Analysis & Implementation of Sinkhole Attack in RPL

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

In

Computer Science & Engineering/Information Technology

By

Kartik Chaudhary (141259)
Apoorv Jain (141276)

Under the supervision of

Mr. Arvind Kumar

To

Department of Computer Science & Engineering & Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **"Analysis & Implementation of Sinkhole Attack in RPL "** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** submitted in the department of Computer Science & Engineering**,** Jaypee University of Information Technology Waknaghat is an authentic record of our own work carried out over a period from July 2017 to May 2018 under the supervision of **Mr. Arvind Kumar** Assistant Professor, Department of Computer Science & Engineering. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Kartik Chaudhary (141259)
Apoorv Jain (141276)

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

Mr. Arvind Kumar

Assistant Professor

Department of Computer Science & Engineering

Dated:

ACKNOWLEGEMENT

We are grateful & indebted to Mr. Arvind Kumar, Assistant professor, Department of Computer Science & Engineering for his help & advice in completion of this project report.

We also express our deep sense of gratitude & appreciation to our guide for his constant supervision, inspiration & encouragement right from the beginning of this Project report.

We also want to thank our parents & friends for their immense support & confidence upon us. We deem it a pleasant duty to place on record our sincere & heartfelt gratitude to our project guide for his long sightedness, wisdom & co-operation which helped us in tackling crucial aspects of the project in a very logical & practical way.

Kartik Chaudhary (141259)
Apoorv Jain (141276)

TABLE OF CONTENTS

LIST OF IMAGES

LIST OF TABLES

LIST OF ABBREVIATIONS

1.	LP	Linear Placement
2.	OF	Object Function
3.	LLN	Low Power & Lossy Network
4.	RPL	Routing Protocol
5.	HPC	Historical Power Consumption
6.	WSN	Wireless Sensor Network
7.	PC	Power Consumption
8.	RDC	Radio Duty Cycle
9.	DAO	Destination Advertisement Object
10.	IEEE	Institute of Electrical & Electronics Engineers

ABSTRACT

"IOT is one of the emerging technology in today's era. In this day to day "physical objects" connects by the means of various technology that are existing today. IOT is developing very rapidly. But there has been always uncertainty about the securities and privacy which affect the sustainability of IOT. This Project involves analyzing and implementing the Sinkhole attack in RPL.

We implement the attack on any of the node in the network. The attacked node will try to attract more traffic than regular. By decreasing the parent rank in the code, once is enable to derive the traffic on one of the node. We will analyse the performance in attack and without attack scenario on the basis of "change in network topology, power consumption analysis, average radio duty cycle analysis and the sensor graph."

CHAPTER 1
INTRODUCTION

1.1 The Internet-of-Things

1.1.1 About IoT

"The Internet of things is the interconnection of physical contraptions, vehicles (moreover insinuated like "associated devices" & "smart devices"), structures & diverse things—installed with gadgets, software's, sensors, actuators, & framework arrange that enable these articles to accumulate & exchange data."IoT is fundamentally associating every one of the gadgets of our everyday life to the internet in order to make our life less dem&ing.

"Things," in IoT alludes to each gadget from lights to vehicles & so on of our everyday life.

These things are associated with a system in order to improve our labor.

Scientists look "Things" as an "indistinguishable mix of equipment, programming, information, & administration ". These gadgets assemble data from the adjacent condition.

IoT is the main research subject as these days center is moving towards omnipresent registering from conventional figuring.

Iot covers distinctive parts of transportation, development, therapeutic sciences, home machines, shopping encounters & so on.

Image 1.1 IoT Application

1.1.2 Growth of the IoT

IOT is one of the recent technology & an expansive number of individuals approximately 87% have even not known about it. In any case, From ATM's in 1980's, IOT is being the part of life of us. In future, by 2020 we will have approximately 1 Billion articles associated to IOT and Web.

Market of smart watches has grown rapidly in the past few years showing the integration of IOT.

It is obvious that IOT is look after & gadgets associated with web will increment definitely.

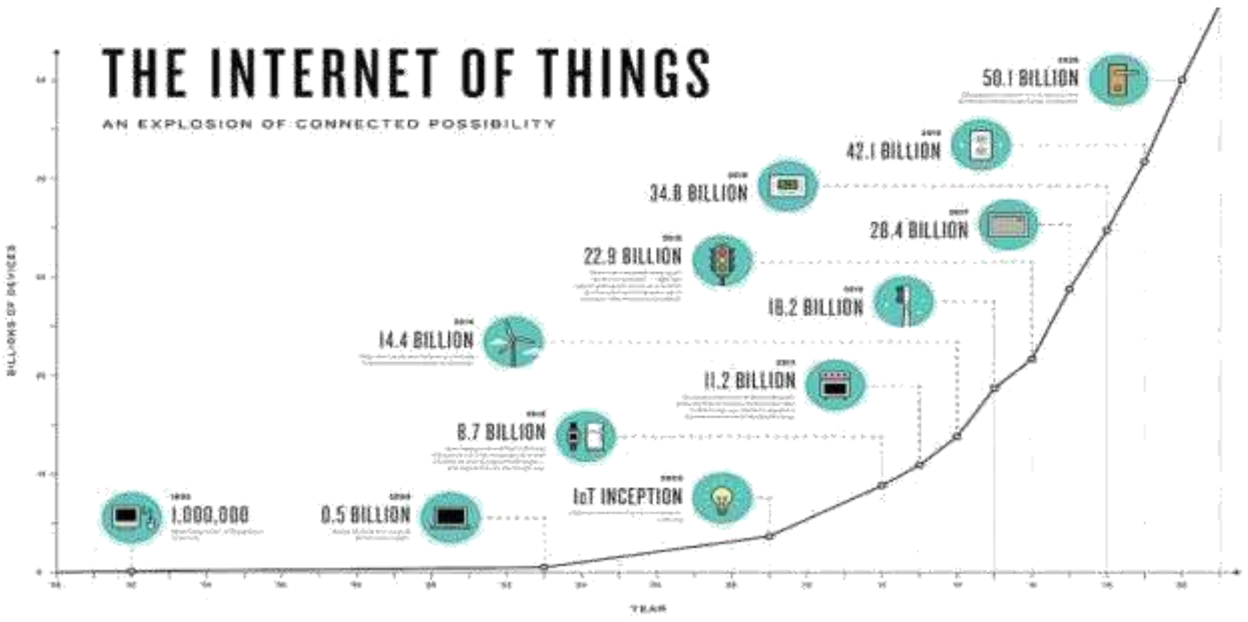


Image 1.2 IoT Growth

1.1.3 Long time to value

IoT assignments can take quite a while. From business case change or change to check of thought for full-scale rollout, every time of the technique can be burdened with inconveniences. In light of our work with clients, we suggest that affiliations take after a five-sort out strategy to confine the time required to pass on their IoT specialists, while improving the entry on these activities.

In today scenario, there are lots of project involved in IOT in government and private sector. The plans of smart cities, Free Wi-Fi zones .

1.1.4 Scalability

Another snag associations keep running into is the trouble of growing their IoT venture after some time. Numerous organizations effectively execute an answer with a couple of hundred gadgets in a single area, just to find that it won't scale to help several thousands or even a great many gadgets crosswise over a wide range of areas.

1.2 Problem Statement

Analysis & Implementation of Sinkhole Attack & examination of change in radio environment, power analysis & topology change in normal radio duty cycle

1.3 Aim

The objective is to implement Sinkhole Attack & analyze the attack on lossy & low power network in RPL.

1.4 Methodology

Stimulate the attack in perspective of the RPL. The simulation is performed on a LINUX based test system COOJA Simulator. The strategy is - attack on the nodes that makes a route brokenly by responding fake system information to the information source & squares data through the broken route.

CHAPTER 2
LITERATURE SURVEY

A literature review is a way to assess & decipher all accessible source significant to a specific research question. Its fundamental point is to show a reasonable assessment of the examination territory of enthusiasm by leading a thorough & auditable philosophy. The primary motivation behind writing audit is to locate the applicable writing about the SinkHole Attack on Low Power and Lossy Networks & their system conventions with the end goal of foundation think about, abridge the current work & recognize the hole in the flow explore.

2.1 6LoWPAN

"6LoWPAN, pronounced slowpan, is an acronym of IPv6 over Low Power Wireless Personal Area Network. It is the name of a concluded working group in the internet area of IETF."

It has encapsulation and mechanism of header that enables the IPv6 to receive and deliver packet over IEEE network. Main work horses for the LAN, MAN and WAN are the IPv4 and IPv6 for the data delivery.

2.2 RPL

"RPL is a Distance Vector IPv6 routing protocol for low power & lossy networks that manage the Destination Oriented Directed Acyclic Graph (DODAG) building process utilizing a target work & an arrangement of measurements/limitations." The target work utilizes a mix of measurements & requirements to Image the 'best' way.

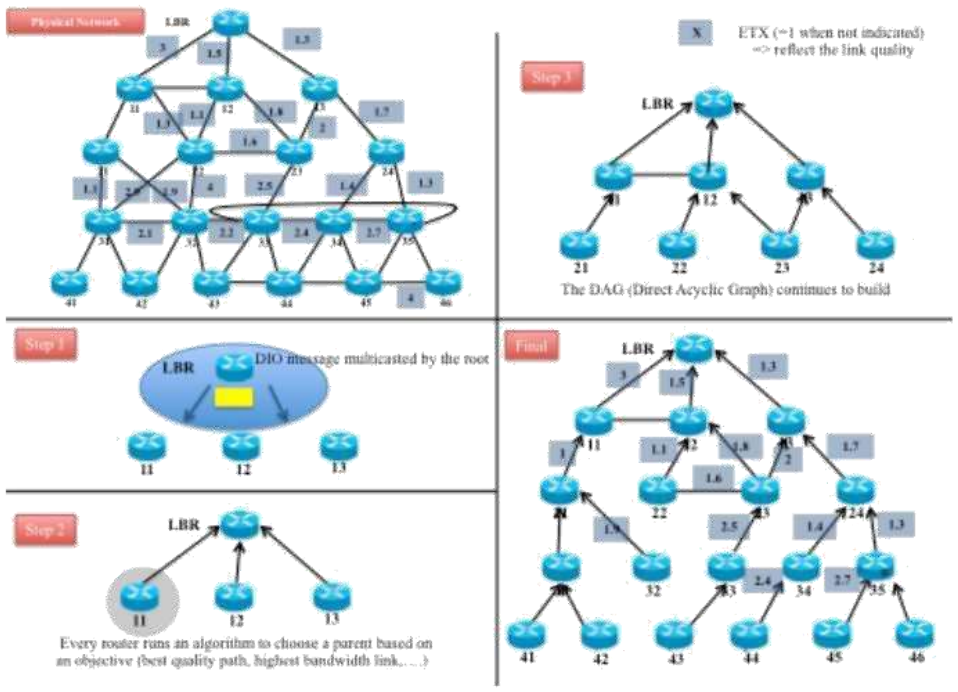
The target work is the key towards the development of the DODAG in light of some system imperatives.

