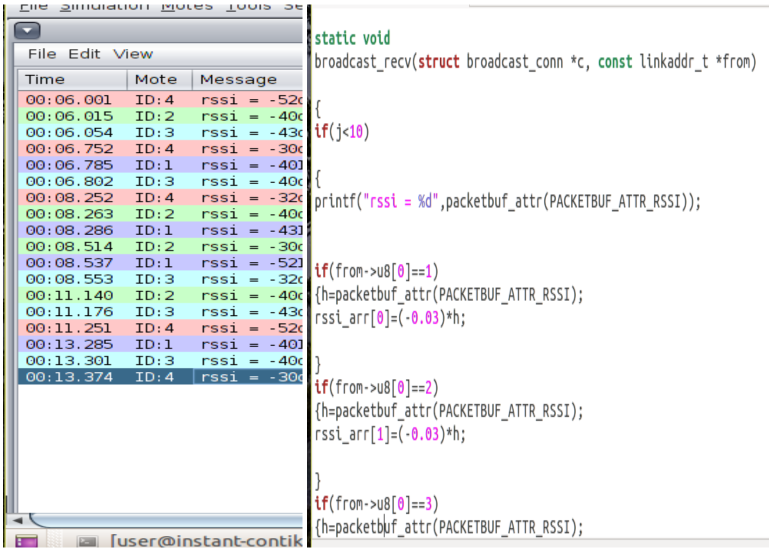


Fig.10 RSSI 1 &RSSI 2*

packetbuf_attr(PACKETBUF_ATTR_RSSI) will return the **RSSI** value associated with the packet.

packetbuf_set_attr(PACKETBUF_ATTR_RSSI,8) will set the **RSSI** attribute packet of the packet to be 8.

```
enum {
  PACKETBUF_ATTR_NONE,//or  integer value=0

  /* Scope 0 attributes: used only on the local node. */
  PACKETBUF_ATTR_CHANNEL,//or  integer value=1
  PACKETBUF_ATTR_NETWORK_ID,//or  integer value=2
  PACKETBUF_ATTR_LINK_QUALITY,//or  integer value=3
  PACKETBUF_ATTR_RSSI,//or  integer value=4
  PACKETBUF_ATTR_TIMESTAMP,//or  integer value=5
  PACKETBUF_ATTR_RADIO_TXPOWER,//or  integer value=6
  PACKETBUF_ATTR_LISTEN_TIME,//or  integer value=7
  PACKETBUF_ATTR_TRANSMIT_TIME,//or  integer value=8
  PACKETBUF_ATTR_MAX_MAC_TRANSMISSIONS,//or  integer value=9
  PACKETBUF_ATTR_MAC_SEQNO,//or  integer value=10
  PACKETBUF_ATTR_MAC_ACK,//or  integer value=11

  /* Scope 1 attributes: used between two neighbors only. */
  PACKETBUF_ATTR_RELIABLE,//or  integer value=12
  PACKETBUF_ATTR_PACKET_ID,//or  integer value=13
  PACKETBUF_ATTR_PACKET_TYPE,//or  integer value=14
  PACKETBUF_ATTR_REXMIT,//or  integer value=15
  PACKETBUF_ATTR_MAX_REXMIT,//or  integer value=16
  PACKETBUF_ATTR_NUM_REXMIT,//or  integer value=17
  PACKETBUF_ATTR_PENDING,//or  integer value=18

  /* Scope 2 attributes: used between end-to-end nodes. */
  PACKETBUF_ATTR_HOPS,//or  integer value=19
  PACKETBUF_ATTR_TTL,//or  integer value=20
  PACKETBUF_ATTR_EPACKET_ID,//or  integer value=21
  PACKETBUF_ATTR_EPACKET_TYPE,//or  integer value=22
  PACKETBUF_ATTR_ERELIABLE,//or  integer value=23

  /* These must be last */
  PACKETBUF_ADDR_SENDER,//or  integer value=24
  PACKETBUF_ADDR_RECEIVER,//or  integer value=25
  PACKETBUF_ADDR_ESENDER,//or  integer value=26
  PACKETBUF_ADDR_ERECEIVER,//or  integer value=27

  PACKETBUF_ATTR_MAX //or  integer value=28
};
```

Fig.11 PacketBuffer

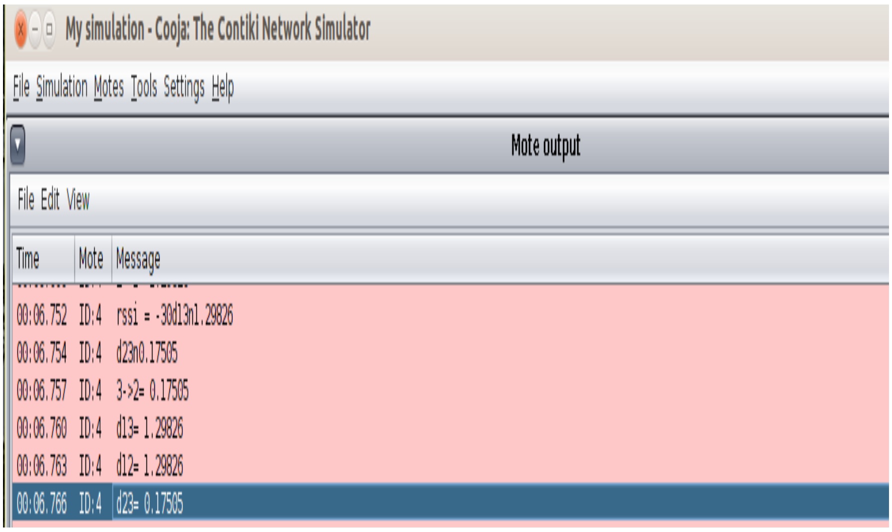


Fig.12 Distance between nodes

- For finding out among which two nodes the packet has been transferred,we are using the function ‘from->u8[0]’ which tell us about the initial node of packet transfer.

