Analysis and Implementation of DAO Inconsistency 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

Akhil Sharma (131313)
Vishwas Malik (131324)

Under the supervision of

Mr. Arvind Kumar

to

Department of Computer Science & Engineering and 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 **"Preliminary Analysis of DAO Inconsistency 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, Waknaghat is an authentic record of our own work carried out over a period from August 2016 to May 2017 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.

Akhil Sharma (131313) Vishwas Malik (131324)

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

(Supervisor Signature)
Mr. Arvind Kumar
Assistant Professor
Department of Computer Science And Engineering
Dated:

ACKNOWLEGEMENT

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.

Akhil Sharma (131313)

Vishwas Malik (131324)

Computer Science and Engineering

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

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

Internet of Things (IoT) has been a very trending research topic in the recent times, where physical objects interconnect as a result of conjunction of various existing technologies. IoT is rapidly developing, but there have been uncertainties about its security and privacy which can affect its sustainability. The network layer which is both wired or wireless is exposed to many kinds of attacks. Because of the openness of these wireless channels, communications can be easily monitored.

In this project we introduce a DAO Inconsistency Attack in RPL networks in which an attacker makes malicious node which uses the "F" flag to make RPL routers remove legitimate downward routes and thus isolates nodes from DODAG Graph. This F flag is present in the rpl-extension-header.c and by modifying this flag this attack is implemented.

Mainly we modified the functions update_header_empty() and verify_header() to implement this attack.

In this project, a Low power and lossy Network is to be constructed, and analyze the results from the simulation of the DAO inconsistency by using the cooja simulator.

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.

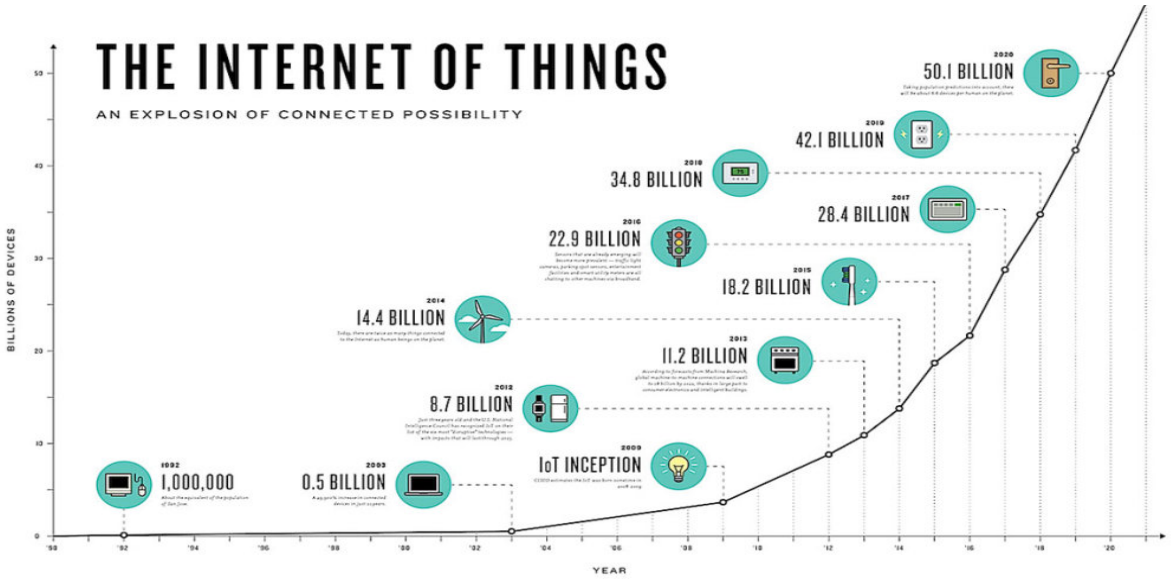

Figure 1.2 Growth of IoT

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 DAO inconsistency attack whose identification is a major challenge. As we know that this particular attack isolates the nodes that are associated with it thus the desired outcome of the network is not what we expect it to be and creates a problem of incomplete network which further hinders the collection of information from the desired isolated node.

1.2 Problem Statement

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

1.3 Aims and Objectives

The aim is to analyze a low power and lossy network in RPL.

1.4 Methodology

In this project we simulate a DAO Inconsistency attack based on RPL on the Linux based simulator COOJA. Our prime methodology is to detect the malicious node which sets the "F" Flag of the Data Packet and send it back to the node from where it received the Data Packet making the nodes below it isolated from the DODAG.

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 structure and finally the different types of network topologies.

In Chapter 2 we would be providing with the basic terminology about the different research paper read by us. We would be providing with facts and figures about different concepts we studied in those 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 DAO inconsistency attack on basis of which we would use the malicious node.

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 DAO inconsistency attack upon low power and lossy networks 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].

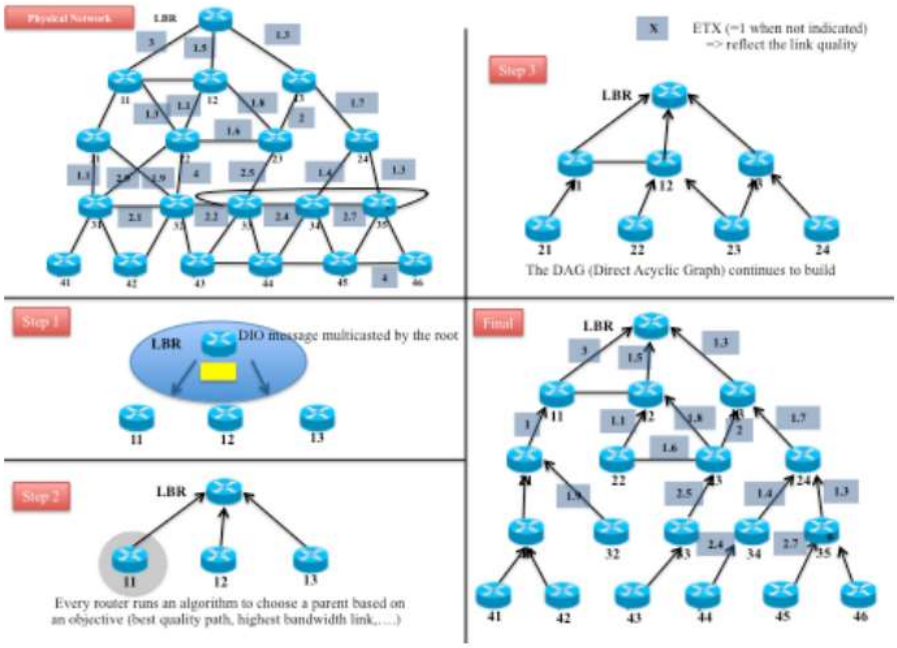

Figure 2.1 DODAG Building Process

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.

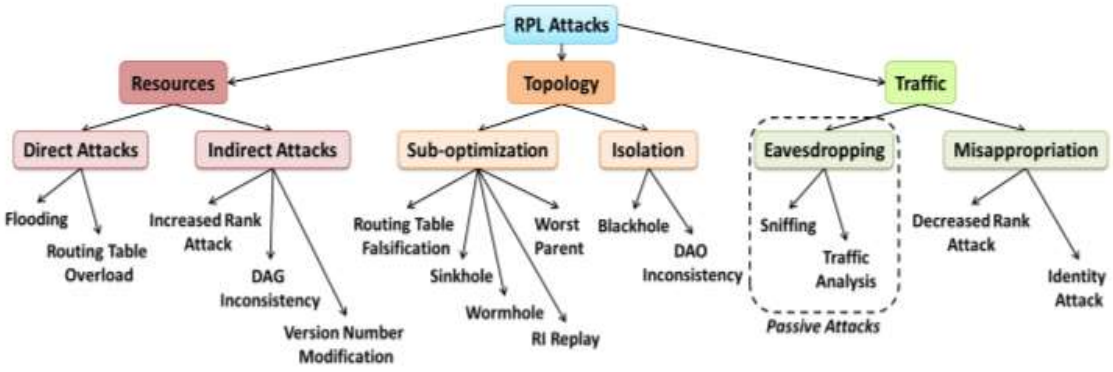

Figure 2.2 Taxonomy of Attacks

2.3.1 DAO Inconsistency Attack

A DAO Inconsistency Attack in RPL networks is in which an attacker makes malicious node which uses the "F" flag to make RPL routers remove legitimate downward routes and thus isolates nodes from DODAG Graph. This F flag is present in the rpl-extension-header.c and by modifying this flag this attack is implemented.

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 Use of RPL Network

In DAO inconsistency attack, a malicious node isolates all the nodes associated with it. By doing this, the malicious node cannot collect any DAO messages from the higher rank node linked with it. Simulations are done using Cooja Simulator that consists of the collection of all network protocols. To simulate DAO inconsistency attack a code is written using C language which involves the implementation of DODAG which comes under RPL networking. Having implemented the routing protocol which simulates the DAO inconsistency attack, network performance is compared with and without the attack in the network.

3.2 Network Topologies

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

- Grid Placement of Nodes (fig 3.1)
- Linear Placement of Nodes (fig 3.4)

3.2.1 Grid Placement of Nodes

We implemented our attack on a Grid placement of 35 nodes in which Node 35 was the sink node and we implemented our attack on Node 8 in first scenario as shown in fig 3.2 which was renamed as Node 36 and on node 28 in second scenario as shown in fig 3.3 which was also renamed as node 36. We made node 8 and node 28 as malicious by changing the code of the rpl-ext-header.c and rpl-icmp6.c.

