

CONNECTIVITY BASED BOUNDARY RECOGNITION 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

Saurav Kumar (141213)
Divyanshoo Sharma(141229)

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**

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled “**Coonectivity Based Boundary Recognition 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 Wknaghat 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))**

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

(Student Signature)
Saurav kumar, 141213

(Student Signature)
Divaynshoo sharma,141229

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

(Supervisor Signature)
Dr. Shailendra Shukla
Assistant Professor (Senior Grade)
Computer Science Department
Dated:

ACKNOWLEDGMENT

This may seem long but the task of our project work both theoretically and practically may not have been completed without the help guidance and mental support of the following persons.

Firstly, we would like to thanks my guide Assistant Professor (Senior Grade), Department of CSE Jaypee University of Information technology Wagnaghat, **Dr. Shailendra Shukla** sir who provided us with the idea and related material for the project proposal. He indeed guided us to do the task for our thesis in such a way that it seems to be research work. His continuous monitoring to support us and our research work encouraged us a lot for doing our thesis in very smooth manner.

Secondly, we would like to thanks **our Parents** who have always been with us for inspiring us and thirdly, we would like to thanks **God** for keeping us energetic, healthy and enthusiastic

Signature of the student
Saurav Kumar:
Date

Signature of the student.....
Divyanshoo sharma:
Date

Table of Content

Certificate	(i)
Acknowledgement	(ii)
Table of Content	(iii)
List of Abbreviations	(v)
List of Figures	(vi)
List of Tables	(vii)
Abstract	(ix)
Chapter 1 Introduction	
1.1 Introduction	1
1.2 Problem Statement	5
1.3 Objectives	5
1.4 Methodology	6
1.5 Organization	7
Chapter-2 Literature Survey	
2.1 Distance Based Boundary Recognition	8
2.2 Connectivity Based Boundary Recognition	11
Chapter-3 System Design	
3.1 System Development	15
3.2 Platform used for simulation	18
3.3 Algorithm	20
3.4 Practical issues	24
Chapter-4 Performance Analysis	
4.1 Proposed Approach	26
4.2 Algorithm for the INITIALIZE (n_0) function	26
4.3 Flowchart of INITIALIZE function	28
4.4 Practice Run of the INITIALIZE function	29
4.5 Analysis for the sparse network	36
4.6 Implementation In Cooja Simulator	37

4.7 Simulation On The Dense Network	41
4.8 Implementation In Cooja	42
4.9 Analysis For The Dense Network	43
4.10 code	44
CHAPTER -5 Conclusion	
5.1 Conclusion	49
5.2 Future Scope	49

List of Abbreviations

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

List Of Figures

S.No.	Figure Number	Page Number
1.	Fig 1: Differentiating between boundary and interior node	2
2.	Fig 2: Overview of methods used for boundary recognition	6
3.	Fig 3: The Proposed Approach For Boundary Extraction	15
4.	Fig 4: Boundary cycle corresponding to the boundary node	16
5.	Fig5. The external and internal geometric boundaries (solid curves) of a wireless sensor network. The dashed lines denote the communication link.	17
6	Fig6. Contiki operating ssystem an IPv6 routing protocol on 41 nodes in the Cooja Contiki network simulator	20
7	Fig 7. Decision on each node after executing INITIALIZE(no) function	21
8	Fig 8. Value of ft after executing the erosion operation	22
9	Fig 9. Tight inner and outer boundaries	22
10	Fig10. INITIALIZE(n0) algorithm	27
11	Fig11: Flow chart to depict the working of INITIALIZE (n0) function	28
12	Fig 12. Deployed wireless sensor network and 1-hop neighbour set for each node	29
13	Fig 13: 2-hop neighbour set of each node in the deployed network	30
14	Fig 14: Practice run of the INITIALIZE function on node numbered 13	31
15	Fig 15. Practice run of the code showing the values of variables at each iteration	32
16	Fig 16. Practice run on the node numbered 13	33
17	Fig 17. Node numbered 13 is identified as actual interior node and node numbered 15 is identified as suspected boundary node	34

18	Fig 18. Console output for the INITIALIZE function	35
19	Fig 19. console output after the execution of erosion function	36
20	Fig 20. Implementation in cooja simulator, nodes with LED red on are on boundary whereas the nodes with LED Blue are interior node	37
21	Fig 21. Mote output window, node with id 5,9,17 and 13 are boundary node	38
22	Fig.22: Creating Mote Of Sky Types	39
23	Fig.23: Firmware selected	39
24	Fig 24: Network created	40
25	Fig 25: Decision on node with ID:13	41
26	Fig 26. Implementation in cooja simulator, nodes with LED red on are on boundary whereas the nodes with LED Blue are interior node	42
27	Fig27. Mote output window showing decision on nodes	43

LIST OF TABLES

S.No.	Table Number	Page Number
1.	Table 1: Existing approaches for boundary recognition	12

Abstract

In this project, we handle the issue of boundary recognition by actualizing the algorithm proposed by which depends on basic, conveyed and connectivity based approach. This algorithm looks at the 2-hop iso-contour of every node. In particular, the proposed algorithm settles on a harsh choice on a speculated boundary node by looking at its 2-hop iso-contour and afterward refines the choice in view of a heuristic activity (otherwise called the disintegration task), which fundamentally decreases the measure of the presumed boundary node set. All the more critically, boundary cycles comparing to inward and external boundary are recognized and give important information to different applications. An intensive assessment demonstrates that our algorithm is material to a large portion of the disseminated WSNs. Lastly we analyse the simulation and deduce the accuracy of the algorithm for two different wireless sensor networks.

Chapter-1

INTRODUCTION

1.1 Introduction

Wireless sensor networks (WSNs), included large number of little and reasonable (sensor) nodes with compelled figuring power, restricted memory, and short battery lifetime, can be utilized to screen and gather information in an area of intrigue.

Wireless sensor networks are extensively applied in many fields such as battlefield surveillance, target tracking environment monitoring, and health care. Most of the above applications are mission-critical and require complete coverage of the monitored area so that the events under observation can be precisely detected.

In the wake of being sent, a wireless sensor network (WSN) stays reliant on human intercessions as it routinely should be kept up and nourished with new parameters, for example, refreshed communications pathways and positions and abilities of recently included sensors. In severe and remote regions, sensors need to work unattended and thus devoid of the necessary human maintenance. Accordingly, averting and repairing abandons caused by battery consumption and decimation by creatures or potentially spatial occasions are relatively inconceivable. These deformities may at last prompt losing correspondence network in a few sections of the WSN.

Failure of boundary nodes degrade the overall performance of the wireless sensor networks(WSN) such as decreasing the connectivity, unbalancing the WSNs load, and aggravating the burden of hole border nodes in data forwarding.

As it is relatively difficult to keep the presence of gaps in light of irregular occasions particularly in dynamic WSN open air environments, a few activities have concentrated on constraining and keeping the impacts of these voids.

What are Boundary Nodes?

Nodes performing observation of target territory are called boundary nodes. They are required to stay mindful for occasion acknowledgment like articles moving in and moving out the territory under surveillance. An arrangement of boundary nodes is indicated by BNs, because of dynamic association in observation they experiences speedy vitality depletion, results to a shorter lifetime.

Since sensor nodes are haphazardly sent in a network, nodes' shutdown, or a natural deterrent may happen because of which holes can be framed in the network, building sets of unavailable nodes and leaving revealed regions. ‘

Furthermore, in like manner they can cause the non-achievement of directing algorithms. While perceiving either the nodes on the boundary of void or on the network's boundary; revealed regions will be recognized and could be repaired by an incremental development of new sensors, the already specified acknowledgment furthermore empowers the routing protocols to recognize and pass these holes.

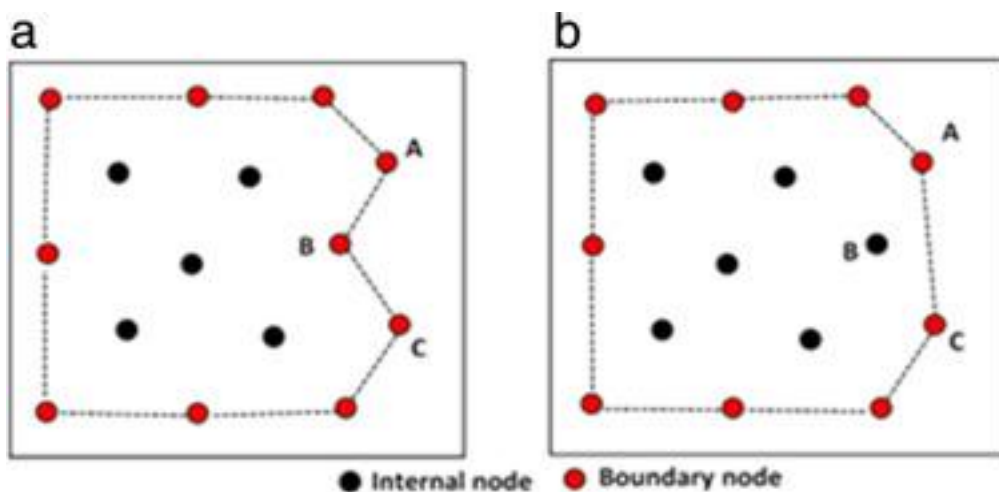


Fig1.Differentiating between boundary and interior node^[1]

Application and Motivation

Limit nodes ordinarily play a more essential role than rest of the nodes. From one viewpoint, limit nodes not just specifically connect with the external condition, for example, events coming in or moving out of the locale checked by the WSN, any correspondence with the outside condition is executed with the help of boundary nodes and help to obtain statistics about the WSN structure.

Then again, because of the interaction between the limits of a WSN and its physical condition, for example, design of the floor in a building, a guide of a transportation organize, territory varieties, and deterrents, boundary nodes are vital for monitoring the WSN shape which shows huge highlights of the fundamental condition.

Rather than the external boundary of a WSN, boundaries of inward openings (named internal boundaries) are basic markers of the general soundness of a WSN, for example, inadequate scope and network. In this way, boundary recognition is of awesome significance for different WSN applications.

Some of the applications of the boundary recognition include:

Node's energy conservation:

If a sensor node can self-distinguish its location on a coverage boundary, it can consequently tune its technique to wake up neighbouring nodes to fill in the coverage gap to preserve vitality.

Localization of sensor nodes in the Wireless Sensor Network.

Geographic Routing

Geographic routing is a routing principal that depends on geographic position data to send a message to the geographic location of the destination instead of utilizing the system address, it can be extremely effective if all the coverage boundaries can be distinguished already.

Boundary Recognition Approach

Impressive endeavors have been put resources into creating limit recognition algorithms, bringing about three classes of techniques: geometric, statistical and topological strategies.

Geometry based strategies employ the geometrical tools to identify the limit nodes and coverage voids with the presumption that nodes are aware of their own locations. In spite of the fact that the geometric strategies can discover more exact boundary nodes than the rest two classes, the necessity of hub area data restrains its application, particularly in extensive scale WSNs.

Statistical strategies usually identify limit nodes by assuming that the nodes in WSNs abide by the probability distribution. The measurable techniques more often than not make suppositions about the probability distribution of the node deployment, and afterward probabilistically distinguish limit nodes in view of some factual properties under certain system conditions. Although good results can be achieved when the node degree in WSNs is smooth, the method is prone to being invalid when the node degree fluctuates sharply. One noteworthy shortcoming of the statistical techniques is the impossible necessity on node distribution and thickness.

The topological techniques utilize topological properties, for example, connectivity information to identify boundary nodes. particularly this approach is applied to the location-free environment, recognize boundary nodes and coverage holes by exploiting the connectivity information of nodes. Typically, the topological techniques have higher bundle control overheads than the rest two classes due to collecting connectivity information from neighbouring nodes, however, don't require node location information and for the most part, beat the statistical strategies. In this paper, we center around perceiving precise boundaries utilizing minimal effort connectivity information in WSNs.

1.2 Problem Statement

- Determine suspected boundary nodes in a wireless sensor network(WSN)
- Performing disintegration (heuristic) task on the arrangement of suspected limit nodes, to get the real boundary nodes.
- Identifying the tight inner and outer boundaries and boundary cycles.

1.3 Objectives

- To do an exhaustive assessment and demonstrate that the proposed calculation is applicable in appropriated WSNs.
- To implement the Connectivity based boundary recognition algorithm proposed by ^[1].
- Extending the proposed algorithm's approach to 3-D networks. The existing algorithm works only for 2-D networks and is unable to detect boundaries for 3-D networks.^[1]
- Coping with Continuous changing behavior of network i.e. mobility of the sensor nodes.
- Implementing the algorithm on the wireless sensor nodes in cooja simulator.

1.4 Methodology

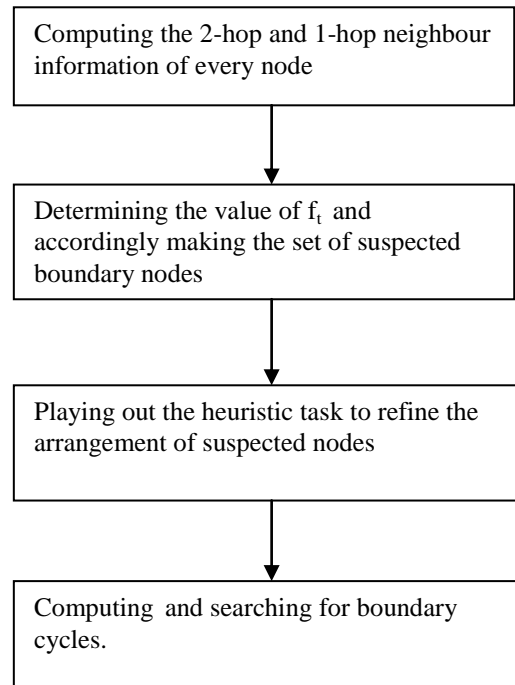


Fig2: Overview of methods used for boundary recognition

The algorithm proposed to detect boundary nodes utilizes connectivity information of each node. At earlier stages the 1-hop and 2-hop information of each node is determined. Then a function $\text{INITIALIZE}(n_0)$ is executed to compute the value of f_t which determines whether a node is an actual interior node or is a suspected boundary nodes.

