

Analysis and 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 and Engineering/Information Technology

By

Mehul Singh (131300)

Under the supervision of

Mr. Arvind Kumar

To



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology Wagnaghat,
Solani-173234, Himachal Pradesh**

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled “**Analysis and Implementation of Sinkhole Attack in RPL** ” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of Computer Science & Engineering, Jaypee University of Information Technology Wagnaghat is an authentic record of our own work carried out over a period from July 2016 to December 2016 under the supervision of **Mr. Arvind Kumar** Assistant Professor, Department of Computer Science And Engineering. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Mehul Singh (131300)

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

Mr. Arvind Kumar
Assistant Professor
Department of Computer Science and Engineering
Dated:

ACKNOWLEDGEMENT

We are grateful and indebted to Mr. Arvind Kumar, Assistant professor, Department of Computer Science And Engineering for his help and advice in completion of this project report. We also express our deep sense of gratitude and appreciation to our guide for his constant supervision, inspiration and encouragement right from the beginning of this Project report. We also want to thank our parents and friends for their immense support and confidence upon us. We deem it a pleasant duty to place on record our sincere and heartfelt gratitude to our project guide for his long sightedness, wisdom and co-operation which helped us in tackling crucial aspects of the project in a very logical and practical way.

Mehul Singh
(131300)

TABLE OF CONTENTS

Serial Number	Topics	Page Numbers
	Candidate's Declaration.....	i
	Acknowledgement.....	ii
	Table Of Contents.....	iii
	List of Figures.....	v
	List of Tables.....	vi
	List of Abbreviations.....	vii
	Abstract.....	viii
1	Introduction.....	1
1.1	The Internet of Things.....	1
1.1.1	About IOT.....	1
1.1.2	Growth of the IOT.....	2
1.1.3	Long Time to Value.....	3
1.1.4	Challenges.....	3
1.2	Problem Statement.....	4
1.3	Objective.....	4
1.4	Methodology.....	4
1.5	Organization of Project Report.....	4
2	Literature Survey.....	5
2.1	6LoWPAN.....	5
2.2	RPL.....	5
2.2.1	DODAG Building Process.....	6
2.2.2	Storing and Non Storing Nodes.....	7
2.2.3	Loop Avoidance and Loop Detection.....	8
2.2.4	Global and Local Repair.....	8
2.3	Taxonomy of Attacks in RPL based Internet of Things.....	9
2.3.1	Sinkhole Attack.....	9
3	System Development.....	10
3.1	Use of RPL Network.....	10
3.2	Network Topology.....	10
3.2.1	Random Placement of Nodes.....	10
3.2.2	Linear Placement of Nodes.....	12
4	Performance Analysis.....	13
4.1	Attack on Random Placement of Nodes.....	13
4.1.1	Analysis of Sensor Map.....	13
4.1.2	Analysis of average power consumption.....	14
4.1.3	Analysis of average radio duty cycle.....	19
4.1.4	Analysis of power history.....	20
4.1.5	Detection.....	22

4.2	Attack on Linear Placement of Nodes.....	23
4.2.1	Analysis of Sensor Map.....	23
4.2.2	Analysis of average power consumption.....	25
4.2.3	Analysis of average radio duty cycle.....	27
4.2.4	Analysis of power history.....	29
4.2.5	Detection.....	30
5	Conclusion.....	31
5.1	Conclusions.....	31
5.2	Future Scope.....	31
6	References.....	32
7	Appendices.....	34
7.1	Code Snippets.....	34
7.1.1	Sender.c File.....	34
7.1.2	Sink.c File.....	40

LIST OF FIGURES

Serial Number	Figure Name	Page Numbers
1.1	Application of IOT.....	2
1.2	Growth of IOT.....	3
2.1	DODAG Building Process.....	7
2.2	Taxonomy of Attacks.....	9
3.1	Random Placement of Nodes without attack.....	11
3.2	Random Placement of Nodes with attack on Node 16/21.....	11
3.3	Linear Placement of Nodes without attack.....	12
3.4	Linear Placement of Nodes with attack on Node 16/21.....	12
4.1	Network without Attacked Node.....	14
4.2	Network with Attack Node.....	14
4.3	Average Power Consumption without Attack Node.....	17
4.4	Average power consumption with attack node 16/21.....	18
4.5	Average radio duty cycle without attack node.....	19
4.6	Average radio duty cycle with attack node 16/21.....	20
4.7	Historical power consumption of node 16 without attack.....	21
4.8	Historical power consumption of node 16 with attack.....	21
4.9	Mote output of Node 16 without Attack.....	22
4.10	Mote output of Node 16 with Attack.....	22
4.11	Network without Attacked Node.....	23
4.12	Network with Attack Node.....	24
4.13	Average Power Consumption without Attack Node.....	26
4.14	Average power consumption with attack node 5/11.....	27
4.15	Average radio duty cycle without attack node.....	28
4.16	Average radio duty cycle with attack node 5/11.....	28
4.17	Historical power consumption of node 5 without attack.....	29
4.18	Historical power consumption of node 5 with attack.....	29
4.19	Mote output of Node 5 without Attack.....	30
4.20	Mote output of Node 5 with Attack.....	30

LIST OF TABLES

Serial Number	Table Name	Page Numbers
2.1	MOP field description.....	8
4.1	Total Packets received by each Node.....	15
4.2	Analysis of Power Consumption without attack.....	16
4.3	Analysis of Power Consumption with attack on Node 16.....	17
4.4	Total Packets received by each Node.....	24
4.5	Analysis of Power Consumption without attack.....	25
4.6	Analysis of Power Consumption with attack on Node 5/11.....	26

LIST OF ABBREVIATIONS

1. IP	Internet Protocol
2. IEEE	Institute of Electrical and Electronics Engineers
3. LLN	Low power and Lossy network
4. RPL	Routing Protocol for LLN
5. DIO	DODAG Information Object
6. DAO	Destination Advertisement Object
7. TCP	Transmission Control Protocol
8. DODAG	Destination Oriented Directed Acyclic Graph
9. WSN	Wireless Sensor Network
10. OF	Objective Function

ABSTRACT

The routing protocol for low power and lossy networks (RPL) is recommended by internet engineering task force (IETF) for IPv6 based Low power personal area network (6LowPAN). RPL is proactive routing protocol for internet of things that has applications in smart homes, smart cities and smart world. RPL creates directed acyclic graphs (DAG) of the network topology.

In this project we introduce a sinkhole attack in which the attacker's aim is to attract more traffic from a particular area through a compromised node. In RPL this attack can be easily performed through the manipulation of the rank. By falsifying (decreasing) the rank of a node one can make it attract more traffic and thus sinkhole attack can be implemented. Because of this falsified advertisement, the malicious node is more frequently chosen as parent by the other nodes.

Then we have analyzed the topology, average power, average radio duty cycle and the power history of the malicious node.

From the analysis we conclude that the power and radio duty cycle due to malicious node increases.

CHAPTER 1

INTRODUCTION

1.1 The Internet-of-Things

1.1.1 About IoT

As discussed in [1] [2] [4] “The Internet of things (stylized Internet of Things or IoT) is the internetworking of physical gadgets, vehicles (additionally alluded to as "associated gadgets" and "smart gadgets"), structures and different things—embedded with electronics, software’s, sensors, actuators, and system network that empower these articles to gather and trade information.” In 2013 the Global Standards Initiative on Internet of Things (IoT-GSI) characterized the IoT as “the foundation of the data society.” IoT is basically connecting all the devices of our day to day life to internet so as to make our life easier.

"Things," in IoT refers to every device from lights to vehicles etc. of our day to day life. These things are connected to a network so as to simplify our life work.

Researchers looks "Things" as an "inseparable blend of hardware, software, data and service ". These devices gather information from the nearby environment.

IoT is the leading research subject as nowadays focus is shifting towards ubiquitous computing from traditional computing.

IoT covers different aspects like transportation, construction, medical sciences, home appliances, shopping experiences etc.



Figure 1.1 Applications of IoT

1.1.2 Growth of the IoT

IoT is one of the leading research topic as it is new and a large no of people around 87% have even not heard of it. But Iot has been in our life from very long as ATMs which date back to 1980's. It is predicted that around Billion objects will be connected to the internet by 2020 out of which around 250 million will be vehicles.

Even the market of smart watches has grown drastically in the past few years.

So it is evident that IoT is in demand and devices connected to the internet are going to increase drastically.