2.2.1 Process of Building DODAG

The way of developing the DODAG graphs starts at the end node or parent node, that is designed by framework overseer. The routing algorithm steering convention gives an arrangement of the new Internet Control Message Protocol packets to inhibit graph related data through various hubs. The 4 DODAG messages or packet are down below :-

- DODAG Information Solicitation (DIS)
- DODAG Information Object (DIO)
- DODAG Destination Advertisement Object (DAO)
- DODAG Destination Advertisement Object Acknowledgment (DAO ACK)

The building of the diagram process begins when the sink node begins broadcasting its data utilizing the DODAG Information Object (DIO) message. The adjacent hubs shall get & process DODAG Information Object (DIO) messages from every single node & settle on their choice whether to interface or not founded on specific standards. Once the hub has joined a diagram it has a course toward the chart root. The diagram sink is named as the 'parent' of the hub. At that point, the hub registers its 'rank' which determines its situation in the system. On the off chance that it's anything but a leaf hub then it shall send DODAG Information Object messages again to its adjacent nodes. On the off chance that the hub is a "leaf hub", it just joins the diagram & doesn't send any DIO message. This procedure will proceed until the point when the leaf hub is come to. This undulating impact fabricates the diagram beats from the root to the leaf hubs where the procedure ends. In this development, every hub of the diagram has a directing passage towards its parent & the leaf hubs can send an information parcel the distance to the base of the graph by simply sending the bundle to its prompt parent.



Physical Network
LBR
Step 1
DIO message multicasted by the root
Step 2
Every router runs an algorithm to choose a parent based on an objective (best quality path, highest bandwidth link, ...)
Step 3
ETX (=1 when not indicated) => reflect the link quality
The DAG (Direct Acyclic Graph) continues to build
Final

Image 2.1Building of DODAG

DODAG Destination Advertisement Object packets are utilized to transmit data of a node upward towards its root node(parent's node). As every node gets the DODAG Destination Advertisement Object packet, it forms data & includes a section in the steering table. This procedure proceeds till the point when a data achieves the sink & an entire way till the sink..

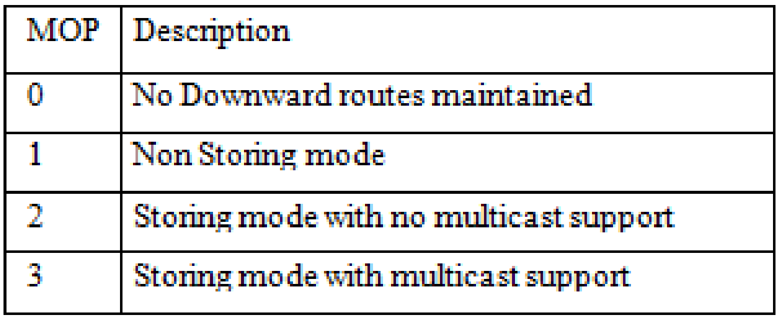
The routing algorithm additionally underpins "**P2P** correspondence to any node to some another hub." The hub sends a bundle to other node inside to system, parcel ventures up to a typical root & soon thereafter it is sent in the 'downside' course goal.

2.2.2 Storing Nodes & Non - Storing Nodes

In General, there are 2 nodes type : -
1.Storing Nodes
2.Non-Storing nodes.

Non Storing Nodes doesn't store the routing entries in the table due to memory constraints where as the Storing Nodes are able to store the routing entries in the routing table of every single node.

It can be achieved by setting the mode in RPL packet header MOP bit. The 4 bit of MOP is as below: -



MOP	Description
0	No Downward routes maintained
1	Non Storing mode
2	Storing mode with no multicast support
3	Storing mode with multicast support

Table2.1 Description of MOP Field

2.2.3 Avoidance and Detection in Loops

In General, Loops are the sequential instruction that are performed continuously until a specific condition is reached. Therefore it is an good practise to locate the loops. They are mainly in shape due to missing of synchronization among the nodes

The two principles determined in are:

• Max_Dept Rule, which expresses that a node can't make a node its parent if its rank is greater than present node

• No node can change its rank itself in order to gain more traffic than usual.

2.2.4 Repair Mechanism

In General, we have 2 mechanism of repair which are supported in RPL

- Global Repair
- Local Repair

Local Repair - It is performed at the time of failure of the links or when there is no path among the hubs or node. Where as Global Repair is performed after the launch of Local Repair.

While Global Repair is executed after Local repair. Global repair is repair mechanism that builts the graph from the very beginning. It is executed after the Local Repair to keep monitoring and maintaining the shape of graph.

2.3 Taxonomy of Attacks in RPL based Internet of Things

Talked about that the RPL convention intended for IPv6 is presented to a wide assortment of assaults & for the most part separated into three classes. The main classification of assaults focuses on the depletion of system assets, for example, memory, vitality & power. The second class of assaults in RPL focuses on the system topology. The assaults on topology is additionally partitioned into two classifications: sub-advancement assaults & segregation assaults. The third classification focuses on the RPL arrange movement..

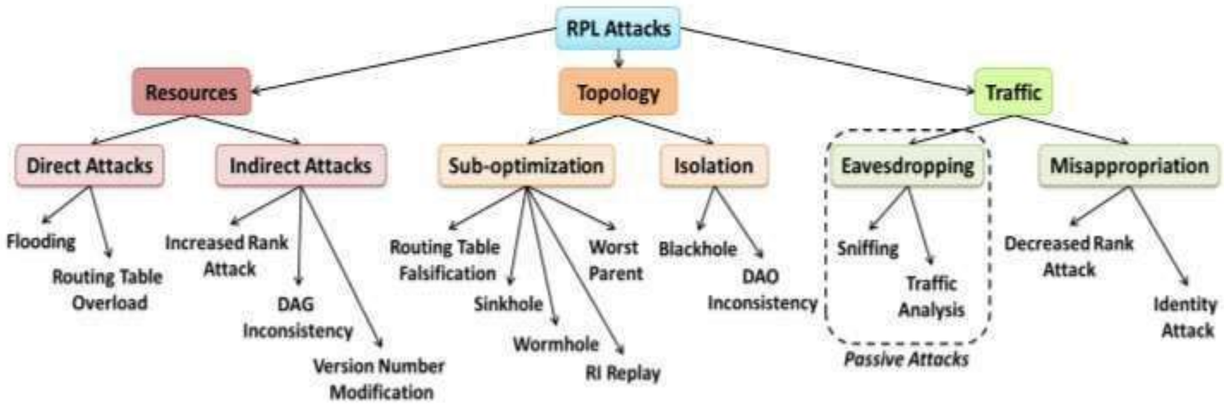


Image 2.2 RPL Attacks Taxanomy

2.3.1 Sinkhole Attack

In this, the main aim of the attacker is to make a node defected or attacked so that it can attract more traffic than it would have attracted in the normal environment. In LLNs, this attack can easily implemented by altering the ranks of the node. If we can decrease the rank of a node, we are able to attract more traffic on the given node. Thus this attack is called as Sink Hole Attack. The attacked node is chosen as parents by many of the child nodes present in the environment.

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 Use of RPL Network

In this, the main aim of the attacker is to make a node defected or attacked so that it can attract more traffic than it would have attracted in the normal environment. In LLNs, this attack can easily implemented by altering the ranks of the node. If we can decrease the rank of a node, we are able to attract more traffic on the given node. Thus this attack is called as Sink Hole Attack. The attacked node is chosen as parents by many of the child nodes present in the environment.

3.2 Topologies

There are basically two types of network topology : -

1. Random Placement of Nodes
2. Linear Placement of Nodes

3.2.1 Random Placement of Nodes

We create 20 nodes in the environment. 19 are the sender nodes and 1 will be the sink node. We placed 20 nodes randomly in the radio environment. The attack is executed on the node 16. We can control the ranks by changing the code by decreasing the parent rank. The attacked node will attract more traffic.

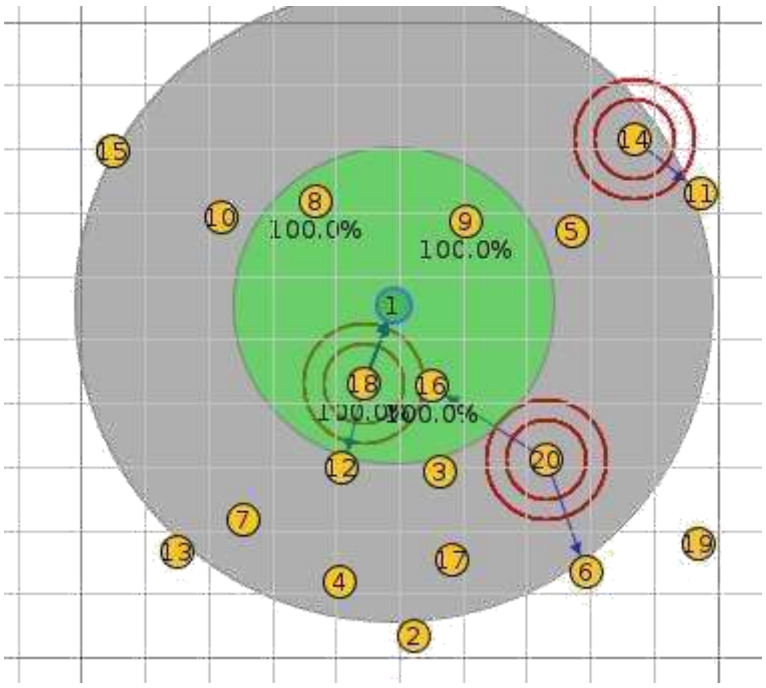


Image 3.1 Randomly Placed Nodes

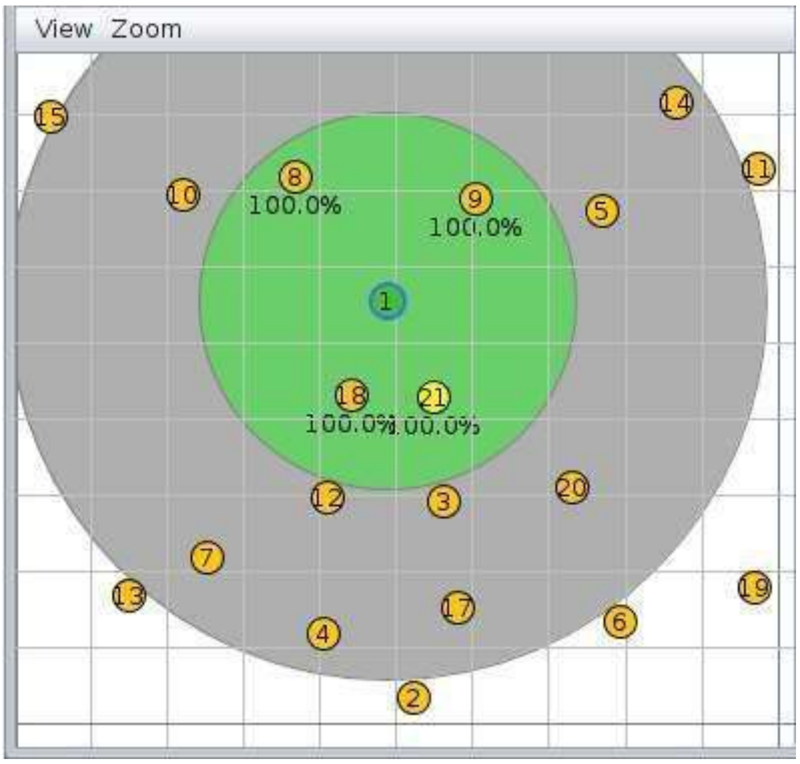


Image 3.2 Radio Environment of Randomly Placed when attack

3.2.2 Linear Placement of Nodes

We create a new simulation having linear placement. We create 9 sender nodes from node 2 to node 10. We create sink node as node 1. We executed sinkhole attack on this linear placement and change the node 5 as the new sink node by changing the rank in the function. Node 5 attracts more traffic than usual.