After the execution of $\text{INITIALIZE}()$ function, these steps follow:

- After the execution of the function on every node, the final set of suspected boundary node is obtained.
- The suspected boundary nodes in the set are far more than the actual boundary nodes.
- An erosion activity is performed to refine the arrangement of suspected boundary node.

- After determining the value of f_d the boundary cycles are identified which corresponds to both tight inner and tight external boundaries.
- The remote channel considered here, fulfills the UDG (unit plate diagram) model where two nodes are associated if and just if their separation is at generally 1.

1.5 Organization

This report is organized into five chapters.

In chapter 1, Introduction, the genesis of the problem, problem statement and methodology followed by different objectives.

In Chapter 2, Literature review presents several existing algorithms for boundary recognition in WSN

In chapter 3, A Solution for boundary recognition using connectivity information of sensor nodes is given.

In the chapter 4, The proposed algorithm is presented with its analysis both computational and mathematical.

Chapter 5 consist of Conclusion and Future scope

Chapter-2

LITERATURE SURVEY

Boundary recognition algorithms mainly have two approaches one is distance based approach which use the geometrical value between sensor nodes and the another approach is connectivity based approach which use the neighbour information of nodes.

2.1 DISTANCE BASED BOUNDARY RECOGNITION:

It initially decides the boundary of an arrangement of nodes which can be considered as an arrangement of seeds. At that point, it decides the greatest hop distances around every seed and analyzes whether the shape around a seed frames a closed cycle. Just nodes that are close to the boundaries are distinguished and they are not expressly associated as a polygon.

Following methods based on distance based boundary recognition technique:

a. **ABBD (Angle Based Boundary Recognition Algorithm) ALGORITHM** ^[11]:

It is produced to identify boundaries utilizing the separation between neighbour nodes. The fundamental thought of this algorithm is to test whether every sensor node is secured by the polygon shaped by its neighbour nodes. A sensor node will be proclaimed non-boundary node, in the event that it is covered by no less than three nodes.

Some essential highlights of this algorithm are right off the bat it depends on 1-hop neighbour data, besides it delivers less system overhead and ultimately the algorithm is valuable in improving the lifetime of limit nodes. The multifaceted nature of this algorithm is $O(n/k)$ time and $O(k\delta^3)$ messages. Where n is the quantity of nodes, δ is the quantity of neighbour and k is the quantity of nodes all the while executing nearby calculation.

b. **THE D-LPCN ALGORITHM**^[12]:

DLPCN algorithm is a distributed model of LPCN algorithm proposed by [12]. In this algorithm iteration is performed on every node in the network. In every iteration, any boundary node, aside from the first, picks its closest polar angle node among its neighbours regarding the node found in the past iteration. The important starting node can be thusly chosen using the Minimum Finding algorithm, which has two key purposes of intrigue.

The first is that the algorithm works with an associated network, given as planar or not. Additionally, it considers any blocking condition and contains the essential parts to avoid them. The second favorable position is that the algorithm can decide every one of the boundaries of the diverse associated parts of the network.

In this algorithm vitality utilization of each node was compute and it is discovered that the utilization of the energy in network relies upon the number of boundary nodes i.e hub thickness, Degree of every hub and their number.

c. **EBDG (Energy balanced Data Gathering) ALGORITHM**^[10]:

energy utilization is a natural issue in wireless sensor network depicted by multihop routing and many-to-one traffic pattern, and this uneven vitality dispersing would altogether be able to diminish organize lifetime.

To determine the issue of energy usage in wireless sensor network and amplifying the lifetime of the network an algorithm was proposed by [10]. In this paper, energy utilization adjusting issue was defined as an ideal transmitting information circulation issue by consolidating the thoughts of Corona-based network division and combined directing technique with information aggregation.

The paper firstly presents a confined zone-based routing plan that assured balanced vitality utilization among nodes within each corona. A centralized algorithm with time complexity $O(n)$ (n is the number of coronas) to solve the transmitting information distribution problem went for balancing vitality utilization among nodes in various

coronas is designed. The approach for computing the flawless number of coronas to the extent of maximizing the lifetime of the network is also included in this paper.

d. BRGT(Boundary Recognition via Graph-Theory)ALGORITHM^[5]:

This is a centralized algorithm which uses graph clustering strategy, in which it permits the division of the system into little clusters that evade association openings. At that point, the boundary nodes of every group are recognized utilizing centrality scores.

The discovery of limit nodes is enforced by the combination of adjoining clusters. Just nodes that are on the fringe of a solitary group and are not associated with at least two clusters are chosen as boundary nodes. It isn't reasonable for expansive networks..

Limitation:

Since the distance based boundary discovery systems utilize the Geometrical technique, can discover more precise boundary nodes than the other two categories(Topological and Statistical), the necessity of hub area data constrains its application, particularly in huge scale WSNs.

2.2 CONNECTIVITY BASED BOUNDARY RECOGNITION:

It uses N-hop neighbour information to construct iso-contour of each node. Connectivity-based Boundary Recognition for the topology has demonstrated the extraordinary effect on the execution of such administrations as area, directing, and way arranging in wireless sensor networks. It doesn't require sensor areas and just uses network connectivity information.

Algorithms used for recognition of the boundary nodes include:

a. Funke's algorithm(Topological Hole Recognition)^[6]:

This algorithm builds iso-contours of one common node looks at where the contours are broken and yield boundary nodes with specific certifications. Be that as it may, just nodes close to the boundaries are recognized, and the density prerequisite is fairly high.

In initial segment of this algorithm, there is a straightforward circulated methodology to distinguish nodes close to the boundary of the sensor field as well as near hole boundaries.

The hole identification algorithm is constructed simply with respect to the topology of the correspondence graph, i.e. the main information accessible is which nodes can interact with each other.

b. Research by I. Khan on "Self-recognition scheme for WSN boundary detection (SDBR)" (Oct. 2009)^[9]: In this algorithm, each node can choose to be a boundary or interior one by the development of its 2-hop neighbours' graph. At that point, the node will checks if this graph frames a closed cycle or broken path. The broken way shows the node is dwelling on the boundary, while if there should be an occurrence of the closed cycle the node is set apart as an inner node.

The major drawback of this algorithm is its determination of coarse limits in which a few interior nodes are wrongly perceived as boundary nodes.

c. Research by W.-C. Chu on “Decentralized boundary recognition without location information in wireless sensor networks.” (Apr 2012)^[13]: The algorithm proposed here, relaxes the condition of distinguishing boundary nodes from interior nodes and identifies more nodes as boundary nodes.

d. CBC^[8] : Hsieh et al. depicted the Distributed Boundary Recognition Algorithm which depends on four stages. The principal stage is to choose Closure Nodes (CNs) which generally encase the openings and boondocks of the detecting field. In the second stage, those closure nodes are associated with each other to frame Coarse Boundary Cycles (CBCs) for recognizing every obstruction. The third stage is proposed to find the correct Boundary Nodes (BNs) and interface them to refine the CBCs to be final boundaries.

The last stage is proposed to keep up the uprightness of BNs while the boundary is broken because of nodes failure.

e. DBNS & SBNS ALGORITHM^{[2][3]}:

Both of the algorithms are proposed by Sahoo et al. The SBNS algorithm accept the sink to be a boundary node at that point utilizes the correct hand rule to choose boundary nodes in a sequential way. The procedure is dined by the sink and is rehashed until the beginning node (sink) is returned to.

While DBNS algorithm characterizes extraordinary nodes as boundary nodes at that point interfacing them to frame cycles encasing limits. An outrageous node is characterized as a node that has either most extreme or least value in its directions contrasted with of its one-hop neighbours.

The fundamental downside of these methods is the need of a precise arrange of sensor nodes. Every node must be furnished with a situating gadget, for example, GPS to get its geographical location, which isn't appropriate for small sensors with low vitality utilization.

f. CDCHD ^[5] : Fekete et al. proposed another boundary recognition algorithm called Connectivity-based Distributed Coverage Hole Recognition CDCHD.

The essential thought is that nodes on the boundaries have moderately smaller normal degrees than nodes inside the network. A statistical limit is utilized to recognize boundary nodes and internal nodes.

g. LCA (Least Common Ancestor) :

Wang et al. construct a shortest path tree through flooding the entire network beginning from a seed node. At that point the algorithm looks for cut nodes that are characterized as nodes which have their Least Common Ancestor (LCA) generally far away and their ways to the LCA very much isolated. By utilizing the cut nodes, the algorithm artificially consolidates gaps into a solitary gap to build one composite cycle.

This algorithm has been criticized for the flooding of the whole network and the synchronization of the nodes of the cycle R, which requires more execution time and communication overhead.

Limitations:

In spite of the fact that the last two algorithms are as straightforward as the Funke's, they have normal shortcoming, specifically requiring a high node thickness, recognizing significantly more than actual boundary nodes and not demonstrating the distinguished boundary nodes definitively.

Table 1. A set of existing approaches for detecting boundary nodes.

Protocol	Location awareness	Additional information	Protocol type	Network flooding	Boundary recognition	2D or 3D
DBNS ^[2]	Yes	Coordinates	Distributed	Boundary nodes	Network + Holes	2D
SBNS ^[3]	Yes	Coordinates	Distributed	Boundary nodes	Network	2D
CDCHD ^[4]	No	Distances	Distributed	2-hop neighbours	Holes	2D
BRGT ^[5]	No	Angles	Centralized	The_ whole network	Network + Holes	2D
THD ^[6]	No	Distances	Centralized	The_ whole network	Network + Holes	2D
LCA ^[7]	No	Distances	Distributed	The_ whole network	Network + Holes	2D
CBC ^[8]	No	Distances	Distributed	The_ whole network	Network + Holes	2D
SDBR ^[9]	No	Distances	Distributed	3-hop neighbours	Network + Hole	2D
EBDG ^[10]	No	Angles	Distributed	1-hop neighbour	Network + Hole	2D
ABBD ^[11]	NO	Angles	Distributed	The whole Network	Network + Holes	2D

Chapter-3

SYSTEM DESIGN

3.1 System Development:

Assume that, there are expansive numbers of sensor hubs, say n hubs, are scattered in a 2-D geometric locale where adjacent nodes are associated with each other to shape a Wireless Sensor Network (WSN). For simplicity of introduction, we expect that the remote interchanges take after the UDG display.

- The WSN can be modelled as a graph $G(V, E)$ where the vertex set V represents nodes and the edge set E represents interaction links between set of nodes.
- Without loss of generality, we accept the chart $G(V, E)$ is associated. The implanting that connects the vertexes V to the real nodes in the WSN is meant by $p: V \rightarrow R^2$; the i -hop neighbour nodes of node n_x is indicated by $N_i(n_x)$; $| \cdot |$ processes the cardinality of a set.

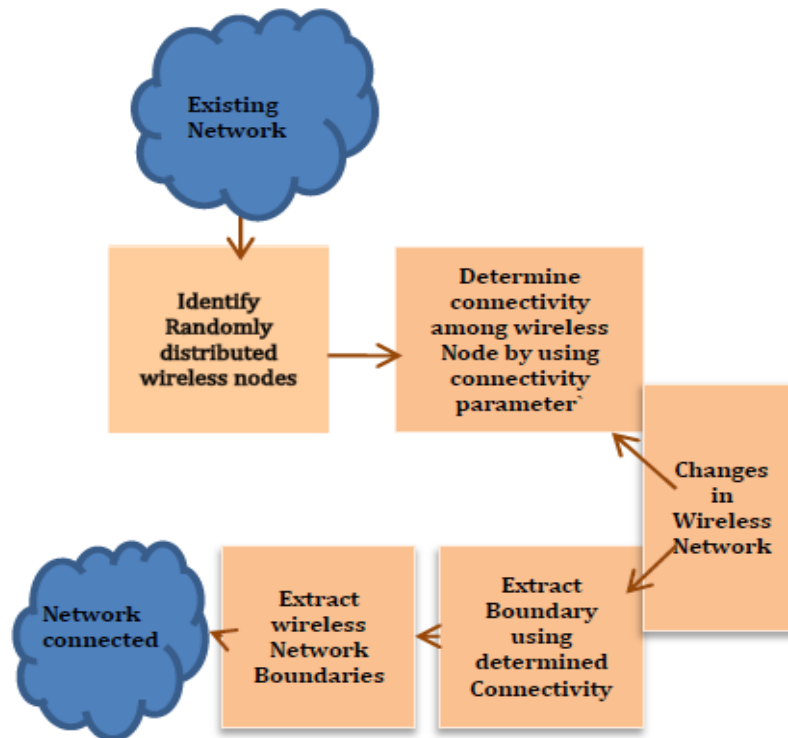


Fig3: The Proposed Approach For Boundary Extraction ^[3]

- The external boundary of the wireless sensor network is the minimum closed curve encasing the graph $G(V, E)$ however not encompassing whatever other closed curve that encases the graph $G(V, E)$.
- Likewise, an interior boundary of the wireless sensor network is the maximal closed curve encasing an opening of the graph $G(V, E)$ yet not being encompassed by whatever other closed curves that encase the like hole.
- Henceforth, an external (or internal) boundary of the WSN is really contained edges, vertices, and fragments of edges in the graph $G(V, E)$ as represented in Figure.
- For clarification, we call the nodes having a place with boundaries as genuine limit hubs and call alternate nodes as real inside nodes. In this manner, utilizing given nearby connectivity data, we will probably around find a limit node set, including genuine boundary nodes as well as some real inside nodes close WSN boundaries, and after that endeavor to lessen the measure of the boundary hub set and further to separate exact limit cycles from the found boundary node set.