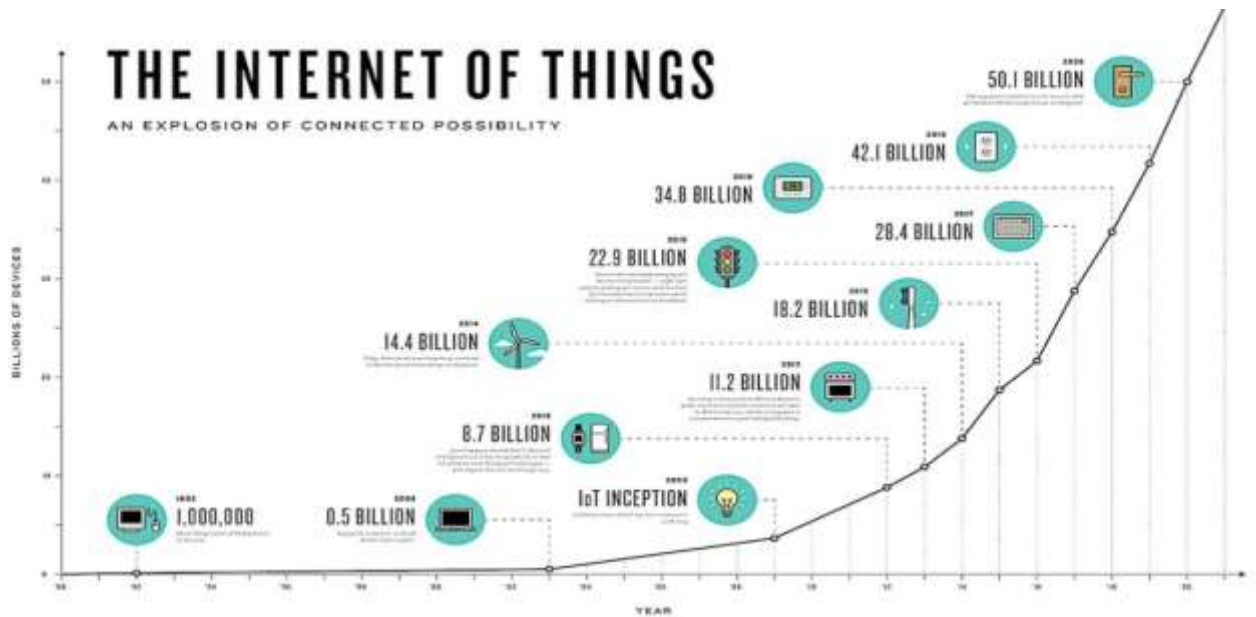


Figure 1.2 Growth of IoT

1.1.3 Long time to value

IoT projects can take quite a while. From business case improvement or development to verification of idea for full-scale rollout, each phase of the procedure can be laden with difficulties. In light of our work with clients, we prescribe that organizations take after a five-stage procedure to limit the time required to convey their IoT ventures, while amplifying the return on these activities.

There are many programs available that helps new organizations to support them for the development of various IoT projects. Even the government is focusing on IoT by implementing new schemes like Smart Cities, free Wi-Fi Zones etc.

1.1.4 Challenges

In this Report we discuss about Sink Hole attack whose identification and avoidance is a major challenge. As we know that this particular attack changes the rank of a particular node, thus making it attract more traffic which in turn changes the topology of the network. This attack if coupled with some other attack like black hole can cause a major challenge for the user.

1.2 Problem Statement

Implementation and detection of Sinkhole attack and analysis of power consumption, change in topology and average radio duty cycle.

1.3 Objective

The aim is to analyze and detect a **Sinkhole** attack on a low power and lossy network in RPL.

1.4 Methodology

In this project we simulate a Sinkhole Attack based on RPL on the Linux based simulator COOJA. Our prime methodology is to attack a node which makes faulty route by responding fake network information to the information source, and intercepts data through that faulty route.

1.5 Organization of Project Report

In Chapter 1 we have discussed about IOT basics, the current growth in this field, the common challenges being faced by persons in implementing the IOT structures.

In Chapter 2 we will be discussing about the basic terminology mentioned in our topic. We will be providing with facts and figures about different concepts we studied about those terms in different research papers.

In Chapter 3 we are going to provide a model of how the project is done on the basis of developments:-

- Analytical
- Experimental
- Statistical

In Chapter 4 we have given a proper analysis of sinkhole attack.

In Chapter 5 we have provided with the conclusion that we derive from our project.

CHAPTER 2

LITERATURE SURVEY

A literature review is a means to evaluate and interpret all available research relevant to a particular research question, or area. Its main aim is to present a fair evaluation of the research area of interest by conducting a rigorous and auditable methodology. The main purpose of our literature review is to find the relevant literature about Sink Hole Attack on LLNs and their network protocols for the purpose of background study, summarize the existing work and identify the gap in the current research.

2.1 6LoWPAN

As discussed in [9][13][14] 6LoWPAN concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices," and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.

The 6LoWPAN group has defined encapsulation and header compression mechanisms that allow IPv6 packets to be sent and received over IEEE 802.15.4 based networks. IPv4 and IPv6 are the work horses for data delivery for local-area networks, metropolitan area networks, and wide-area networks such as the Internet.

2.2 RPL

As discussed in [5][10] "RPL is a Distance Vector IPv6 routing protocol for LLNs that dictate the Destination Oriented Directed Acyclic Graph (DODAG) building process using an objective function and a set of metrics/constraints." The objective function uses a combination of metrics and constraints to compute the 'best' path.

The objective function is the key towards the formation of the DODAG based on some network constraints.

2.2.1 DODAG Building Process

The building process of the graph starts at the root node or the sink node, which is configured by the system administrator. The RPL routing protocol provides a set of new ICMPv6 control messages to communicate graph related information between different nodes. These messages are:

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

The graph building process starts when the root starts broadcasting its information using the DIO message. The neighboring nodes will receive and process DIO messages from all the nodes and makes their decision whether to connect or not based on certain rules. Once the node has joined a graph it has a route toward the graph root. The graph root is termed as the ‘parent’ of the node. Then the node computes its ‘rank’ which specifies its position in the network. If it is not a leaf node then it will again send DIO messages to all its neighboring nodes. If the node is a “leaf node”, it simply joins the graph and does not send any DIO message. This process will continue until the leaf node is reached. This rippling effect builds the graph edges out from the root to the leaf nodes where the process terminates. In this formation each node of the graph has a routing entry towards its parent and the leaf nodes can send a data packet all the way to root of the graph by just forwarding the packet to its immediate parent. The various steps of the graph building process are represented in Figure 2.1[5].

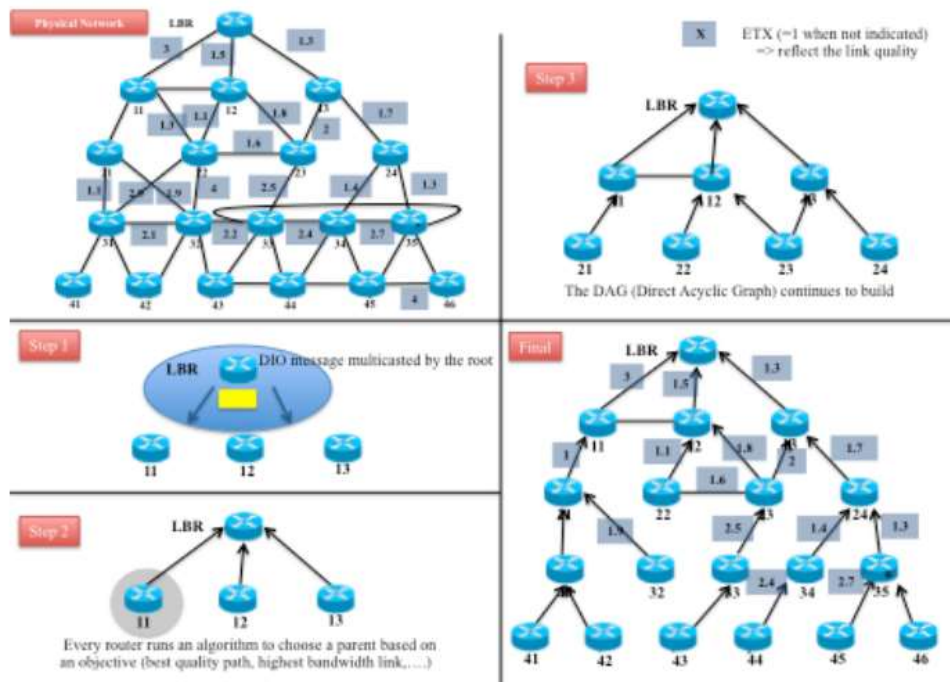


Figure 2.1 DODAG Building Process

DAO messages are used to send the nodes information in the upward direction towards its parent. As each node receives the DAO message, it processes the information and adds an entry in the routing table. This process continues until the information reaches the root and a complete path to the root is setup.

RPL also supports “point-to-point (P2P) communication from any node to any other node in the graph.” When a node sends a packet to another node within the network, the packet travels ‘upwards’ to a common parent at which point it is forwarded in the ‘down’ direction to the destination.

2.2.2 Storing and Non-storing nodes

Basically there are two types of nodes i.e. storing and non-storing nodes.

In storing nodes due to some memory constraints it is unable to store routing entries in the routing table while in case of storing nodes it will store the routing table at each node.

This mode can be set by using the MOP bit of RPL packet header. MOP is a 4 bit field.

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 MOP field description

2.2.3 Loop Avoidance and Loop Detection

Loops are a major concern in any network, therefore it is important to detect and avoid them. Loops are basically formed due to changes in some topology and lack of synchronization between nodes.