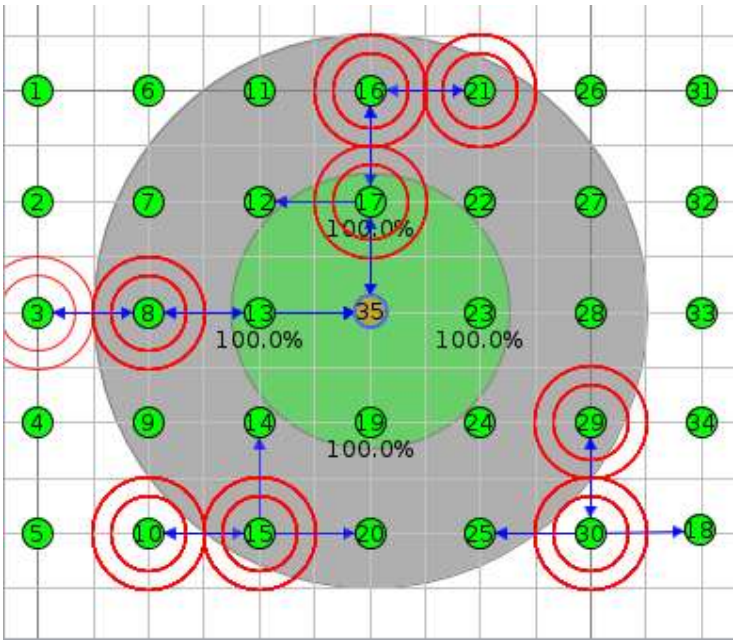

Figure 3.1 Grid Placement of Nodes without Attack

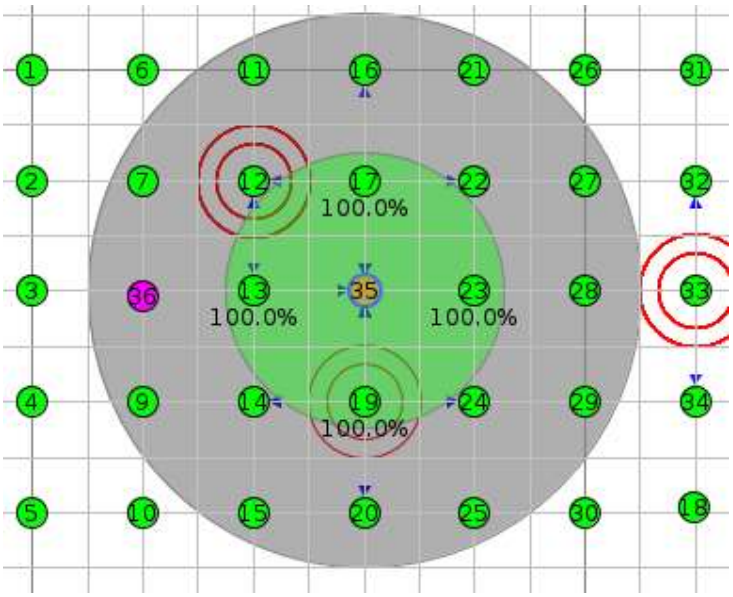

Figure 3.2 Grid Placement of Nodes with Attack (Node 8/36)

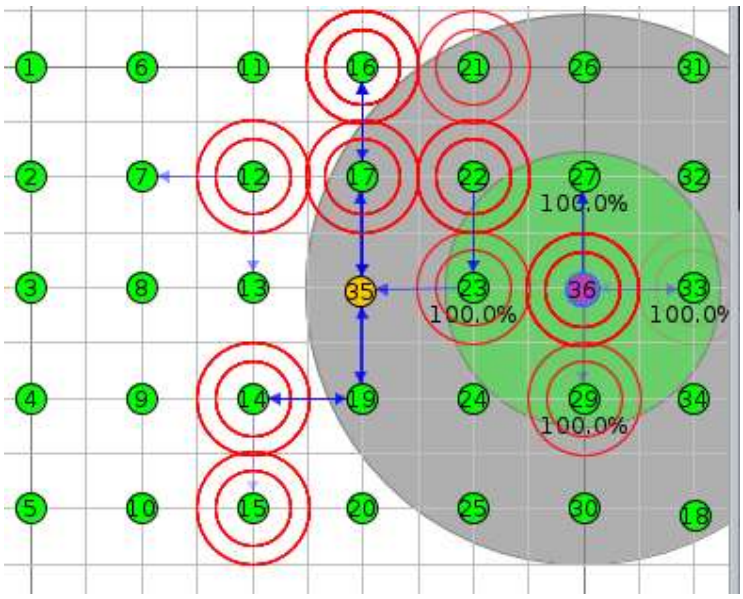

Figure 3.3 Grid Placement of Nodes with Attack (Node 28/36)

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 3.5. We made node 5 as malicious by changing the code of rpl-ext-header.c and rpl-icmp6.c.

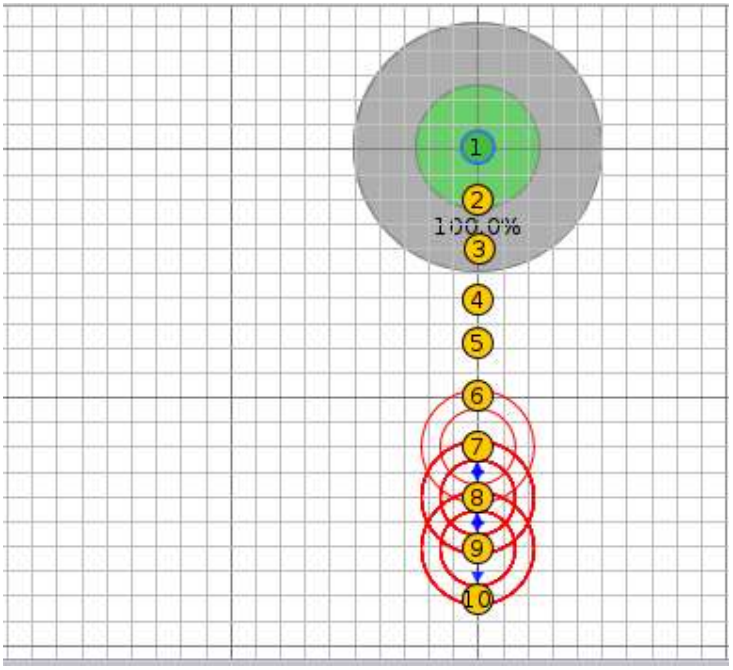

Figure 3.4 Linear Placement of Nodes without Attack

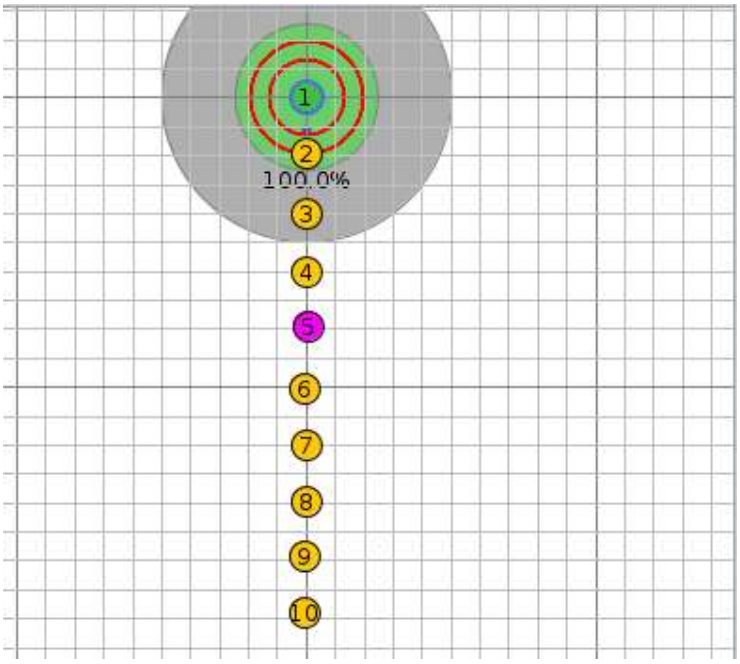

Figure 3.5 Linear Placement of Nodes with Attack (Node 5/11)

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

4.1 Attack on Grid Placement of Nodes

4.1.1 Scenario 1 (With Attack on Node 8)

4.1.1.1 Analysis of Sensor Map

"Sensor map depicts the relationship between parent and child nodes."

In the given scenario Node 35 is the Sink node and Node 8 has Node 3 and Node 7 as its child node and its parent is Node 13 as shown in fig 4.1.

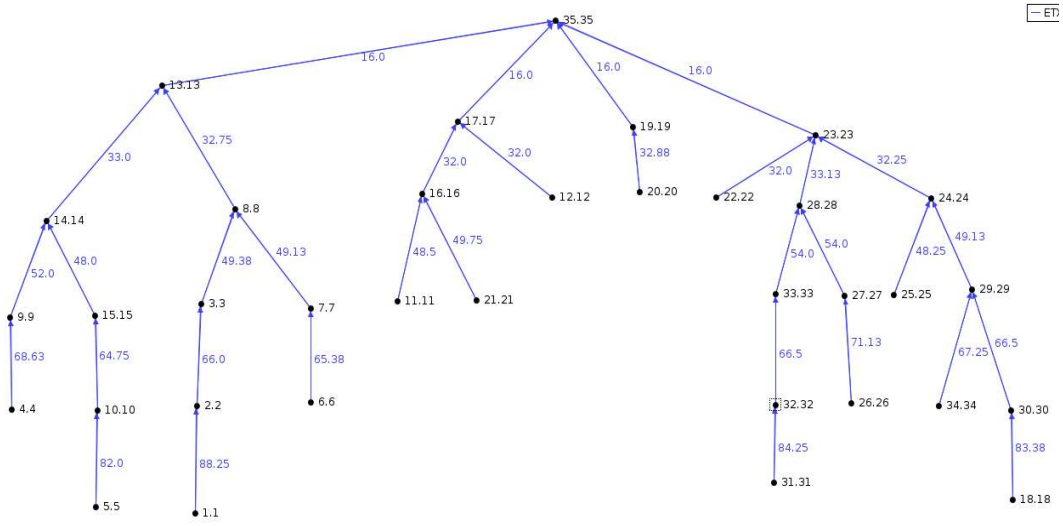

Figure 4.1 Network without Attacked node

Now we implement our attack on Node 8 as Node 36 by attacking it and making it a malicious node. As you can observe from the figure 4.2 that now Node 36 (Actually Node 8) has now no child. This confirms that our attack has been successfully implemented since now Node 36 has isolated its children from the root and the network topology has changed.