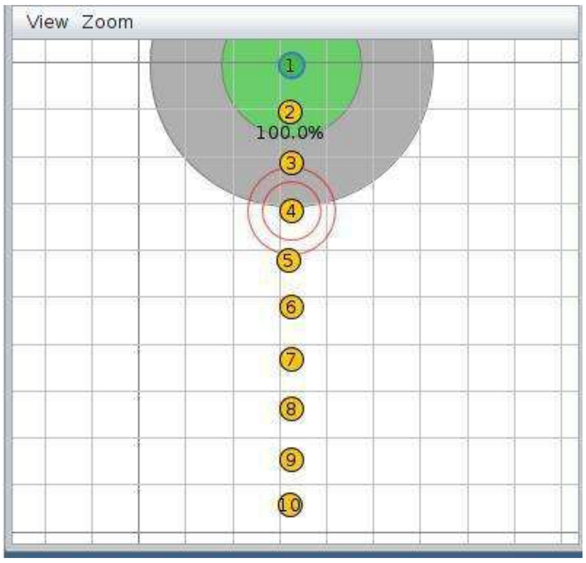


Image 3.3 Linearly nodes placed environment

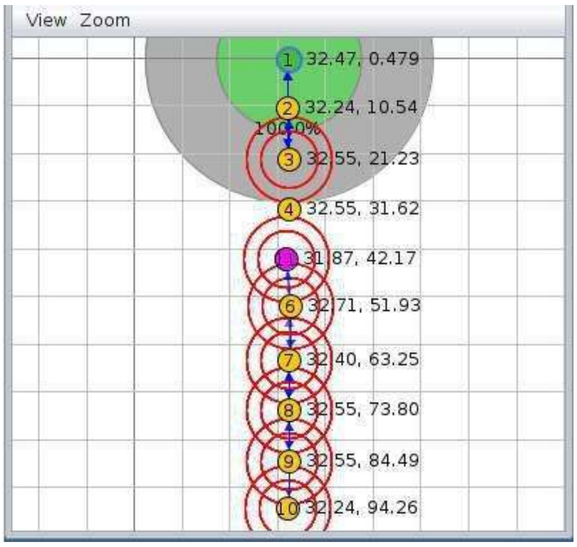


Image 3.4 Radio Environment when attacked

CHAPTER 4
PERFORMANCE ANALYSIS

We analyzed network in 2 different topology:-
1. Random Placement of Nodes
2. Linear Placement of Nodes

After the implementation of Sinkhole Attack, we analyse the performance on following categories: -

1. Sensor Map
2. Avg. Power Consumption(PC)
3. Avg. Radio Duty Cyle(RDC)
4. Historical Power Consumption(HPC)
5. Detection of Sinkhole Attack

4.1 Attack on Random Placement of Nodes

4.1.1 Sensor Map Analysis

Senor Map shows the relation between the Child Node & the Parent Node

In the below scenario, we have Node 1 and Node 16.Node - 1 is the Sink node & Node 16 is a child node having no child. The parent of Node 16 is Node 18 as depict in Image below

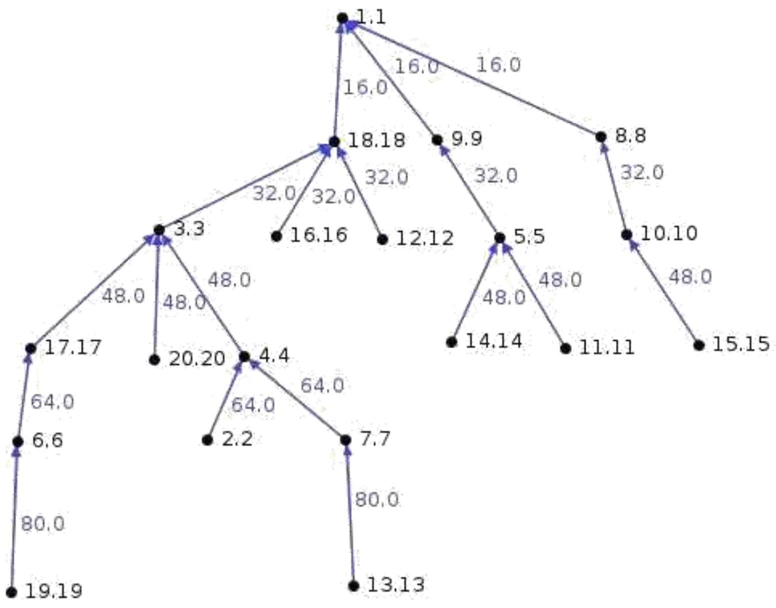


Image 4.1Normal Network without Sink Hole Attack

We implemented Sink Hole attack on node 16. We can see from the Image 4.2 that Node 11 has become Node 21 and now have 3 child which are Node 3, Node 12 and Node 20 as its child. This shows that attacked node has more traffic and the network topology has been changed. This shows the successful implementation of attack.

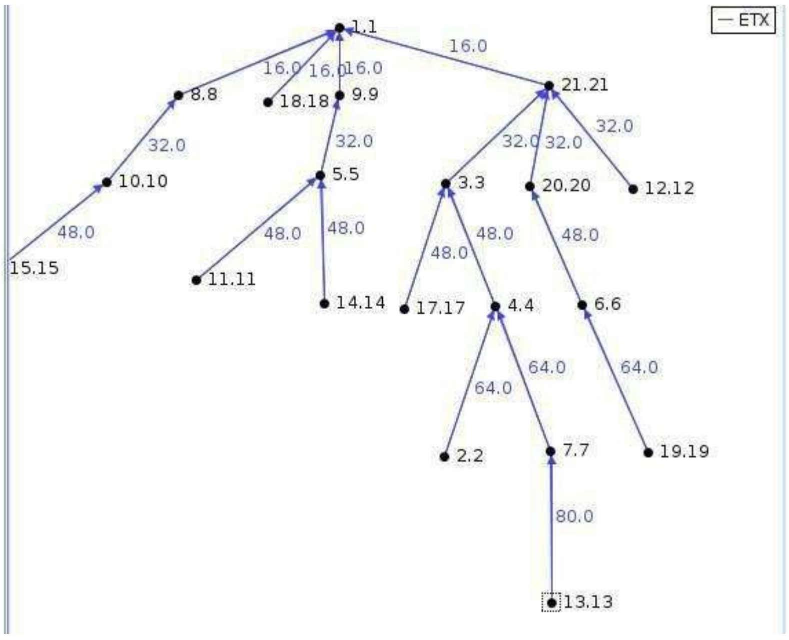


Image 4.2 Network after Attack on node 16

Table 4.1 Nodes with packets received with and without attack

Node	Without Attack	With Attack
1	0	0
2	15	12
3	12	14
4	11	11
5	12	12
6	12	12
7	12	12
8	12	12
9	11	12
10	13	14
11	11	12
12	12	12
13	9	13
14	12	10
15	11	14
16	12	13
17	13	13
18	12	11
19	13	14
20	13	10

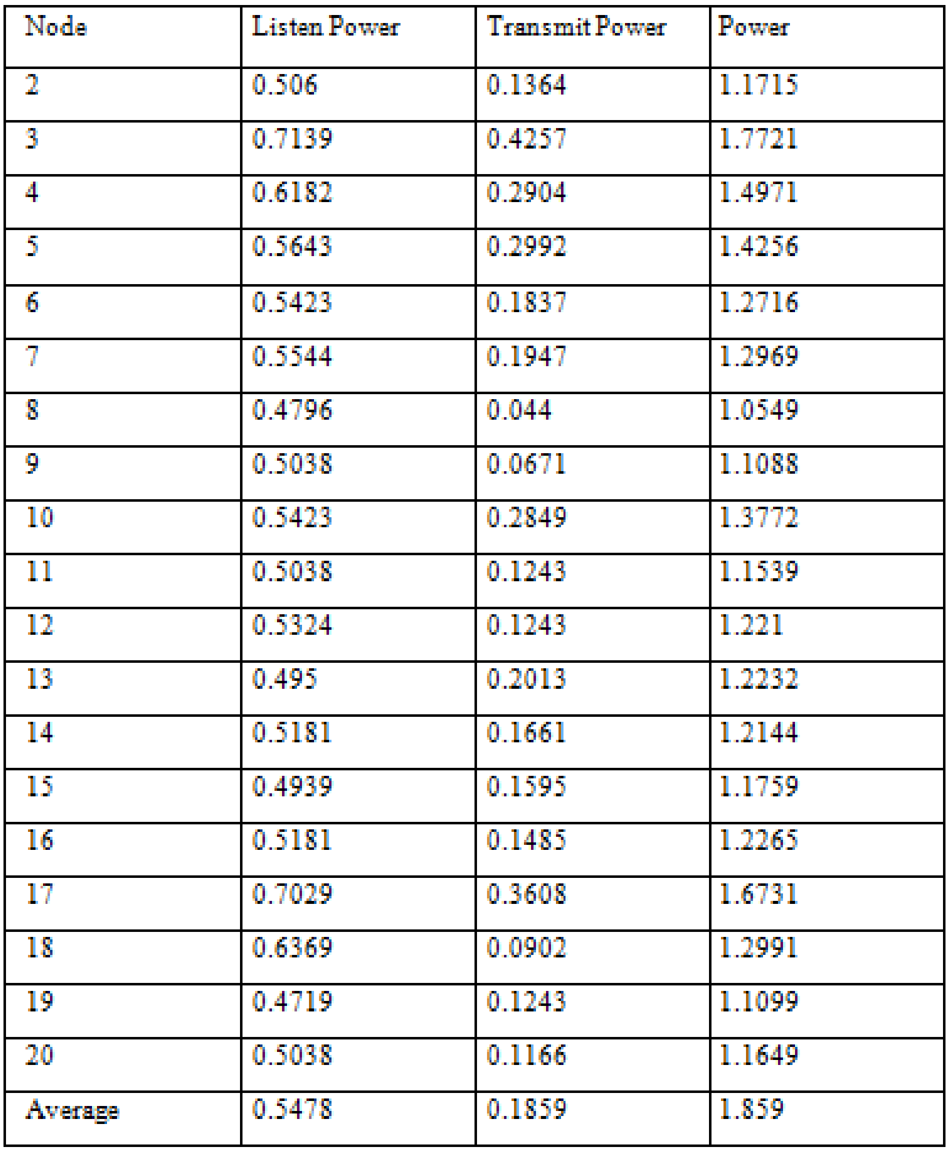
This table shows that packet the number of packet received by Node 16 has increased by 1 i.e. from 12 to 13. Thus Node 16 is attracting more traffic in the attacked mode as compared to the without attack environment.

4.1.2 Average Power Consumption Analysis

“The PC profiling can be defined as a method of Parametering the nodes PC in accordance to their workload.”

In the table below, the power of Node 16 is 1.2265mW. There is no attack implemented.