RPL provides certain mechanism for loop avoidance and detection based on some rules. The two rules specified in the RPL are:

- Max_Dept Rule, which states that a node can't make a node its parent whose rank is more than the current node
- A node is not allowed to change its rank to attract more traffic.[5]

2.2.4 Global and Local Repair

There are basically two repair mechanisms that RPL supports namely Global and Local Repair.

Local Repair is launched when there is a link failure or no path is found between nodes.

While Global Repair is launched after local repair, because after local repair is launched shape of the graph may start to change therefore global repair is required to maintain the shape of the graph. Global Repair constructs the graph from the scratch.

2.3 Taxonomy of Attacks in RPL based Internet of Things

[6][7] Discussed that the RPL protocol designed for IPv6 is exposed to a wide variety of attacks and mainly divided into three categories. The first category of attacks targets the exhaustion of network resources such as, memory, energy and power. The second category of attacks in RPL targets the network topology. The attacks on topology is further divided into two categories: sub-optimization attacks and isolation attacks. The third category targets the RPL network traffic.

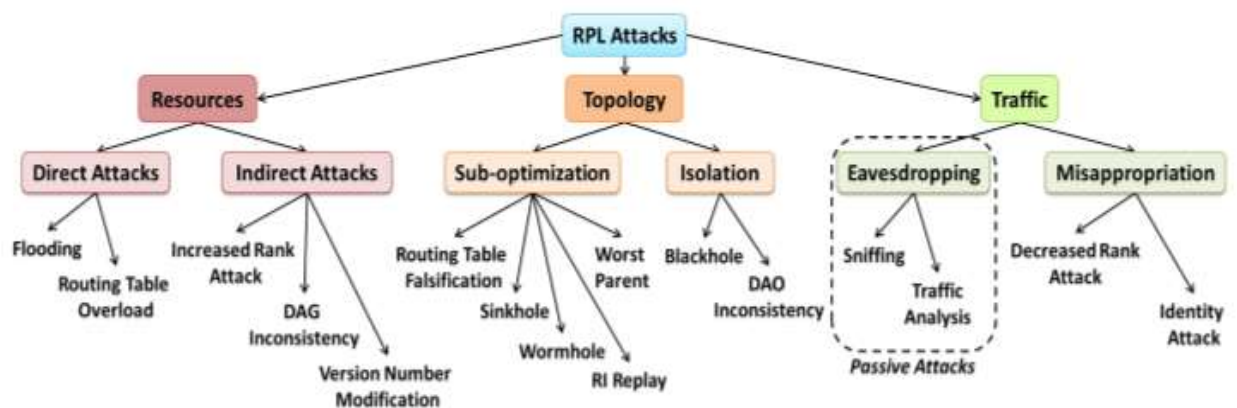


Figure 2.2 Taxonomy of Attacks

2.3.1 Sinkhole Attack

In a sinkhole attack, the attacker's aim is to attract more traffic from a particular area through a compromised node. In RPL this attack can be easily performed through the manipulation of the rank. By falsifying (decreasing) the rank of a node one can make it attract more traffic and thus sinkhole attack can be implemented. Because of this falsified advertisement, the malicious node is more frequently chosen as parent by the other nodes.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Use of RPL Network

In a sinkhole attack, a malicious node decreases its rank. By doing this, the malicious node can attract more traffic. Simulations are done using Cooja Simulator that consists of the collection of all network protocols. To simulate sinkhole attack, a new protocol is added into the Cooja Simulator. The protocol is written using C language which involves the implementation of DODAG which comes under RPL networking. Having implemented the routing protocol which simulates the sinkhole, network performance is compared with and without sinkholes in the network.

3.2 Network Topologies

We implemented our attack on two topologies which are discussed below:

- Random Placement of Nodes (fig 3.1)
- Linear Placement of Nodes (fig 3.3)

3.2.1 Random Placement of Nodes

Firstly we implemented our attack on a random Placement of 20 nodes in which Node 1 was the sink node and we implemented our attack on Node 16 which was renamed as Node 21 as shown in fig 3.2. We made node 16 as malicious by changing the code of the objective function that was used to form the node. In the code we basically changed the `calculate_rank` function so that the node communicates its faulty rank and thus attracting more traffic through it.

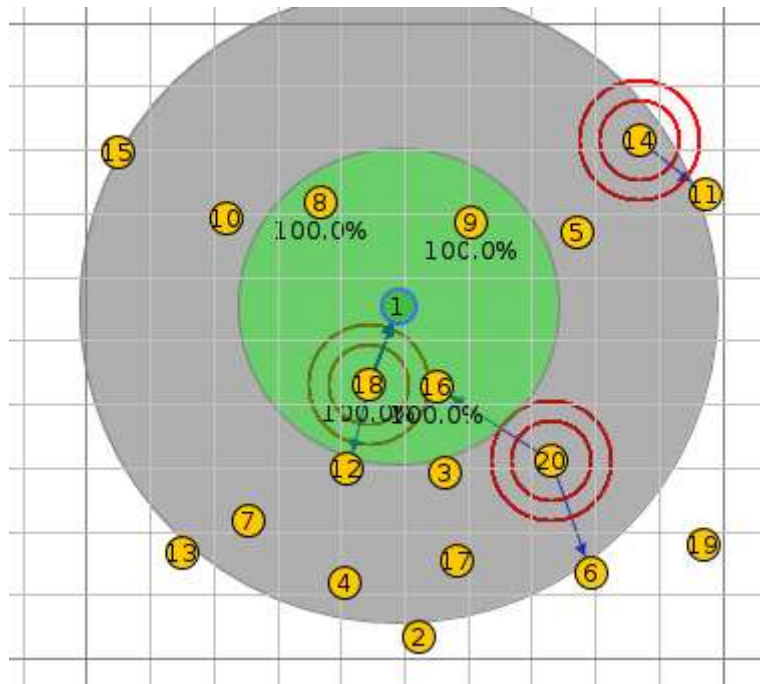


Figure 3.1 Random Placement of Nodes without Attack



Figure 3.2 Random Placement of Nodes with Attack on Node 16/21

3.2.2 Linear Placement of Nodes

Then we implemented our attack on a linear placement of 10 nodes in which Node 1 was the sink node and we implemented our attack on Node 5 which was renamed as Node 11 as shown in fig 3.4. We made node 5 as malicious by changing the code of the objective function that was used to form the node. In the code we basically changed the calculate_rank function so that the node communicates its faulty rank and thus attracting more traffic through it.

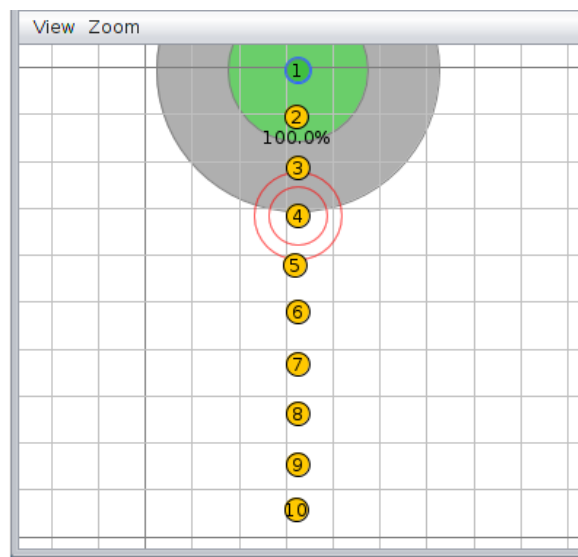


Figure 3.3 Linear Placement of Nodes without Attack

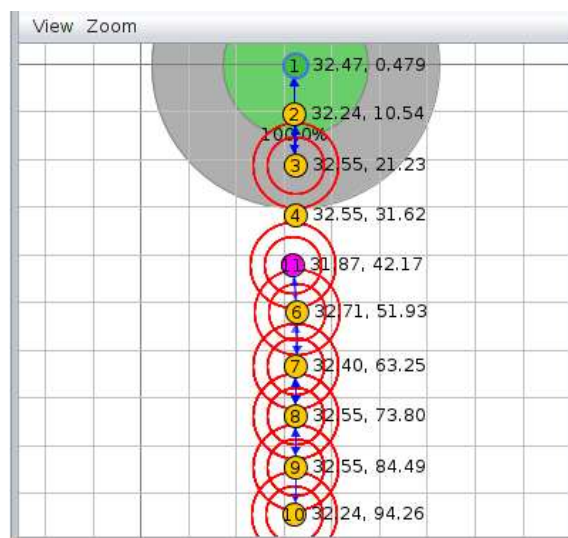


Figure 3.4 Linear Placement of Nodes with Attack on Node 5/11

CHAPTER 4

PERFORMANCE ANALYSIS

In this Chapter we analyzed a network in two different radio environments.

We will discuss some of the differences that we observed in both the scenarios like:-

- Sensor Map
- Average Power Consumption
- Average Radio Duty Cycle
- Power History
- Detection

4.1 Attack on Random Placement of Nodes

4.1.1 Analysis of Sensor Map

“Sensor map depicts the relationship between parent and child nodes.”

