MITIGATING SINKHOLE ATTACK ON LOW-POWER AND LOSSY NETWORKS WITH TRAFFIC AWARE SCHEDULING ALGORITHM USING DUAL PARENT MECHANISM

Mr. Tay Zar Bhone Maung

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Electrical Engineering
Department of Electrical Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2021

การบรรเทาผลของการโจมตีแบบซิงค์โฮลบนโครงข่ายกำลังต่ำและมีการสูญเสียที่ใช้อัลกอริทึมการจัดสรรตามปริมาณการใช้โดยใช้กลไกพาเรนต์คู่

นายเท ซา โบน หม่อง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

Thesis Title	MITIGATING SINKHOLE ATTACK ON LOW-POWER AND LOSSY NETWORKS WITH TRAFFIC AWARE SCHEDULING ALGORITHM USING DUAL PARENT MECHANISM
By	Mr. Tay Zar Bhone Maung
Field of Study	Electrical Engineering
Thesis Advisor	Associate Professor LUNCHAKORN WUTTISITTIKULKIJ, Ph.D.

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Engineering

-- Dean of the FACULTY OF ENGINEERING

(Professor SUPOT TEACHAVORASINSKUN, D.Eng.)

THESIS COMMITTEE

-- Chairman

(Professor WATIT BENJAPOLAKUL, Ph.D.)

-- Thesis Advisor

(Associate Professor LUNCHAKORN WUTTISITTIKULKIJ, Ph.D.)

-- Examiner

(Assistant Professor PASU KAEWPLUNG, Ph.D.)

-- External Examiner

(Pisit Vanichchanunt, Ph.D.)

เท ซา โบน หม่อง :
การบรรเทาผลของการโจมตีแบบซิงค์โฮลบนโครงข่ายกำลังต่ำและมีการสูญเสียที่ใช้อัลกอริทึมการจัดสรรตามปริมาณการใช้โดยใช้กลไกพาเรนต์คู่. (MITIGATING SINKHOLE ATTACK ON LOW-POWER AND LOSSY NETWORKS WITH TRAFFIC AWARE SCHEDULING ALGORITHM USING DUAL PARENT MECHANISM) อ.ที่ปรึกษาหลัก : ลัญฉกร วุฒิสิทธิกุลกิจ