Table 4.2PC Analysis without attack



Node	Listen Power	Transmit Power	Power
2	0.506	0.1364	1.1715
3	0.7139	0.4257	1.7721
4	0.6182	0.2904	1.4971
5	0.5643	0.2992	1.4256
6	0.5423	0.1837	1.2716
7	0.5544	0.1947	1.2969
8	0.4796	0.044	1.0549
9	0.5038	0.0671	1.1088
10	0.5423	0.2849	1.3772
11	0.5038	0.1243	1.1539
12	0.5324	0.1243	1.221
13	0.495	0.2013	1.2232
14	0.5181	0.1661	1.2144
15	0.4939	0.1595	1.1759
16	0.5181	0.1485	1.2265
17	0.7029	0.3608	1.6731
18	0.6369	0.0902	1.2991
19	0.4719	0.1243	1.1099
20	0.5038	0.1166	1.1649
Average	0.5478	0.1859	1.859

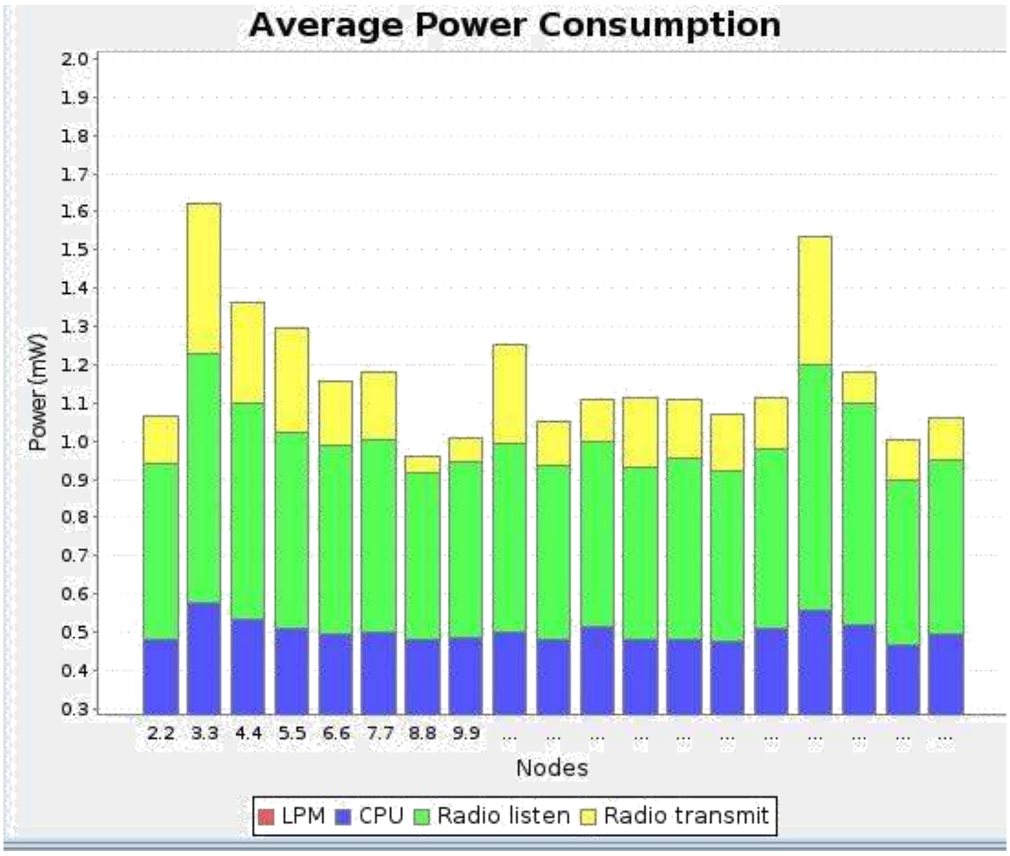


Image 4.3Avg. PC of each node without attack

In the Table given below, we have attack implemented on Node 16. We can see that power consumed by it is 1.342mW . This shows that Power consumption has increased by 0.116mW of Node 16.

Table 4.3 PC Analysis when attacked

Node	Listen Power	Transmit Power	Power
2	0.480	0.212	1.186
3	0.798	0.780	2.201
4	0.616	0.298	1.459
5	0.543	0.336	1.405
6	0.516	0.203	1.221
7	0.542	0.246	1.296
8	0.453	0.070	1.013
9	0.467	0.083	1.047
10	0.508	0.294	1.306
11	0.463	0.128	1.074

12	0.530	0.125	1.175
13	0.452	0.186	1.116
14	0.482	0.174	1.143
15	0.448	0.137	1.061
16	0.601	0.100	1.342
17	0.556	0.178	1.264
18	0.466	0.046	1.006
19	0.455	0.157	1.087
20	0.632	0.385	1.561
Average	0.527	0.218	1.242

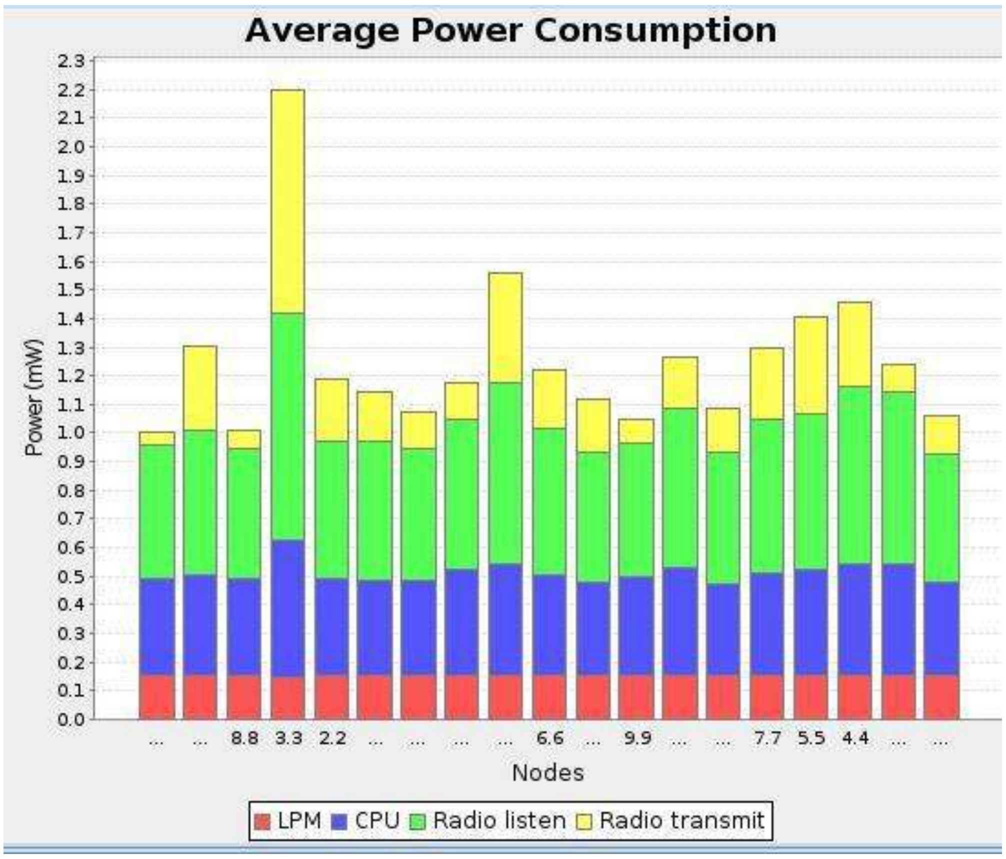


Image 4.4 Avg. PC of each node with Attack

4.1.3 Average Radio Duty Cycle Analysis

“Duty Cycle is the ratio of the transmitted signal's on the air time to the total operating time during the measurement period. It is essential since it identifies with top and normal power in the assurance of aggregate vitality yield..”

The average radio duty cycle from the figure below comes out to be 1.04 of node 16 when there is no attack on it.

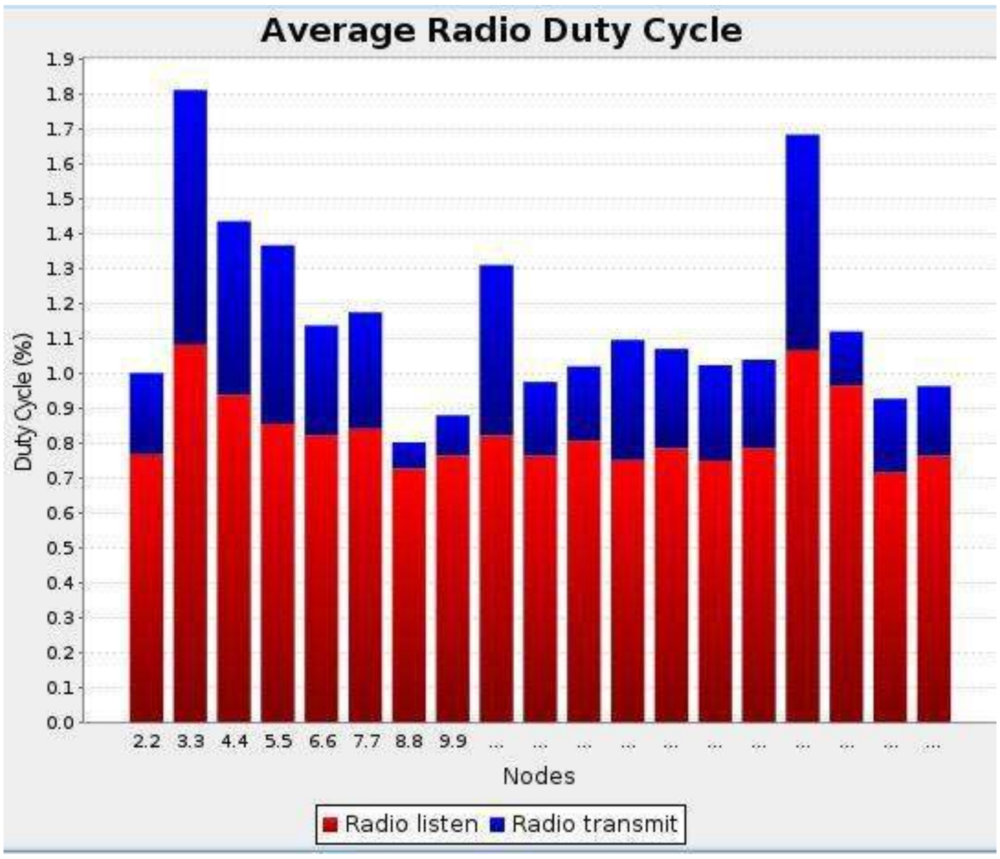


Image 4.5 Avg RDC without attack

When we implement sinkhole attack on Node 16, from the image below we can notice that its radio cycle changes from 1.04 to 1.2.

So we can observe that Avg. RDC of attacked node incremented by 0.16.