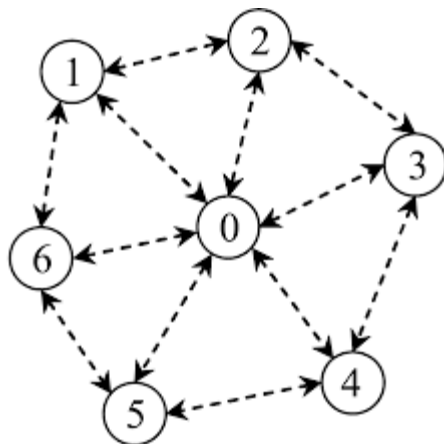


Fig 4. Boundary cycle corresponding to the boundary node

Analytical Model Development

Considering the below given geometric boundaries, we can readily define the boundary nodes as follows.

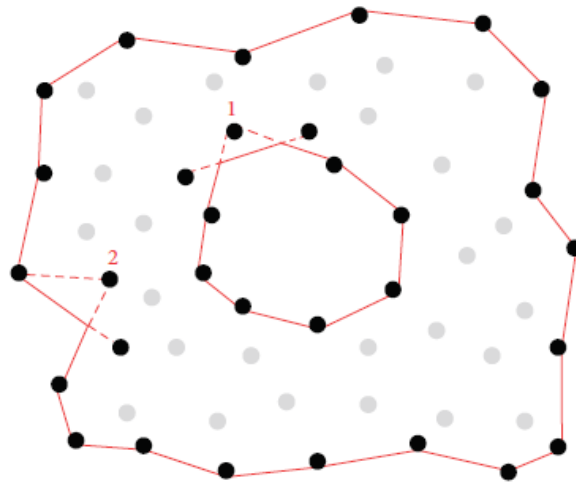


Fig5. The external and internal geometric boundaries (solid curves) of a wireless sensor network. The dashed lines denote the communication link.

Initially, nodes associated to boundaries are certainly boundary nodes.

Further, if any portion is associated with a boundary, edges and hubs having a place with this boundary won't have the capacity to shape a (shut) boundary cycle comparing to this boundary, so additional nodes must be joined as boundary nodes.

For instance, nodes 1 and 2 in Figure3 don't lie on any boundaries but on the other hand are viewed as boundary nodes to frame boundary cycles. Rather than boundary nodes, the various nodes are named inside nodes.

On a basic level, the assignment of boundary (and opening) identification is to distinguish boundary nodes and after that develop relating boundary cycles.

3.2 Platform used for simulation

Contiki:

- It is an operating system for networked, memory-compelled frameworks with an attention on the low-power wireless Internet of Things (IoT) gadgets. Surviving utilizations for Contiki incorporate system for street lighting, sound checking for smart cities communities, radiation observing, and cautions.
- Contiki picked up the popularity because of its inherent TCP/IP stack and lightweight preemptive scheduling over occasion driven bit which is an extremely spurring highlight for IoT.
- Contiki gives multitasking and an inherent Internet Protocol Suite (TCP/IP stack), yet requires just around 10 kb of RAM (random access memory) and 30 kb of ROM (read-only memory). Whole system, including a GUI, needs around 30 kb of RAM.
- Contiki is intended to keep running on kinds of equipment gadgets which are extremely compelled in memory, energy, computation power, and interactive data transmission. A classic Contiki system has memory of the order of kbs, a power budget of the order of milliwatts, computing speed estimated in megaHertz, and correspondence data transfer capacity on the order of many kbps. Such frameworks incorporate numerous sorts of embedded systems.
- Contiki gives three system components: the TCP/IP stack which gives IPv4 organizing, the uIPv6 stack, which gives IPv6 organizing, and the Rime stack, which is an arrangement of custom lightweight networking protocols for low-power wireless network.

Simulation:

- The Contiki framework incorporates a network simulator called Cooja, that recreates network of Contiki nodes.
- The nodes may have a place with both of three classes: imitated nodes, where the whole equipment of every node is emulated.
- Cooja nodes, where the Contiki code for the node is arranged for and executed on the simulation host.
- Java nodes, where the conduct of the hub must be re-implemented as a Java class. One Cooja simulation might contain a blend of nodes from any of the three predefined classes. Emulated nodes can likewise be utilized to incorporate non-Contiki nodes in a simulated network.

Programming Model

- To perform proficiently on little memory frameworks, the Contiki programming model depends on protothreads.
- Protothreads are collectively scheduled.
- A Contiki process must always provide control back to the kernel at regular intervals of time.

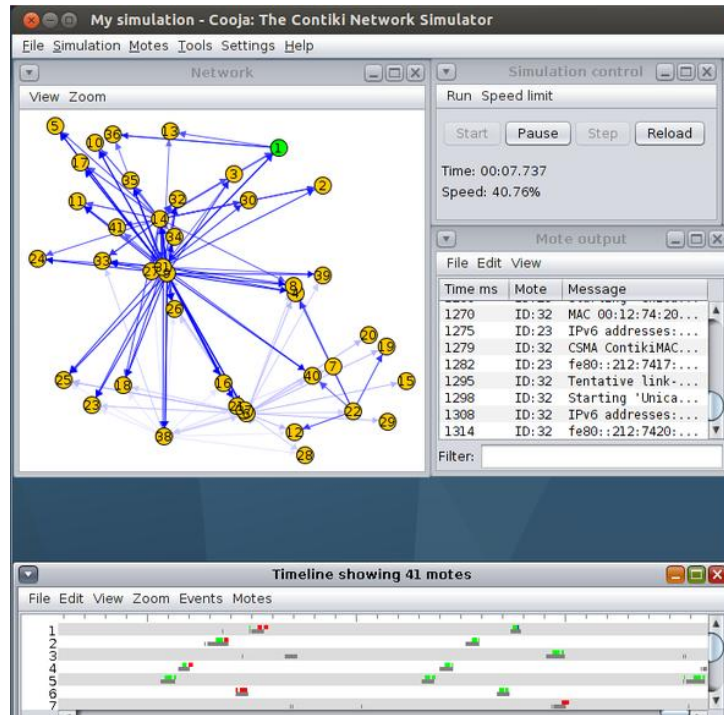


Fig6. Contiki operating system an IPv6 routing protocol on 41 nodes in the Cooja Contiki network simulator

3.3 ALGORITHM

The essential thought of the proposed algorithm is right off the bat recognizing a set of questionable boundary nodes, at that point clarifying them, and ultimately searching them for actual boundary cycles.

The outline is listed below.

- Every node, represented by n_0 , keeps up a neighbourhood variable, signified by f_i , which is introduced in light of its 2-hop iso-contour separated from the subgraph of $G(V,E)$.

- As per the estimation of f_t , it is generally chosen whether node n_0 is a limit node: $f_t=1$ demonstrates that node n_0 is probably going to be a limit node and is in this way named as a suspected boundary node, while $f_t=0$ shows that node n_0 is recognized to be an inside node, as appeared in Figure.

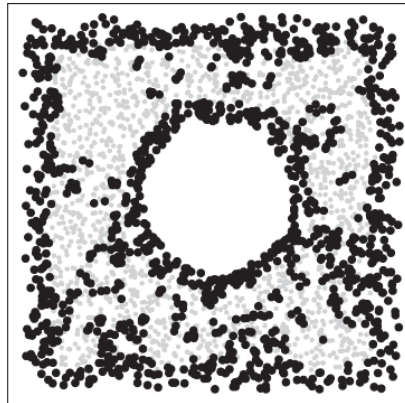


Fig 7. Decision on each node after executing INITIALIZE(n_0) function

- In the wake of trading the estimations of f_t between sets of neighbouring nodes, each presumed boundary node locally plays out a erosion task to refresh the decision on the value of its f_t as indicated by the segment of distinguished inside nodes in its 1-hop neighbour nodes, to be specific, clarifying the rough decision made in the previous step, as delineated in Figure.

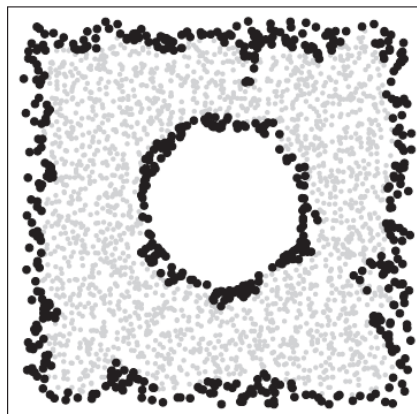


Fig 8. Value of f_t after executing the erosion operation

- Each speculated limit node, say node n_0 , keeps up another local variable, indicated by f_d , to separate itself from the other suspected boundary nodes.
- The suspected boundary nodes helpfully scan for boundary cycles in light of their estimations of f_d . Therefore, a set of tight boundary cycles comparing to internal and external boundaries are acquired, as appeared in Figure.

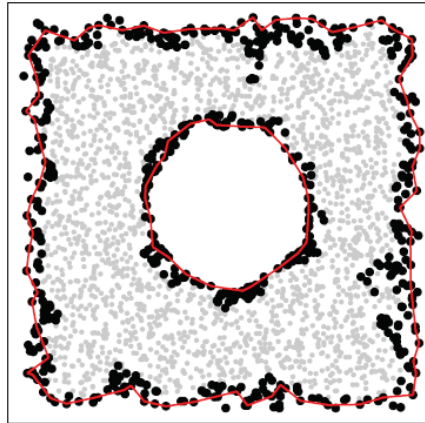


Fig 9. Tight inner and outer boundaries

3.3.1 Initializing the local variable f_t

- Here, a node say n_0 load the value its local f_t using the return value of the function, INITIALIZE (n_0).
- As per the underlying estimation of f_t , a rough judgment regarding whether node n_0 is probably going to be a boundary hub is made, so the nodes in the WSN are assembled into a speculated boundary node set with $f_t= 1$ and an inside node set with $f_t= 0$.
- The function INITIALIZE (n_0) distinguishes node n_0 as a presumed limit hub if a closed way encasing hub n_0 isn't found in its 2-hop iso-contour; for effortlessness, we state this condition as the CLOSED-CYCLE condition.

- In the usage of the function INITIALIZE (n_0), the parameter L_{th} indicates the insignificant dimension of all conceivable closed ways that fulfill the CLOSED-CYCLE condition.
- As per the CLOSED-CYCLE condition, hubs with $f_t=0$ are certainly genuine inside hubs, however hubs with $f_t=1$ are just plausible to be real limit hubs and in this way are named as presumed boundary nodes.
- One can promptly derive that the underlying estimations of f_t for genuine limit hubs are dependably 1, while the underlying qualities for real inside hubs take either 0 or 1, suggesting that a segment of real inside hubs are disgracefully found as presumed limit hubs, particularly in low-thickness WSNs. Henceforth, surprisingly the CLOSED-CYCLE condition is simply adequate.

3.3.2 Executing the heuristic operation

- After the instatement of f_t at every node, the quantity of associated boundary recognition is dependably times with that of genuine limit node, which forces unfavourable effects on limit discovery.
- Perceptions show that a major of 1-hop neighbour nodes of real boundary nodes have $f_t=1$, yet some genuine inside nodes, which are contiguous real boundary nodes or which have moderately scanty neighbourhoods, frequently end up having $f_t=1$, and a dominant part of 1-hop neighbour hubs of these hubs have $f_t=0$.
- In perspective of this component, for any hub with $f_t=1$, if the proportion of its 1-hop neighbour hubs with $f_t=0$ is over a limit, signified by E_{th} , this hub is profoundly plausible to be a genuine inside hub, so its f_t ought to be set to 0.
- We plan an erosion task to understand the refinement of f_t at suspected boundary nodes.

3.3.3 Finding boundary cycles

To introduce our outcome definitively, we endeavor to acquire exact boundary Cycles relating to both inward and external boundary of the wireless sensor networks.

3.4 PRACTICAL ISSUES

Working with Realistic Channel Models

The algorithm is proposed in the idealized UDG show.

In spite of the fact that the UDG demonstrate is basic, commonsense remote interchanges can never fulfill it, and, henceforth, more reasonable channel models, for instance, the QUDG show, are produced.

With the QUDG show, it is conceivable that the 2-hop iso contours of some real limit hubs can frame closed cycles in order to fulfill the CLOSED-CYCLE condition, and, subsequently, they are erroneously recognized as inside hubs, which corrupts the nature of the proposed algorithm.

Given that a real boundary node is recognized as an inside hub due to $f_t = 0$, it can be watched that the lion's share of its 1-hop neighbour hubs are distinguished as speculated limit hubs with $f_t = 1$, however given a genuine inside hub with $f_t = 0$, it is of little likelihood that the lion's share of its 1-hop neighbour node are distinguished as presumed boundary nodes.

It can be presumed that, for any recognized inside hub with $f_t = 0$, if the proportion of the quantity of its 1-hop neighbour nodes with $f_t = 0$ to the aggregate number of its 1-hop neighbour nodes is underneath a limit, it will probably be a real neighbour node and its f_t ought to be set to 1.

Chapter-4

PERFORMANCE ANALYSIS

4.1 Proposed Approach

The proposed approach can be divided into three phases:

- 1) In the first phase we execute a function INITIALIZE (n_0) on each node of the network. The value returned by this function initializes the value of f_i .
- 2) Second phase comprises of heuristic operation which refines the set of suspected boundary nodes, by computing E_{th} for each suspected boundary node.
- 3) In the last phase, we present our result in a meaningful way and look for the CLOSED-CYCLE condition to find the tight inner and outer boundaries.