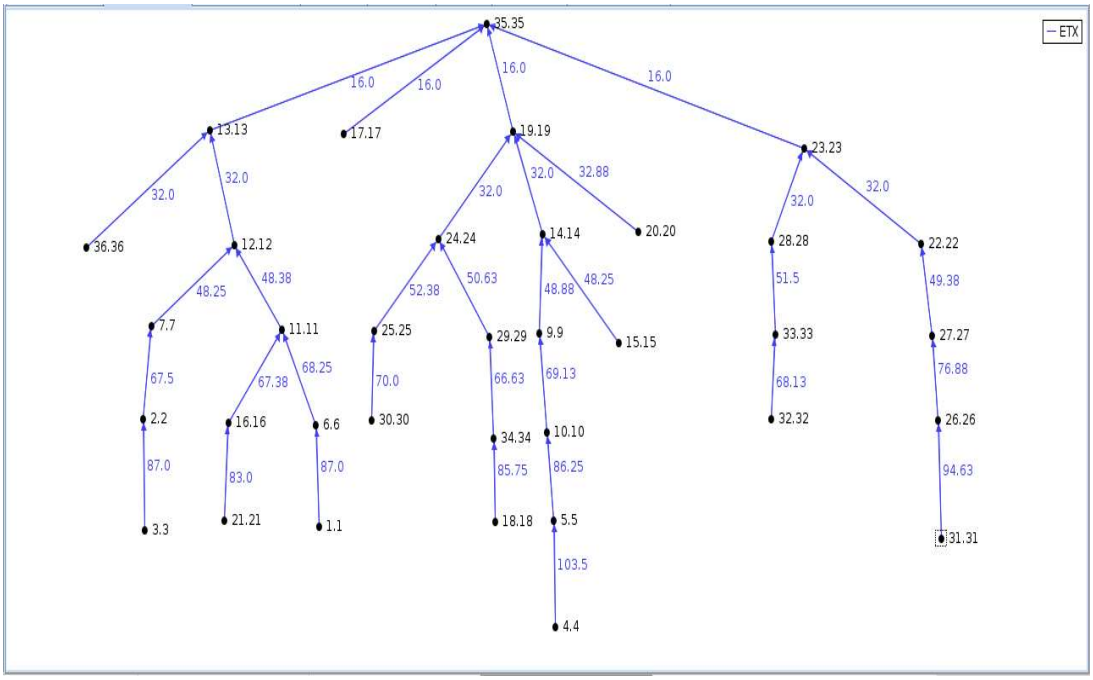

Figure 4.2 Network with Attacked node 8

Table 4.1 Total Packets received by each Node

Node	Without Attack	With Attack
1	28	12
2	28	13
3	20	13
4	29	14
5	29	13
6	28	14

7	27	13
8/36	28	13
9	29	13
10	28	13
11	29	14
12	20	12
13	28	11
14	28	13
15	29	14
16	28	14
17	20	13
18	29	14
19	29	14
20	29	13
21	29	14
22	26	13
23	29	13
24	29	14
25	29	12
26	29	14
27	29	13
28	28	13
29	28	14
30	20	13
31	28	13
32	20	13
33	29	11
34	28	11
35	0	0

As you can see from the table 4.1 that no. of packets received by Node 8/36 has decreased from 28 to 13 because of no children associated with it.

4.1.1.2 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 given scenario, there is no attack implemented on Node 8. The power consumed by Node 8 is 1.527mW from table 4.2.

Table 4.2 Analysis of Power Consumption without attack

Node	Listen Power	Transmit Power	Power
1	0.510	0.103	1.121
2	0.465	0.132	1.077
3	0.484	0.109	1.096
4	0.494	0.227	1.227
5	0.556	0.328	1.405
6	0.536	0.248	1.305
7	0.713	0.244	1.514
8	0.634	0.348	1.527
9	0.531	0.254	1.296
10	0.499	0.234	1.240
11	0.528	0.219	1.263
12	0.730	0.547	1.850
13	0.623	0.415	1.588
14	0.572	0.339	1.438
15	0.464	0.136	1.084
16	0.532	0.286	1.332
17	0.519	0.222	1.250
18	0.466	0.146	1.095
19	0.511	0.288	1.321
20	0.536	0.260	1.313
21	0.465	0.150	1.107
22	0.467	0.133	1.080
23	0.449	0.124	1.063
24	0.551	0.245	1.315
25	0.720	0.553	1.845

26	0.557	0.276	1.357
27	0.511	0.155	1.164
28	0.483	0.197	1.180
29	0.473	0.141	1.114
30	0.661	0.170	1.373
31	0.648	0.458	1.666
32	0.574	0.344	1.443
33	0.535	0.269	1.320
34	0.510	0.103	1.121
Avg	0.549	0.249	1.364

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

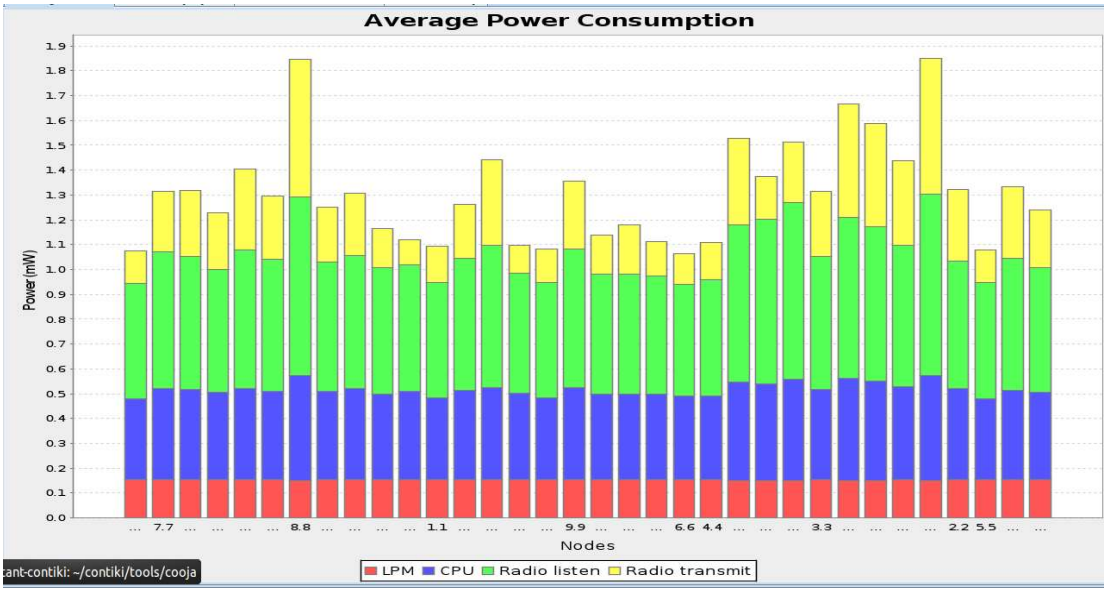

Figure 4.3 Average Power Consumption without Attacked Node

In the given scenario, there is DAO inconsistency attack implemented on Node 8. The power consumed by Node 8 is 1.213mW from table 4.3. So, we can observe that due to the attack the power consumed by Node 8 has decreased by 0.314mW since node 8 has no child and the number of packets received by it has decreased.

Table 4.3 Analysis of Power Consumption with attack on Node 8

Node	Listen Power	Transmit Power	Power
1	0.478	0.180	1.148
2	0.502	0.194	1.199
3	0.496	0.220	1.214
4	0.560	0.245	1.316
5	0.620	0.471	1.642
6	0.562	0.253	1.325
7	0.559	0.291	1.367
8	0.487	0.238	1.213
9	0.750	0.472	1.780
10	0.637	0.464	1.646
11	0.816	0.915	2.339.
12	0.737	0.478	1.789
13	0.600	0.134	1.263
14	0.812	0.772	2.192
15	0.619	0.292	1.427
16	0.642	0.422	1.602
17	0.548	0.116	1.163
18	0.461	0.180	1.129
19	0.742	0.281	1.585
20	0.521	0.260	1.290
21	0.583	0.321	1.428
22	0.717	0.666	1.965
23	0.655	0.193	1.379
24	0.878	1.035	2.552
25	0.565	0.186	1.256
26	0.629	0.521	1.704
27	0.592	0.320	1.448
28	0.589	0.292	1.413
29	0.654	0.350	1.541
30	0.498	0.223	1.224
31	0.506	0.260	1.264
32	0.502	0.178	1.183

33	0.500	0.200	1.205
34	0.525	0.230	1.265
Avg	0.604	0349	1.484

Fig 4.4 shows the power consumed by all the nodes in the graphical format with attack on Node 8/36.