- Using law of cosines,we have calculated the angles formed between the sides.

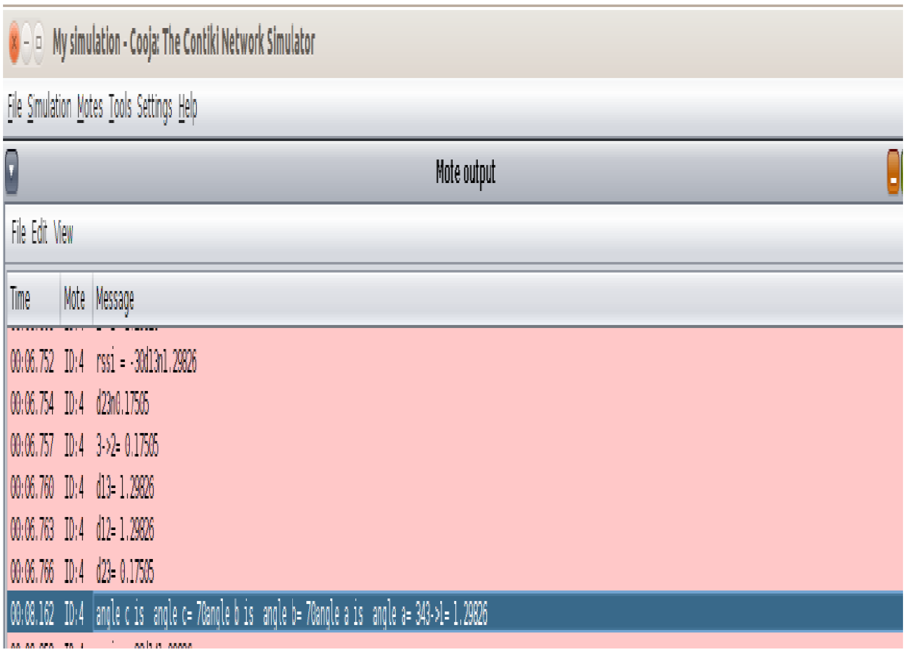


Fig.13 Angles

- Finally we have used this information to find whether a node lies inside the triangle or it is a boundary node.

RESULT AND PERFOMANCE ANALYSIS

4.1 RESULT

In accordance to our algorithm,for detecting whether a node is boundary node or not,we create a network of nodes and then with the help of a reference node we check if it is an interior point or exterior point (boundary node).

- Scenario 1:To check whether the node is an interior node

- Here we have created a network by placing the node inside the network.

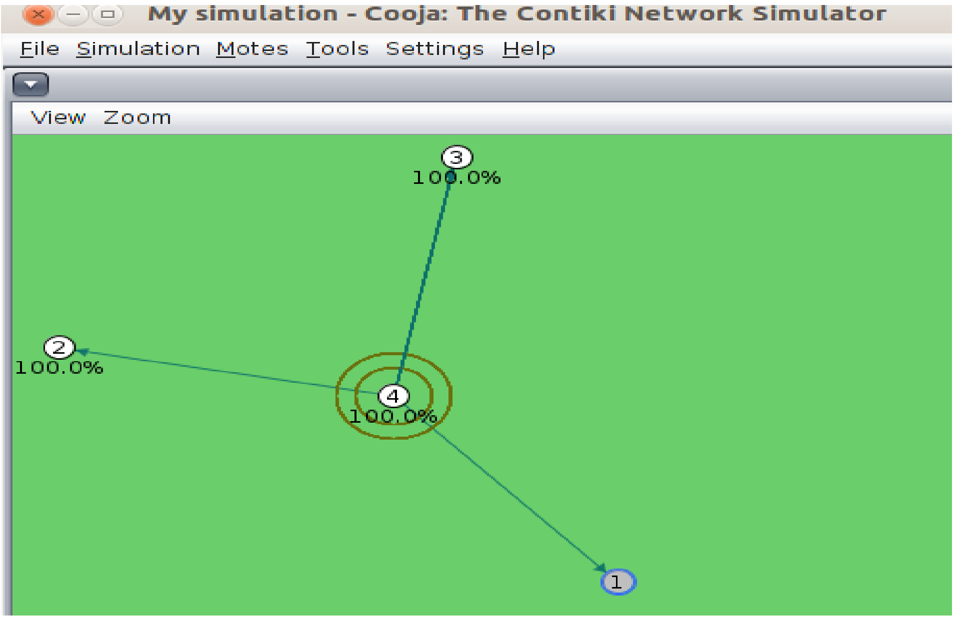
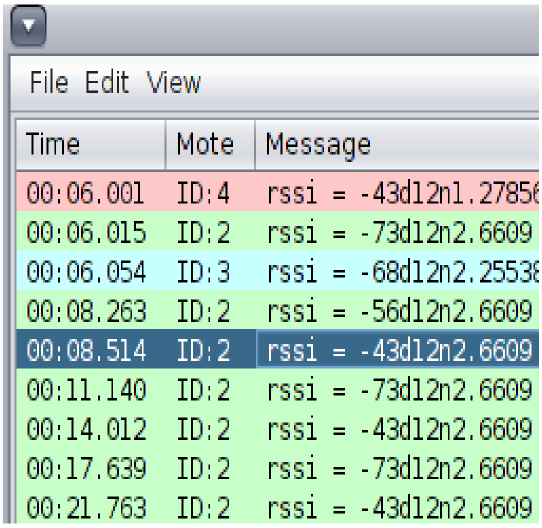
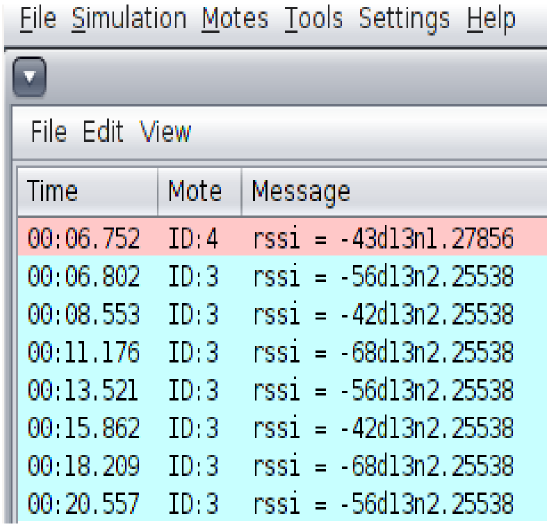