4.2 Algorithm for the INITIALIZE (n_0) function

The algorithm gives the set of suspected boundary node and this is the first step of the whole algorithm proposed by [1]

Initialize(n_0)

1. Construct $G_2(V_2, E_2)$ based on $G(V, E)$, where $V_2 = \{v \mid v \in N_2(n_0)\}$ &
 $E_2 = \{e_{ij} \mid i, j \in V_2 \wedge e_{ij} \in E\}$
2. Index nodes in V_2 by $1, 2, \dots, |V_2|$
3. Roadmap \leftarrow Integer array of size $|V_2|$ with all zeros.
4. $V_c \leftarrow V_2$
5. Expanding_path \leftarrow Empty stack
6. While $|V_c| > 0$ do
7. $n_r \leftarrow$ An arbitrary node in V_c
8. $V_c \leftarrow V_c - n_r$
9. push(expanding_path, n_r)
10. Roadmap(n_r) $\leftarrow 1$
11. step $\leftarrow 2$
12. While true do
13. Candidate_nodes $\leftarrow V_2 \cap N_1(n_r)$
14. $n_m \leftarrow$ The node in candidate_nodes and satisfying $\text{Roadmap}(n_m) \leq \text{Roadmap}(\text{candidate_nodes})$
15. If $\text{Roadmap}(n_r) - \text{Roadmap}(n_m) > L$ then
16. return 0;
17. endif
18. Candidate_nodes $\leftarrow V_c \cap N_1(n_r)$
19. if $| \text{Candidate_nodes} | > 0$ then
20. $V_c \leftarrow (V_c - \text{Candidate_nodes})$
21. Roadmap(Candidate_nodes) \leftarrow step
22. step \leftarrow step + 1
23. $n_s \leftarrow$ An arbitrary node in Candidate_nodes
24. push(expanding_path, n_s)
25. else if step > 2 then
26. step \leftarrow step - 1
27. recovery_nodes = $\{n \mid \text{Roadmap}(n) = \text{step}\}$
28. Roadmap(recovery_nodes) $\leftarrow 0$
29. $V_c \leftarrow (V_c + \text{recovery_nodes} - n_r)$
30. POP(expanding_path)
31. $n_r \leftarrow$ The top node in expanding_path.
32. else
33. break;
34. endif
35. end while
36. return 1
37. end while

Fig10. INITIALIZE(n_0) algorithm

4.3 Flowchart of INITIALIZE function

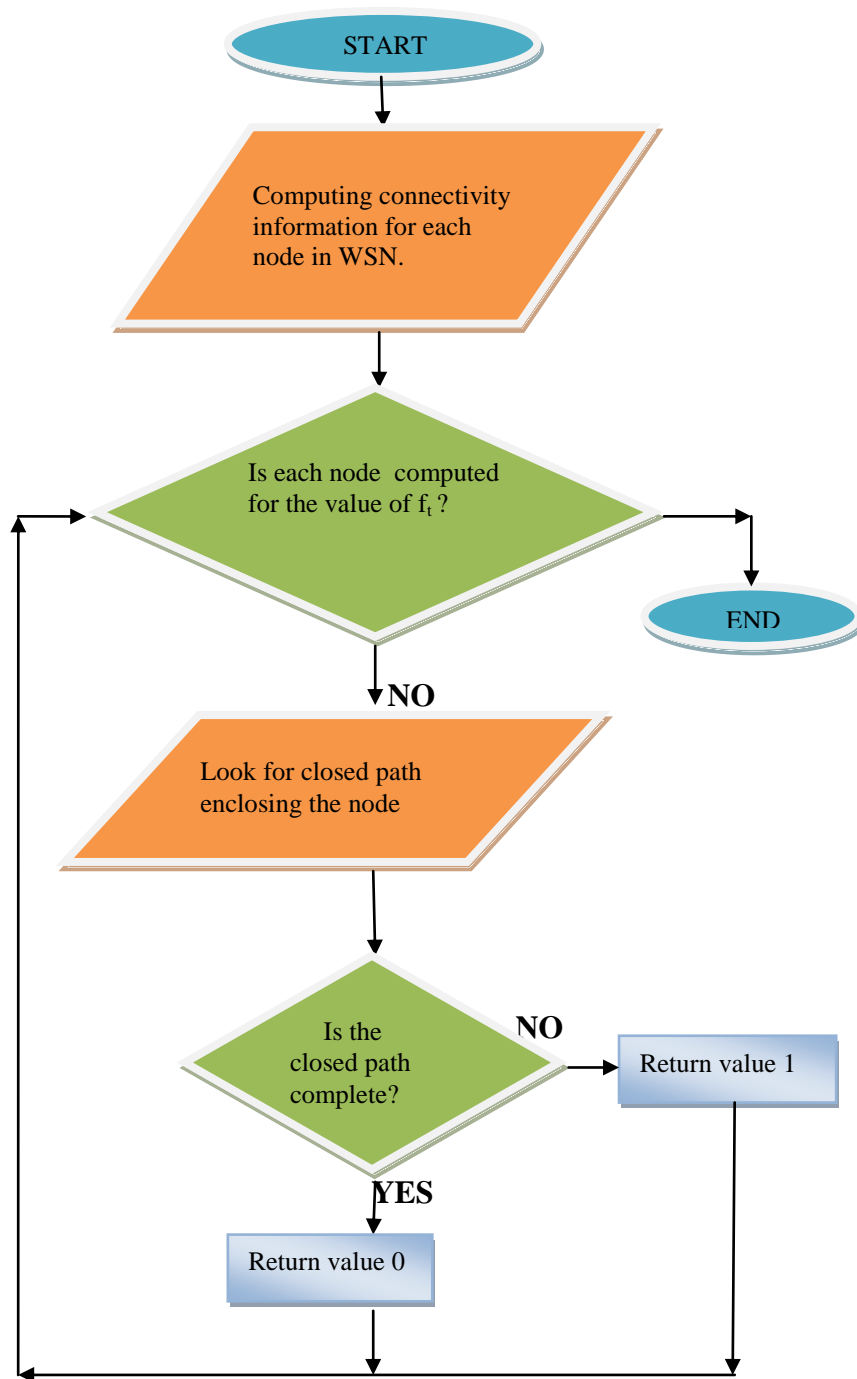


Fig11: Flow chart to depict the working of INITIALIZE (n_0) function

4.4 Practice Run of the INITIALIZE function

We carry out simulations in different scenarios, with the goal to evaluate the performance of the proposed algorithm with respect to the node density and distribution, the wireless channel model.

First, we dry run the INITIALIZE function which will initiate the variable f_i for each node in the wireless sensor network.

For the dry run we consider a network of wireless sensor nodes, the network is sparse. (Shown in the figure).

We need to find out 1 and 2-hop neighbour information for each node.

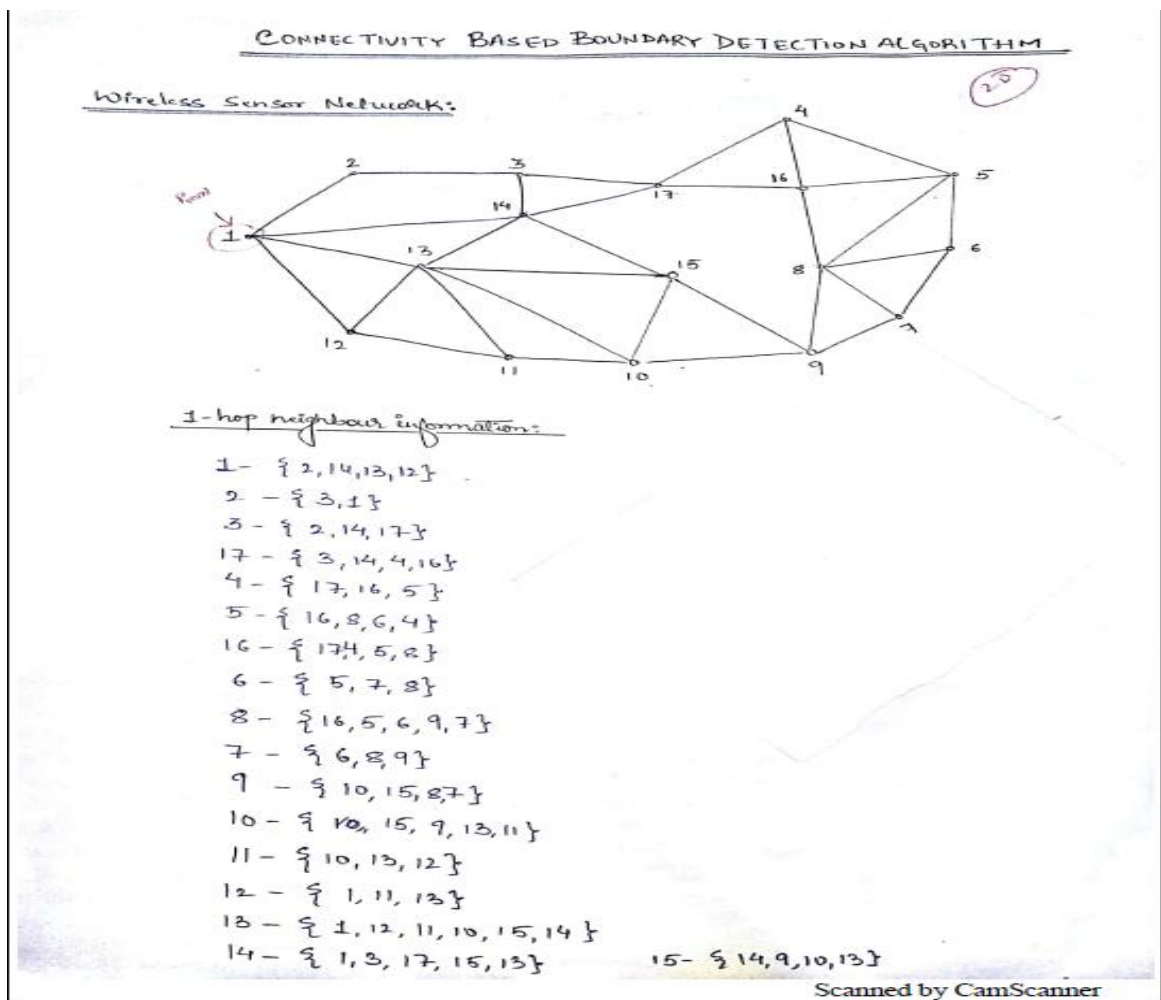


Fig 12. Deployed wireless sensor network and 1-hop neighbour set for each node

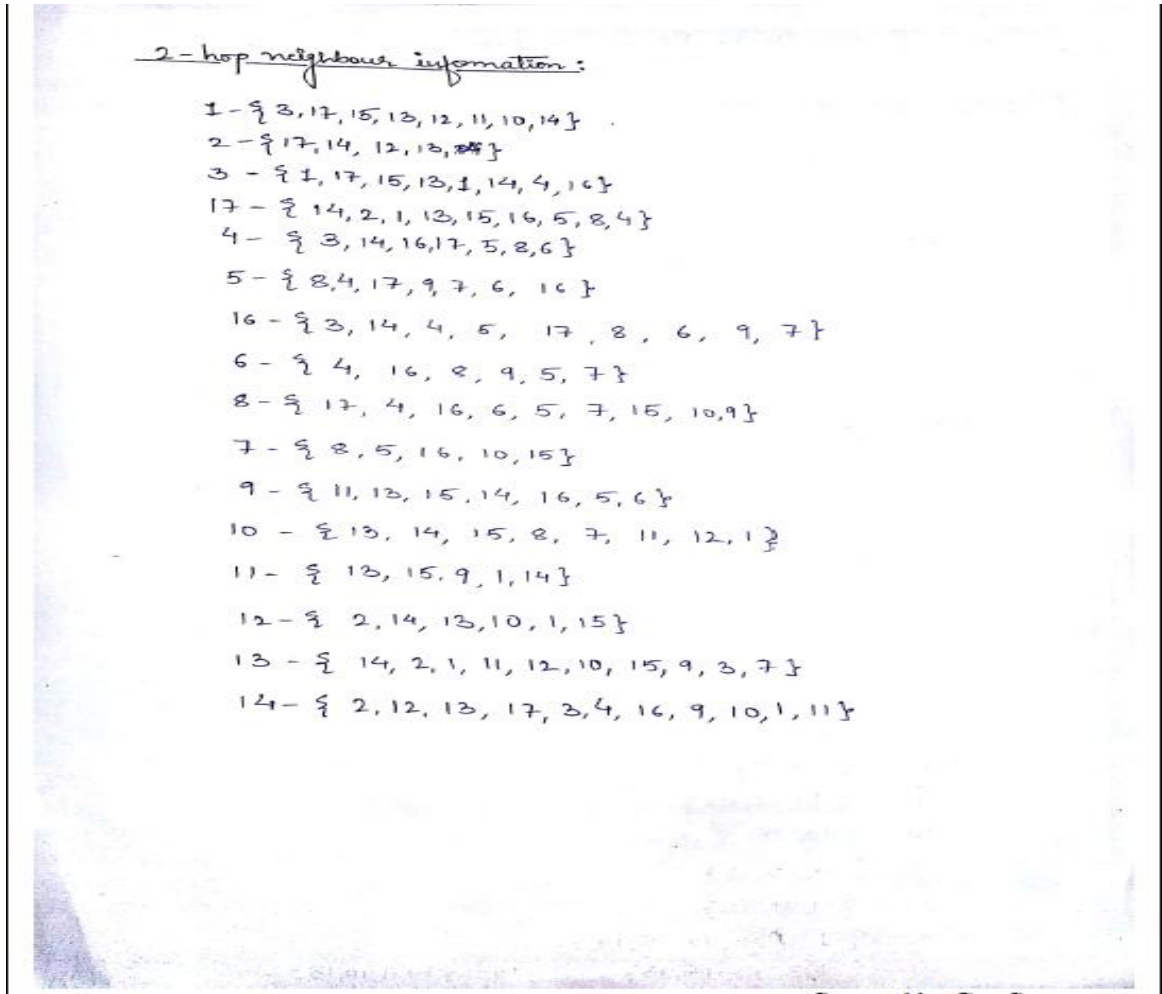


Fig 13: 2-hop neighbour set of each node in the deployed network

After gathering the neighbour information we practice run the INITIALIZE function code on two nodes of the deployed network that are node 13(interior) and 5(exterior).

The basic idea is to check whether the 2-hop isocontour for a node is broken or is completely surrounded or covered by the network.