In the scenario given below, Node 1 is the Sink node and Node 16 has no child node and its parent is Node 18 as shown in Fig 4.1.

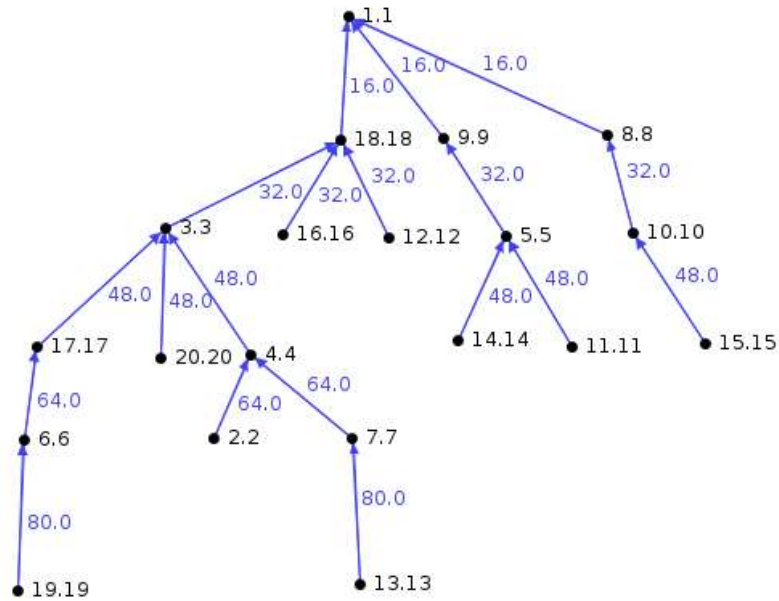


Figure 4.1 Network without Attacked node

Now we implement our attack on Node 16 by attacking it and making it a malicious node. As you can observe from the figure 4.2 that now Node 21 (Actually Node 11) has now Node 3, Node 12 and Node 20 as its child. This confirms that our attack has been successfully implemented since now Node 21 attracts more traffic as compared to before and the network topology has changed.

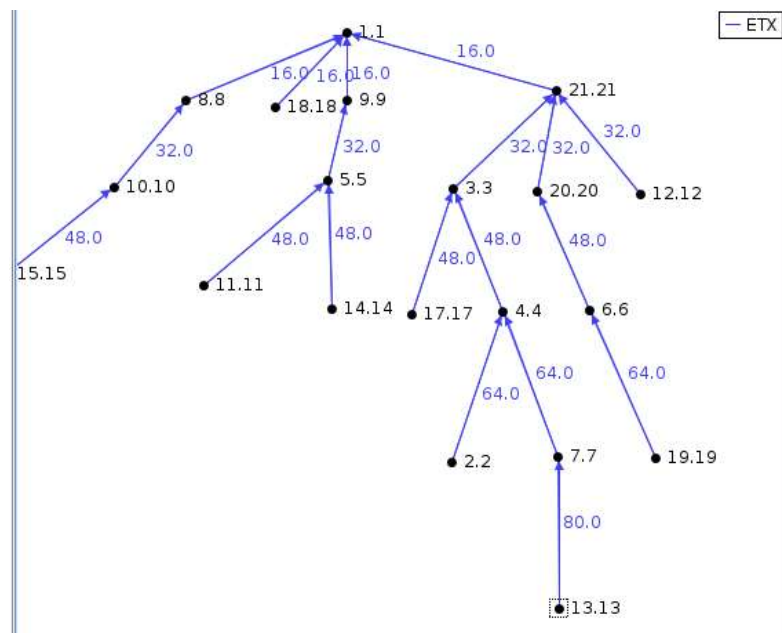


Figure 4.2 Network with Attacked node 16/21

Table 4.1 Total Packets received by each Node

Node	Without Attack	With Attack
1	0	0
2	14	12
3	12	14
4	12	11
5	13	13
6	13	12
7	13	12
8	12	13
9	11	12
10	13	14
11	11	12
12	12	12
13	9	13
14	12	10
15	11	14
16/21	11	12
17	14	13
18	14	11
19	12	14
20	14	10

As you can see from the table 4.1 that no. of packets received by Node 16 has increased from 11 to 12 because of its new child. This shows that Node 16 attracts more traffic when we implement our attack as compared to the normal environment.

4.1.2 Analysis of Average Power Consumption

“We define power consumption profiling as the process of parameterizing a network node (or nodes) power consumption in terms of its (their) workload.”

In the scenario given below, there is no attack implemented on Node 16. The power consumed by Node 16 is 1.115mW from table 4.2.

Table 4.2 Analysis of Power Consumption without attack

Node	Listen Power	Transmit Power	Power
2	0.460	0.124	1.065
3	0.649	0.387	1.611
4	0.562	0.264	1.361
5	0.513	0.272	1.296
6	0.493	0.167	1.156
7	0.504	0.177	1.179
8	0.436	0.040	0.959
9	0.458	0.061	1.008
10	0.493	0.259	1.252
11	0.458	0.113	1.049
12	0.484	0.113	1.110
13	0.450	0.183	1.112
14	0.471	0.151	1.104
15	0.449	0.145	1.069
16	0.471	0.135	1.115
17	0.639	0.328	1.521
18	0.579	0.082	1.181
19	0.429	0.113	1.009
20	0.458	0.106	1.059
Average	0.498	0.169	1.169

Fig 4.3 shows the power consumed by all the nodes in the graphical format without attack.

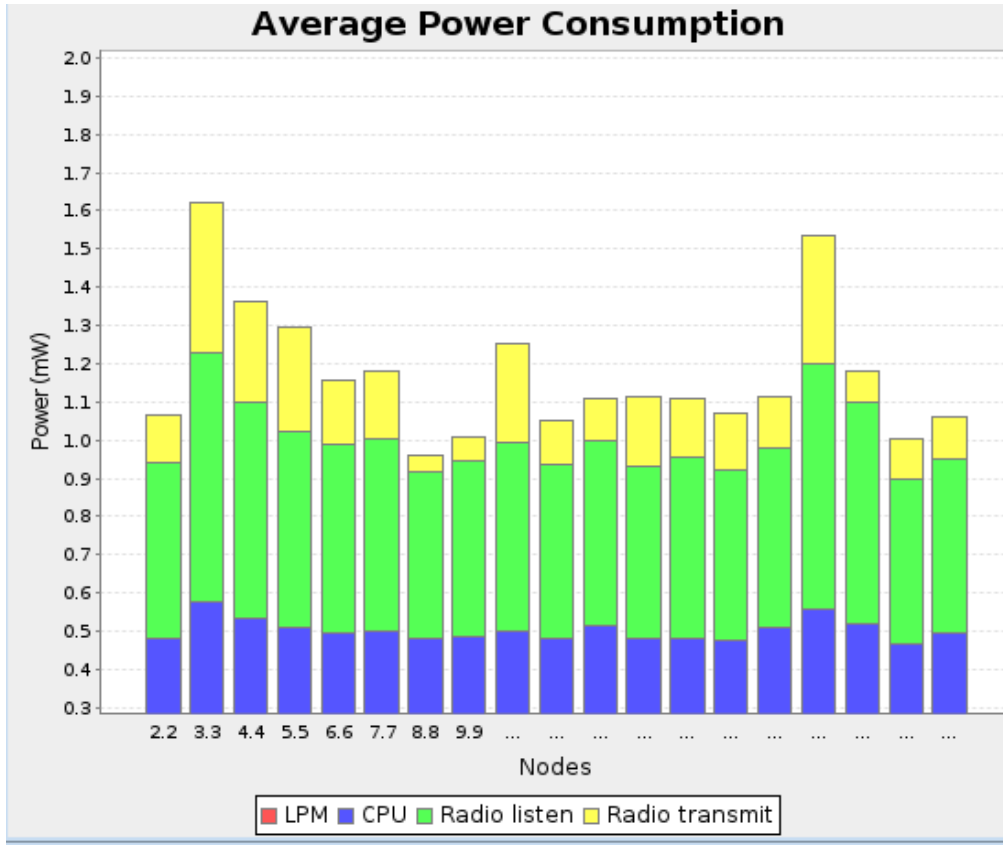


Figure 4.3 Average Power Consumption without Attacked Node

In the scenario given below, there is Sinkhole attack implemented on Node 16. The power consumed by Node 16 is 1.242mW from table 4.3. So, we can observe that due to the attack the power consumed by Node 16 has increased by 0.127mW.

Table 4.3 Analysis of Power Consumption with attack on Node 16

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
16/21	0.601	0.100	1.242
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

Fig 4.4 shows the power consumed by all the nodes in the graphical format with attack on Node 16/21.