โครงข่ายกำลังงานต่ำและมีการสูญเสียเป็นโครงข่ายข่ายที่เราเตอร์และอุปกรณ์อินเตอร์เน็ตของทุกสรรพสิ่งทั้งหมดทำงานโดยใช้กำลังงานหน่วยความจำ และ พลังงานในการคำนวณ อย่างจำกัด เนื่องจากโครงสร้างที่มีข้อจำกัดของโครงข่ายกำลังงานต่ำและมีการสูญเสีย เช่น การมีทรัพยากรที่จำกัด การเชื่อมต่อมีการสูญเสีย และ ขาดความปลอดภัยเชิงกายภาพ การโจมตีด้านความปลอดภัยสามารถเกิดขึ้นได้เมื่อมีการกำหนดเส้นทางในโครงข่าย โปรโตคอลการกำหนดเส้นทางสำหรับโครงข่ายกำลังงานต่ำและมีการสูญเสียข้อมูล หรือ RPL ได้ถูกพัฒนาเพื่อตอบสนองความต้องการของแอปพลิเคชันที่หลากหลายในด้านโครงข่ายเซ็นเซอร์ไร้สา และ อินเตอร์เน็ตของทุกสรรพสิ่ง โหนดเซ็นเซอร์บางตัวในโครงข่าย RPL ไม่แข็งแกร่งพอที่จะทนต่อการโจมตีต่างๆ เช่น การโจมตีแบบซิงค์โฮล การโจมตีประเภทนี้สามารถสร้างความเสียหายให้กับโครงข่ายได้ด้วยตัวเองหรือในจุดเชื่อมร่วมกับการโจมตีอื่น ๆ ผู้โจมตีสามารถสร้างพฤติกรรมการโจมตีได้อย่างง่ายดาย และ อาจทำให้เกิดการแยกตัวอย่างรุนแรงจากโครงข่าย และ เกิดการสูญเสียแพ็กเก็ตที่รับส่งของการจราจรในโครงข่าย เนื่องจากความเสียหายของการโจมตีแบบซิงค์โฮลในโครงข่าย RPL นั้นใหญ่มากและทำให้เกิดผลกระทบสูงต่อโครงข่าย ตำแหน่งของโหนดโจมตีจึงมีความสำคัญมากในเครือข่าย และสามารถทำให้โครงข่ายแยกตัวขนาดใหญ่และสูญเสียเปอร์เซ็นต์สูงของการจราจรที่สูญเสีย วิทยานิพนธ์นี้ศึกษาความเสียหายของการโจมตีแบบซิงค์โฮลในโครงข่าย RPL และเสนอหนทางที่ง่ายและมีประสิทธิภาพสูงขอกลไกการป้องกันเพื่อบรรเทาการโจมตีซิงค์โฮล วิธีการนำเสนอนั้นสร้างรูปแบบพาเรนต์เชิงคู่สำหรับโหนดย่อยแต่ละโหนดในโครงข่ายเมื่อติดตั้งโทโพโลยี ซึ่งเป็นวิธีที่มีประสิทธิภาพในการป้องกันการโจมตีแบบซิงค์โฮล วิทยานิพนธ์นี้ยังใช้การปรับสมดุลการจราจรของโครงข่ายโดยใช้อัลกอริทึมการจัดการการรับรู้จราจร หรือ TASA การใช้ TASA ในโทโพโลยีโครงข่าย RPL เป็นวิธีที่ดีในการจัดการปริมาณการจราจรทั้งหมดของโครงข่ายการเก็บข้อมูล หลีกเลี่ยงการชนกันระหว่างการส่งข้อมูลของโหนดย่อยและโหนดหลัก และลดเวลาและความล่าช้าของโครงข่าย ผลลัพธ์แสดงให้เห็นว่าสามารถบรรเทาการโจมตีซิงค์โฮลและให้งทั้งโครงข่ายได้อย่างเต็มที่ และวิทยานิพนธ์นี้ได้เปรียบเทียบจำนวนช่องเวลาและการสูญเสียแพ็กเก็ตในกลไกทั้งสอง โดยมีและไม่มีพาเรนต์คู่ ภายใต้การโจมตีแบบซิงค์โฮล และ รายละเอียดของการเปรียบเทียบช่องเวลาของพฤติกรรมการโจมตีหนึ่งอย่างเพื่อตรวจสอบความถูกต้องของวิธีการและการจำลองที่นำเสนอ สุดท้ายนี้สามารถสรุปได้ว่ากลไกป้องกันการโจมตีจากซิงค์โฮลแบบพาเรนต์คู่สามารถทำงานได้ดีโดยการตรวจสอบผลการเปรียบเทียบ

สาขาวิชา	วิศวกรรมไฟฟ้า	ลายมือชื่อนิสิต ....
ปีการศึกษา	2564	ลายมือชื่อ อ.ที่ปรึกษาหลัก ....

6170503621 : MAJOR ELECTRICAL ENGINEERING

KEYWORD: Routing Protocol, Low-Power and Lossy Networks, Security, Sinkhole Attack, Dual-Parent Mechanism, Traffic Aware Scheduling Algorithm, RPL, IoT

Tay Zar Bhone Maung : MITIGATING SINKHOLE ATTACK ON LOW-POWER AND LOSSY NETWORKS WITH TRAFFIC AWARE SCHEDULING ALGORITHM USING DUAL PARENT MECHANISM. Advisor: Assoc. Prof. LUNCHAKORN WUTTISITTIKULKIJ, Ph.D.