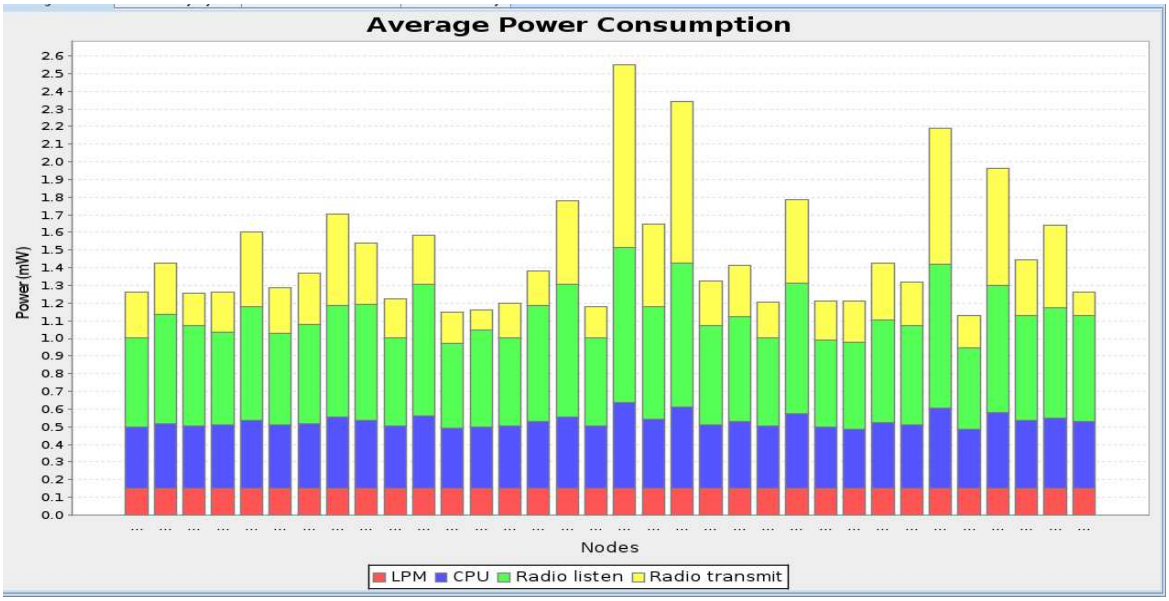

Figure 4.4 Average Power Consumption with Attacked Node 8/36

4.1.1.3 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 given scenario, there is no attack implemented on Node 8. The Average Radio Duty Cycle of Node 8 is 2.01 from fig 4.5.

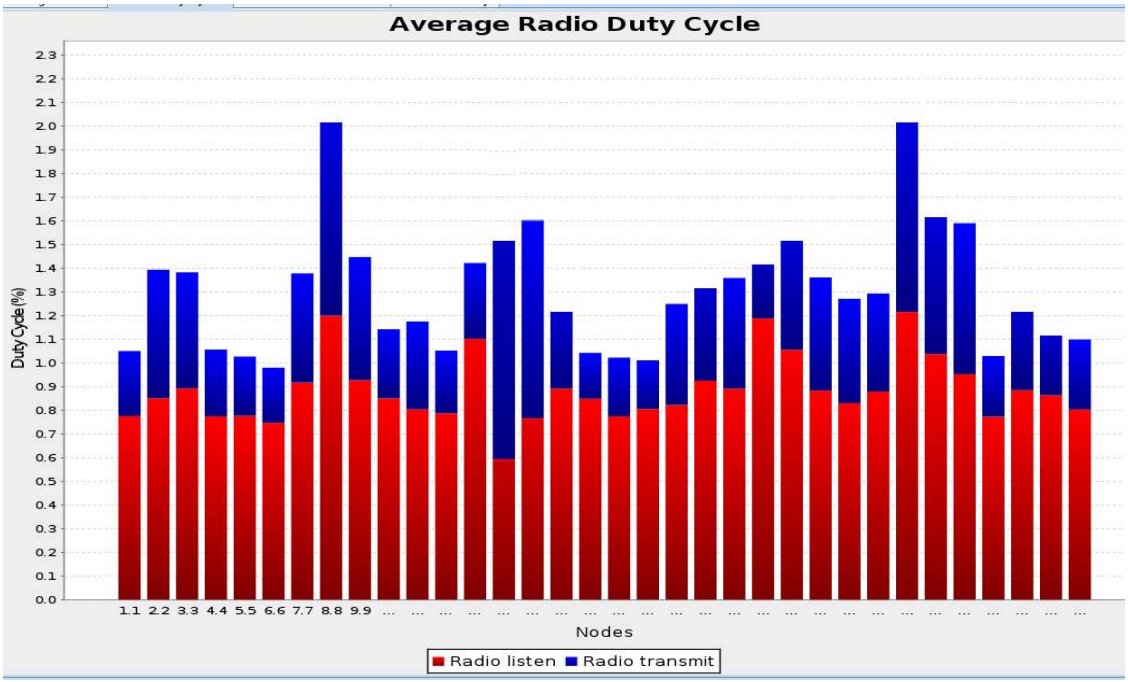

Figure 4.5 Average Radio Duty Cycle without Attacked Node

In the given scenario, there is DAO inconsistency attack implemented on Node 8. The Average Radio Duty Cycle of Node 8 is 1.25 from fig 4.6. So, we can observe that due to the attack the Average Radio Duty Cycle of Node 8 has decreased by a factor of 0.76.

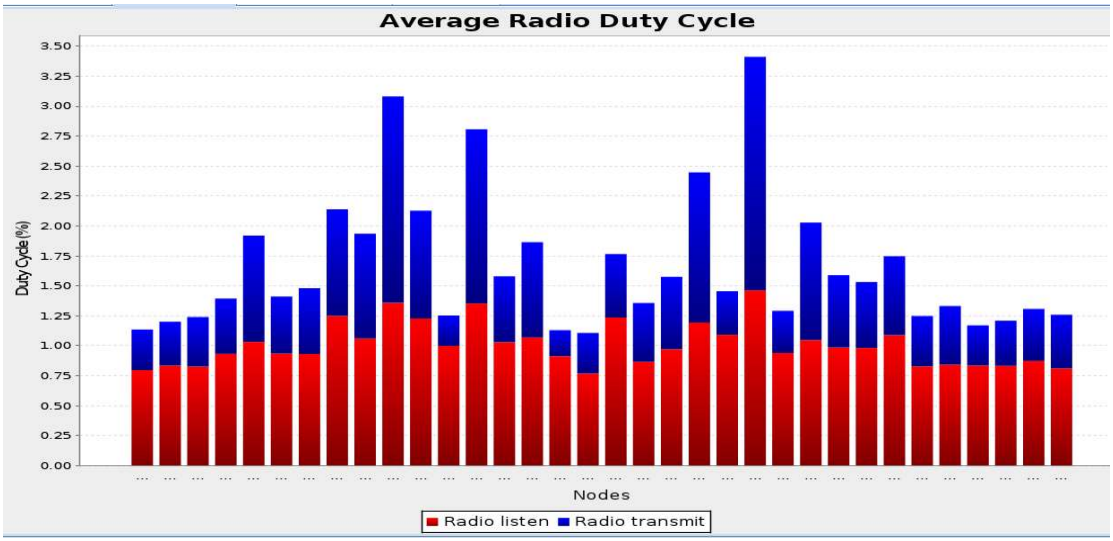

Figure 4.6 Average Radio Duty Cycle with Attacked Node 8/36

4.1.1.4 Analysis of Power History

In the given scenario, there is no attack implemented on Node 8. Node 8's initial power consumption is high and even the overall power consumption is high because it has number of children to which the root can send packets through Node 8 as shown in fig 4.7.

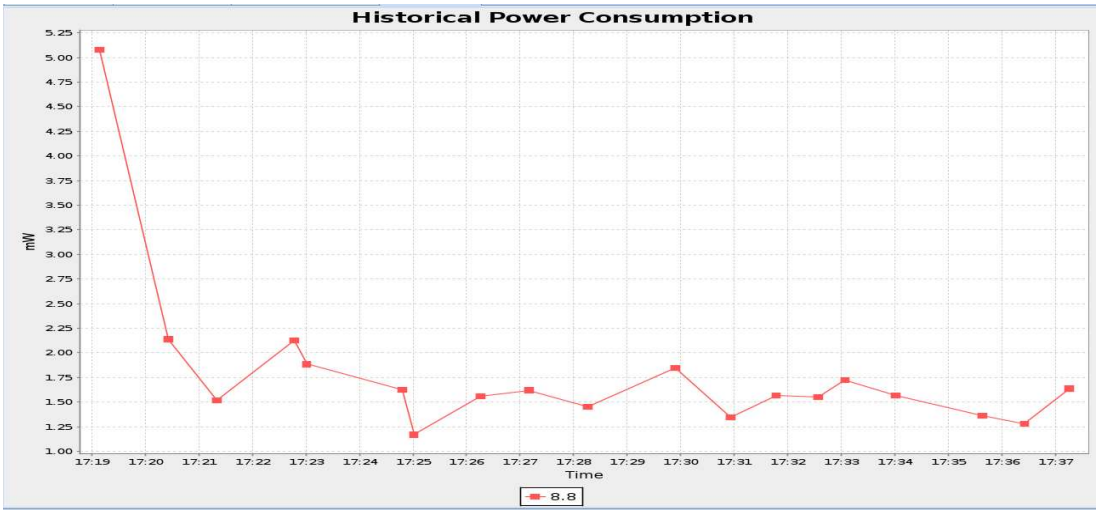

Figure 4.7 Historical Power Consumption of Node 8 without Attack

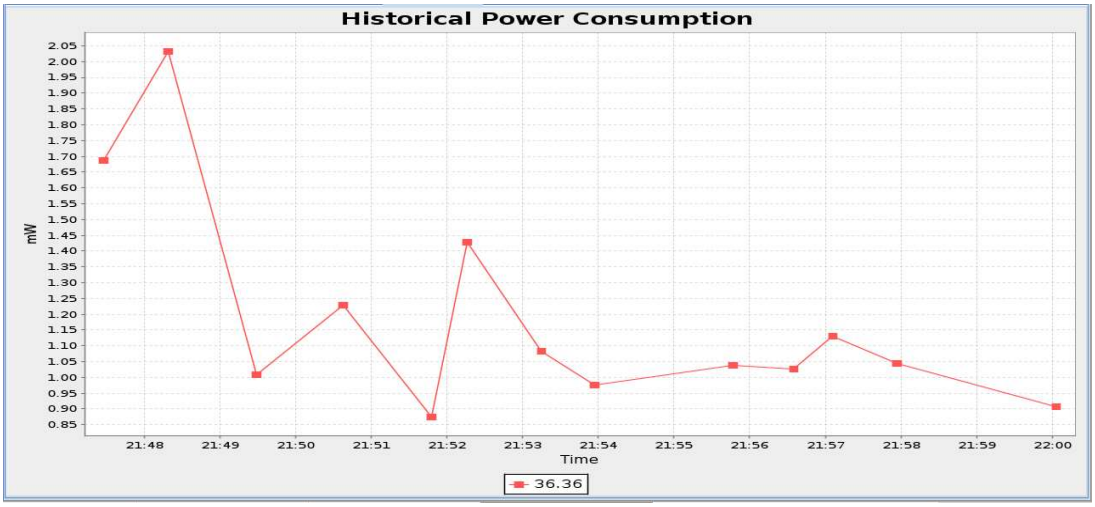

Figure 4.8 Historical Power Consumption of Node 8 with Attack

In the scenario given above Fig 4.8, there is DAO inconsistency attack implemented on Node 8. Node 8's initial power consumption is high but the overall power consumption is low because it has no child to which it can forward the packets.