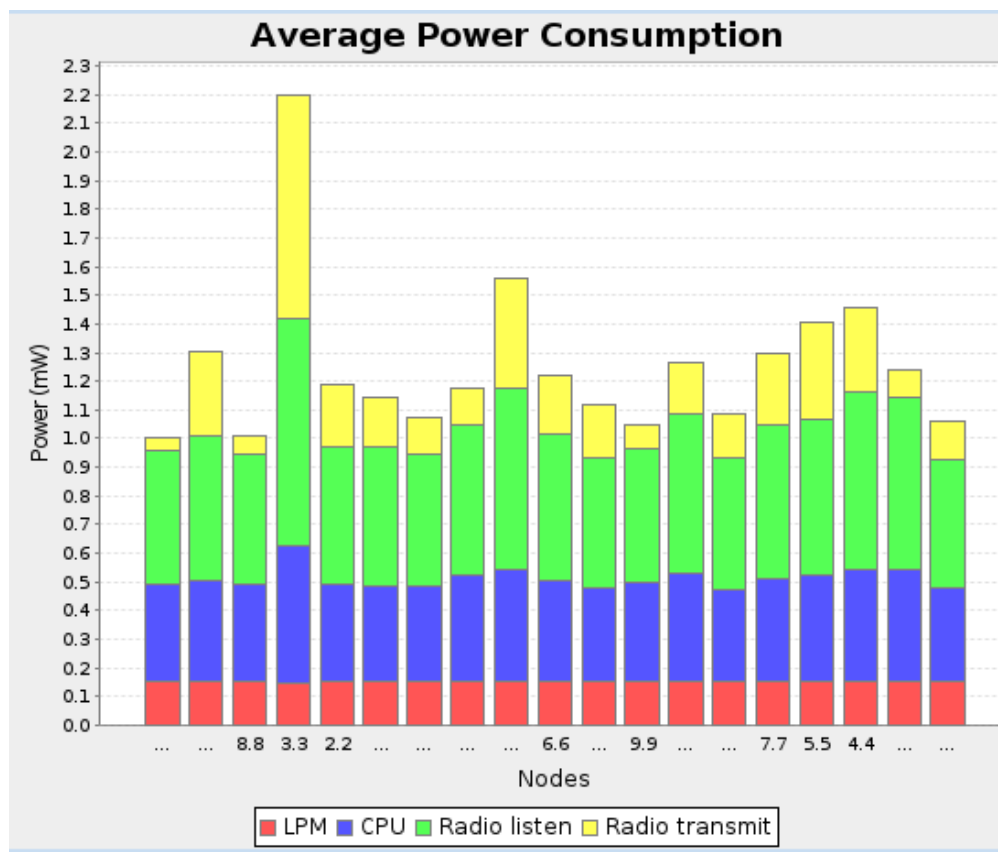


Figure 4.4 Average Power Consumption with Attacked Node 16/21

4.1.3 Analysis of Average Radio Duty Cycle

“Duty cycle (or duty factor) is a measure of the fraction of the time a radar is transmitting. It is important because it relates to peak and average power in the determination of total energy output.”

In the scenario given below, there is no attack implemented on Node 16. The Average Radio Duty Cycle of Node 16 is 1.06 from fig 4.5.

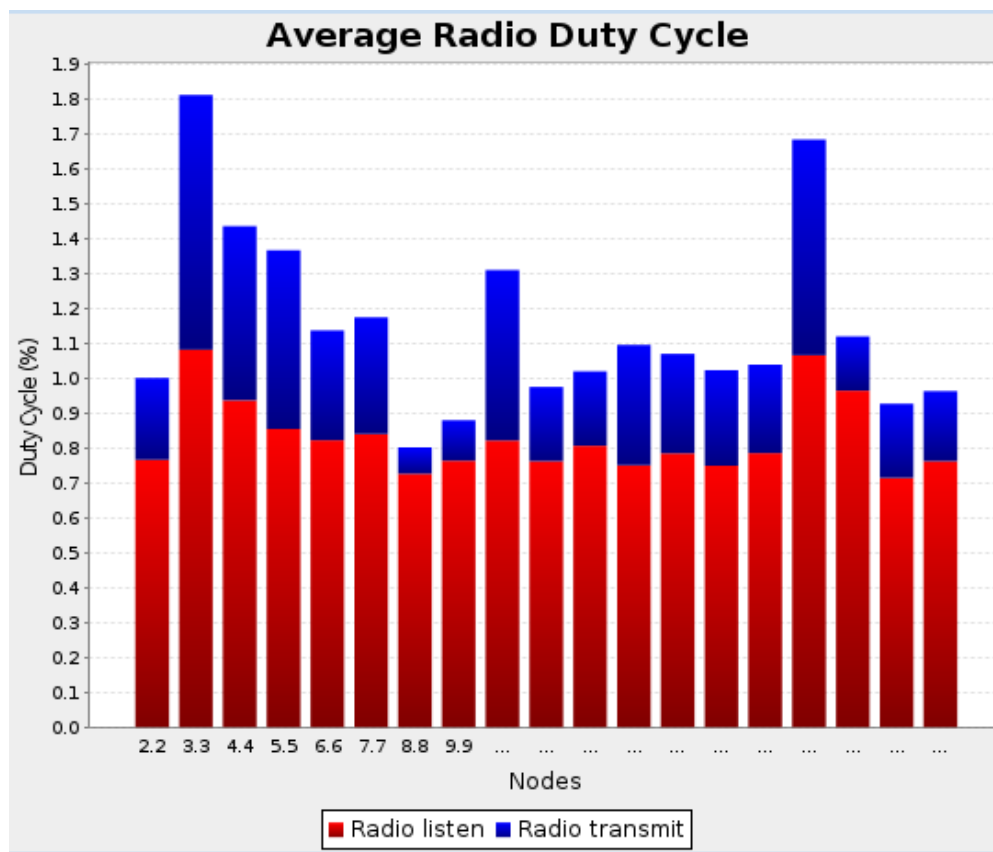


Figure 4.5 Average Radio Duty Cycle without Attacked Node

In the scenario given below, there is Sinkhole attack implemented on Node 16. The Average Radio Duty Cycle of Node 16 is 1.2 from fig 4.6. So, we can observe that due to the attack the Average Radio Duty Cycle of Node 16 has increased by a factor of 0.14.

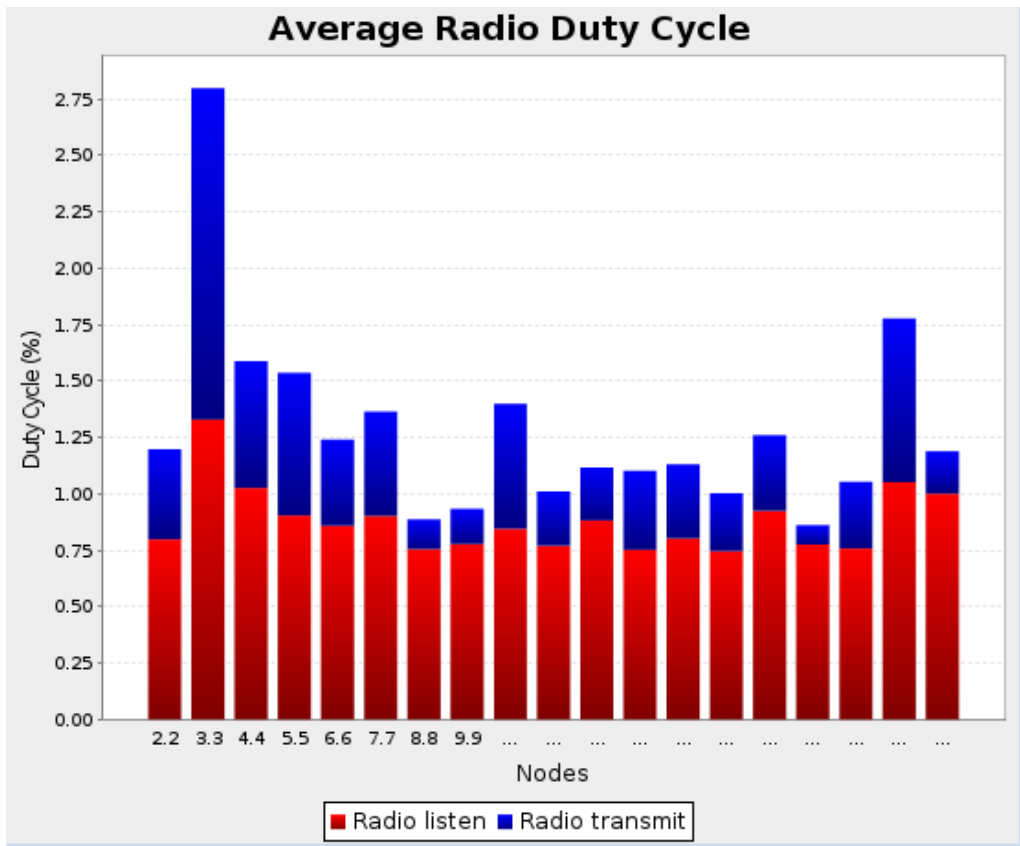


Figure 4.6 Average Radio Duty Cycle with Attacked Node 16/21

4.1.4 Analysis of Power History

In the scenario given below, there is no attack implemented on Node 16. Node 16's initial power consumption is high but overall power consumption is less because it has no child as shown in fig 4.7.