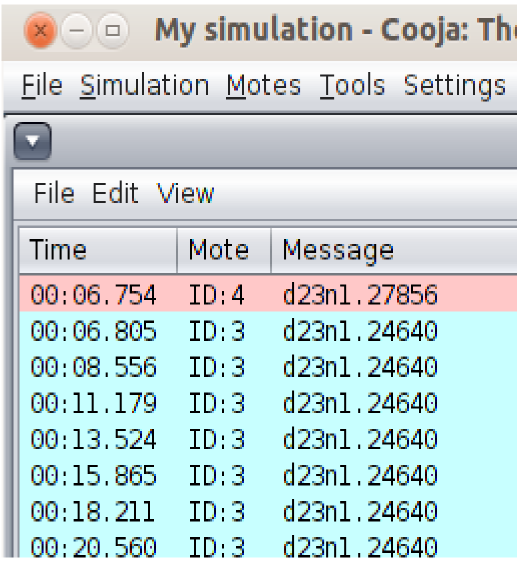
Fig.14 Inside_region



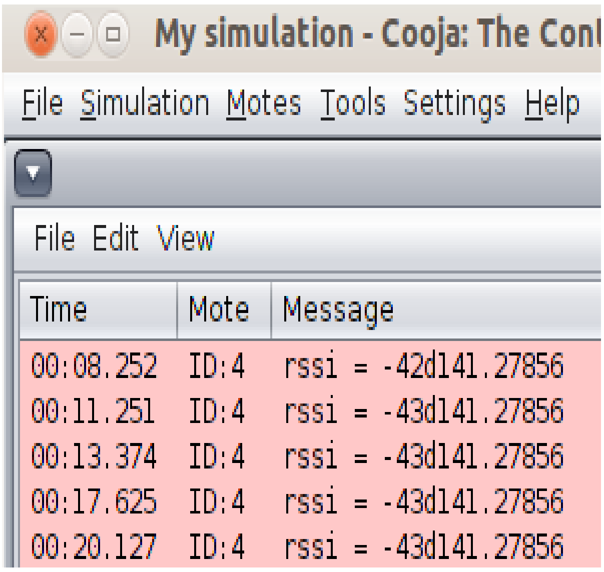
This show the distance b/w node 1 & node 2



This show the distance b/w node 1 &node 3

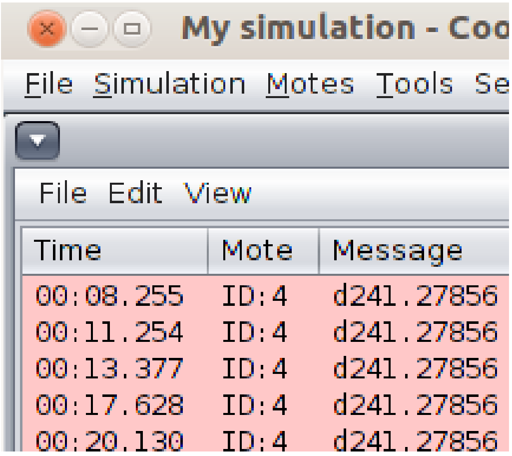


This show the distance b/w node 2 & node 3

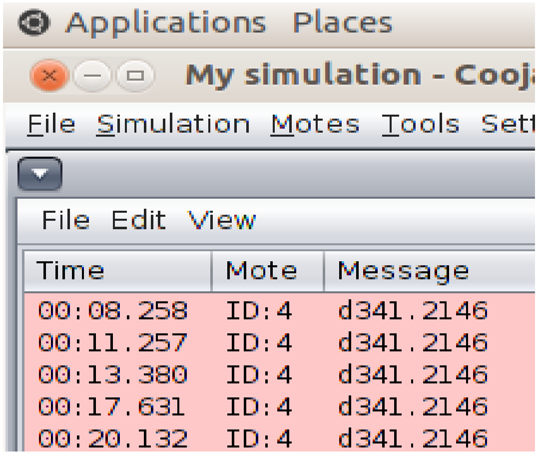


This show the distance b/w node 4 & node 2

Fig.15 Distance b/w nodes



This show the distance b/w node 2 &node 4



This show the distance b/w node 3 &node 4

We have used the simulator to compute the Received Signal Strength Indicator value & distance between nodes.Now using this data,we take use of a C compiler to find out the angle between nodes and further check whether the point lies inside or outside our given network.The use of Cooja for calculating the angle is not feasible as Cooja is not efficient with complex calculations.