4.1.2 Scenario 2 (With Attack on Node 28)

4.1.2.1 Analysis of Sensor Map

"Sensor map depicts the relationship between parent and child nodes."

In the given scenario, Node 35 is the Sink node and Node 28 has Node 33 and Node 27as its child node and its parent is Node 23 as shown in fig 4.9.

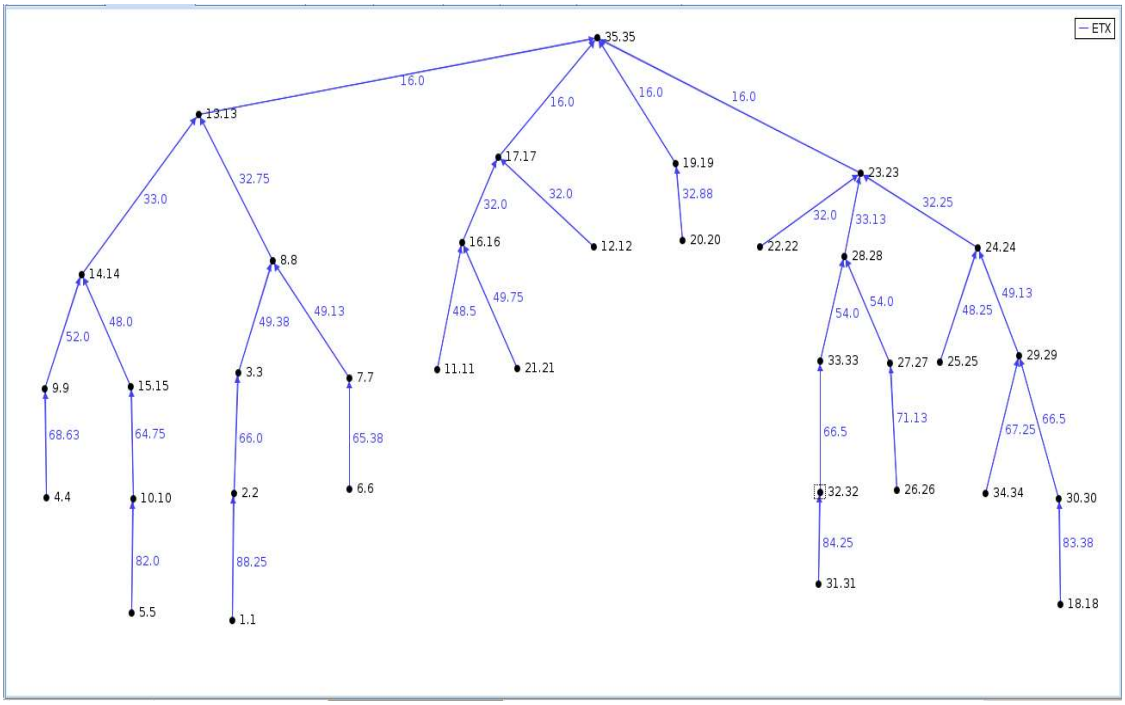

Figure 4.9 Network without Attacked node

Now we implement our attack on Node 28 by attacking it and making it a malicious node. As you can observe from the figure given below that now Node 36 (Actually Node 11) has no child now as shown in fig 4.10. This confirms that our attack has been successfully implemented since now Node 28 attracts less traffic as compared to before and the network topology has changed.

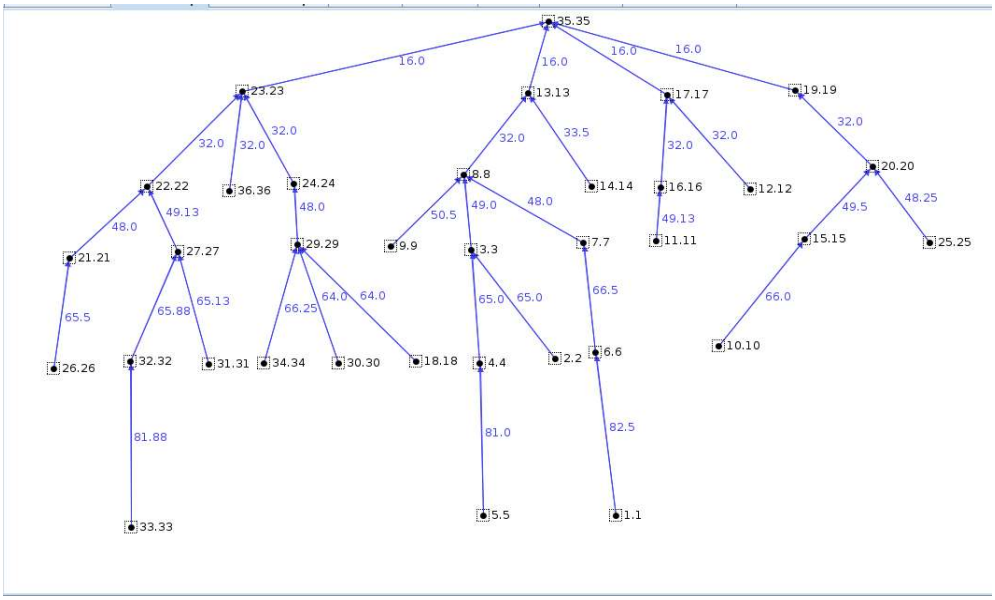

Figure 4.10 Network with Attacked node 28

Table 4.4 Total Packets received by each Node

Node	Without Attack	With Attack
1	28	21
2	28	25
3	20	25
4	29	24
5	29	23
6	28	24
7	27	24
8	28	25
9	29	24
10	28	24
11	29	23
12	20	24
13	28	24
14	28	24
15	29	25

16	28	25
17	20	24
18	29	23
19	29	25
20	29	24
21	29	25
22	26	25
23	29	24
24	29	24
25	29	25
26	29	25
27	29	24
28	28	23
29	28	23
30	20	25
31	28	24
32	20	23
33	29	25
34	28	24
35	0	0

As you can see from the table 4.4 that no. of packets received by Node 28 has decreased from 28 to 23 because of no child which results in less traffic through Node 28. This shows that Node 28 attracts more less when we implement our attack as compared to the normal environment.

4.1.2.1 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 given scenario, there is no attack implemented on Node 28. The power consumed by Node 28 is 1.180mW from table 4.5.

Table 4.5 Analysis of Power Consumption without attack

Node	Listen Power	Transmit Power	Power
1	0.510	0.103	1.121
2	0.465	0.132	1.077
3	0.484	0.109	1.096
4	0.494	0.227	1.227
5	0.556	0.328	1.405
6	0.536	0.248	1.305
7	0.713	0.244	1.514
8	0.634	0.348	1.527
9	0.531	0.254	1.296
10	0.499	0.234	1.240
11	0.528	0.219	1.263
12	0.730	0.547	1.850
13	0.623	0.415	1.588
14	0.572	0.339	1.438
15	0.464	0.136	1.084
16	0.532	0.286	1.332
17	0.519	0.222	1.250
18	0.466	0.146	1.095
19	0.511	0.288	1.321
20	0.536	0.260	1.313
21	0.465	0.150	1.107
22	0.467	0.133	1.080
23	0.449	0.124	1.063
24	0.551	0.245	1.315
25	0.720	0.553	1.845
26	0.557	0.276	1.357
27	0.511	0.155	1.164
28	0.483	0.197	1.180
29	0.473	0.141	1.114
30	0.661	0.170	1.373
31	0.648	0.458	1.666
32	0.574	0.344	1.443

33	0.535	0.269	1.320
34	0.510	0.103	1.121
Avg	0.549	0.249	1.364

Fig 4.11 shows the power consumed by all the nodes in the graphical format without attack.

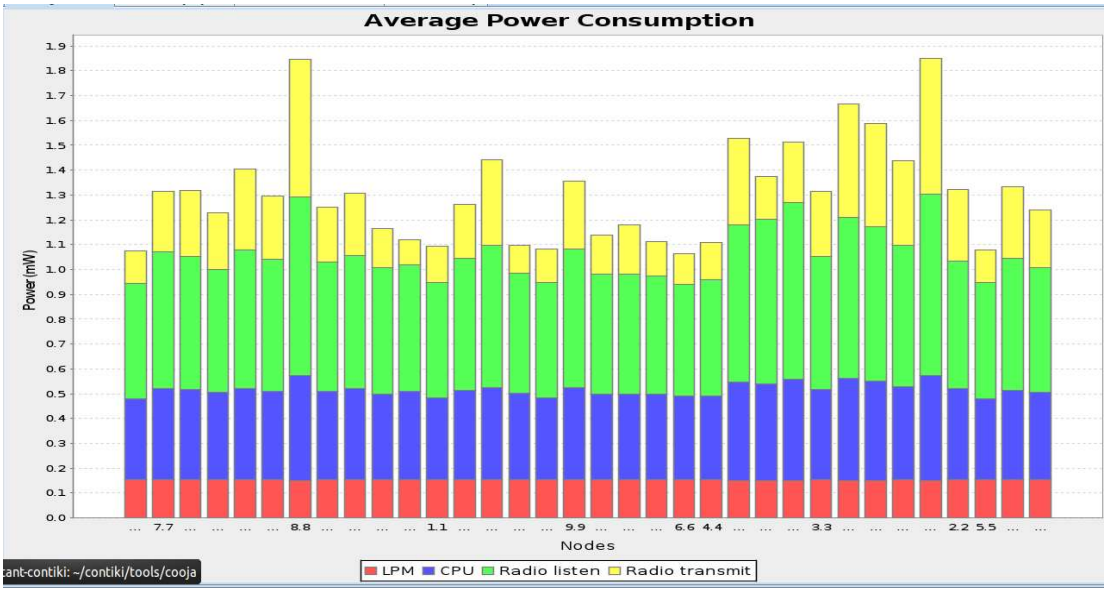

Figure 4.11 Average Power Consumption without Attacked Node

In the given scenario, there is DAO inconsistency attack implemented on Node 28. The power consumed by Node 28 is 1.005mW from table 4.5. So, we can observe that due to the attack the power consumed by Node 28 has decreased by 0.175mW since node 28 has no child and the number of packets received by it has decreased.