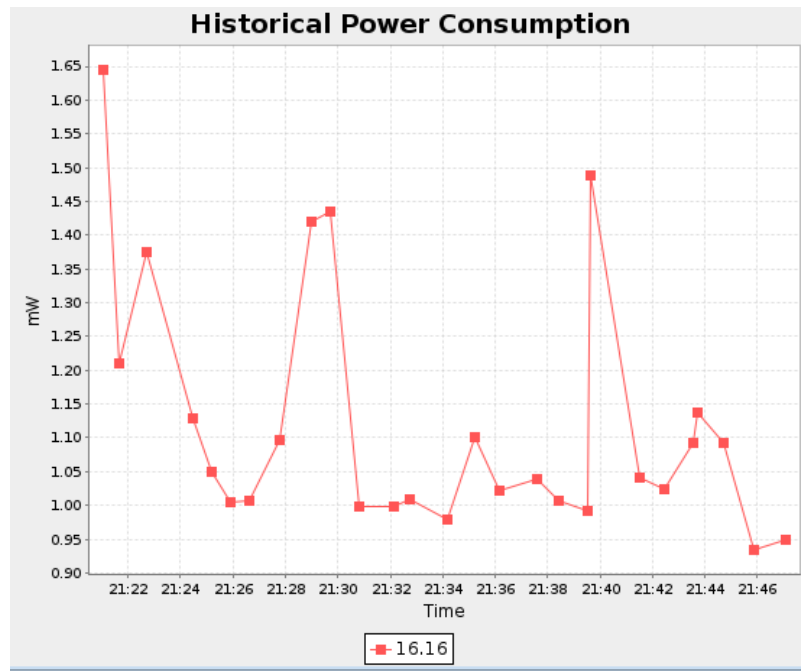


Figure 4.7 Historical Power Consumption of Node 16 without Attack

In the scenario given below, there is Sinkhole attack implemented on Node 16. Node 16's initial power consumption is high and even the overall power consumption is also high as it has many nodes as its children which can send their packets through it as shown in fig 4.8.

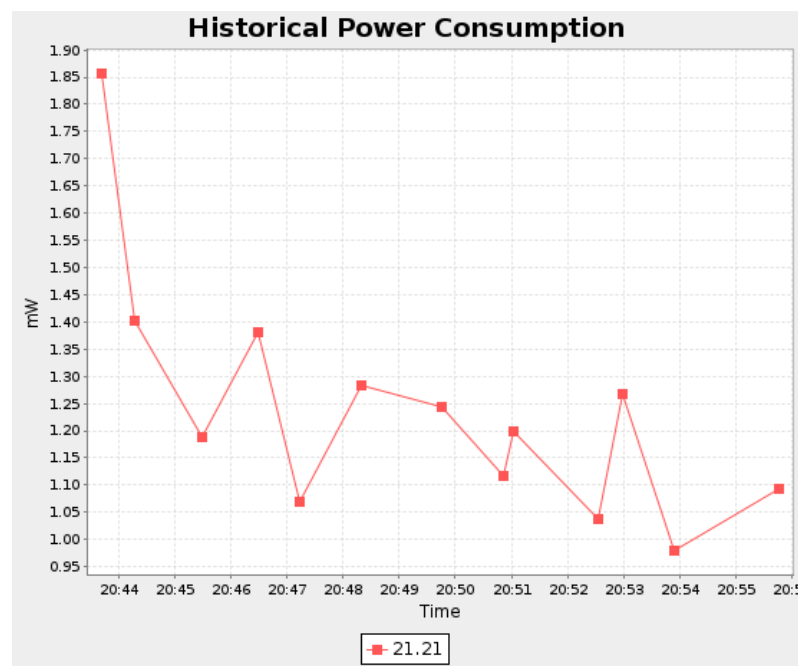


Figure 4.8 Historical Power Consumption of Node 16/21 with Attack

4.1.5 Detection

Algorithm for the detection of Sink Hole Attack

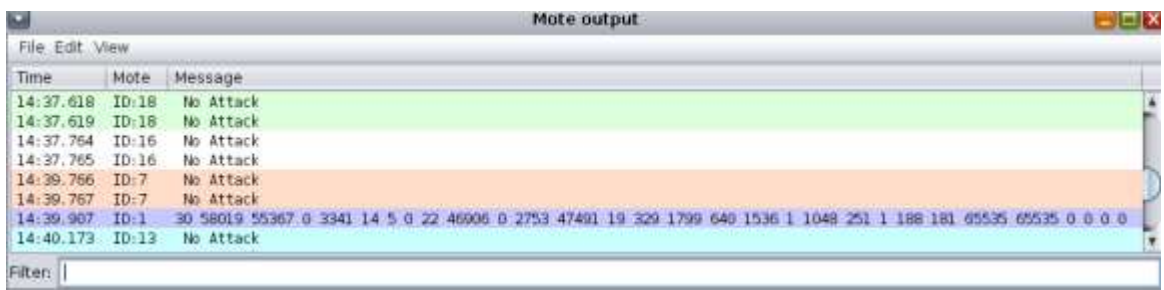
for all Nodes

if Node.rank \leq Node.Parent.Rank

then Sink Hole Attack

else No Attack

When we implemented this algorithm, then the results can easily be seen in the mote output window as you can see in the fig 4.9.



The screenshot shows a window titled "Mote output" with a menu bar (File, Edit, View) and a table of log entries. The table has three columns: Time, Mote, and Message. The entries are as follows:

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
14:39.967	ID:1	30 58019 55367 0 3341 14 5 0 22 46906 0 2753 47401 19 329 1799 640 1536 1 1048 251 1 188 181 65535 65535 0 0 0 0
14:40.173	ID:13	No Attack

At the bottom of the window, there is a "Filter:" label followed by an empty text input field.

Figure 4.9 Mote output of Node 16 without Attack

In the fig 4.10 it is shown that no attack was implemented on Node 16.



The screenshot shows a window titled "Mote output" with a menu bar (File, Edit, View) and a table of log entries. The table has three columns: Time, Mote, and Message. The entries are as follows:

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

At the bottom of the window, there is a "Filter:" label followed by the text "Sink" in the input field.

Figure 4.10 Mote output of Node 16/21 with Attack

And as you can see in the above figure Sink Hole attack is implemented on Node 21 which was originally Node 16.

So this implies that our detection algorithm worked properly.

4.2 Attack on Linear Placement of Nodes

4.2.1 Analysis of Sensor Map

“Sensor map depicts the relationship between parent and child nodes.”

In the scenario given below, Node 1 is the Sink node and Node 5 has no child node and its parent is Node 4 as shown in fig 4.11.

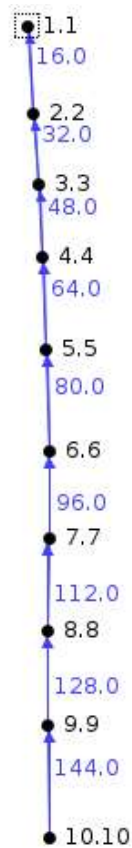


Figure 4.11 Network without Attacked node

Now we implement our attack on Node 5 by attacking it and making it a malicious node by changing its rank. But as in the linear placement of nodes Node 5/11 has only Node 4 and Node 6 in its listening vicinity, therefore there is no change in the topology since all nodes wants to be grounded as shown in fig 4.12.

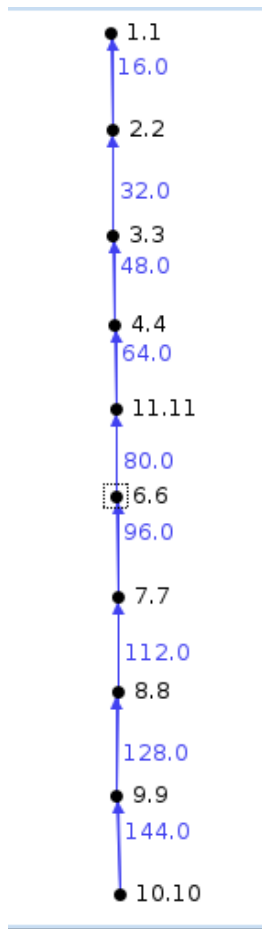


Figure 4.12 Network with Attack on Node 5/11

Table 4.4 Total Packets received by each Node

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

As you can see from the table 4.4 that no. of packets received by Node 5 has increased from 16 to 17. This shows that Node 5 attracts more traffic when we implement our attack as compared to the normal environment.

4.2.2 Analysis of Average Power Consumption

“We define power consumption profiling as the process of parameterizing a network node (or nodes) power consumption in terms of its (their) workload.”

In the scenario given below, there is no attack implemented on Node 5. The power consumed by Node 5 is 1.939mW from table 4.5.

Table 4.5 Analysis of Power Consumption without attack

Node	Listen Power	Transmit Power	Power
1	0	0	0
2	0.531	0.064	1.103
3	0.667	0.516	1.741
4	0.636	0.314	1.480
5	0.691	0.673	1.939
6	0.627	0.398	1.558
7	0.622	0.412	1.569
8	0.572	0.306	1.402
9	0.494	0.240	1.237
10	0.443	0.116	1.028
Avg	0.587	0.338	1.451

Fig 4.13 shows the power consumed by all the nodes in the graphical format without attack.