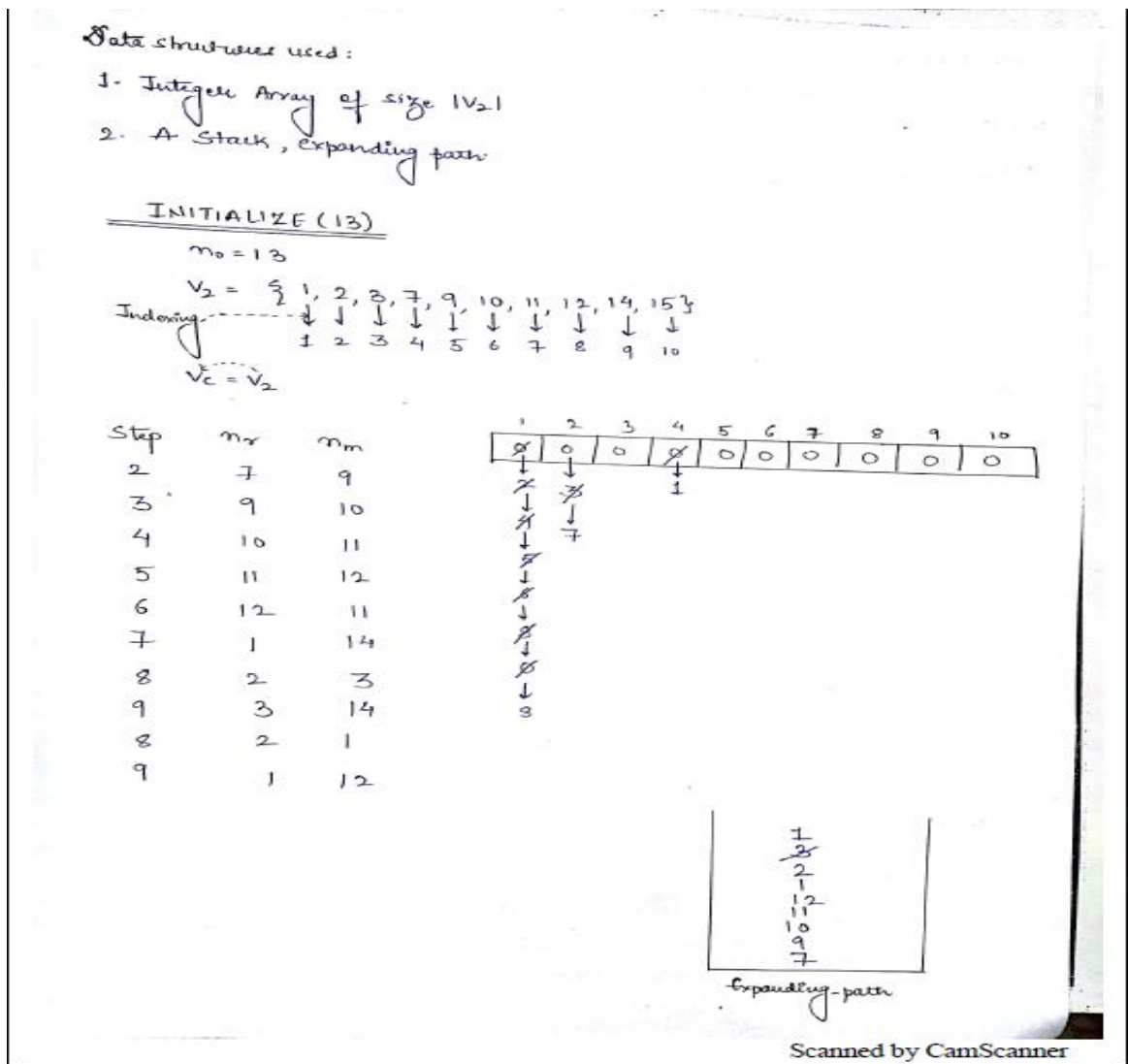


Fig 14. Practice run of the INITIALIZE function on node numbered 13

Value of the variable n_r is changing after every iteration.

Each new value of n_r is used to find the candidate nodes set (means nodes which are common in the 2-hop neighbour set of the node on which the code is executed and the nodes which appear in the 1-hop neighbour set of the node having value n_r), so that the decision on the node, whether its interior or boundary node can be made. Value of n_r is also used to determine the recover node set which adds into the set V_c whose value is the check condition for the while loop. As soon as the decision is made for the node, the variable f_i is initialised and the program ends.


```

while (Vc > 0)
    Vc = {1, 2, 3, 9, 10, 11, 12, 14, 15}
while (1)
    1-time: candidate_nodes = V2 ∩ N1(7) = V2 ∩ {6, 8, 9}
           = {9}
           candidate_nodes = Vc ∩ N1(7)
           = {9}
           Vc = {1, 2, 3, 10, 11, 12, 14, 15}
           nr ← 9
    2-time: candidate_nodes = V2 ∩ N1(9) = V2 ∩ {15, 8, 13, 14} ∩ {10, 15, 8, 7}
           = {10, 15, 8, 7}
           candidate_nodes = Vc ∩ N1(9)
           = {10, 15}
           Vc = {1, 2, 3, 11, 12, 14}
           nr ← 10
    3-time: candidate_nodes = V2 ∩ N1(10) = V2 ∩ {15, 9, 13, 11}
           = {15, 9, 11}
           candidate_nodes = Vc ∩ N1(10)
           = {11}
           Vc = {1, 2, 3, 12, 14}
           nr ← 11
    4-time: candidate_nodes = V2 ∩ N1(11) = V2 ∩ {10, 13, 12}
           = {10, 12}
           candidate_nodes = Vc ∩ N1(11)
           = {12}
           Vc = {1, 2, 3, 14}
           nr ← 12

```

Fig 15. Practice run of the code showing the values of variables at each iteration

$$\begin{aligned} \text{5-time: candidate nodes} &= V_2 \cap N_1(12) = V_2 \cap \{1, 11, 13\} \\ &= \{1, 11\} \end{aligned}$$

$$\begin{aligned} \text{Candidate nodes} &= V_C \cap N_1(12) \\ &= \{1, 2, 3, 14\} \cap \{1, 11, 13\} \\ &= \{1\} \end{aligned}$$

$$V_C = \{2, 3, 14\}$$

$$\begin{aligned} \text{6-time: candidate node} &= V_2 \cap N_1(1) = V_2 \cap \{2, 14, 13, 12\} \\ &= \{2, 14, 12\} \end{aligned}$$

$$\begin{aligned} \text{Candidate nodes} &= V_C \cap N_1(1) \\ &= \{2, 14\} \end{aligned}$$

$$V_C = \{3\}$$

$$nr \leftarrow 2$$

$$\begin{aligned} \text{7-time: candidate node} &= V_2 \cap N_1(12) = V_2 \cap \{3, 1\} \\ &= \{1\} \end{aligned}$$

$$\begin{aligned} \text{Candidate nodes} &= V_C \cap N_1(2) \\ &= \{3\} \end{aligned}$$

$$V_C = \emptyset$$

$$nr \leftarrow 3$$

$$\begin{aligned} \text{8-time: candidate node} &= V_2 \cap N_1(3) = V_2 \cap \{2, 14, 17\} \\ &= \{2, 14\} \end{aligned}$$

$$\text{Candidate nodes} = \emptyset$$

$$\text{Success nodes} = \{1\}$$

$$V_C = \{1\}$$

$$nr \leftarrow 2$$

$$\begin{aligned} \text{9-time: candidate nodes} &= V_2 \cap N_1(2) \\ &= \{3, 1\} \cap V_2 \\ &= \{1, 3\} \end{aligned}$$

$$\begin{aligned} \text{Candidate nodes} &= V_C \cap N_1(2) \\ &= \{1\} \cap \{1, 3\} \\ &= \{1\} \end{aligned}$$

$$V_C = \emptyset$$

$$nr \leftarrow 1$$

Fig 16. Practice run on the node numbered 13

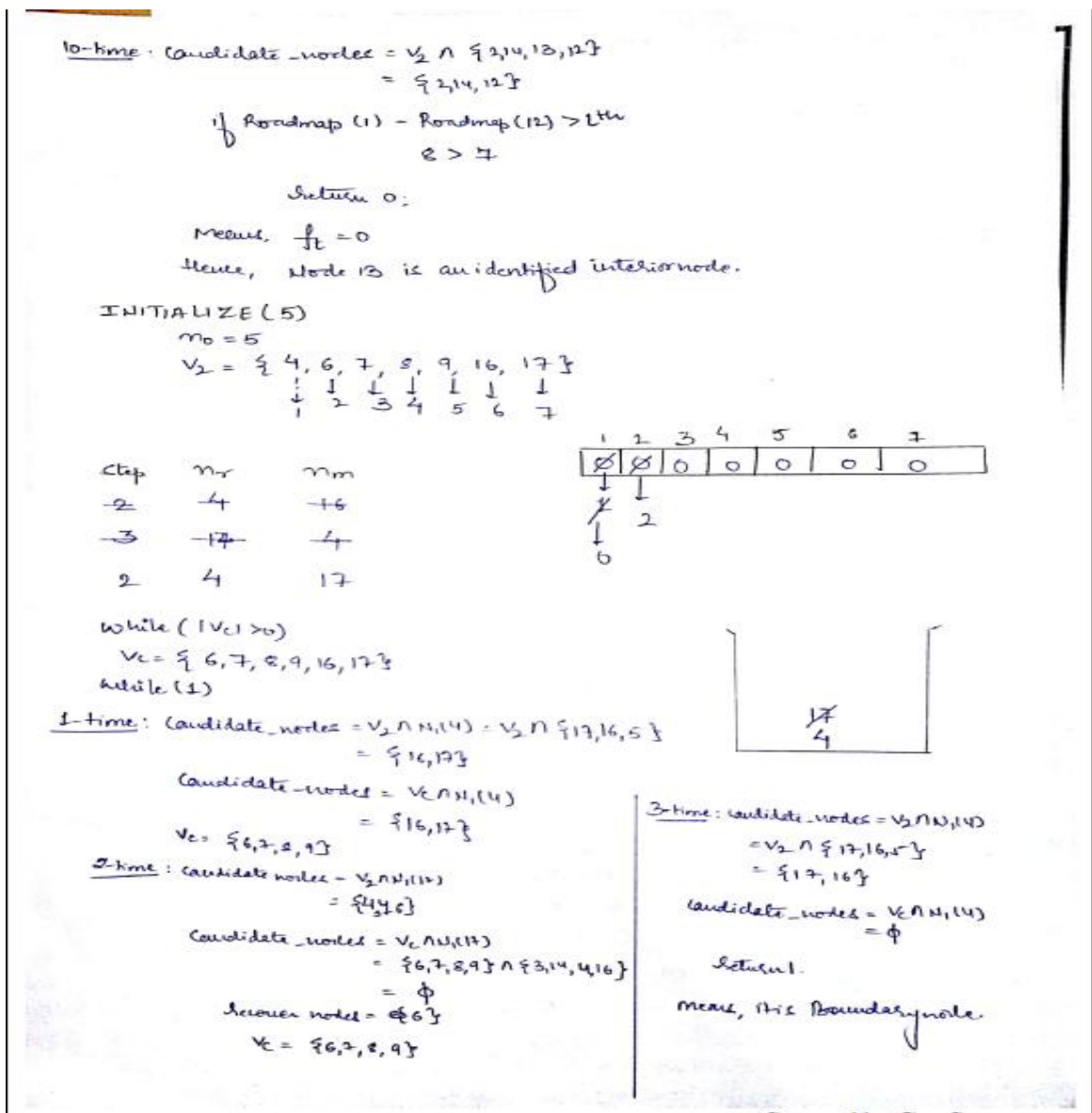
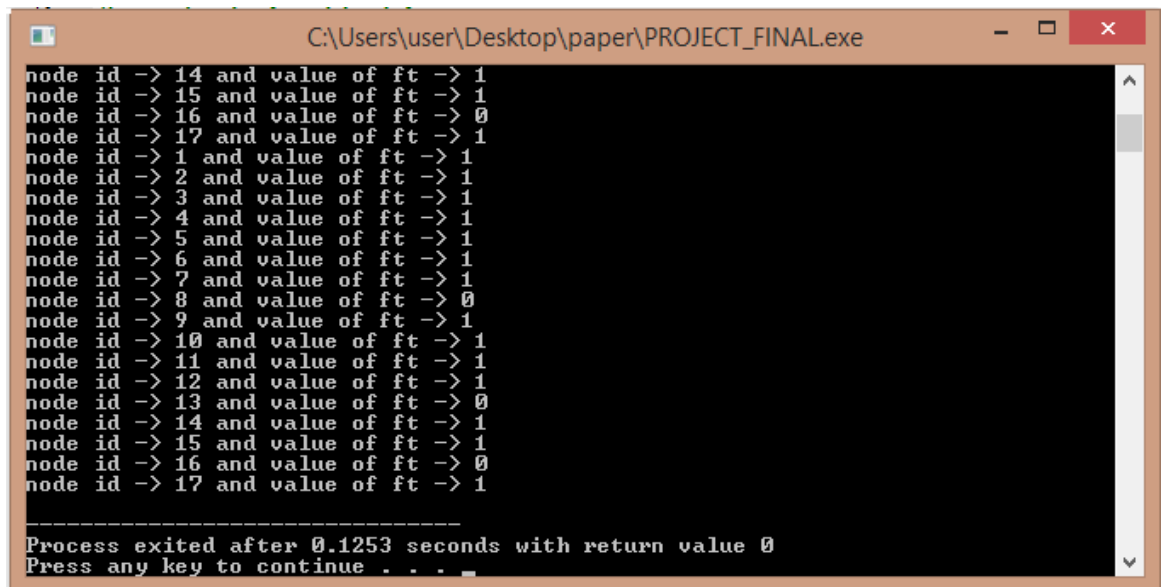


Fig 17. Node numbered 13 is identified as actual interior node and node numbered 15 is identified as suspected boundary node

From the practice run we can infer that the results or the decision given by the INITIALIZE function is in accordance with the deployed network. Here it should be noted that the any node identified as boundary node is only suspected to be boundary node and needs a refinement operation however nodes identified as interior nodes are the actual interior nodes and need no refinement decision or operation.