Table 4.5 Analysis of Power Consumption with attack on Node 28

Node	Listen Power	Transmit Power	Power
1	0.454	0.178	1.121
2	0.498	0.194	1.197
3	0.561	0.268	1.347
4	0.481	0.179	1.157
5	0.448	0.135	1.063
6	0.472	0.142	1.106

7	0.578	0.295	1.402
8	0.795	0.702	2.096
9	0.537	0.247	1.304
10	0.491	0.160	1.148
11	0.471	0.167	1.133
12	0.459	0.122	1.077
13	0.645	0.195	1.379
14	0.539	0.254	1.309
15	0.583	0.339	1.446
16	0.541	0.290	1.347
17	0.499	0.089	1.092
18	0.461	0.136	1.088
19	0.541	0.120	1.174
20	0.622	0.502	1.670
21	0.528	0.256	1.293
22	0.673	0.446	1.678
23	0.679	0.295	1.538
24	0.613	0.412	1.576
25	0.512	0.137	1.142
26	0.475	0.156	1.126
27	0.519	0.163	1.192
28	0.431	0.109	1.005
29	0.560	0.301	1.395
30	0.465	0.128	1.084
31	0.447	0.110	1.046
32	0.476	0.156	1.130
33	0.450	0.140	1.074
34	0.466	0.127	1.085
Avg	0.528	0.225	1.265

Fig 4.12 shows the power consumed by all the nodes in the graphical format with attack on Node 28/36.

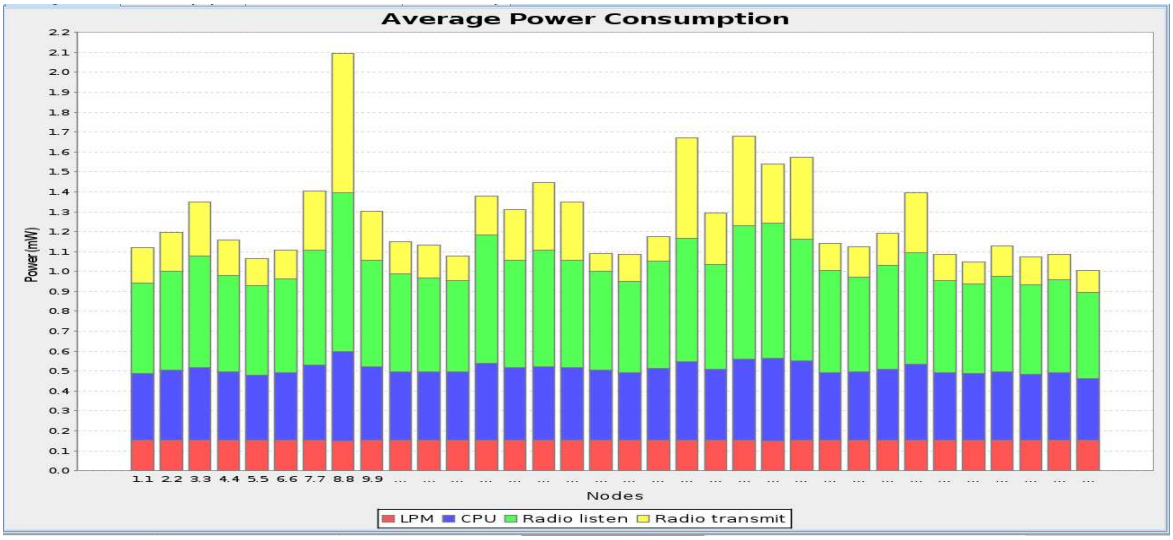

Figure 4.12 Average Power Consumption with Attacked Node 28

4.1.2.3 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 given scenario, there is no attack implemented on Node 28. The Average Radio Duty Cycle of Node 28 is 1.3 from fig 4.13.

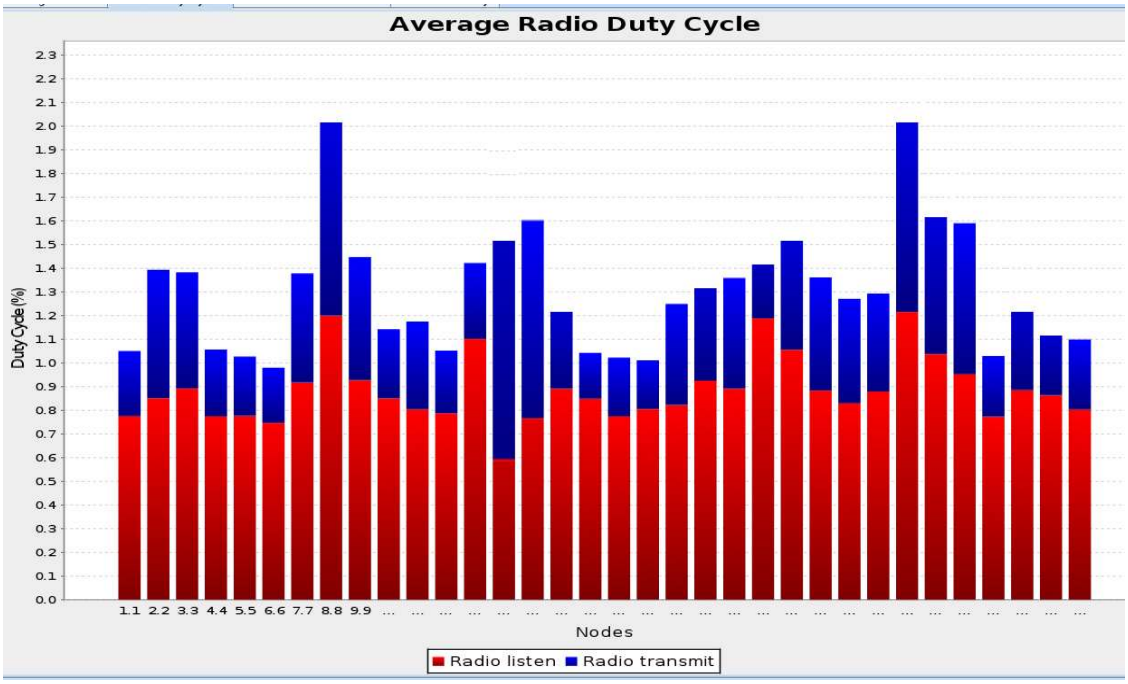

Figure 4.13 Average Radio Duty Cycle without Attacked Node

In the given scenario, there is DAO inconsistency attack implemented on Node 28. The Average Radio Duty Cycle of Node 28 is 0.9 from fig 4.14. So, we can observe that due to the attack the Average Radio Duty Cycle of Node 28 has decreased by a factor of 0.4.

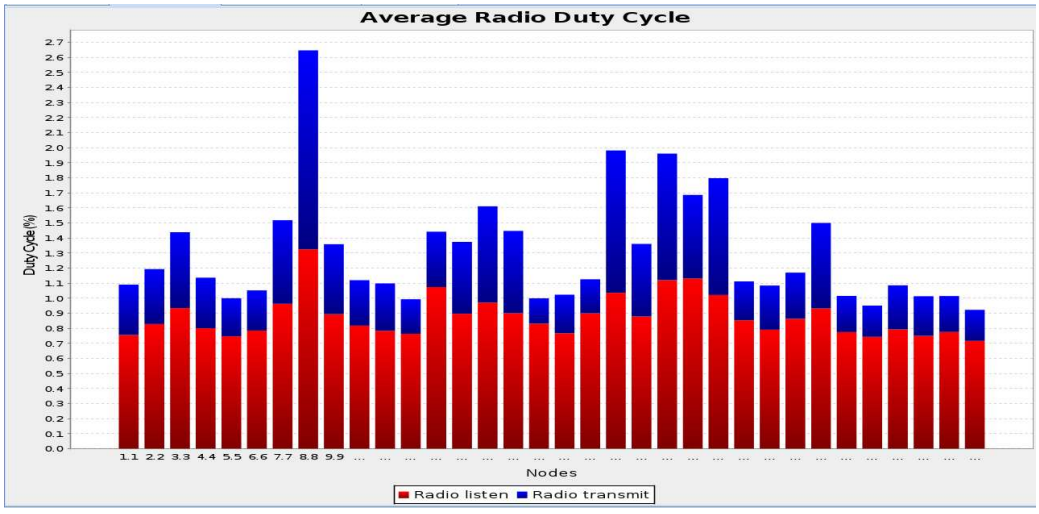

Figure 4.14 Average Radio Duty Cycle with Attacked Node 28

4.1.2.4 Analysis of Power History

In the given scenario, there is no attack implemented on Node 28. Node 28's initial power consumption is high and even the overall power consumption is high because it has number of children to which the root can send packets through Node 28 as shown in fig 4.15.