Low-Power and Lossy Networks (LLN) are networks where all the routers and IoT devices are working on a limited power, memory, and computational energy. Due to the constrained structures of LLN networks such as limited resources, lossy connection and lack of physical security, security attacks can occur when routing in an LLN network. The Routing Protocol for Low-Power and Lossy Networks (RPL) was developed to meet the needs of multiple applications in the fields of Wireless Sensor Networks (WSN) and Internet of Things (IoT). Some sensor nodes in a RPL network are not strong enough to withstand a variety of attacks, such as a sinkhole attack. This type of attack can damage the network by itself or in conjunction with other attacks. The attacker can easily create attacking behavior and can cause serious isolation from the network and loss of delivered packets of the network traffic. As the damage of sinkhole Attack in the RPL network is very big and it makes a high impact to the network, position of the attack node is very important in the network, and it can make a huge network isolation and loss the high percentage of traffic loss. This thesis studies the damage of Sinkhole attack in RPL networks and proposed the simple and very effective way of defense mechanism to mitigate that sinkhole attack. Our proposed method, making a dual-parent formation for each child node in the network when the topology is set up, is the effective way to defense the Sinkhole Attack. This thesis also implements the traffic load balancing of the network by applying Traffic Aware Scheduling Algorithm (TASA). Applying the TASA in the RPL network topology is a good way to concern the total traffic load of our data acquisition network, avoid the collision between child nodes and parent node transmission and reduce the time and delay of the network. Results show that we can mitigate the sinkhole attack and fully deliver the total traffics of the network. And this thesis compares the number of time slots and packets loss in both mechanisms, with and without dual parent, under sinkhole attack and details of time slots comparison of one attacking behavior to check the correctness of our proposed method and simulation. Finally, we can conclude that our dual-parent sinkhole attack defense mechanism is worked well by checking the comparisons results.

Field of Study:	Electrical Engineering	Student's Signature ....
Academic Year:	2021	Advisor's Signature ....

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Associate Professor Dr. Lunchakorn Wuttisittikulkij, for his patient and wise counsel. My experience at Chulalongkorn University was made more enjoyable by your advice and knowledge.

Without the assistance and support of the Chulalongkorn University scholarship program for ASEAN countries to study for a Master's Degree, my research would not have been possible. Additionally, I want to thank the Department of Electrical Engineering for their financial assistance.

I am grateful to my colleagues from Smart Wireless Communication Ecosystem Research Unit who gave valuable suggestions and advice during our meeting.

I also would like to thank to committee members of my thesis examinations for your valuable comments and suggestion.

I want to express my sincere gratitude to Dr. Thiri Thitsar Khaing for her assistance during the initial stages of this master's thesis journey.

I also want to express my gratitude to my parents, brother, and sister for their financial and emotional support. While I was a student at Chulalongkorn University, my friends supported me. I want to express my gratitude to Dr. Ei Ei Tun, Dr. Ei Ei Mon, Dr. Hsu Mon Lai Aung, Mr. Htain Lynn Aung and Ms. Phoo Phoo Thet Lyar Tun for their unwavering assistance in helping me apply for the scholarship and finish my dissertation. I want to express my gratitude to Dr. Sanika Wijayasekara and Mr. Pruk Sasithong for their insightful advice and assistance. Finally, I'd want to express my gratitude to Mr. Soe Ye Htet, Mr. Ye Moe Myint, Ms. Nang Htet Htet Aung, and Mr. Aung Myo Htut for their support throughout our discussion and assistance with my thesis and would like to thank to my beloved person who, in the last stage of my struggle, gave me emotional support and encouragement.

Tay Zar Bhone Maung

TABLE OF CONTENTS

LIST OF TABLES

List of Abbreviations

6LoWPAN	IPv6 over Low Power Personal Area Network
DAO	DODAG Advertisement Object
DAO-ACK	DODAG Advertisement Object Acknowledgement
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acrylic Graph
DTSN	Destination Advertisement Trigger Sequence Number
ETX	Expected Transmission Count
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPv6	Internet Protocol version 6
LLN	Low Power and Lossy Networks
LBR	Low Power Border Router
M2P	Multipoint to Point
MAC	Medium Access Control
MOP	Mode of Operation
MRHOF	Minimum Rank with Hysteresis Objective Function
OF	Objective Function
OSPF	Open Shortest Path First
P2M	Point to Multipoint
P2P	Point to Point
RPL	Routing Protocol for Low Power and Lossy Networks
TASA	Traffic Aware Scheduling Algorithm
TSCH	Time Synchronized Channel Hopping
UNS	Unheard Node Set