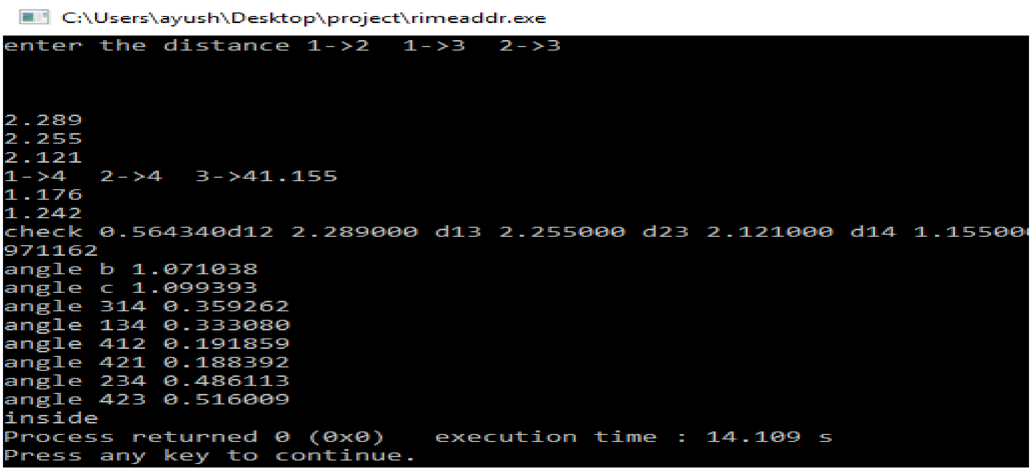


Fig.16 Interior node proof

➢ <u>Scenario 2</u>:To check whether the node is an exterior node

- Here we have created a network by placing the node outside the network.

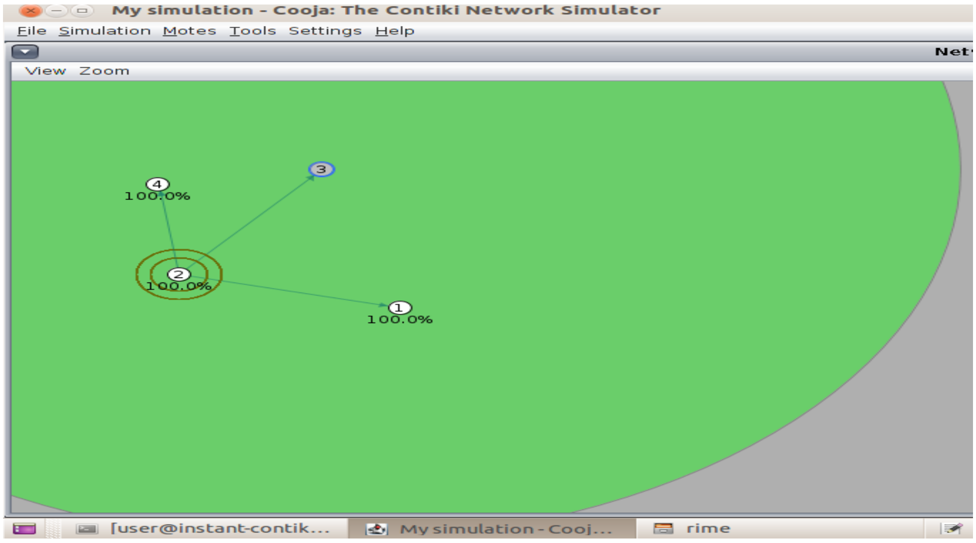
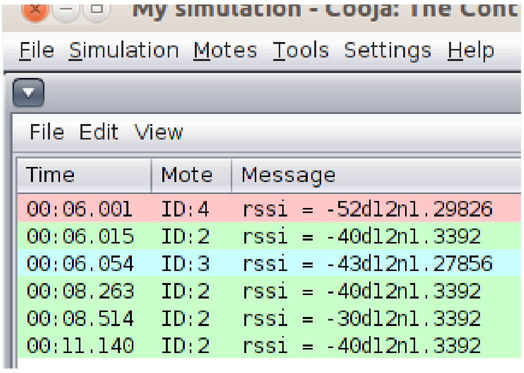
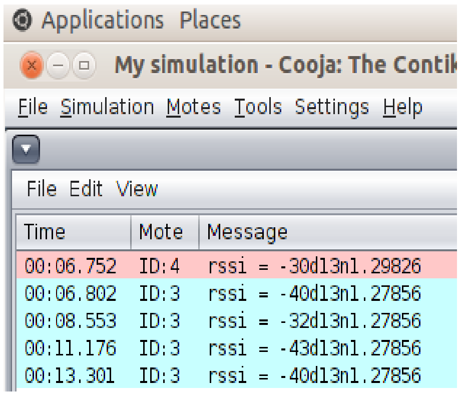


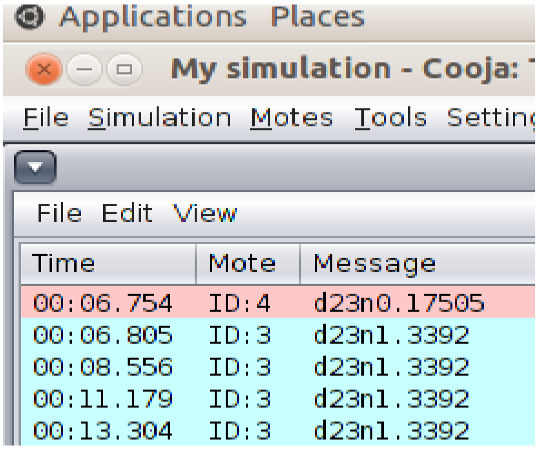
Fig.17 Outside region



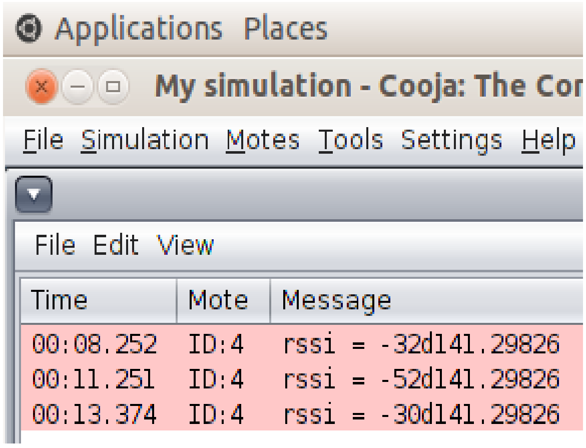
This shows the distance b/w node 1 & node 2