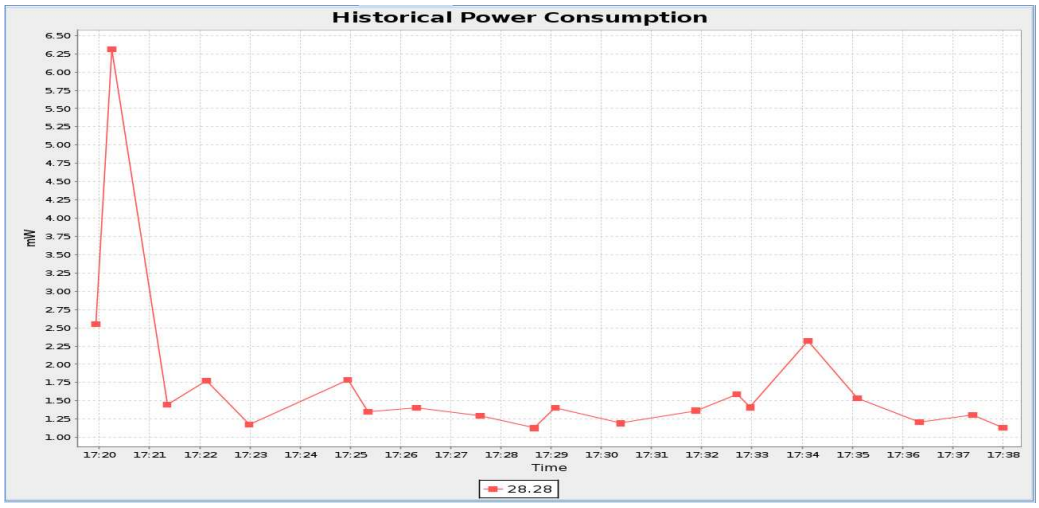

Figure 4.15 Historical Power Consumption of Node 28 without Attack

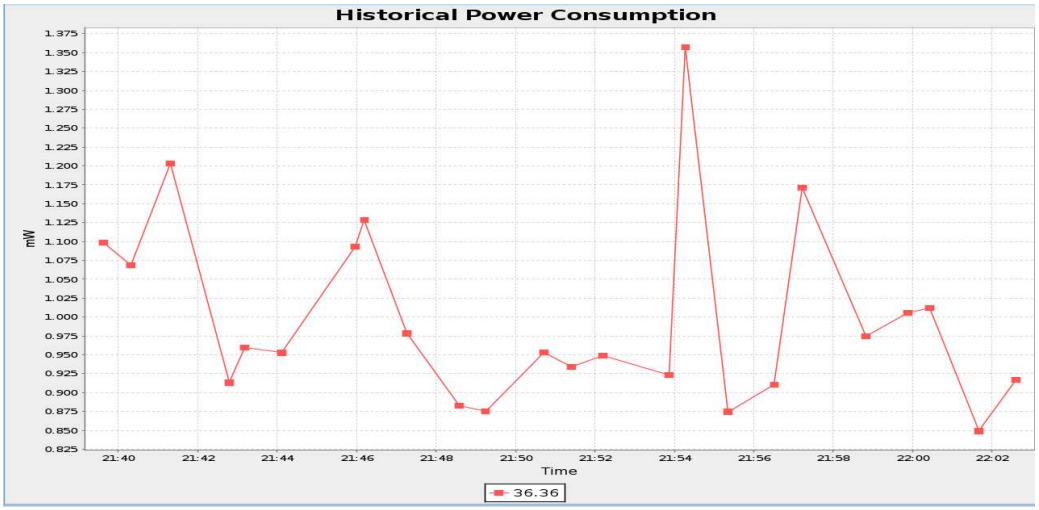

Figure 4.16 Historical Power Consumption of Node 28 with Attack

In the scenario given above Fig 4.16, there is DAO inconsistency attack implemented on Node 28. Node 28's initial power consumption is high but the overall power consumption is low because it has no child to which it can forward the packets.

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 given scenario, Node 1 is the Sink node and Node 5 has Node 6 as its child node and its parent is Node 4 as shown in fig 4.17.

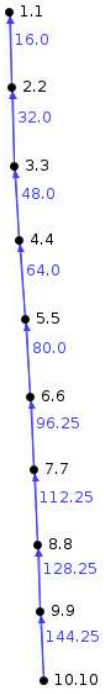

Figure 4.17 Network without Attacked node

Now we implement our attack on Node 5 by attacking it and making it a malicious node. As you can observe from the figure given below that now Node 5 now has no child as shown in fig 4.18. This confirms that our attack has been successfully implemented since

now Node 5 has isolated its children and their subnetwork and the network topology has changed.

Figure 4.18 Network with Attacked node 5

Table 4.7 Total Packets received by each Node

Node	Without Attack	With Attack
1	0	0
2	116	62
3	115	63
4	115	61
5	115	63
6	114	-
7	114	-
8	116	-
9	116	-
10	114	-

As you can see from table 4.7 that no. of packets received by Node 5 has decreased from 115 to 63 because of no children associated with it. This shows that Node 5 has isolated its children and their subnetwork 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 given scenario, there is no attack implemented on Node 5. The power consumed by Node 5 is 1.877mW from table 4.8.

Table 4.8 Analysis of Power Consumption without attack

Node	Listen Power	Transmit Power	Power
1	**0**	**0**	**0**
2	**0.521**	**0.066**	1.077
3	**0.692**	**0.567**	1.812
4	**0.649**	**0.307**	1.473
5	**0.682**	**0.637**	1.877
6	**0.623**	**0.382**	1.526
7	**0.591**	**0.340**	1.445
8	**0.527**	**0.228**	1.251
9	**0.489**	**0.215**	1.198
10	**0.447**	**0.120**	1.038
Avg	**0.580**	**0.318**	1.411

Fig 4.19 shows the power consumed by all the nodes in the graphical format without attack.

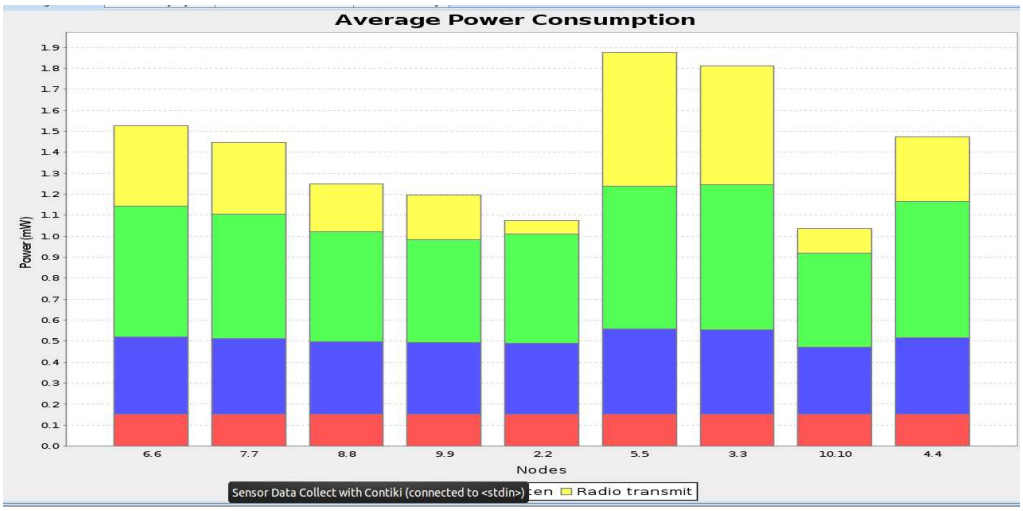

Figure 4.19 Average Power Consumption without Attacked Node

In the given scenario, there is DAO inconsistency attack implemented on Node 5. The power consumed by Node 5 is 0.990mW from table 4.8. So, we can observe that due to the attack the power consumed by Node 5 has decreased by 0.887mW.

Table 4.8 Analysis of Power Consumption with attack on Node 10

Node	**Listen Power**	**Transmit Power**	**Power**
1	**0**	**0**	**0**
2	**0.439**	**0.045**	0.956
3	**0.499**	**0.233**	1.229
4	**0.480**	**0.133**	1.097
5	**0.428**	**0.098**	0.990
6	-	-	-
7	-	-	-
8	-	-	-
9	-	-	-
10	-	-	-
Avg	**0.461**	**0.127**	1.068

Fig 4.20 shows the power consumed by all the nodes in the graphical format with attack on Node 5/11.

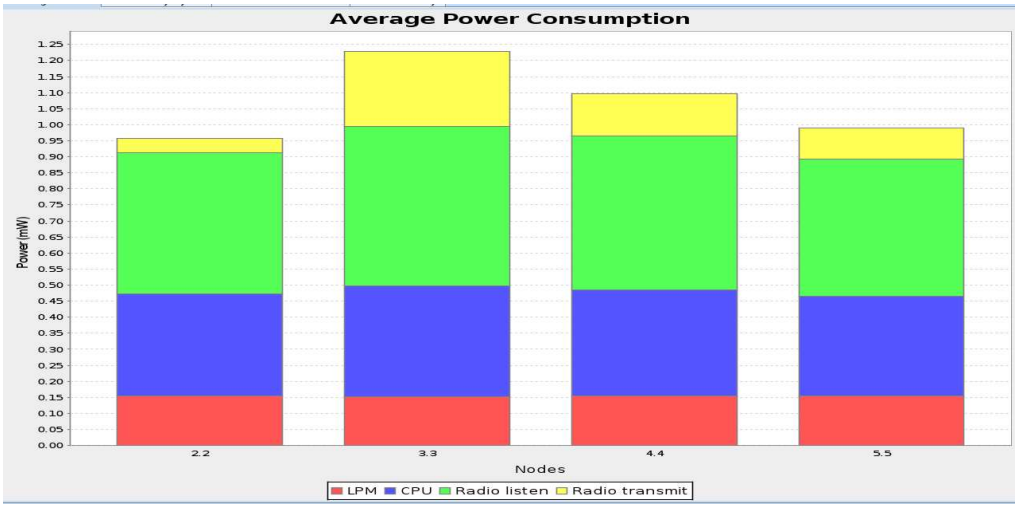

Figure 4.20 Average Power Consumption with Attacked Node 5

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 given scenario, there is no attack implemented on Node 5. The Average Radio Duty Cycle of Node 5 is 2.32 from fig 4.21.