VERA	Version Number and Rank Authentication
WSN	Wireless Sensor Networks

Chapter 1

Introduction

In recent years, along with the rise of the usage of Internet of Things (IoT) devices, the applications of wireless ad-hoc networks and sensor networks for smart systems have increased. In a wireless ad-hoc network, many sensor devices (nodes) can be installed easily to collect the data and each sensor node connects to all its surrounding nodes for data forwarding and successful communication. When a smart system like environmental monitoring is implemented, sensor nodes and interconnected devices are connected wirelessly. On the other hand, with the increasing demand for cyber security, the communication links in an IoT wireless sensor network need to be secured. With the implementation of IoT, the physical objects in the real world can relate to each other to share information and communicate in real-time with a higher degree of performance as well as security. Therefore, finding the secure path between the two sensor nodes and sending the collected data from one node to another trusted one are big problems in the wireless routing protocol.

Routing Protocol for Low Power and Lossy Networks (RPL) is the Distance Vector Routing Protocol for Low Power and Lossy Networks (LLN) where many Internet of Things (IoT) devices are implemented through the wireless connection. That kind of Wireless Sensor Networks (WSN) are very popular nowadays because that is convenient to use and can be set up easily for various purposes. When a WSN network is built up, a routing protocol for IoT devices is played a vital role in effective, reliable, and successful communication among sensor nodes. RPL meets that kind of criteria for low-power sensor nodes that have constrained memory and limited energy.

Because of wireless connection and low power working procedure, RPL cannot work well for the strong security issue. Many kinds of attacks can happen in the RPL network based on resources, network topology and network traffic. Among them, a sinkhole attack is the kind of network topology, and it can create easily in an RPL network but the damage of a sinkhole attack to the network is excessively big.

1.1: Motivation

Increasing building many smart systems; smart parking, smart healthcare system, smart farm, smart city, etc. nowadays, the application of Wireless Sensor Networks (WSN) is exceedingly popular. In WSN, many routers, gateways, sensor nodes and network accumulators are involved to build up a smart system or smart environment. Some WSN networks are built up as strong networks which have many powerful sensor nodes and routers for network connection, but some networks are working with low power and limited energy nodes. These kinds of networks are called Low Power and Lossy Networks (LLN). LLN works with lower power sensor nodes which are constrained energy, power, and memory nodes for wireless connection.

Routing Protocol for Low Power and Lossy Network (RPL) is the originating routing protocol for low-power IoT devices. RPL has been proposed by the Internet Engineering Task Force (IETF) as a standard routing protocol for LLN networks and 6LoWPAN (IPv6 over Low Power Personal Area Network) networks. The working procedure of RPL is simple and quite efficient for low-power networks and RPL is one of the best routing protocols for data transportation and routing in IoT devices.

Applying RPL protocol in a wireless sensor network is a good way to set up to create a smart system, especially for low power networks. RPL works proficiently with limited power sensor nodes, and it is very useful for massive data transmission.

1.2: Problem Statement

Building a smart wireless system with many sensor nodes is easy to implement at this moment but creating a secure network that has a secure connection and no malicious node in the network is quite challenging for a low-power lossy network. For example, in a smart data acquisitions network, creating a reliable connection between two sensor nodes and forwarding the sensed data from one sensor node to another node have existed till now. A sensor node forwards the data packet to its up-level node to deliver to the server or user application. When the up-level sensor node does not forward the data to the next up-level node, the server does not get the useful data and users do not know what is happening in the real situation outside.

Moreover, so many attackers are still waiting to get useful data from our sensor network, or they want to make trouble in our data transmission. In the RPL network, there are many kinds of attacks are figured out due to their attack behaviors. An attack can easily happen in the RPL network compared with other normal networks. Because of the network structure of RPL, some attacks make serious damage to the network in data forwarding and acquisition.