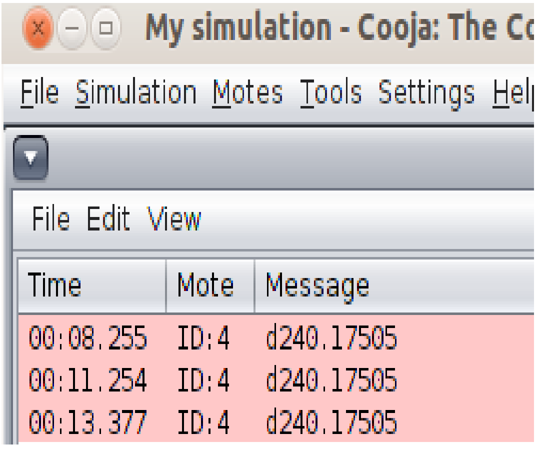
This shows the distance b/w node 1 & node 3



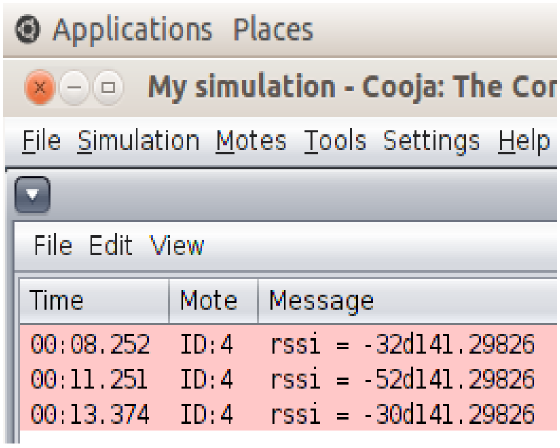
This shows the distance b/w node 2& node 3



This shows distance b/w node 1 & node 4



This shows the distance b/w node 2 & node 4



This shows the distance b/w node 3 & node 4

Fig.18 Distance b/w nodes

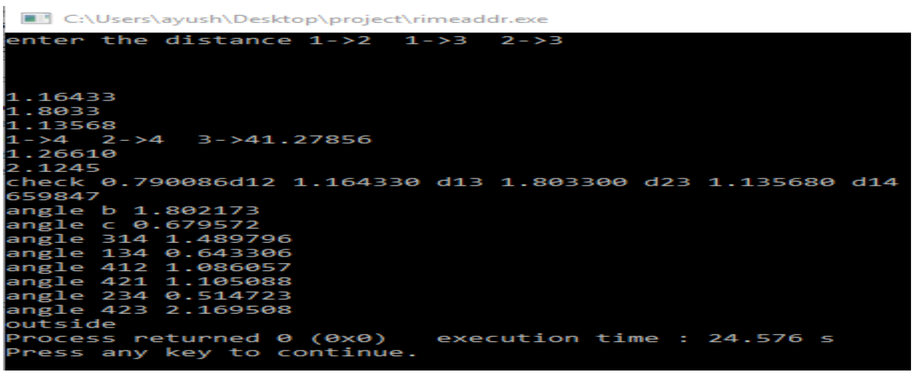


Fig 19 Exterior node

➢ <u>Scenario 3</u>:To check whether the reference nodes forms a triangle

- Here we have created a network and placed the reference nodes randomly.We check whether the network forms a triangle.

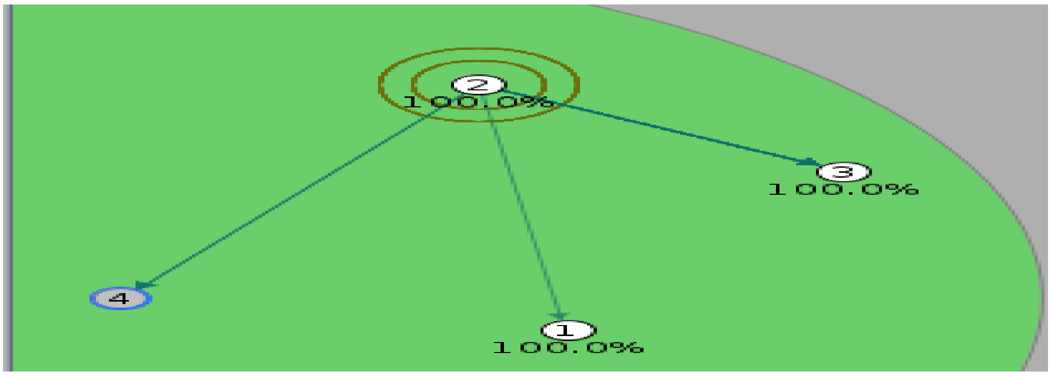
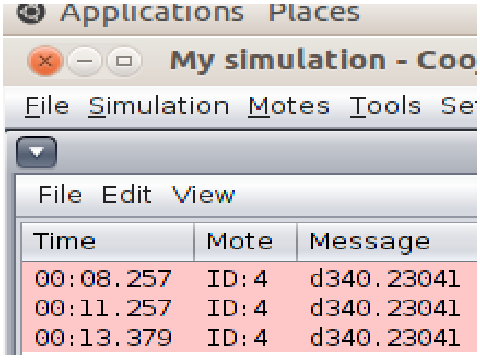
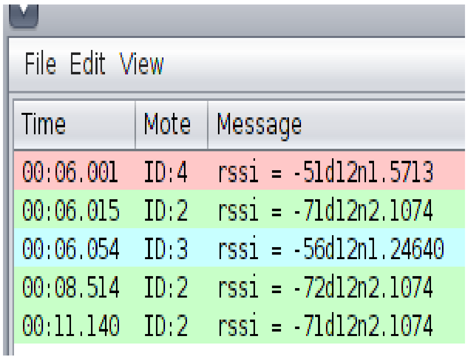