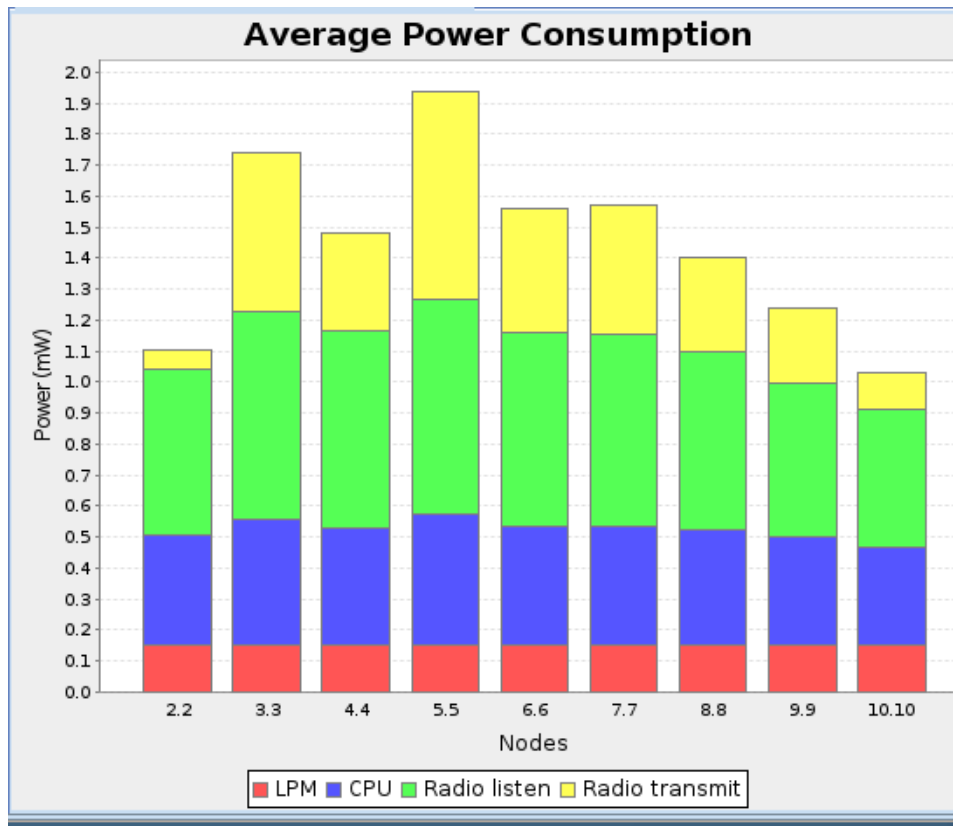


Figure 4.13 Average Power Consumption without Attacked Node

In the scenario given below, there is Sinkhole attack implemented on Node 5/11. The power consumed by Node 11 is 1.234mW from table 4.6. So, we can observe that due to the attack the power consumed by Node 11 has decreased by 0.705mW.

Table 4.6 Analysis of Power Consumption with attack on Node 5/11

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

Fig 4.14 shows the power consumed by all the nodes in the graphical format with attack on Node 5/11.

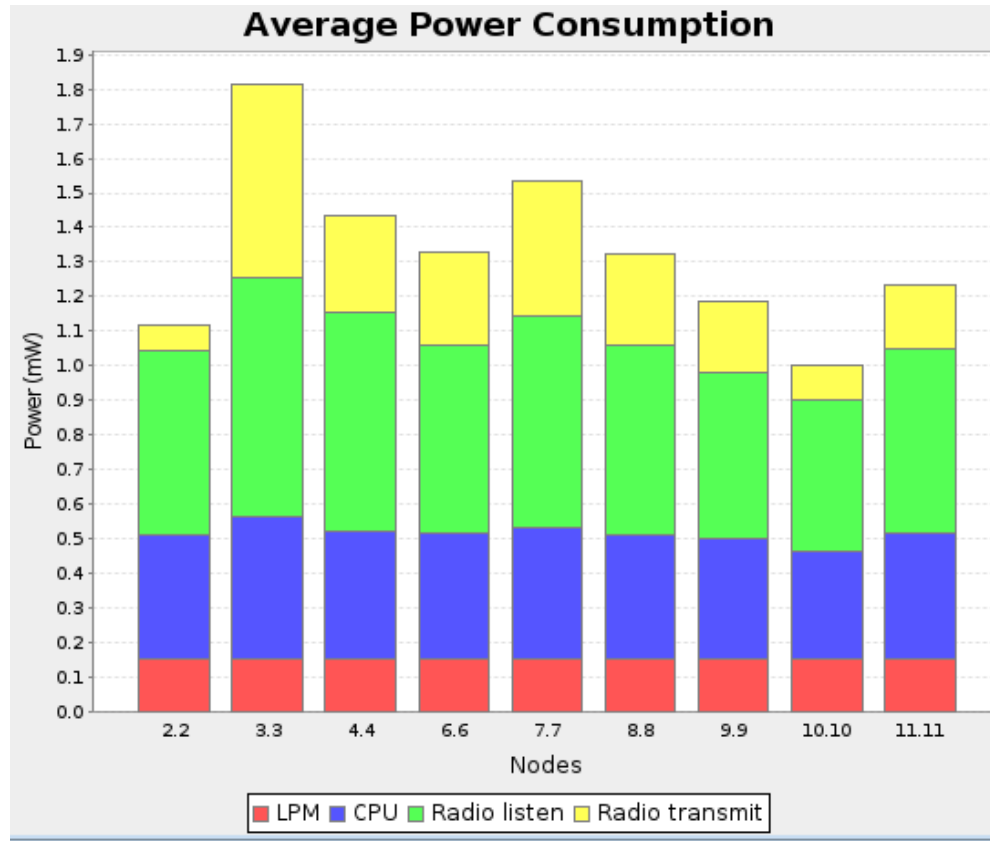


Figure 4.14 Average Power Consumption with Attack on Node 5/11

4.2.3 Analysis of Average Radio Duty Cycle

“Duty cycle (or duty factor) is a measure of the fraction of the time a radar is transmitting. It is important because it relates to peak and average power in the determination of total energy output.”

In the scenario given below, there is no attack implemented on Node 5. The Average Radio Duty Cycle of Node 5 is 2.35 from fig 4.15.

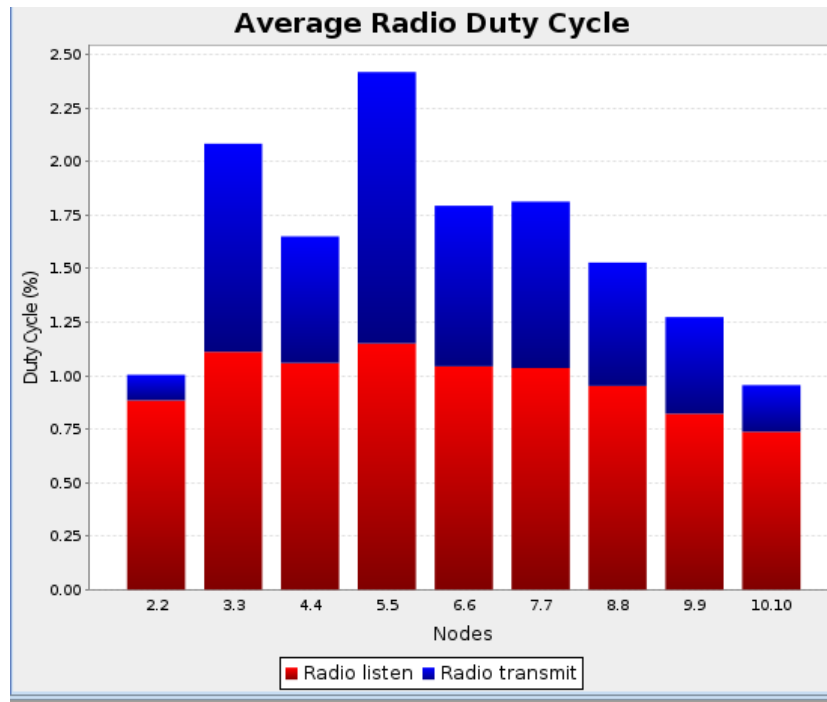


Figure 4.15 Average Radio Duty Cycle without Attacked Node

In the scenario given below, there is Sinkhole attack implemented on Node 5/11. The Average Radio Duty Cycle of Node 11 is 1.25 from fig 4.16. So, we can observe that due to the attack the Average Radio Duty Cycle of Node 11 has decreased by a factor of 1.1.