To obtain the result or value of f_i for each node of the deployed network we run the code in the c (programming language) compiler i.e. DevBlocks and see the results in the console output.



```
C:\Users\user\Desktop\paper\PROJECT_FINAL.exe
node id -> 14 and value of ft -> 1
node id -> 15 and value of ft -> 1
node id -> 16 and value of ft -> 0
node id -> 17 and value of ft -> 1
node id -> 1 and value of ft -> 1
node id -> 2 and value of ft -> 1
node id -> 3 and value of ft -> 1
node id -> 4 and value of ft -> 1
node id -> 5 and value of ft -> 1
node id -> 6 and value of ft -> 1
node id -> 7 and value of ft -> 1
node id -> 8 and value of ft -> 0
node id -> 9 and value of ft -> 1
node id -> 10 and value of ft -> 1
node id -> 11 and value of ft -> 1
node id -> 12 and value of ft -> 1
node id -> 13 and value of ft -> 0
node id -> 14 and value of ft -> 1
node id -> 15 and value of ft -> 1
node id -> 16 and value of ft -> 0
node id -> 17 and value of ft -> 1
-----
Process exited after 0.1253 seconds with return value 0
Press any key to continue . . .
```

Fig 18. Console output for the INITIALIZE function

From the results we can infer that the INITIALIZE function identified node numbered 14 and 15 as suspected boundary node although they are actual interior nodes. This anomaly is because, majority of 1-hop neighbour of these two nodes are boundary nodes.

To refine the decision on these nodes we execute the erosion function. We check and verify the results for each node after the execution of erosion or heuristic operation.

```

C:\Users\user\Desktop\paper\PROJECT_FINAL.exe
node id -> 14 and value of ft -> 1
node id -> 15 and value of ft -> 1
node id -> 16 and value of ft -> 0
node id -> 17 and value of ft -> 1
node id -> 1 and value of ft -> 1
node id -> 2 and value of ft -> 1
node id -> 3 and value of ft -> 1
node id -> 4 and value of ft -> 1
node id -> 5 and value of ft -> 1
node id -> 6 and value of ft -> 1
node id -> 7 and value of ft -> 1
node id -> 8 and value of ft -> 0
node id -> 9 and value of ft -> 1
node id -> 10 and value of ft -> 1
node id -> 11 and value of ft -> 1
node id -> 12 and value of ft -> 1
node id -> 13 and value of ft -> 0
node id -> 14 and value of ft -> 1
node id -> 15 and value of ft -> 1
node id -> 16 and value of ft -> 0
node id -> 17 and value of ft -> 1
-----
Process exited after 0.1253 seconds with return value 0
Press any key to continue . . .

```

Fig 19. console output after the execution of erosion function

From the console output we can infer that still the decision on the node numbered 14 and 15 does not match with the deployed network.

4.5 Analysis for the sparse network

From the results obtained after the compilation of the INITIALIZE and erosion function we deduce that the algorithm did not provide the correct output for the node numbered 14 and 15.

Out of the total 17 nodes deployed as the wireless sensor node, the algorithm gave incorrect decision for 2 nodes (node numbered 14 and node numbered 15)

According to this network the accuracy of the algorithm for a sparse network is:

$$\text{Accuracy} = \frac{\text{Number of nodes correctly detected}}{\text{Total number of nodes}} * 100 \%$$

$$\text{Accuracy} = (15 / 17) * 100 \% = 88 \% \text{ (approx)}$$

4.6 Implementation in Cooja simulator

Implementation in cooja simulator aim at the recognition of Boundary nodes in the deployed network and making the sensor aware about the position (at the boundary) by blinking the respective LED of the nodes.

Following are the screenshots of the implementation in cooja simulator.

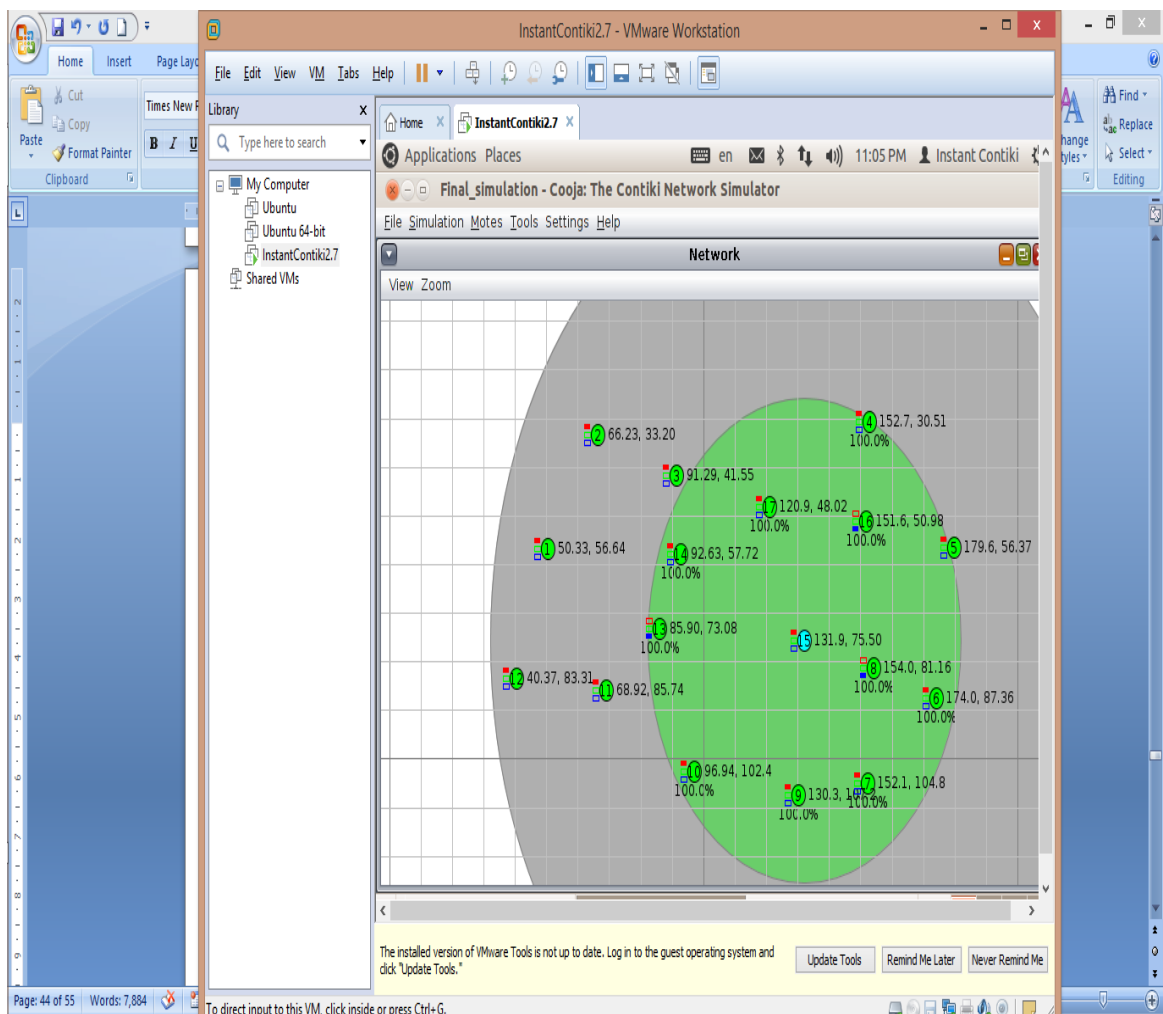


Fig 20. Implementation in cooja simulator, nodes with LED red on are on boundary whereas the nodes with LED Blue are interior node

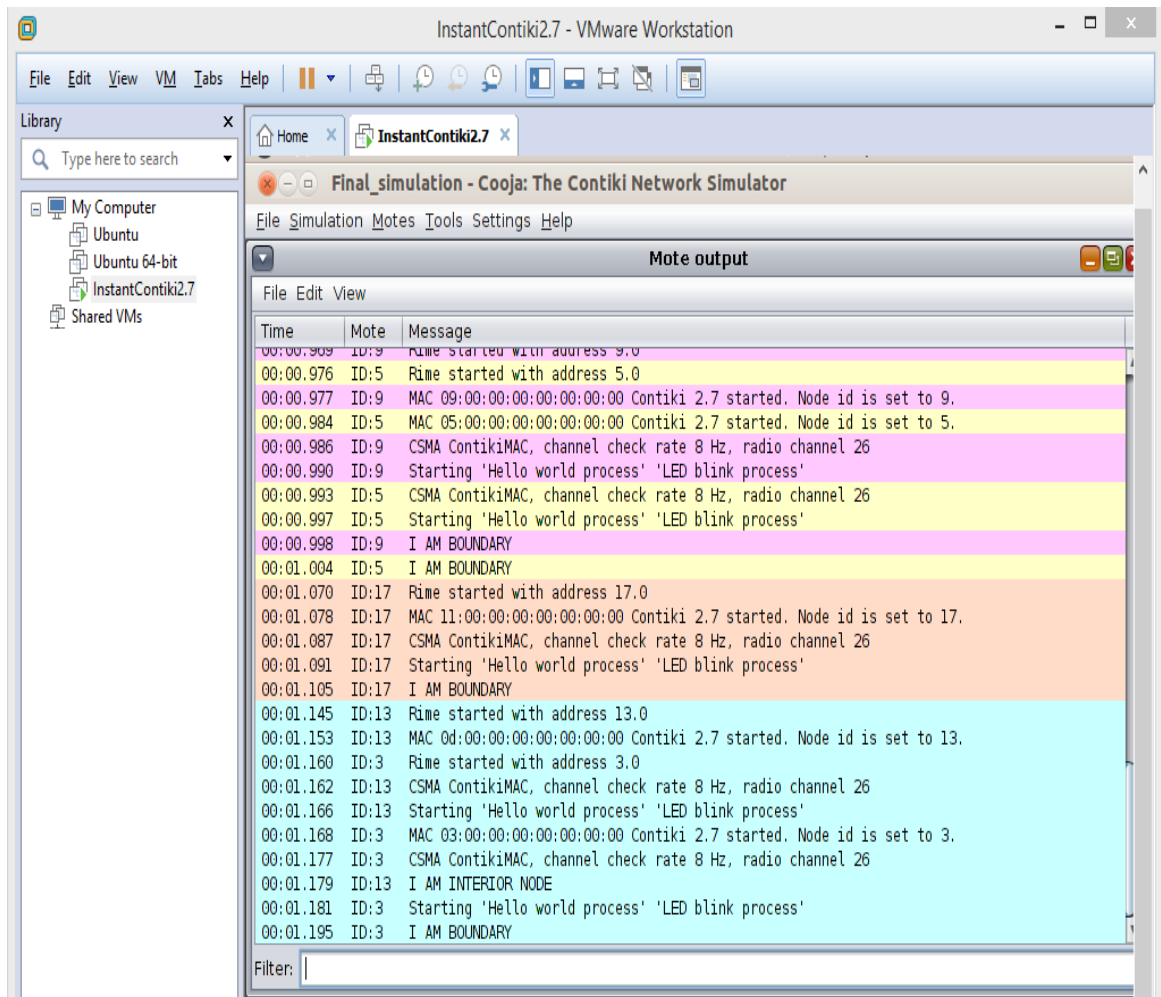


Fig 21. Mote output window, node with id 5,9,17 and 13 are boundary node

The deployment of wireless sensor nodes in real world can be realized with the help of simulators. The simulators depict the deployment of sensor nodes in a real life network and allow us to compute various parameters and characteristics of the nodes.

Here we are using Cooja simulator which relies on contiki operating system for its execution.

Before starting the simulation we need to store our code in the directory of contiki and after that the code must be first compiled on the terminal in order to make it function.

The following steps have been performed for the above simulation to occur:

Step1: A new simulation is created with 17 nodes of mote type “sky”.