To create a secure sensor network, there have many protections, applying protection mechanisms and using security features are available. We can use various data encryption algorithms, hashing algorithms and node authentication security features in our sensor network to make a secure connection. Implementing these kinds of security features in normal networks is not a serious issue. But it is a big concern for RPL networks for using high-security features because RPL works in low-power and lossy networks. Many sensor nodes in the RPL network are low power, constrained memory and limited energy. So, applying for the big security program in sensor nodes is not easy, and that can use a lot of power consumption to run the program. Avoiding serious harmful attacks, e.g., Sinkhole Attacks in RPL with a simple protection mechanism and making the successful data transmission within a minimum timeframe are good challenging in the research field of RPL network.

1.3: Objective

This thesis's main objective is to investigate how to mitigate the negative effect of sinkhole attacks in a secure low power and lossy network routing protocol where a few dozens of low-power internet of things (IoT) devices are connected wirelessly. We first identify the amount of damage measured in terms of packets loss due to the sinkhole attack at different levels of nodes in the networks. Then we propose to avert the sinkhole attack in an effective manner by ensuring that every member nodes are reachable by the root node through at least two parent nodes. A network simulator is designed and implemented specifically for this study and used extensively to determine whether dual-parent mechanism can effectively advert the sinkhole attack with respect to the traffic loss and packet delivery time.

1.4: Scope of the Thesis

The scope of this thesis is as follow:

1. To implement a network simulator to simulate routing protocol based on TASA for low-power and lossy networks for normal operation and under a Sinkhole attack, one of the serious topology attacks of RPL that makes network isolation at different level in the networks.
2. To propose a dual-parent concept as an effective mechanism to mitigate the impact of sinkhole attack, where traffic loss and packet delivery time can be maintained despite the attack.

3. To numerically compare the results of traffic loss of the network under network with and without dual-parent mechanism under sinkhole attack.

1.5: Contributions

Sinkhole attack is one kind of topology attack in an RPL network, and it attacks the network internally and makes the sub-network that will be isolated from the main network. It is seriously damaging to those networks which employ data acquisition and data transmission.

Dual-parent network formation is verified as an effective way to combat against the sinkhole attack in the RPL networks. It is simple to create but a very effective way to avoid the attack. For working in Low Power and Lossy Network (LLN) structure, implementation of dual parent for every child node in the network is no need to consume extra power for protection from attack. By combining the Traffic Aware Scheduling Algorithm (TASA) algorithm with dual-parent implementation, sensor nodes in the RPL network can work very effective way for power consumption and can avoid the traffic conjunction of the data transmission in the network.

1.6: Literature Review

For detecting the internal attack like a Sinkhole attack in the RPL network, there are two Intrusion Detection System (IDS) approaches **[1]**, agent cluster based, or specification-based IDS and agent distributed based or anomaly-based IDS. For first approach, Le et al **[2]** proposed a specification-based IDS in 2016, in that each IDS agent works as a cluster-based type. The IDS agent is placed in the center of the

DODAG to cut down the overhead on the root node and its surrounding member nodes, and this approach is adopted from the SVELTE. For second approach anomaly-based IDS, Raza et al **[3]** conducted a newly intrusion detection system called SVELTE. They used three main modules; first is to collect the information of RPL network and rebuild the network, second is to analyse the data and detect intrusion and the last is looks like a mini firewall to filter undesirable traffic. All three modules are placed in the border router and nodes of the RPL network. In 2015, Cervantes et al. **[4]** proposed another anomaly-based IDS called INTI. For mitigation of Sinkhole attack, the INTI analyses the devise performance of every node by working together with three units such as watchdog, reputation, and trust approach.