Fig.20 Reference nodes



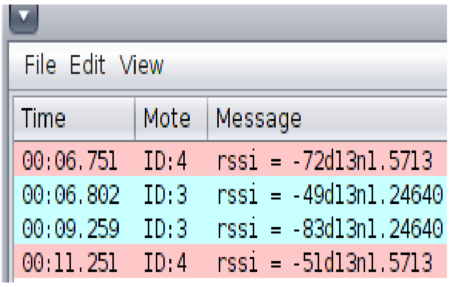
Time	Mote	Message
00:08.257	ID:4	d340.23041
00:11.257	ID:4	d340.23041
00:13.379	ID:4	d340.23041

This shows the distance b/w node 3 & node 4



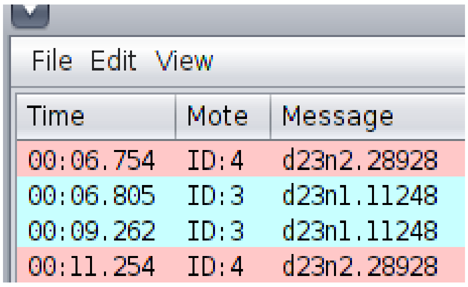
Time	Mote	Message
00:06.001	ID:4	rssi = -51d12n1.5713
00:06.015	ID:2	rssi = -71d12n2.1074
00:06.054	ID:3	rssi = -56d12n1.24640
00:08.514	ID:2	rssi = -72d12n2.1074
00:11.140	ID:2	rssi = -71d12n2.1074

This shows the distance b/w node 1 &node 2



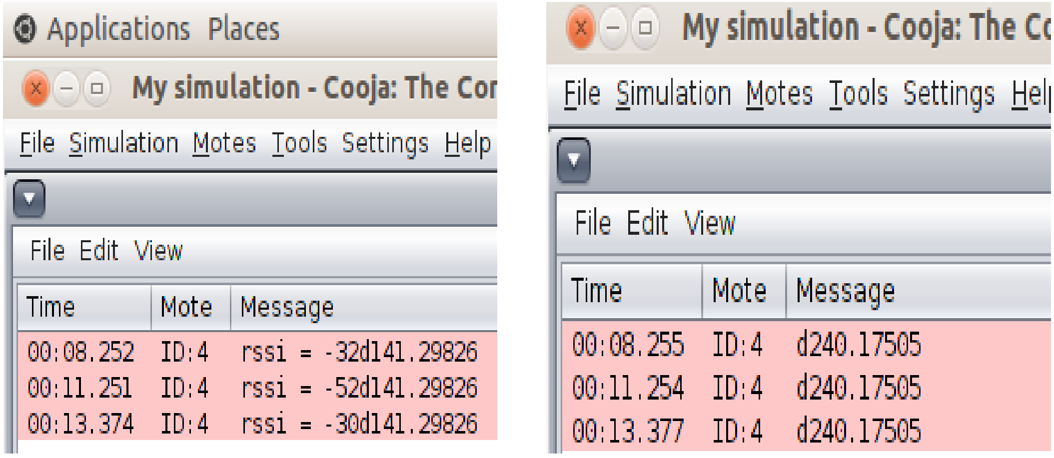
Time	Mote	Message
00:06.751	ID:4	rssi = -72d13n1.5713
00:06.802	ID:3	rssi = -49d13n1.24640
00:09.259	ID:3	rssi = -83d13n1.24640
00:11.251	ID:4	rssi = -51d13n1.5713

This shows the distance b/w node 1 & node 3



Time	Mote	Message
00:06.754	ID:4	d23n2.28928
00:06.805	ID:3	d23n1.11248
00:09.262	ID:3	d23n1.11248
00:11.254	ID:4	d23n2.28928