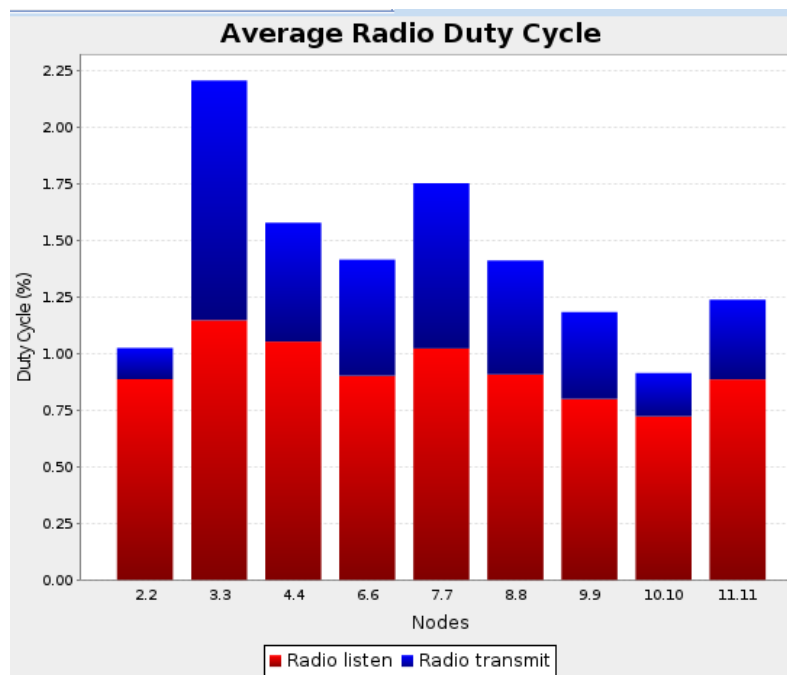


Figure 4.16 Average Radio Duty Cycle with Attack on Node 5/11

4.2.4 Analysis of Power History

In the scenario given below, there is no attack implemented on Node 5. Fig 4.17 shows the historical power consumption of Node 5 without attack.

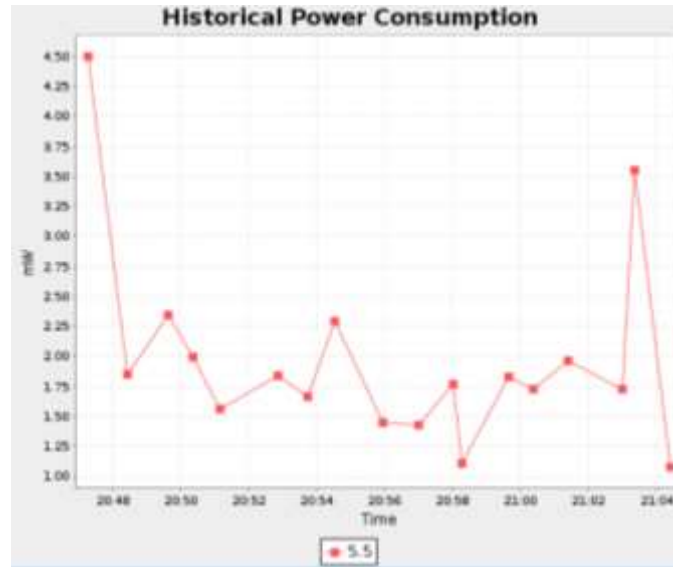


Figure 4.17 Historical Power Consumption of Node 5 without Attack

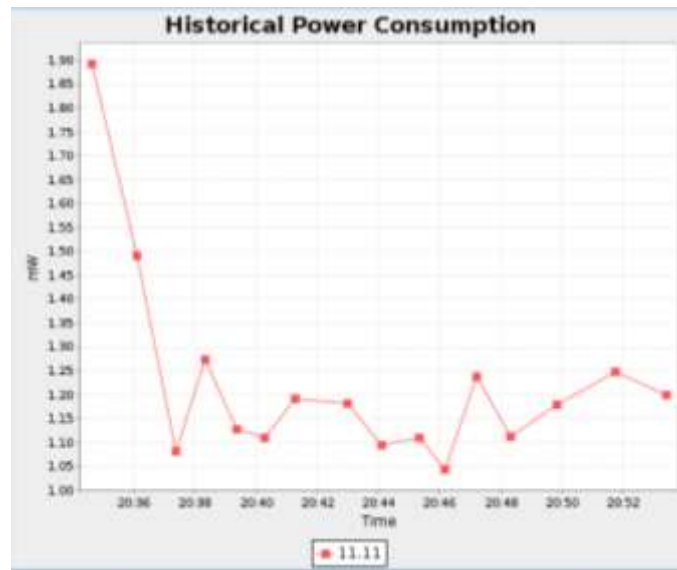


Figure 4.18 Historical Power Consumption of Node 5 with Attack

In the scenario given, there is Sinkhole attack implemented on Node 5. Since there is no change in topology therefore the power consumed by Node 5 is low in attack case as compared to normal case as shown in fig 4.18.

4.2.5 Detection

Algorithm for the detection of Sink Hole Attack

for all Nodes

if Node.rank \leq Node.Parent.Rank

then Sink Hole Attack

else No Attack

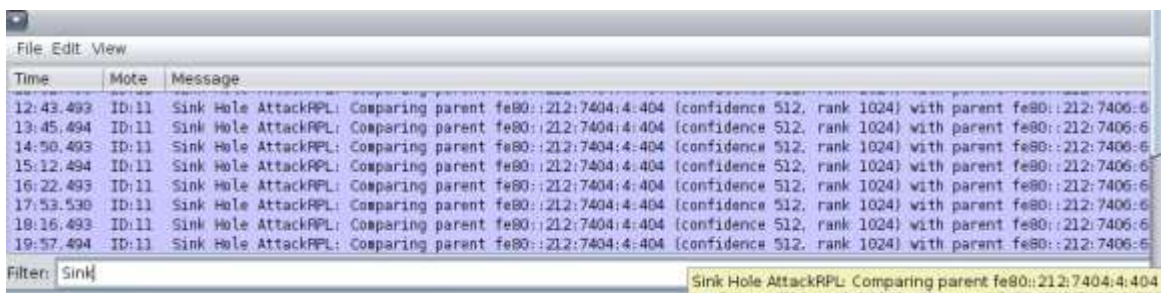
When we implemented this algorithm, then the results can easily be seen in the mote output window as you can see in the fig 4.19 given below.



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	No Attack
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	No Attack
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

Figure 4.19 Mote output of Node 5 without Attack

In the fig 4.20 it is shown that no attack was implemented on Node 5.



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::212:7406:6:606
13:45.494	ID:11	Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606
14:50.493	ID:11	Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606
15:12.494	ID:11	Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606
16:22.493	ID:11	Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606
17:53.530	ID:11	Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606
18:16.493	ID:11	Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606
19:57.494	ID:11	Sink Hole AttackRPL: Comparing parent fe80::212:7404:4:404 (confidence 512, rank 1024) with parent fe80::212:7406:6:606

Figure 4.20 Mote output of Node 5 with Attack

And as you can see in the above figure Sink Hole attack is implemented on Node11 which was originally Node 5.

So this implies that our detection algorithm worked properly.

CHAPTER 5

CONCLUSION

5.1 Conclusions

We implemented and detected the Sink Hole attack on two different topologies i.e. random placement of nodes and linear placement of node and successfully analyzed it by analyzing its sensor map, power, radio duty cycle and no of packets both in attacked environment and normal environment. In case of random placement we can observe major changes in the topology of the network while in case of linear placement there was no change in the topology of the network because all the nodes in the network wants to be grounded and since in linear arrangement all nodes only have a single parent in its preferred parent list therefore on changing the rank of one of the node it will not affect the network because no other node can make the faulty node as its parent.

We even implemented an algorithm for the detection of our attack which was successfully implemented as can be seen from the mote output window.

5.2 Future Scope

Study for understanding wireless network security in LLNs for attacks such as Worm hole will be done in the near future, and also our research for preventing sinkhole attack will be done. Developing skills to use Cooja is also important for us to deal with these networks.

CHAPTER 6

REFERENCES

- [1] Vikrant Negi internet of thing, seminar report ,2008,pp. 1-4.
- [2] Shi Yan-rong, Hou Tao, Internet of Things key technologies and architectures research in information processing in Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE), 2013
- [3] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini and Imrich Chlamtac, Internet of Things: Vision, applications and 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 Bertrand, Cedric Chauvenet, RPL: The IP routing protocol designed for low power and 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 and Lossy Networks, Internet Draft draft-ietf-roll-rpl-17 Retrieved, June 2015.
- [11] Contiki Operating Systems Website: <http://www.contiki-os.org> Retrieved:, 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 and Yuan L, 6LoWPAN: a study on QoS security threats and 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_handler(void)
{
    if(uiplib_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)
{
#ifdef 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);
    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, 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_handler();
    }
}

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_handler(void)
{

```

```

uint8_t *appdata;
linkaddr_t sender;
uint8_t seqno;
uint8_t hops;

if(uiplib_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;
    printf("DATA recvd '%u' from %u \n", seqno, sender.u8[0]);
    collect_common_recvd(&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");

#ifdef 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);
    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_handler();
    } else if (ev == sensors_event && data == &button_sensor) {
        PRINTF("Initiaing global repair\n");
        rpl_repair_root(RPL_DEFAULT_INSTANCE);
    }
}

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

```