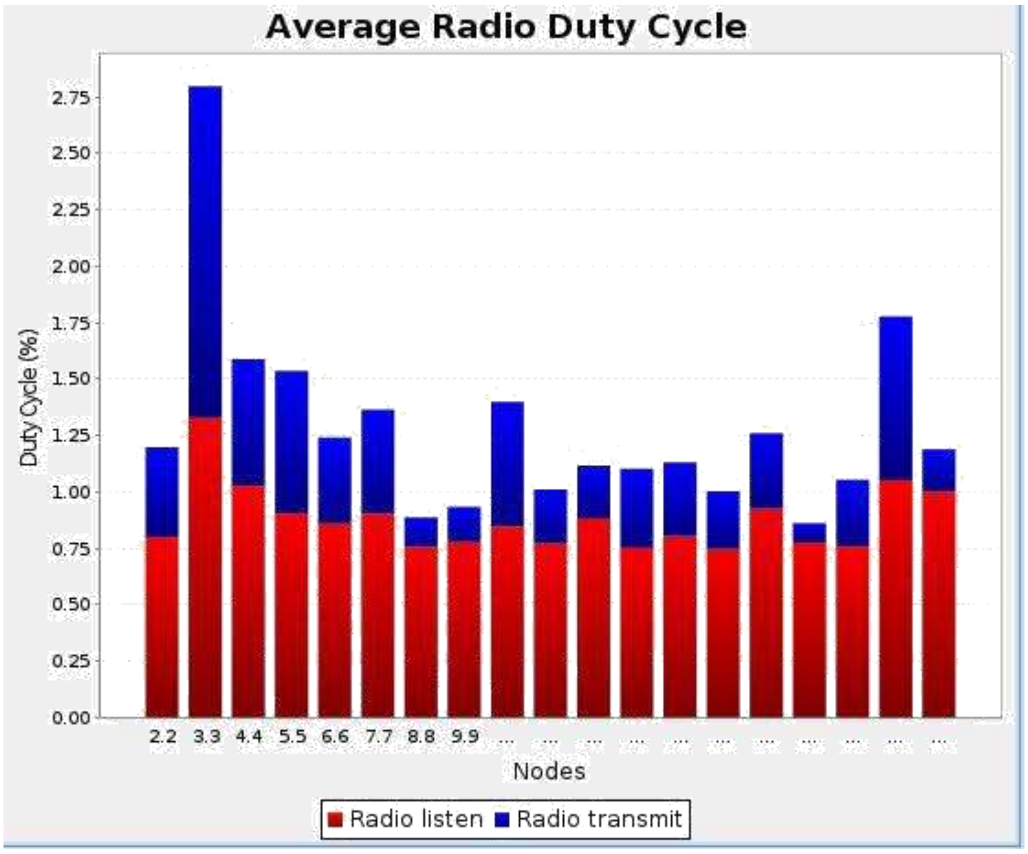


Image 4.6 Avg. RDC with Attack

4.1.4 Power History Analysis

We have implemented no attack on any of the node, we observe that initial PC of the Node 16 is high as compare to its overall PC. This is due to the fact that this node doesn't have any child.

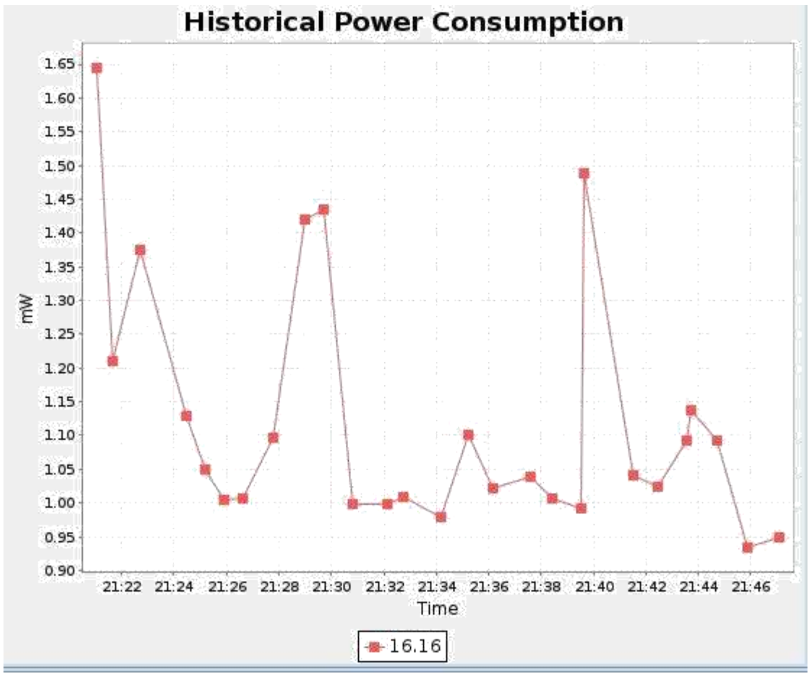


Image 4.7 HPC of Node 16

When we implement sinkhole attack on Node 16, it gets more traffic attract towards it due to the presence of child node. The child node start attracting towards the Node 16 which makes not only initial PC high but also overall PC high.

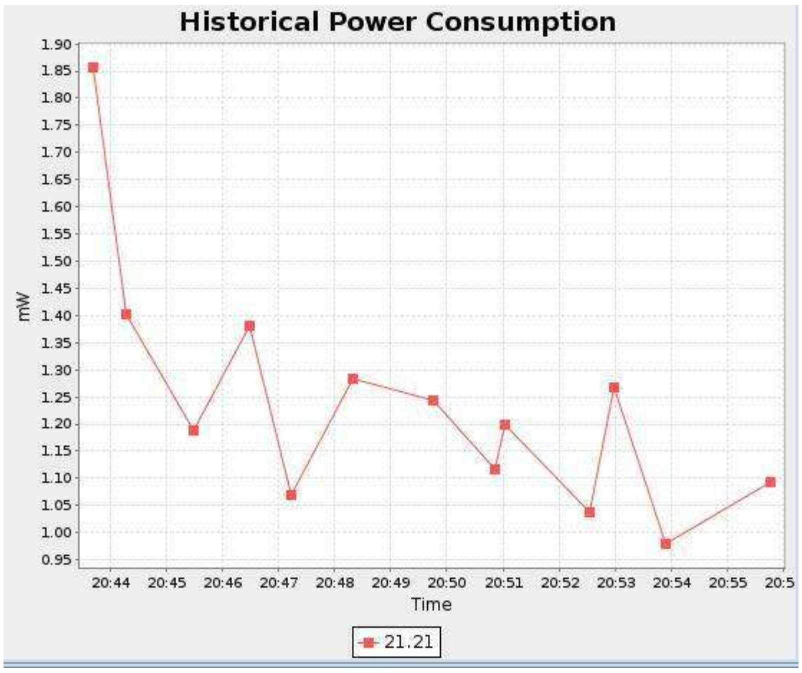


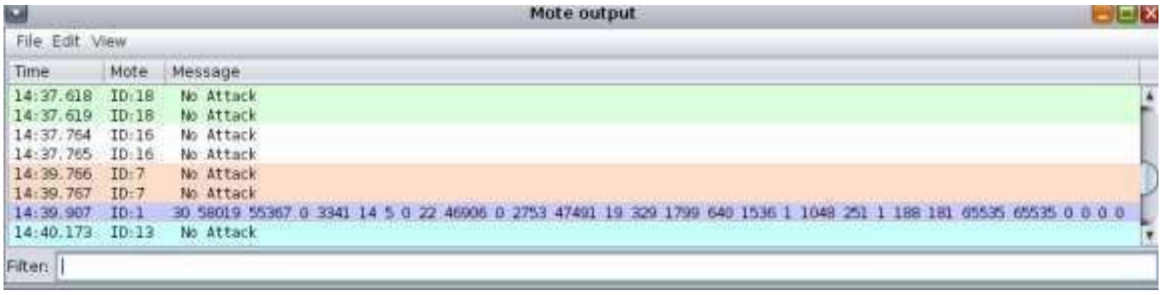
Image 4.8 HPC of Node16 after attack

4.1.5 Detection of Sink Hole

Detection Algorithm

```
If( Node.rank <= Node.Parent.Rank)
Then
 {Sink Hole Attack}
Else
 {No Attack}
```

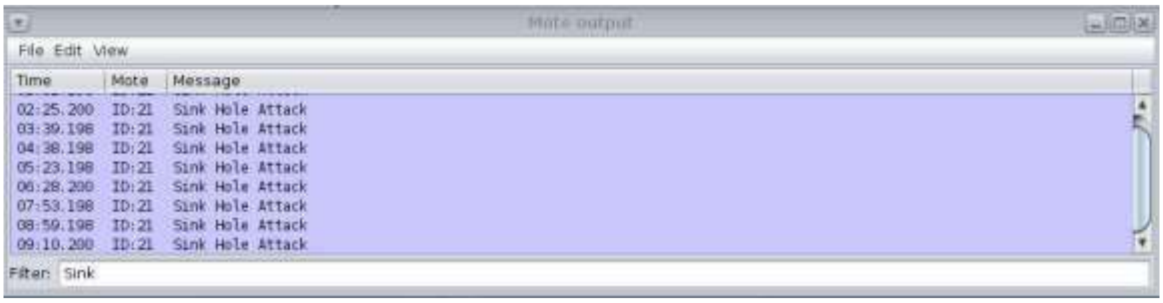
The output can be seen in the mote output window



```
Time        Mote    Message
14:37.618   ID:18   No Attack
14:37.619   ID:18   No Attack
14:37.764   ID:16   No Attack
14:37.765   ID:16   No Attack
14:39.766   ID:7    No Attack
14:39.767   ID:7    No Attack
14:39.907   ID:1    30 58019 55367 0 3341 14 5 0 22 46906 0 2753 47491 19 329 1799 640 1536 1 1048 251 1 188 181 65535 65535 0 0 0 0
14:40.173   ID:13   No Attack
```

Image 4.9 Output without attack

Output window shows that there is no attack.



```
Time        Mote    Message
02:25.200   ID:21   Sink Hole Attack
03:39.198   ID:21   Sink Hole Attack
04:38.198   ID:21   Sink Hole Attack
05:23.198   ID:21   Sink Hole Attack
06:28.200   ID:21   Sink Hole Attack
07:53.198   ID:21   Sink Hole Attack
08:59.198   ID:21   Sink Hole Attack
09:10.200   ID:21   Sink Hole Attack
Filter: Sink
```

Image 4.10 Output after attack

Output window shows that there is sinkhole attack on the given node.

Therefore, the detection algorithm works.

4.2 Attack on Linear Placement of Nodes

4.2.1 Sensor Map Analysis

Senor Map shows the relation between the Child Node & the Parent Node

In the below scenario, we have Node 1 and Node 5.Node - 1 is the Sink node & Node 5 is a child node having no child. The parent of Node 5 is Node 4 as depict in Image below

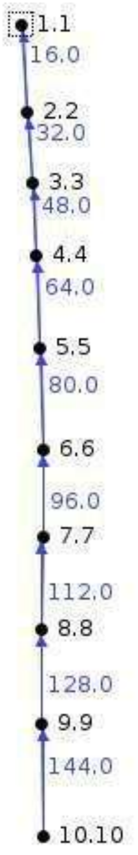


Image 4.11 Normal Linear Placement

When we attack the Node 5 by changing the rank in the objective function, the network topology doesn't changed as all the nodes wants to be grounded

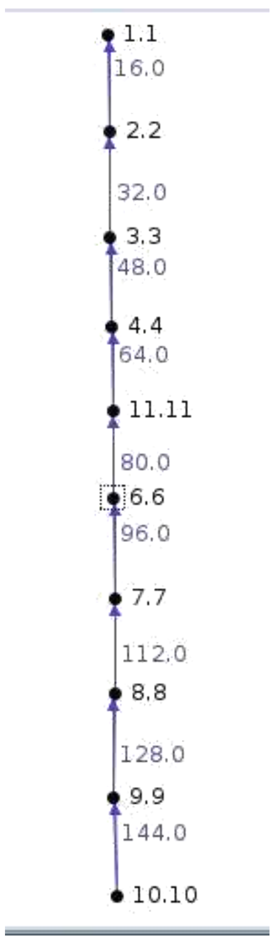


Image 4.12 Attack Linear Placement

Node	Without Attack	With Attack
1	0	0
2	18	18
3	17	17
4	16	18
5/11	16	17
6	18	18
7	18	17
8	15	16
9	17	17
10	17	17