This shows the distance b/w node 2 &node 3



This shows the distance b/w node 1 & node 4

This shows the distance b/w node 2 &node 4

Fig.21 Distance b/w nodes

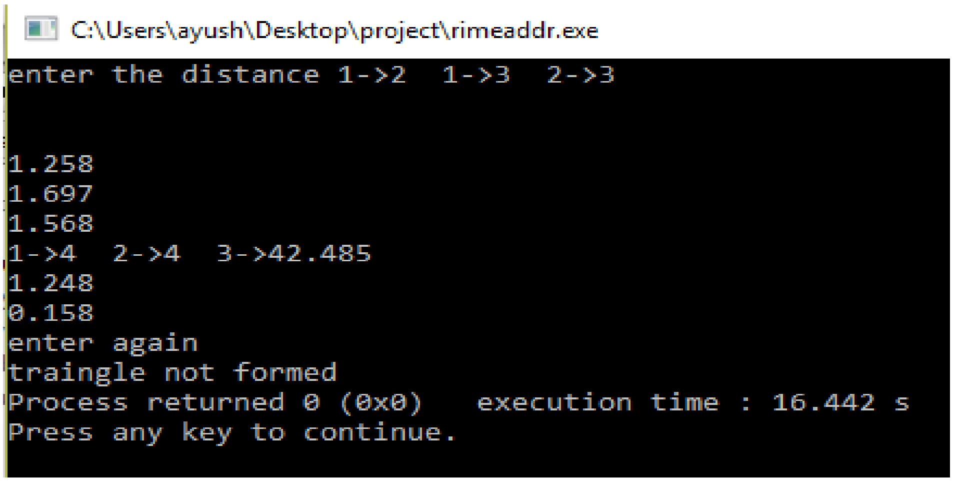


Fig.22 Triangle error

The triangle identity, " Sum of two sides is always greater than the third side" , does not hold true here.As the sum of side (2,4) + side(3,4) > side(2,3).

CHAPTER-5

CONCLUSION

The angle based boundary detection is an effective way to determine the eligibility of a node to be interior node or boundary node.At times,it falters while scaling RSSI value as a unit of distance or its inability to form a triangle.

Despite this,the boundary node detection has vast applications in the field of surveillance and virtual co-ordinate assignement.The method we use is handy as it does not require a static network and flourishes when used with a mobile network.The benefits to reap are immense,it just needs exploring further.

CHAPTER-6

FUTURE SCOPE

The angle based boundary detection algorithm has a large room for future improvements.Most of the boundary detection algorithms are predominantly working on 2-d networks.So to develop an algorithm that handles the 3-d structure has big incentives.Further,our algorithm works only for a triangle.There is a scope to develop an algorithm that works on larger network structure.Lastly,our algorithm uses the approximation technique to calculate the distance between nodes using RSSI,so a more accurate method will guarantee a more accurate result.

REFERENCES

[1] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *INFOCOM'04*, Hong Kong, China, March
2004.

[2] A. Ghosh. Estimating coverage holes and enhancing coverage in mixed sensor networks. In *LCN'04*, Washington, DC, Nov. 2004.

[3]Chi Zhang, Yanchao Zhang, Student Member, IEEE, and Yuguang Fang, Senior Member, IEEE.Detecting Coverage Boundary Nodes in Wireless Sensor Networks,Ft. Lauderdale, FL, USA

[4]Baoqi Huang, WeiWu, Guanglai Gao, and Tao Zhang,Recognizing Boundaries in Wireless Sensor Networks Based on Local Connectivity Information

[5]Jitender S. Deogun, Saket Das, Haitham S. Hamza, and Steve Goddard,An Algorithm for Boundary Discovery in Wireless Sensor Networks

[6] S. Benkirane, A.B. Hssane, M.L. Hasnaoui, M.D.E. Ouadghiri, M. Laghdir,Performance analysis of routing protocols for mobile wireless sensor network in an environment with obstacles using NCTUns tools, in:IEEE Next Generation Networks and Services, 2012, pp. 81–86.

[7] L. Zou, M. Lu, Z. Xiong, A distributed algorithm for the dead-end problem of location-based routing in sensor networks, IEEE Trans.Veh. Technol. 54 (4) (2005) 1509–1522.

[8] Wei-Cheng Chu, Kuo-Feng Ssu,Location-free boundary detection in mobile wireless sensor networks with a distributed approach

[9] Shailendra Shukla, Rajiv Misra,Angle Based Double Boundary Detection in Wireless Sensor Networks

[10] Ahmed M. Khedra, Walid Osamya, Dharma P. Agrawal,Perimeter discovery in wireless sensor networks

APPENDIX

```
#include "contiki.h"

#include "net/rime/rime.h"

#include "random.h"

#include "dev/button-sensor.h"

#include "dev/leds.h"

#include<math.h>

#include <stdio.h>


float square_r(float n)

{

float d;

int s;

for(s = 1; s*s <= n; s++);

s--;

float x;

for( d = 0.001; d < 1.0; d+= 0.001)

{

x = (float)s + d;

if((x*x > (float)n))

{

x -= 0.001;
```

```
break;
}
}
return x;
}
```