Dvir et al. **[5]** offered a new Sinkhole against mechanism called VERA (version number and rank authentication) in 2011. When a DODAG is create, the node sent its rank value in the DIO message and this rank value shows the individual position of the node that is how it is closer to the root node. So, an attacker node can modify this rank value to the lower value than real rank value to impersonate to the other nodes. To prevent the forging true rank value or obtaining lower rank value, VERA uses a one-way hashing method that is sequence and strict the changing rank value form the root to the nodes. The rooted node has already sent hash value to all members DODAG nodes while they are creating a DODAG. A node checks the hash value that id changed or not by the previous node when it received an encrypted rank value that was put in DIO message. In 2012, Weekly et al. **[6]** proposed another Sinkhole defence mechanism that was based on parent fail-over technique. The DODAG rooted node make an unheard node set (UNS) field that was added in the DIO message as an extra to prevent the alternation of DODAG information. To fix some drawbacks of VERA, in 2013, TRAIL **[7]** was conducted. Perrey et al. proposed a new topology authentication in RPL that was TRAIL which presents a more powerful version of VERA and designs to minimize a network message exchange and resource power consumption. Iuchi et al. in **[8]** proposed a new DIO message-based [1][65]Sinkhole against mechanism in 2015. Their system is based on selecting a secure parent method and a child might choose its parent that might be a legitimate node base on the standard threshold value. But a node that has extremely lower rank value than a threshold, then this node will be skipped for choosing as preferred parent by others. In 2020, Zaminkar et al. [48] presented a novel approach to against the sinkhole attack by rating and ranking on operation of the node in the network. Their system, SoS-RPL has better performance results comparing with other methods, but they did not test on power consumption that is important for low-power IoT devices.

Table 1 Literature Review of State of Arts

Mechanism	**Security Protocol**	**Techniques**	**Testing**	**Details**
Specification cluster-based IDS	Detection	IDS: with three specification-based algorithms,	Simulation in Cooja	In this paper, authors propose a specification-based IDS agent system that is placed in each same size cluster of the network. To detect the five kinds of internal attacks, the proposed system works with three IDS algorithms. In each cluster, the algorithm 1 and 2 of the IDS agent extracts the states and transitions of the sensor nodes and algorithm 3 works based on the collected data to detect the attacks. Because of its cluster-based system, there might need more IDS agents and due to the centralization, there may have a high probability of IDS failure rate.
SVELTE	Detection	IDS: with 6LoWPAN Mapper, Intrusion detection	Simulation in Cooja	In this paper, the authors implement an Intrusion Detection System (IDS), is called SVELTE, with three main modules to detect the sinkhole attack.

		component, Mini firewall		They simulated their system with three types of attacking behaviors not only in lossless but also in lossy network. The positive rate of SVELTE is good when the network is small, but it has a pretty low result when the network is increase and the power consumption rate is a little bit higher.
INTI	Detection	IDS: watchdog, reputation, and trust strategies	Simulation in Cooja	This paper is about a new Intrusion Detection System named INTI to detect, prevent and make an isolation sinkhole attack in the network. To compare the former IDS system, SVELTE, the authors consider the concept of mobility of attacker node in their implementation and show the comparison results.
VERA	Protection	Hash Chains: Hash function (SHA-1) MAC function Digital signature	Proposed	In this paper, the authors propose a new security service that prevents any misbehaving node from illegitimately increasing the Version Number and compromise illegitimate decreased Rank values.
Parent fail-over	Detection	Rank authentication, Parent fail-over	Simulation in a custom-built	The authors present a combine method of Rank Authentication with one-way hash function and Parent fail-over technique, which uses a special set (UNS) like a blacklist to protect the Sinkhole attack in routing. The Parent fail-over has a little problem to create a set of unheard nodes and the result shows that there have not too much different when compared with no defense method. But the combination of two techniques has a good result as a result.