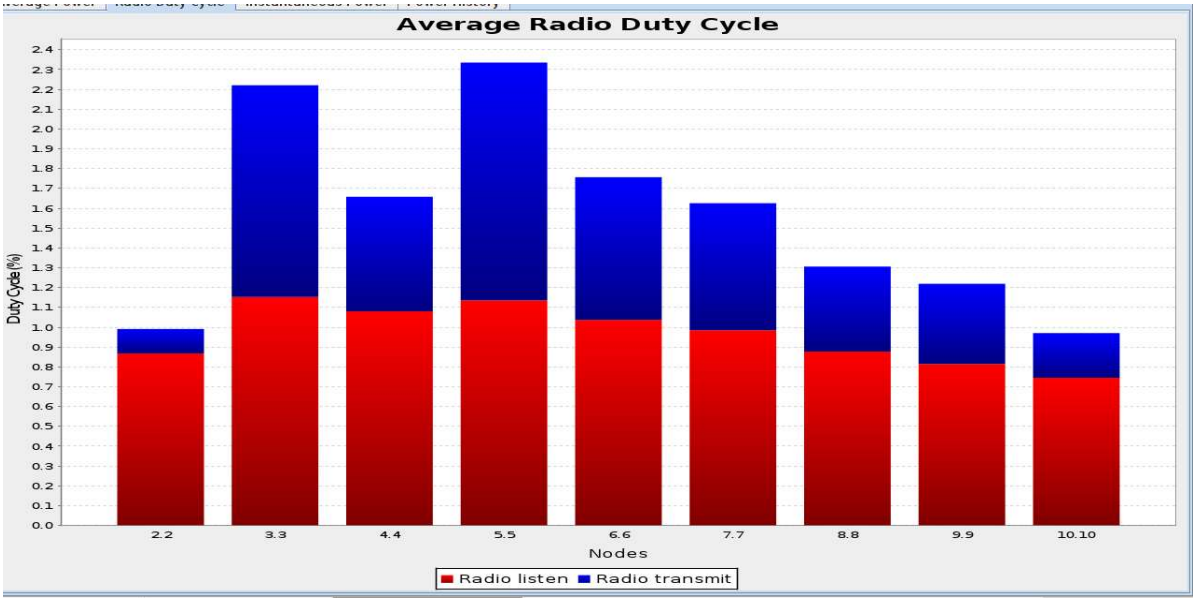

Figure 4.21 Average Radio Duty Cycle without Attacked Node

In the given scenario, there is DAO inconsistency attack implemented on Node 5. The Average Radio Duty Cycle of Node 5 is 0.90 from fig 4.22. So, we can observe that due to the attack the Average Radio Duty Cycle of Node 5 has decreased by a factor of 1.32.

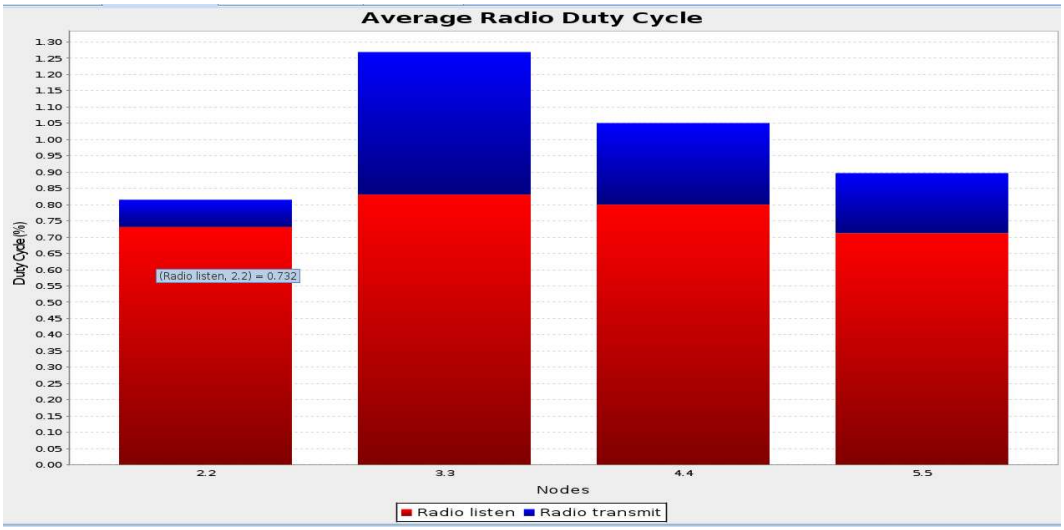

Figure 4.22 Average Radio Duty Cycle with Attacked Node 5

4.1.4 Analysis of Power History

In the given scenario, there is no attack implemented on Node 5. Node 5's initial power consumption is high and overall power consumption is also high because it has a child which can send its packet through Node 5 as shown in fig 4.23.

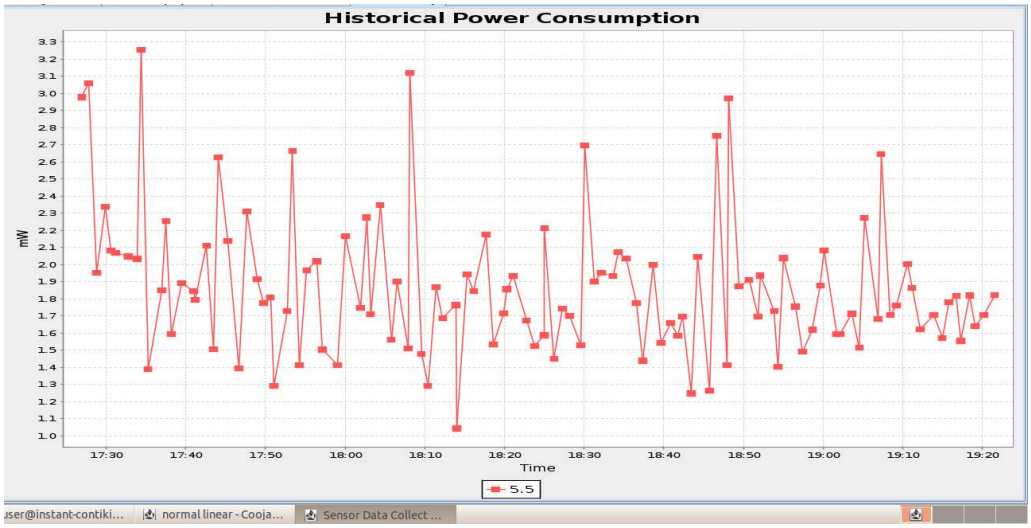

Figure 4.23 Historical Power Consumption of Node 5 without Attack

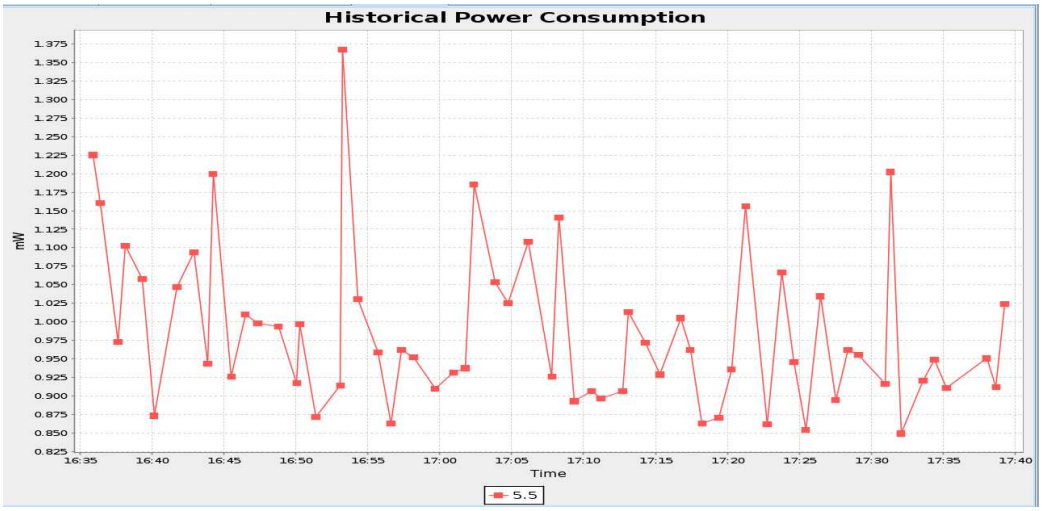

Figure 4.24 Historical Power Consumption of Node 5 with Attack

In the given scenario, there is DAO inconsistency attack implemented on Node 5 as shown in fig 4.24. Node 5's initial power consumption is low and overall power consumption is also low because it is now a leaf node, therefore it cannot forward any packets.

CHAPTER 5

CONCLUSIONS

5.1 Conclusions

We implemented the DAO inconsistency attack on two different topologies and successfully analyzed it by analyzing its sensor map, power, radio duty cycle and no of packets both in attacked environment and normal environment. Now taking the case of Grid placement of node we observed that whenever the attacking node is introduced in the network it isolates the subnetwork below it, but the nodes of the subnetwork lies in the vicinity of the nodes connected to the root node, thus the isolated sub graph gets again connected to the network through the nearby node according to the applied metrics in objective function. Now in the second case that is Linear Placement of nodes when the subgraph or node below the attacking node gets isolated then none of the other node lies in the vicinity of it thus the subgraph gets totally isolated from the network and cannot reconnect to the network. So from these observation we can conclude that Grid Placement of Nodes is better than Linear Placement of nodes as in case of Grid Placement we can

reconnect to the network just by increasing the cost of the network that is in terms of metrics applied but the same is not possible for Linear placement .

5.2 Future Scope

Study for understanding wireless network security in low power and lossy networks for attacks such as Sink Hole (an extension of DAO Inconsistency Attack). Also, we would continue our analysis of relation of RPL with IPv6 Neighbor Discovery, with 6LoWPAN. Our project for reducing power consumption in DAO Inconsistency Attack will be taken up. Developing skills to use Cooja is also important for us to deal with these networks.

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

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