Table 4.4 Nodes with packet received with and without attack in LP

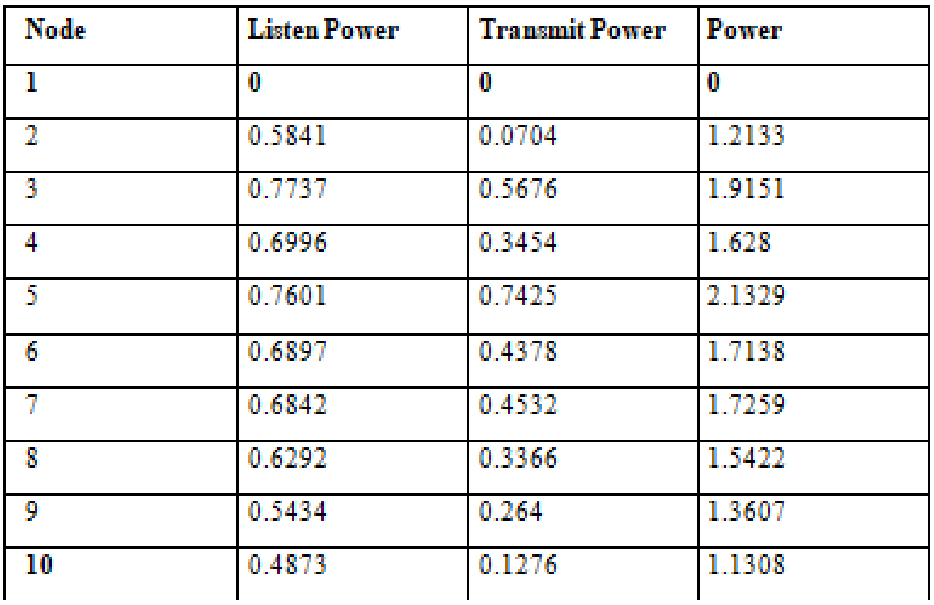
This table shows that packet the number of packet received by Node 5 has increased by 1 i.e. from 16 to 17. Thus Node 5 is attracting more traffic in the attacked mode as compared to the without attack environment.

4.2.2 Average Power Consumption Analysis

“The PC profiling can be defined as a method of Parametering the nodes PC in accordance to their workload.”

From the table we observe that PC of Node 5 is 2.1329mW.

Table 4.5 PC analysis without attack in LP



Node	Listen Power	Transmit Power	Power
1	0	0	0
2	0.5841	0.0704	1.2133
3	0.7737	0.5676	1.9151
4	0.6996	0.3454	1.628
5	0.7601	0.7425	2.1329
6	0.6897	0.4378	1.7138
7	0.6842	0.4532	1.7259
8	0.6292	0.3366	1.5422
9	0.5434	0.264	1.3607
10	0.4873	0.1276	1.1308

The figure below depicts the Avg. PC in the graphical format

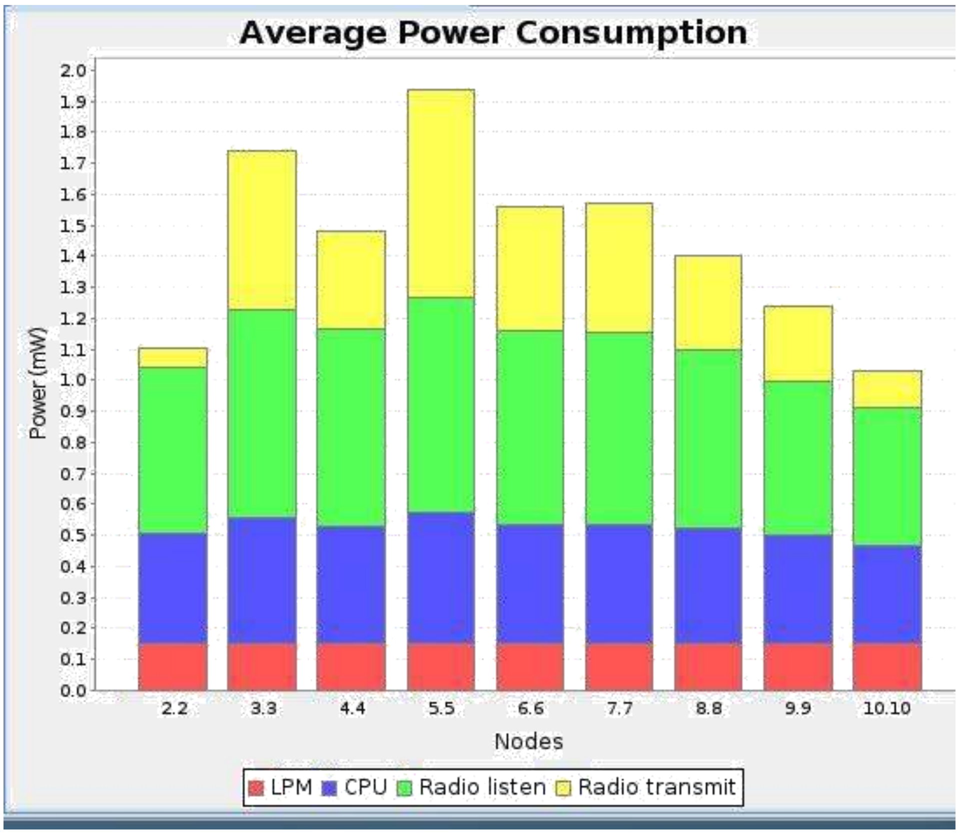


Image 4.13 Avg. PC when the node is attacked

From the table we observe that after attack, the PC of Node 5 has decreased significantly from 2.1329mW to 1.234mW. The total decrement is of 0.9089mW.

Table 4.6 PC Analysis when attacked in LP

Node	Listen Power	Transmit Power	Power
1	0	0	0
2	0.533	0.073	1.118
3	0.689	0.562	1.816
4	0.632	0.279	1.435
5/11	0.532	0.187	1.234
6	0.542	0.272	1.331
7	0.614	0.387	1.553
8	0.545	0.267	1.325
9	0.480	0.203	1.185
10	0.434	0.102	1.001
Avg	0.556	0.259	1.331

The figure below depicts the Avg. PC in the graphical format when there is attack

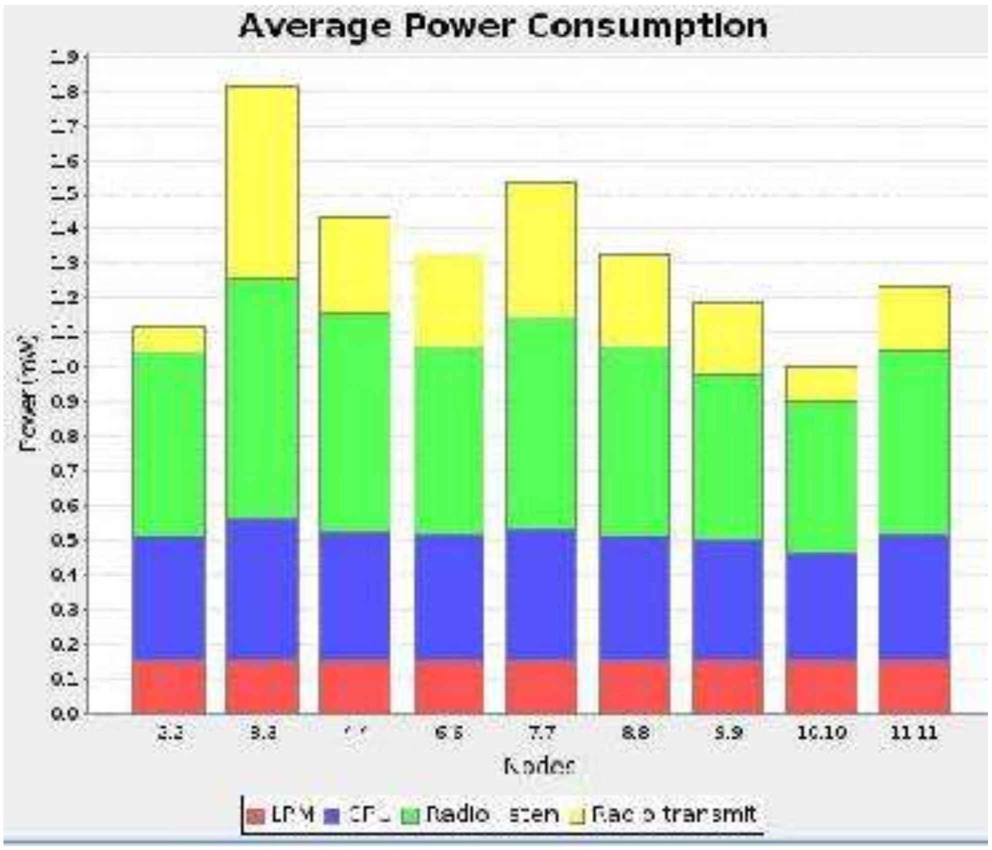


Image 4.14 Avg. PC in Linear Placement when attacked

4.2.3Average Radio Duty Cycle Analysis

“Duty Cycle is the ratio of the transmitted signal's on the air time to the total operating time during the measurement period. It is essential since it identifies with top and normal power in the assurance of aggregate vitality yield..”

The average radio duty cycle from the figure below comes out to be 2.35 of node 16 when there is no attack on it.

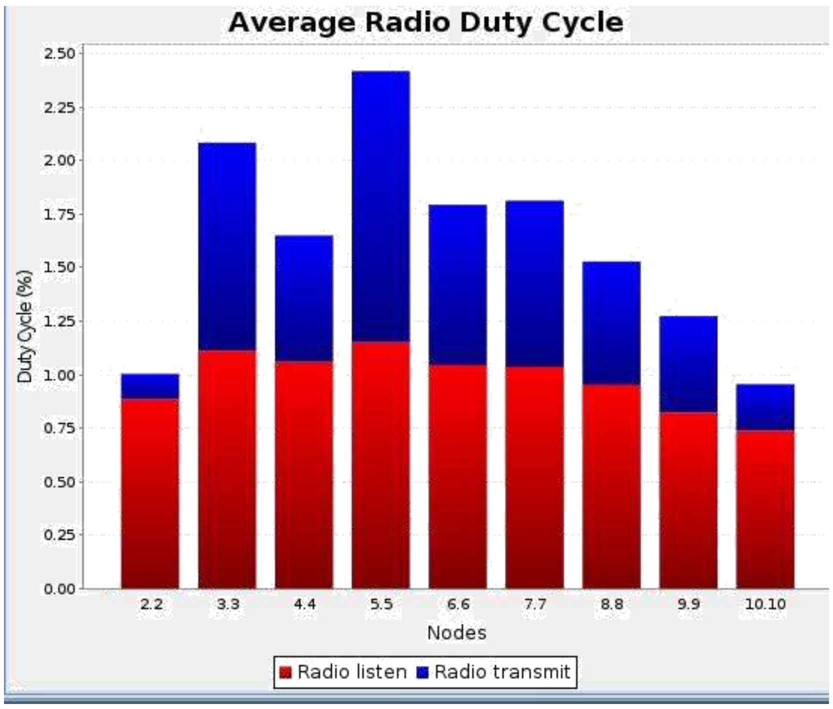


Image 4.15 Avg. RD C without Attack in Linear Placement

When we implement sinkhole attack on Node 5, from the image we can notice that its radio cycle changes from 2.35 to 1.25.
So we can observe that Avg. RDC of attacked node decremented by 1.1.