/*The above function is used to find and print the square root of the number as any math function is not suppoted by our cooja simulator*/

```
typedef enum
{
DEC1 = 10,
DEC2 = 100,
DEC3 = 1000,
DEC4 = 10000,
DEC5 = 100000,
DEC6 = 1000000,

} tPrecision ;

void putFloat( float f, tPrecision p )
{
long i = (long)f ;
printf("%d",i);
f = (f - i) * p ;
```

```
i = abs((long)f) ;
if( fabs(f) - i >= 0.5f )
{
i++ ;
}
printf(".");
printf("%d",i);
printf("\n");
}
/*The above code is used to print the float value of the distance as it is not possible in our simulator*/
int arr[16], iter=0,i,j=0,k=0;
int flag=0;
int h;
float rssi_arr[4],p,z,x,y=0,y2=0;
static int flag1=0,flag2=0,flag3=0;
static float angle_a,angle_b,angle_c;
static float dist12=0,dist13=0,dist23=0,d14=0,d24=0,d34=0,d13n=0,d23n=0,d12n=0;
static float sn=0,so=0,oa=0,ob=0,oc=0;
static int flag4=0,flag5=0,flag6=0;
static int flag1star=0,flag2star=0,flag3star=0,flag4star=0,flag5star=0,flag6star=0;

void putvalue(float n ,int num)
```

```
{
if(flag1==0||flag2==0||flag3==0)
{
if(num==14)
{
dist23=n;
if(dist23!=0)
{
flag3=1;
}}
}
if(flag1!=0&&flag2!=0&&flag3!=0)
{
putFloat(dist23,DEC6);
so=dist12+dist13+dist23;
y=(dist13*dist13+dist23*dist23-dist12*dist12)/(2*dist13*dist23);
y=(7*(1000-(1000*y)));
y=square_r(y);
angle_c=y;
printf("angle c is  ");
printf("angle c= %d",(int)angle_c);
y=(dist23*dist23+dist12*dist12-dist13*dist13)/(2*dist23*dist12);
y=(7*(1000-(1000*y)));
```

```
y=square_r(y);
angle_b=y;
printf("angle b is  ");
printf("angle b= %d",(int)angle_b);
y2=(dist13*dist13+dist12*dist12-dist23*dist23)/(2*dist13*dist12);
y2=(7*(1000-(1000*y2)));
y2=square_r(y2);
angle_a=y2;
printf("angle a is  ");
printf("angle a= %d",(int)angle_a);
}
}
/*The above function finds the angle between the sides of the triangle*/
PROCESS(example_broadcast_process, "Broadcast example");
AUTOSTART_PROCESSES(&example_broadcast_process);
static void
broadcast_recv(struct broadcast_conn *c, const linkaddr_t *from)
{
if(j<10)
{
printf("rssi = %d",packetbuf_attr(PACKETBUF_ATTR_RSSI));
if(from->u8[0]==1)
{
```

```
h=packetbuf_attr(PACKETBUF_ATTR_RSSI);
rssi_arr[0]=(-0.03)*h;
}
if(from->u8[0]==2)
{h=packetbuf_attr(PACKETBUF_ATTR_RSSI);
rssi_arr[1]=(-0.03)*h;
if(from->u8[0]==1)
{
h=packetbuf_attr(PACKETBUF_ATTR_RSSI);
rssi_arr[0]=(-0.03)*h;
}
if(from->u8[0]==2)
{h=packetbuf_attr(PACKETBUF_ATTR_RSSI);
rssi_arr[1]=(-0.03)*h;
}
if(from->u8[0]==3)
{
h=packetbuf_attr(PACKETBUF_ATTR_RSSI);
rssi_arr[2]=(-0.03)*h;
}
if(from->u8[0]==4)
{
h=packetbuf_attr(PACKETBUF_ATTR_RSSI);
```

```
rssi_arr[3]=(-0.03)*h;
}
arr[from->u8[0]-1]=1;
if(j==4&&flag==0){
flag=1;
printf("neighbour list is" );
for(i=0;i<16;i++)
{
if(arr[i]==1)
{
printf(" %d",i+1);
}
}}
for(k=0;k<3;k++)
{
if(rssi_arr[k]==0)
{
break;
}}
if(k==0)
{
putvalue(rssi_arr[1],12);
checkpoint(rssi_arr[1],12);
```

```
printf("1->2= ");

putFloat(rssi_arr[1],DEC6);

putvalue(rssi_arr[2],13);

checkpoint(rssi_arr[2],13);

printf("1->3= ");

putFloat(rssi_arr[2],DEC6);

}

if(k==2)

{

d13n=rssi_arr[0];

printf("d13n");

putFloat(d13n,DEC6);

d23n=rssi_arr[1];

printf("d23n");

putFloat(d23n,DEC6);

putvalue(rssi_arr[1],14);

checkpoint(rssi_arr[1],14);

printf("3->2= ");

putFloat(rssi_arr[1],DEC6);

putvalue(rssi_arr[0],13);

checkpoint(rssi_arr[0],13);

printf("3->1= ");

putFloat(rssi_arr[0],DEC6);
```

```
}
if(k==1)
{
d12n=rssi_arr[0];
printf("d12n");
putFloat(d12n,DEC6);
putvalue(rssi_arr[2],14);
checkpoint(rssi_arr[2],14);
printf("2->3= ");
putFloat(rssi_arr[2],DEC6);
putvalue(rssi_arr[0],12);
checkpoint(rssi_arr[0],12);
printf("2->1= ");
putFloat(rssi_arr[0],DEC6);
}
if(k==3)
{
d14=rssi_arr[0];
printf("d14");
putFloat(d14,DEC6);
d24=rssi_arr[1];
printf("d24");
putFloat(d24,DEC6);
```

```
d34=rssi_arr[2];

printf("d34");

putFloat(d34,DEC6);

checkpoint(rssi_arr[0],40);

checkpoint(rssi_arr[1],41);

checkpoint(rssi_arr[2],42);

putFloat(d34,DEC6);

checkpoint(rssi_arr[0],40);

checkpoint(rssi_arr[1],41);

checkpoint(rssi_arr[2],42);

}

}

}

/* The above is finding and storing RRSi value in an array*/

static void

sent_by_abc(struct broadcast_conn *c, int status, int num_tx)

{

}

static const struct broadcast_callbacks broadcast_call = {broadcast_recv,sent_by_abc};

static struct broadcast_conn broadcast;

/*---------------------------------------------------------------------------*/

PROCESS_THREAD(example_broadcast_process, ev, data)

{
```

```
static struct etimer et;

PROCESS_EXITHANDLER(broadcast_close(&broadcast);)
PROCESS_BEGIN();
for(i=0;i<16;i++)
{
arr[i]=0;
}
for(i=0;i<4;i++)
{
rssi_arr[i]=0;
}
broadcast_open(&broadcast, 129, &broadcast_call);
while(j<11) {
/* 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");
j++;
}
PROCESS_END();
```

```
}

/*----------------------------------------------------------------------------------------------------------*/

/*The above function is the main function and is starting the broadcast process.*/

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