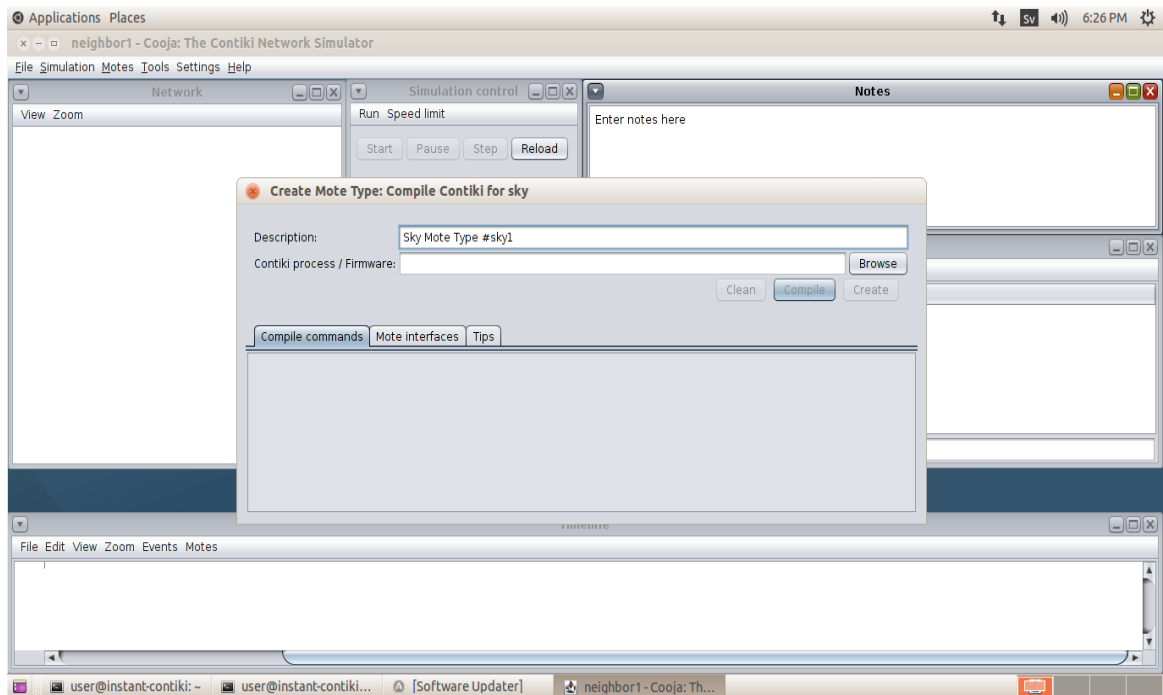


Fig.22: Creating Mote Of Sky Types

Step2: The firmware to compile is Node-recognition.c

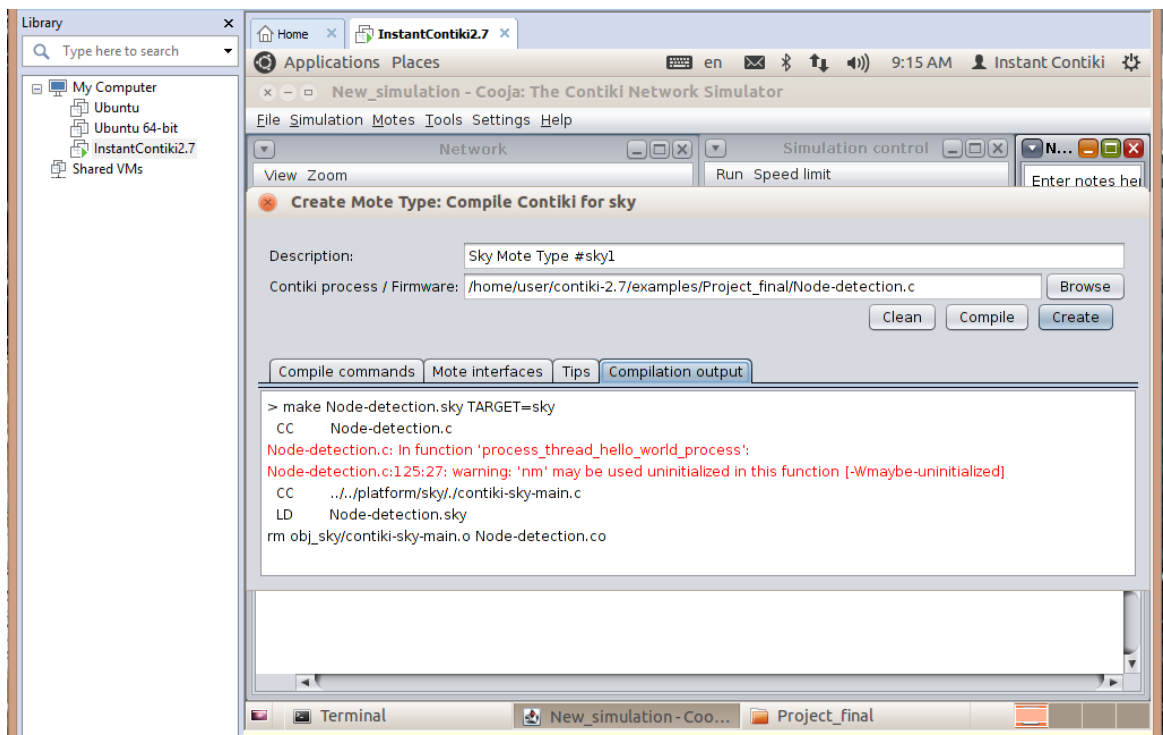


Fig.23: Firmware selected

Step3: After the code in c is compiled successfully a sky file is created. On clicking the create button the required network is created.

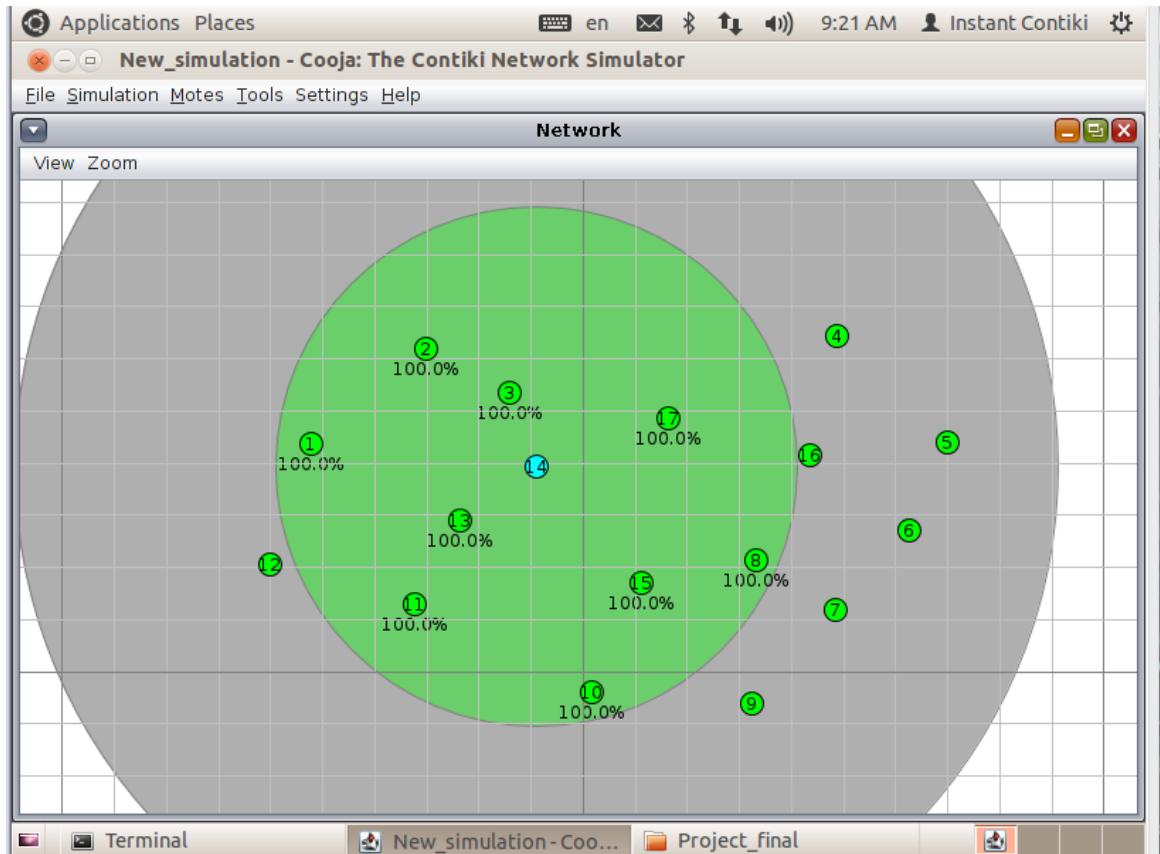


Fig 24: Network created

Step4: To make identification and simulation of sensor nodes much easier we enable the mote type, mote ID's, Radio traffic and Radio Environment from the view option.

Step5: Now we start the simulation by clicking on the start button from the simulation window.

Step6: we can also adjust the speed of the simulation, to make the simulation run faster than the real environment deployment.

Step7: After the code is successfully executed for each node, we can see the decision on the nodes.

Step8: To view the decision on any node we can do so by filtering the node by specifying the mote ID in the mote output window(here it is shown for mote D1).

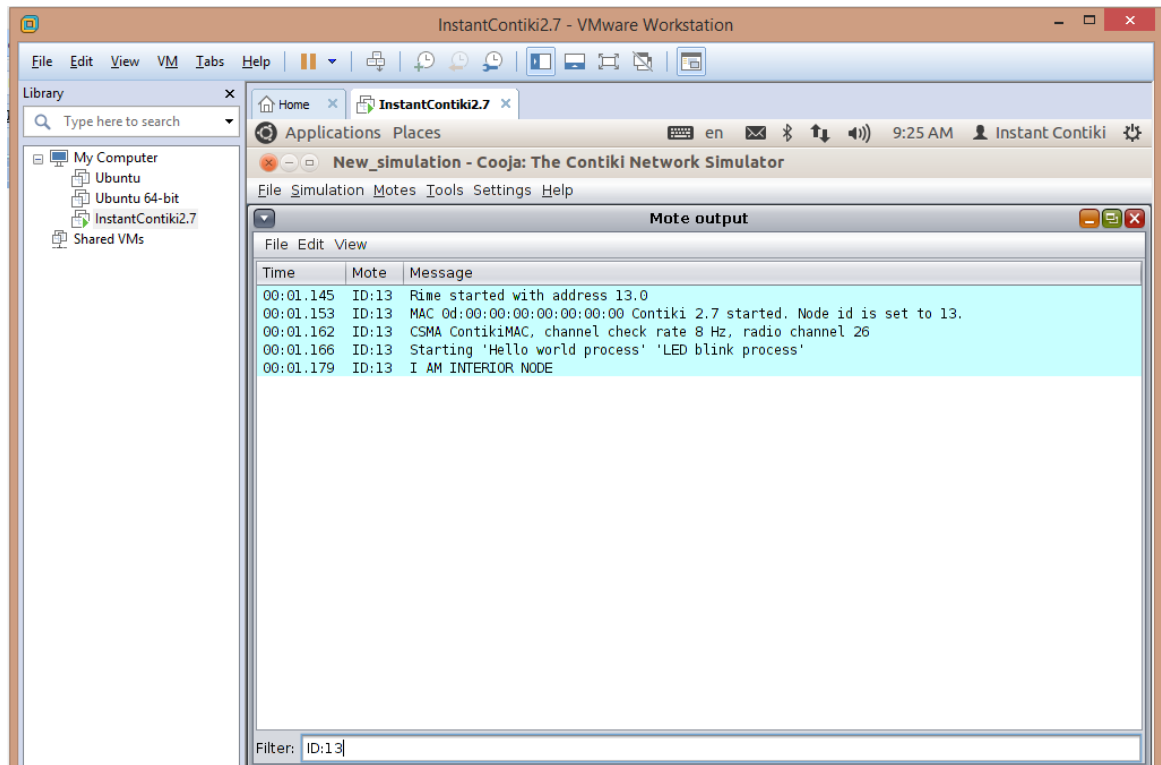


Fig 25: Decision on node with ID:13

4.7 Simulation on dense network

A second simulation is carried out, with aim as previous i.e. detecting boundary nodes and performing erosion operation on it.

The simulation is carried on 58 nodes which are deployed randomly in the network.

After collecting the information for each node's 1 and 2 hop neighbour we compile the code for boundary recognition and make it run on each node.

4.8 Implementation in cooja

Implementation in cooja simulator aim at the recognition of Boundary nodes in the deployed network and making the sensor aware about the position (at the boundary) by blinking the respective LED of the nodes.

Following are the screenshots of the implementation in cooja simulator.

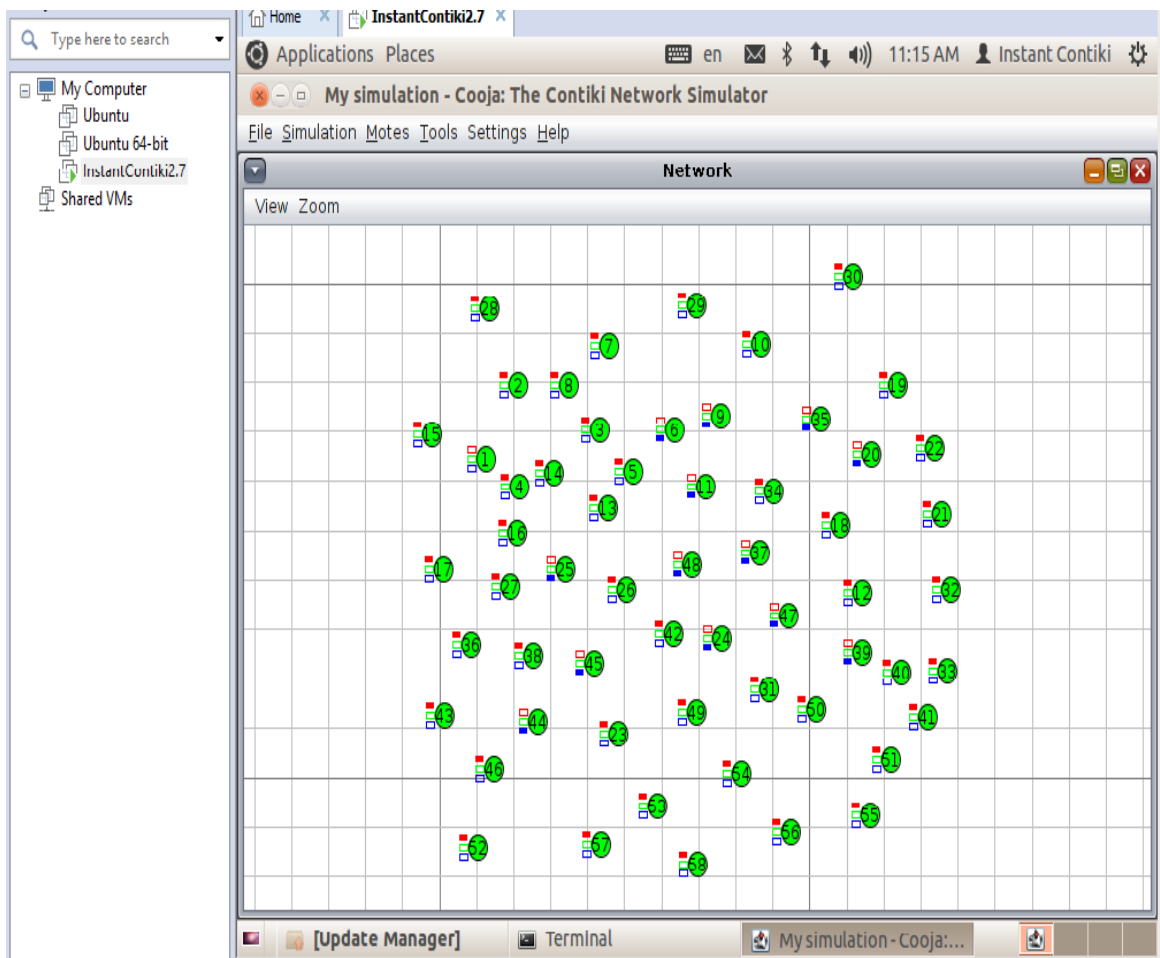


Fig 26. Implementation in cooja simulator, nodes with LED red on are on boundary whereas the nodes with LED Blue are interior node