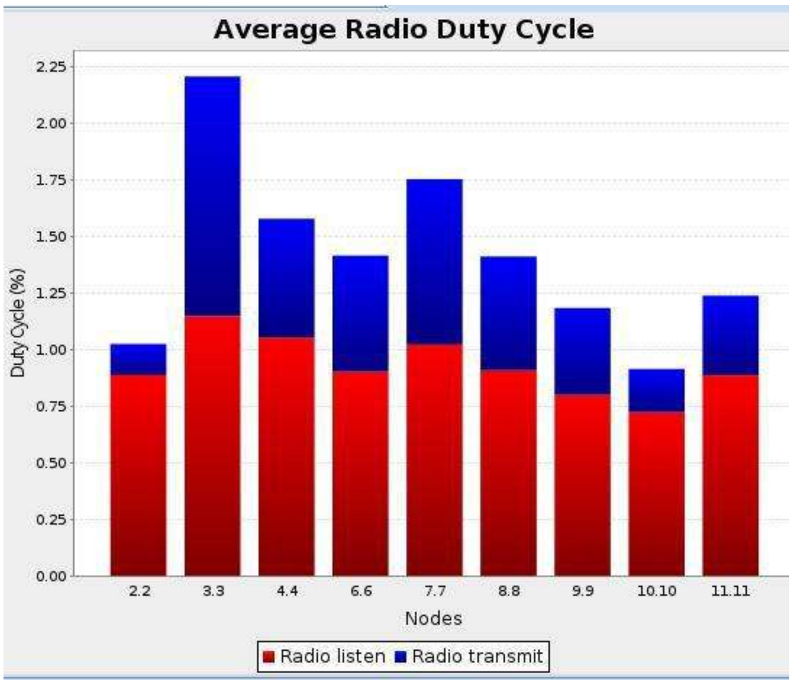


Image 4.16 Avg RDC with attack in Linear Placement

4.2.4 Power History Analysis

The image below depicts the Historical Power Consumption of Node 5 when no attack is implemented.

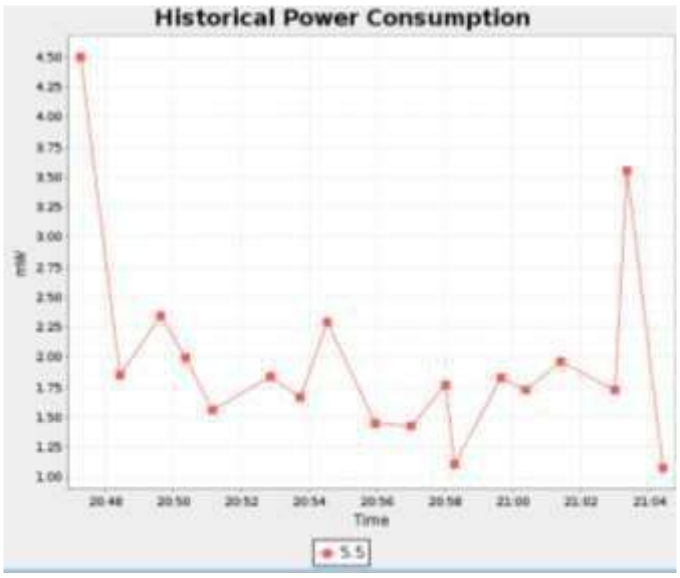


Image 4.17 HPC in Linear Placement without attack

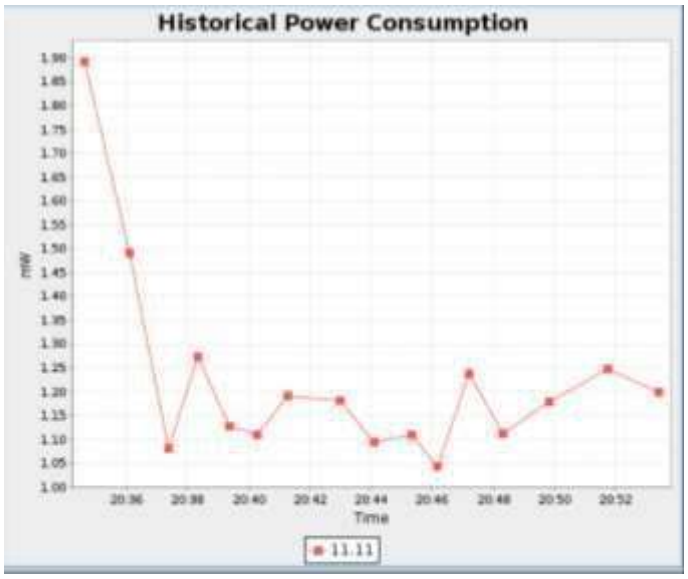


Image 4.18 HPC in Linear Placement with attack

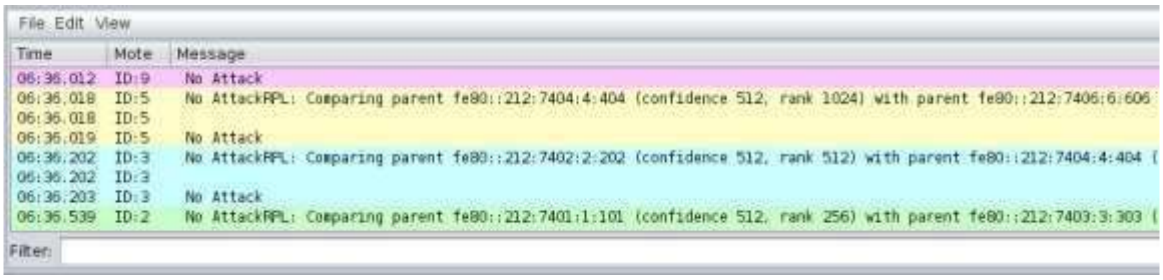
As there is no change in the network topology, we can observe that HPC of Node 5 is little low when the attack is implemented as compared to the normal scenario.

4.2.5 Detection of Sink Hole

Detection - Sinkhole Detection

```
If( Node.rank <= Node.Parent.Rank)
Then
 {Sink Hole Attack}
Else
 {No Attack}
```

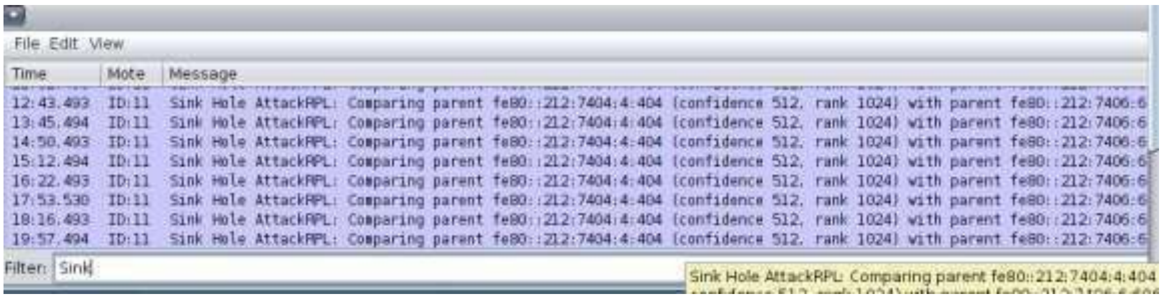
The results can be seen in the output window.



```
File  Edit  View
Time          Mote    Message
06:36.012     ID:9     No Attack
06:36.018     ID:5     No AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606
06:36.018     ID:5
06:36.019     ID:5     No Attack
06:36.202     ID:3     No AttackRPL: Comparing parent fe80::212:7402:2:202 (confidence 512, rank 512) with parent fe80::212:7404:4:404 (
06:36.202     ID:3
06:36.203     ID:3     No Attack
06:36.539     ID:2     No AttackRPL: Comparing parent fe80::212:7401:1:101 (confidence 512, rank 256) with parent fe80::212:7403:3:303 (
Filter:
```

Image 4,19 Window Output in LP

The above figure shows that there is no attack on node.



```
File  Edit  View
Time         Mote    Message
12:43.493    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
13:45.494    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
14:50.493    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
15:12.494    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
16:22.493    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
17:53.530    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
18:16.493    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
19:57.494    ID:11   Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6
Filter: Sink                                          Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404
```

Image 4,20 Window output in LP when attacked

The image above shows that Sink Hole attack is implemented.

Therefore, the algorithm is successfully implemented .

CHAPTER 5
CONCLUSION

5.1 Conclusions

We stimulate the sinkhole attack on Linux based system Cooja Simulator. We perform and executed the Sinkhole Attack on 2 topology which are Random placement of nodes & linear placement of nodes & analyzed it successfully. We analyzed the sensor map, power, radio duty cycle & number of packets, both in the sinkhole attack environment & the normal environment. In Random Placement of Nodes, there is a high shift in the network topology. In Linear Placement of Nodes, there is no major change in the topology as all the nodes are grounded & all the nodes in Linear Placement has a single parent node, so changing the parent rank will not affect all the network as no other node will able to make a faulty parent.

We successfully executed the algorithm to detect the sinkhole attack that can be observe from the screen of Mote Window

5.2 Future Scope

Study for understanding wireless network security in Low Power and Lossy Networks for attacks will be done in the near future, & research in prevention of Sink Hole attack must be done. Cooja Simulation is also an important tool in this network

CHAPTER 6

REFRENCES

[1] Vikrant Negi internet of thing, seminar report ,2008,pp. 1-4.

[2] Shi Yan-rong, Hou Tao, Internet of Things key technologies & architectures research in information processing in Proceedings of the 2nd International Conference on Computer Science & Electronics Engineering (ICCSEE), 2013

[3] Daniele Mior&i, Sabrina Sicari, Francesco De Pellegrini & Imrich Chlamtac, Internet of Things: Vision, applications & research challenges, in Ad Hoc Networks, 2012, pp.1497-1516

[4] Luigi Atzori, Antonio Iera, Giacomo Morabito, The Internet of Things: A Survey, in Computer Networks, pp. 2787-2805

[5] JP Vasseur, Navneet Agarwal, Jonathan Hui, Zach Shelby, Paul Bertr&, Cedric Chauvenet, RPL: The IP routing protocol designed for low power & lossy networks Internet Protocol for Smart Objects, (IPSO) Alliance, April 2011, pp 5-13

[6] Anthea Mayzaud, Remi Badonnel, Isabelle Chrisment, A Taxonomy of Attacks in RPL-based Internet of Thing, in International Journal of Network Security, August 2015, pp 3-12

[7] Arvind Kumar, Rakesh Matam, Shailendra Shukla, Impact of Packet Dropping Attacks on RPL, in Computer Networks, 2016, pp 3-5

[8] Anuj Sehgal ; Anthéa Mayzaud ; Rémi Badonnel ; Isabelle Chrisment ; Jürgen Schönwälder, Addressing DODAG inconsistency attacks in RPL networks, Dec 2014, pp 3-5

[9] Zach Shelby, Carsten Bormann, 6LoWPAN The Wireless Embedded Internet.

[10] T. Winter et. al, RPL: IPv6 Routing protocol for Low power & Lossy Networks, Internet Draft draft-ietf-roll-rpl-17 Retreived, June 2015.

[11] Contiki Operating Systems Website: http://www.contiki-os.org Retreived:, July,2015.

[12] Heddeghem W V, Cross-Layer link estimation for Contiki based wireless sensor networks: PhD Thesis, Vrije University. May,2012.

[13] Geoff Mulligan, The 6LoWPAN architecture, EmNets 2007: Proceedings of the 4th workshop on Embedded networked sensors, ACM, 2007.

[14] Anhtuan L, Jonathan L, Aboubaker L, Mahdi A & Yuan L, 6LoWPAN: a study on QoS security threats & countermeasures Using intrusion detection system approach International Journal of Communication systems, 2012, pp. 1-20.