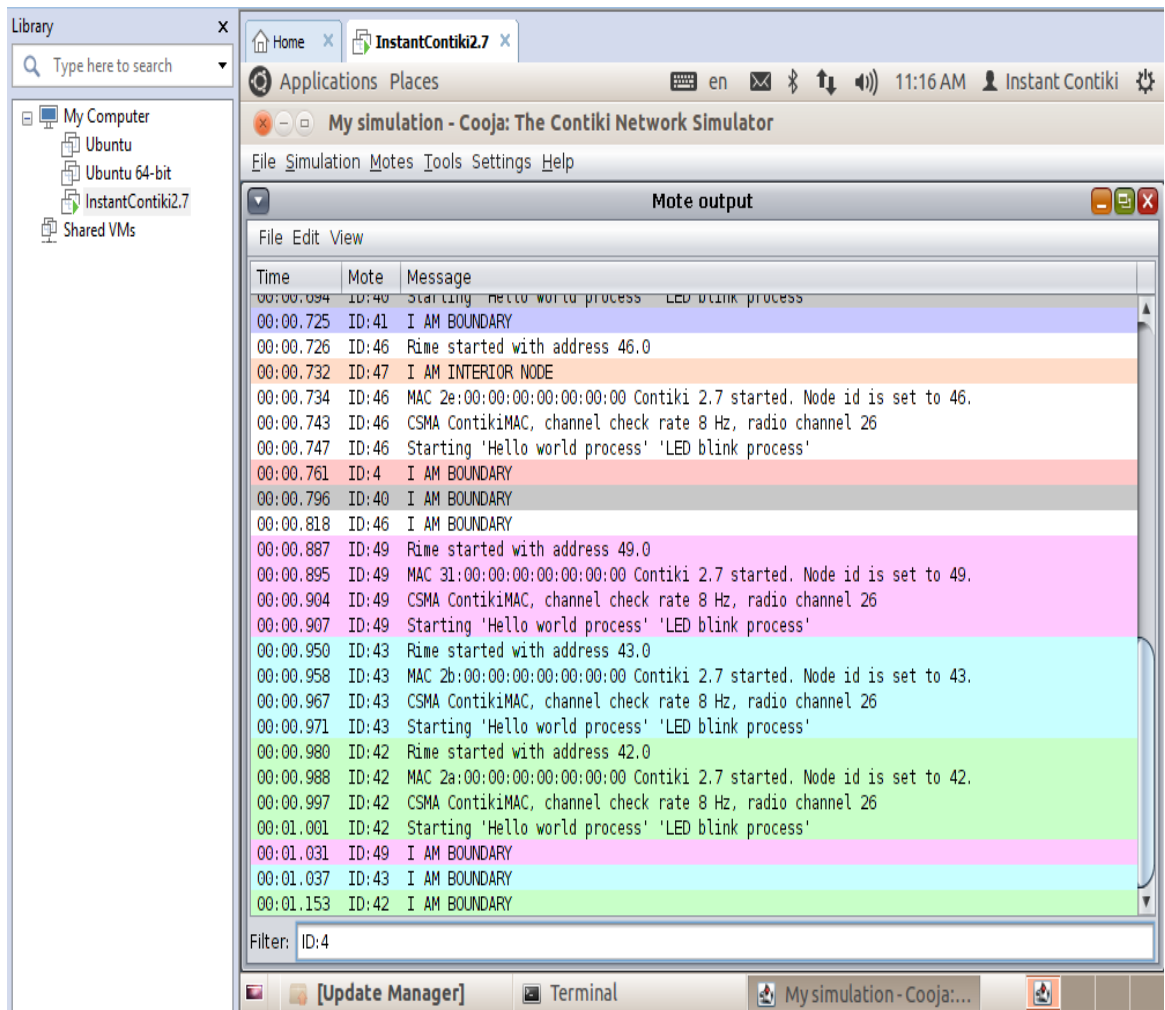


Fig27. Mote output window showing decision on nodes

4.9 Analysis for the dense network

From the results obtained after the compilation of the INITIALIZE and erosion function we deduce that the algorithm did not provide the correct output for some nodes.

Out of the total 58 nodes deployed as the wireless sensor node, the algorithm gave incorrect decision for 23 nodes

This is because actual interior nodes with majority of neighbour as boundary nodes are detected as boundary nodes.

According to this network the accuracy of the algorithm for a sparse network is:

$$\text{Accuracy} = \frac{\text{Number of nodes correctly detected}}{\text{Total number of nodes}} * 100 \%$$

$$\text{Accuracy} = (23 / 58) * 100 \% = 40 \% \text{ (approx)}$$

4.7 Code

```
#include "contiki.h"
#include "dev/leds.h"
#include "node-id.h"
#include <stdio.h>
/*-----*/
int arr_led[50];
/*-----*/
PROCESS(hello_world_process, "Hello world process");
PROCESS(blink_process, "LED blink process");

AUTOSTART_PROCESSES(&hello_world_process,&blink_process);
/*-----*/
/* Implementation of the first process */
PROCESS_THREAD(hello_world_process, ev, data)
{

    PROCESS_BEGIN();

    int ex_path[50],key=0;
    int ar1[17][12] = {{2,14,13,12,0},{3,1,0},{2,14,17,0},{17,16,5,0},
    {16,8,6,4,0},{5,7,8,0},{6,8,9,0},{16,5,6,9,7,0},{10,15,8,7,0},
    {15,9,13,11,0},{10,13,12,0},{1,11,13,0},{1,12,11,10,15,14,0},
    {1,3,17,15,13,0},{14,9,10,13,0},{17,4,5,8,0},{3,14,4,16,0}};

    int ar2[17][12]={3,17,15,13,12,11,10,14,0},{17,14,12,13,0},{1,17,15,13,14,4,16},
    {3,14,16,17,5,8,6},{8,4,17,9,7,6,16,0},{4,16,8,9,5,7,0},{8,5,16,10,15,0},
    {17,4,16,6,5,7,15,10,9,0},{11,13,15,14,16,5,6,0},{13,14,15,8,7,11,12,1,0},
```

```
{13,15,9,1,14,0},{2,14,13,10,1,15,0},{14,2,1,11,12,10,15,9,3,7,0},  
{2,12,13,17,3,4,16,9,10,1,11,0},{1,3,17,15,13,10,8,7,9,11,12,14},  
{3,14,4,5,17,8,6,9,7,0},{14,2,1,13,15,16,5,8,4}};
```

```
int k=0,size =0,temp;  
int inc_xp=0,step,lth=5,size2;
```

```
temp = node_id;
```

```
while(k<12)  
{  
    if(ar2[temp-1][k]!=0)  
        size++;  
  
    else  
        break;  
    k++;  
}
```

```
int road_map[size];  
int p;  
for(p=0;p<size;p++)  
{  
    road_map[p]=0;  
}
```

```
int vc[size];  
int ppt;  
for(ppt=0;ppt<size;ppt++)  
vc[ppt]=ar2[temp-1][ppt];  
size2 = size;
```

```
while(size2 > 0)  
{  
    int nr,value,ttp;  
  
    for(ttp=0;ttp<size;ttp++)  
    {  
  
        if(vc[ttp]!=0)  
            break;  
    }  
    nr = ttp;
```

```
value = vc[nr];  
vc[nr]=0;  
size2--;  
ex_path[inc_xp++]=nr;  
key=inc_xp-1;  
road_map[nr]=1;  
step=2;  
int candidate_nodes[size];  
while(1)
```

```

{
    int cd1=0,g,h,size_cd,nm,min=99;

    for(g=0;g<size;g++)
    {
        for(h=0;h<12;h++)
        {
            if(ar2[temp-1][g]==ar1[value-1][h] && ar1[value-1][h]!=0)
            {
                candidate_nodes[cd1++]=ar2[temp-1][g];
                if(road_map[g] < min)
                {
                    nm = g;
                    min = road_map[g];
                }
                break;
            }
        }
        size_cd = cd1;
        if(road_map[nr]-road_map[nm]>lth)
        {
            printf("I AM INTERIOR NODE\n");
            leds_toggle(LED_BLUE);
            arr_led[node_id]=0;
            return 0;
        }
    }

    int cd2=0,q,w;
    for(q=0;q<size;q++)
    {
        for(w=0;w<12;w++)
        {
            if(vc[q]==ar1[value-1][w] && vc[q]!=0)
            {
                candidate_nodes[cd2++]=vc[q];
            }
        }
        size_cd = cd2;
        if(size_cd>0)
        {
            int a,s;
            for(a=0;a<size;a++)
            {
                for(s=0;s<size_cd;s++)
                {
                    if(vc[a]==candidate_nodes[s])
                    {
                        vc[a]=0;
                        size2--;
                        road_map[a]=step;
                    }
                }
            }
        }
    }
}

```

```

    }
    step++;
    nr = 0;
    int mmp;
    for(mmp=0;mmp<size;mmp++)
    {
        if(candidate_nodes[nr]==ar2[temp-1][mmp])
        {
            nr = mmp;
            break;
        }
    }
    ex_path[inc_xp++] = nr;
    key = inc_xp-1;
    value = ar2[temp-1][nr];
}
else if(step>2)
{
    step--;
    int z,recover_nodes[size],rn=0,size_rn;
    for(z=0;z<size;z++)
    {
        if(road_map[z]==step)
        {
            recover_nodes[rn++]=z;
            road_map[z]=0;
        }
    }
    size_rn=rn;
    int p;
    for(p=0;p<size_rn;p++)
    {
        if(recover_nodes[p]!=nr)
        {
            vc[recover_nodes[p]]=ar2[temp-1][recover_nodes[p]];
            size2++;
        }
    }
    if(key>=0)
    {
        nr = ex_path[key];
        inc_xp--;
        key = inc_xp-1;
    }
}
else
break;
}
printf("I AM BOUNDARY\n");
leds_toggle(LED_RED);
arr_led[node_id]=1;
return 0;

```



```

}
PROCESS_END();
}
/*-----*/
/* Implementation of the second process */
PROCESS_THREAD(blink_process, ev, data)
{

PROCESS_BEGIN();

int ar1[17][12] = {{2,14,13,12,0},{3,1,0},{2,14,17,0},{17,16,5,0},
{16,8,6,4,0},{5,7,8,0},{6,8,9,0},{16,5,6,9,7,0},{10,15,8,7,0},
{15,9,13,11,0},{10,13,12,0},{1,11,13,0},{1,12,11,10,15,14,0},
{1,3,17,15,13,0},{14,9,10,13,0},{17,4,5,8,0},{3,14,4,16,0}};

if(arr_led[node_id]==1)
{
int res,j,count=0,count_i=0;
for(j=0;j<12;j++)
{
res = ar1[node_id-1][j];
if(arr_led[res]==1)
count++;
else if(arr_led[res]==0)
{
count_i++;
count++;
}
else if(res==0)
break;
}
float Eth = 0.00;
Eth = (float) count_i/count;
if (Eth >0.53)
{
arr_led[node_id]=0;
}

}
return 0;
PROCESS_END();
}
/*-----*/

```

Chapter – 5

CONCLUSION

5.1 Conclusion

From the simulations performed we can conclude that the algorithm proposed by Baoqi Huang, Wei Wu, and Tao Zhang detects the boundary node of a deployed network of sensor nodes with an accuracy of 88% in sparse network and 40% accuracy in dense network.

5.2 Future scope

After the boundary nodes are detected correctly, there are further many extensions to the project.

The algorithm can be executed for 3-D networks; here we need to tackle with the z-coordinate of the sensor node with which we realize the 3-D space.

Efforts can be made for finding virtual coordinates for localization of sensor nodes.

References :

1. Baoqi Huang, Wei Wu, Tao Zhang An Improved Connectivity-based Boundary Recognition Algorithm in Wireless Sensor Networks, in 38th Annual IEEE Conference on Local Computer Networks.(2016)
2. P.K. Sahoo, J.P. Sheu, K.Y. Hsieh Target tracking and boundary node selection algorithms of wireless sensor networks for internet services Inform. Sci., 230 (1) (2013)
3. P.K. Sahoo, J.P. Sheu, K.Y. Hsieh Target tracking and boundary node selection algorithms of wireless sensor networks for internet services Inform. Sci., 230 (1) (2013)
4. S.P. Fekete, A. Kroller, D. Pfisterer, S. Fischer, C. Buschmann, Neighbourhood-based topology recognition in sensor networks, in: Proceedings of the 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks, Turku Finland, July 2004
5. G. Destino, G.T. Freitas de Abreu, Network boundary recognition via graph-theory, in: Proceedings of the 5th Workshop on Positioning, Navigation and Communication, 2008
6. S. Funke, Topological hole recognition in wireless sensor networks and its applications, in: Proc of the Joint Workshop on Foundations of Mobile Computing, 2005
7. Y. Wang, J. Gao, J.S.B. Mitchell, Boundary recognition in sensor networks by topological methods, in: Proc of the 12th Annual International Conference on Mobile Computing and Networking, Los angeles, California, 2006
8. K.Y. Hsieh, J.P. Sheu Hole recognition and boundary recognition in wireless sensor networks Personal, Indoor and Mobile Radio Communications, IEEE (2009)

9. I.M. Khan, M.Z. Khan, H. Mokhtar, M. Merabti Enhancements of the self-recognition scheme for boundary recognition in wireless sensor networks *Developments in E-systems Engineering (DeSE)*, IEEE (2011)

10. Haibo Zhang, Hong Shen Balancing Energy Consumption to Maximize Network Lifetime in Data-Gathering Sensor Networks in *IEEE Transactions on Parallel and Distributed Systems* (2009)

11. Shailendra Shukla, Rajiv Misra Angle Based Double Boundary Recognition in Wireless Sensor Networks *JOURNAL OF NETWORKS*, MARCH 2014

12. Massinissa Saoudi, Farid Lalem, Ahcène Bounceur, Reinhardt Euler, M-Tahar Kechadi, Abdelkader Laouid, Madani Bezoui, Marc Sevaux D-LPCN: A distributed least polar-angle connected node algorithm for finding the boundary of a wireless sensor network

13. Research by W.-C. Chu on “Decentralized boundary recognition without location information in wireless sensor networks.” (Apr 2012)