[15] Luigi Atzori, Antonio Iera, Giacomo Morabito, The Internet of Things: A Survey, In Computer Networks, pp. 2787-2805.

CHAPTER 7
APPENDICES

7.1 Code Snippets

7.1.1 Sender.c File

```
#include "contiki.h"
#include "net/uip.h"
#include "net/uip-ds6.h"
#include "net/uip-udp-packet.h"
#include "net/rpl/rpl.h"
#include "dev/serial-line.h"
#if CONTIKI_TARGET_Z1
#include "dev/uart0.h"
#else
#include "dev/uart1.h"
#endif
#include "collect-common.h"
#include "collect-view.h"

#include <stdio.h>
#include <string.h>
#define UDP_CLIENT_PORT 8775
#define UDP_SERVER_PORT 5688

#define DEBUG DEBUG_PRINT
#include "net/uip-debug.h"

static struct uip_udp_conn *client_conn;
static uip_ipaddr_t server_ipaddr;
```

```
/*---------------------------------------------------------------
*/ PROCESS(udp_client_process, "UDP client process");
AUTOSTART_PROCESSES(&udp_client_process, &collect_common_process);
/*---------------------------------------------------------------*/
void
collect_common_set_sink(void)
{
 /* A udp client can never become sink */
}
/*---------------------------------------------------------------*/

void
collect_common_net_print(void)
{
 rpl_dag_t *dag;
 uip_ds6_route_t *r;


 /* Let's suppose we have only one instance */
 dag = rpl_get_any_dag();
 if(dag->preferred_parent != NULL) {
   PRINTF("Preferred parent: ");
   PRINT6ADDR(rpl_get_parent_ipaddr(dag->preferred_parent));
   PRINTF("\n");
 }
 for(r = uip_ds6_route_head();
     r != NULL;
     r = uip_ds6_route_next(r)) {
   PRINT6ADDR(&r->ipaddr);
 }
 PRINTF("---\n");
}
/*---------------------------------------------------------------*/
```

```
static void
tcpip_h&ler(void)
{
 if(uip_newdata()) {
   /* Ignore incoming data */
 }
}
/*---------------------------------------------------------------------------*/
void
collect_common_send(void)
{
 static uint8_t seqno;
 struct {
   uint8_t seqno;
   uint8_t for_alignment;
   struct collect_view_data_msg msg;
 } msg;

  /* struct collect_neighbor *n; */
  uint16_t parent_etx;
  uint16_t rtmetric;
  uint16_t num_neighbors;
  uint16_t beacon_interval;
  rpl_parent_t *preferred_parent;
  rimeaddr_t parent;
  rpl_dag_t *dag;

  if(client_conn == NULL) {
    /* Not setup yet */
    return;
  }
  memset(&msg, 0, sizeof(msg));
  seqno++;
  if(seqno == 0) {
   /* Wrap to 128 to identify restarts */
   seqno = 128;
  }
  msg.seqno = seqno;

  rimeaddr_copy(&parent, &rimeaddr_null);
  parent_etx = 0;

  /* Let's suppose we have only one instance */
  dag = rpl_get_any_dag();
  if(dag != NULL) {
   preferred_parent = dag->preferred_parent;
   if(preferred_parent != NULL) {
    uip_ds6_nbr_t *nbr;
    nbr=uip_ds6_nbr_lookup(rpl_get_parent_ipaddr(preferred_parent));
    if(nbr != NULL) {

   /* Use parts of the IPv6 address as the parent address, in reversed byte order. */
   parent.u8[RIMEADDR_SIZE - 1] = nbr->ipaddr.u8[sizeof(uip_ipaddr_t) - 2];
   parent.u8[RIMEADDR_SIZE - 2] = nbr->ipaddr.u8[sizeof(uip_ipaddr_t) - 1];
   parent_etx = rpl_get_parent_rank((rimeaddr_t *) uip_ds6_nbr_get_ll(nbr)) / 2;
  }
 }
 rtmetric = dag->rank;
 beacon_interval = (uint16_t) ((2L << dag->instance->dio_intcurrent) /
 1000); num_neighbors = RPL_PARENT_COUNT(dag);
} else {
 rtmetric = 0;
 beacon_interval = 0;
 num_neighbors = 0;
}
```

```
  /* num_neighbors = collect_neighbor_list_num(&tc.neighbor_list); */
  collect_view_construct_message(&msg.msg, &parent,
                                 parent_etx, rtmetric,
                                 num_neighbors, beacon_interval);

  uip_udp_packet_sendto(client_conn, &msg, sizeof(msg),
                        &server_ipaddr, UIP_HTONS(UDP_SERVER_PORT));
}

void
collect_common_net_init(void)
{
#if CONTIKI_TARGET_Z1
  uart0_set_input(serial_line_input_byte);
#else
  uart1_set_input(serial_line_input_byte);
#endif
  serial_line_init();
}

/*---------------------------------------------------------------------------*/
static void
print_local_addresses(void)
{
  int i;
  uint8_t state;

  PRINTF("Client IPv6 addresses: ");
  for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
    state = uip_ds6_if.addr_list[i].state;
    if(uip_ds6_if.addr_list[i].isused &&
       (state == ADDR_TENTATIVE || state == ADDR_PREFERRED))
     { PRINT6ADDR(&uip_ds6_if.addr_list[i].ipaddr);
```

```
      PRINTF("\n");
      /* hack to make address "final" */
      if (state == ADDR_TENTATIVE) {
        uip_ds6_if.addr_list[i].state = ADDR_PREFERRED;
      }
    }
  }
}
/*---------------------------------------------------------------------------*/
static void
set_global_address(void)
{
  uip_ipaddr_t ipaddr;

  uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
  uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
  uip_ds6_addr_add(&ipaddr, 0, ADDR_AUTOCONF);

  /* set server address */
  uip_ip6addr(&server_ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 1);

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

  PROCESS_PAUSE();

  set_global_address();

  PRINTF("UDP client process started\n");
```

```
 print_local_addresses();

 /* new connection with remote host */
 client_conn = udp_new(NULL, UIP_HTONS(UDP_SERVER_PORT),
 NULL); udp_bind(client_conn, UIP_HTONS(UDP_CLIENT_PORT));

 PRINTF("Created a connection with the server
 "); PRINT6ADDR(&client_conn->ripaddr);
 PRINTF(" local/remote port %u/%u\n",
      UIP_HTONS(client_conn->lport), UIP_HTONS(client_conn->rport));

 while(1) {
   PROCESS_YIELD();
   if(ev == tcpip_event) {
     tcpip_h&ler();
   }
 }

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

7.1.2 Sink.c File

```
#include "contiki.h"
#include "contiki-lib.h"
#include "contiki-net.h"
#include "net/ip/uip.h"
```

```
#include "net/rpl/rpl.h"
#include "net/linkaddr.h"

#include "net/netstack.h"
#include "dev/button-sensor.h"
#include "dev/serial-line.h"
#if CONTIKI_TARGET_Z1
#include "dev/uart0.h"
#else
#include "dev/uart1.h"
#endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "collect-common.h"
#include "collect-view.h"

#define DEBUG DEBUG_PRINT
#include "net/ip/uip-debug.h"

#define UIP_IP_BUF  ((struct uip_ip_hdr *)&uip_buf[UIP_LLH_LEN])

#define UDP_CLIENT_PORT 8775
#define UDP_SERVER_PORT 5688

static struct uip_udp_conn *server_conn;

PROCESS(udp_server_process, "UDP server process");
AUTOSTART_PROCESSES(&udp_server_process,&collect_common_process); /*-
---------------------------------------------------------------------
*/ void
```

```
collect_common_set_sink(void)
{
}
/*---------------------------------------------------------------------------*/
void
collect_common_net_print(void)
{
  printf("I am sink!\n");
}
/*---------------------------------------------------------------------------*/
void
collect_common_send(void)
{
  /* Server never sends */
}
/*---------------------------------------------------------------------------*/

void
collect_common_net_init(void)
{
#if CONTIKI_TARGET_Z1
  uart0_set_input(serial_line_input_byte);
#else
  uart1_set_input(serial_line_input_byte);
#endif
  serial_line_init();

  PRINTF("I am sink!\n");
}
/*-------------------------------------------------------------------------*/
static void
tcpip_h&ler(void)
{
 uint8_t *appdata;
 linkaddr_t sender;
 uint8_t seqno;
 uint8_t hops;

 if(uip_newdata()) {
   appdata = (uint8_t *)uip_appdata;
   sender.u8[0] = UIP_IP_BUF->srcipaddr.u8[15];
   sender.u8[1] = UIP_IP_BUF->srcipaddr.u8[14];
   seqno = *appdata;
   hops = uip_ds6_if.cur_hop_limit - UIP_IP_BUF->ttl + 1;s
   printf("DATA recv '%u' from %u \n", seqno, sender.u8[0]);
   collect_common_recv(&sender, seqno, hops,
                       appdata + 2, uip_datalen() - 2);
 }
}
/*---------------------------------------------------------------------------*/
static void
print_local_addresses(void)
{
 int i;
 uint8_t state;

 PRINTF("Server IPv6 addresses: ");
 for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
  state = uip_ds6_if.addr_list[i].state;
  if(state == ADDR_TENTATIVE || state == ADDR_PREFERRED) {
   PRINT6ADDR(&uip_ds6_if.addr_list[i].ipaddr); PRINTF("\n");

   /* hack to make address "final" */
   if (state == ADDR_TENTATIVE) {
     uip_ds6_if.addr_list[i].state = ADDR_PREFERRED;
```

```
      }
    }
  }
}
/*---------------------------------------------------------------------------
*/ PROCESS_THREAD(udp_server_process, ev, data) {

 uip_ipaddr_t ipaddr;
 struct uip_ds6_addr *root_if;

 PROCESS_BEGIN();

 PROCESS_PAUSE();

 SENSORS_ACTIVATE(button_sensor);

 PRINTF("UDP server started\n");

#if UIP_CONF_ROUTER
 uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 1); /*
 uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr); */
 uip_ds6_addr_add(&ipaddr, 0, ADDR_MANUAL);
 root_if = uip_ds6_addr_lookup(&ipaddr); if(root_if
 != NULL) {
   rpl_dag_t *dag;
   dag = rpl_set_root(RPL_DEFAULT_INSTANCE,(uip_ip6addr_t *)&ipaddr);
   uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0); rpl_set_prefix(dag, &ipaddr,
   64);
   PRINTF("created a new RPL dag\n");
 } else {
   PRINTF("failed to create a new RPL DAG\n");
 }
```

```
#endif /* UIP_CONF_ROUTER */

 print_local_addresses();

 /* The data sink runs with a 100% duty cycle in order to
   ensure high packet reception rates. */
 NETSTACK_RDC.off(1);

 server_conn = udp_new(NULL, UIP_HTONS(UDP_CLIENT_PORT),
 NULL); udp_bind(server_conn, UIP_HTONS(UDP_SERVER_PORT));

 PRINTF("Created a server connection with remote address ");
 PRINT6ADDR(&server_conn->ripaddr);
 PRINTF(" local/remote port %u/%u\n", UIP_HTONS(server_conn->lport),
       UIP_HTONS(server_conn->rport));

  while(1) {
   PROCESS_YIELD();
   if(ev == tcpip_event) {
    tcpip_h&ler();
   } else if (ev == sensors_event && data == &button_sensor)
    { PRINTF("Initiaing global repair\n");
    rpl_repair_root(RPL_DEFAULT_INSTANCE);
   }
  }

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