TRAIL	Detection	Algorithm based on first-hand principle, Bloom filter	Simulation in RIOT OS	In this paper, a topology authentication scheme named TRAIL is created to fix some weakness of VERA. It is based on VERA but authors reduce the cryptographic workload in their system. Authors apply the cryptographic operation only in root node and set the root node as a trust anchor using with a Bloom filter.
Secure parent	Detection	Algorithm based on rank threshold	Simulation in Cooja	Authors have presented a secure parent selection method by performing the calculation mechanism of rank threshold in the node. The node chooses it parent that is trusted or not based on threshold. The authors proposed to fix the drawback of TRAIL when a child node chose a parent node.
SoS-RPL	Detection	Node rating and ranking	Simulation in SN-3	The authors present an easy and understandable mechanism to detect the sinkhole attack by rating two rank values; rank variation between child node is DV-RANK, and between source node and receive node is DI-RANK. Then they simulated their implementation with 500 nodes and showed the better results compared with other four approaches

1.7: Thesis Layout

There are five chapters in this thesis. Chapter 1 describes thesis motivation, problem statement, objective, contributions, and literature review. Chapter 2 studies background technology such as RPL, Attack in RPL, TASA, etc. Chapter 3 explains the sinkhole attack that is easily creatable but serious damage to the network and discusses the different damages of sinkhole attacks in each level of the network with simulation results. Chapter 4 demonstrates the simulation testing that avoids the sinkhole attack and implements the RPL with TASA to know the traffic of the network and make scheduling in a simulated network. Finally, Chapter 5 will conclude the thesis with future work. References are also outlined at the end of Chapter 5.

Chapter 2
Background

2.1 Internet of Things

Nowadays the usage of IoT devices is more year after year. The Internet of Things (IoT) devices are sensors, actuators and some smart devices which can access internet connection and applied in our everyday activities [1][2]. They are very useful to build a smart environment like smart farming, smart city, smart health-care system. The IoT involves billions of connected devices which are collecting and sharing data between each other. The development of IoT has been taking since last ten years ago but the security issues go down the rapidity of the development of IoT. The IoT architecture was proposed by at least three layers or four or five layers according to different point of views.

In the literature, the architecture of IoT network system was proposed variously by their different interests of the researchers like three-layers based, four or five-layers based, middleware based and services-oriented based. Among them, the three-layers based architecture: 1) Perception Layer, 2) Network Layer and 3) Application Layer is seen commonly in many states of art.

2.2 Low-Power and Lossy Networks

According to **[27]**, Low power and Lossy Network (LLN) is a type of network where all the routers and IoT devices are working on constrained power, memory and computational energy. Generally, LLN routers work with constraints one of those of energy, memory and processing power and all their interconnected nodes are also working on this in wireless medium. In an LLN, there may has a few tens of routers and interconnected devices and up to a hundred or thousands of devices can involve. And the connections between the LLN routers and their interconnected nodes are point to multipoint traffic (P2M), that is from router to all its neighbour nodes, multipoint to point traffic (M2P), from interconnected nodes to router, and point to point traffic (P2P) which is traffic between the interconnected nodes in some special case.

2.3 Internet Protocol version 6 (IPv6) over Low-Power Wireless Personal Area Network

Internet Protocol version 6 (IPv6) over low-power wireless personal area networks (6LoWPAN) is the key protocol for communications over IoT networks. It is a standardized protocol that supports higher layer functions for IEEE 802.15.4 networks, which is characterized by low power and lossy links with scarce resources such as memory and throughput. IoT networks are based on the IEEE 802.15.4 standard, which defines the physical and data link layers of the IoT network stack [4]. From [5], we find that 6LoWPAN enables network connectivity for IPv6 packets over an IP-based infrastructure such as the Internet. This is done through the border router, which is also known as the sink node in an IoT network. 6LoWPAN can also be viewed as a network adaptation layer that allows vertical communications between the Medium Access Control (MAC) layer and the network layer. Its functions include fragmentation and reassembly of datagrams between these two layers as well as header compression on network addresses to allow IPv6 packets to be sent and received over IEEE 802.15.4-based networks. Shown in Figure 1 is the communication paradigm for packets traversing between the IPv6 Internet and IoT enabled network. The network stacks for the Internet and IoT network are also depicted. The IPv6 address that is used by the Internet is compressed into a 6LoWPAN datagram for communications within an IoT network.

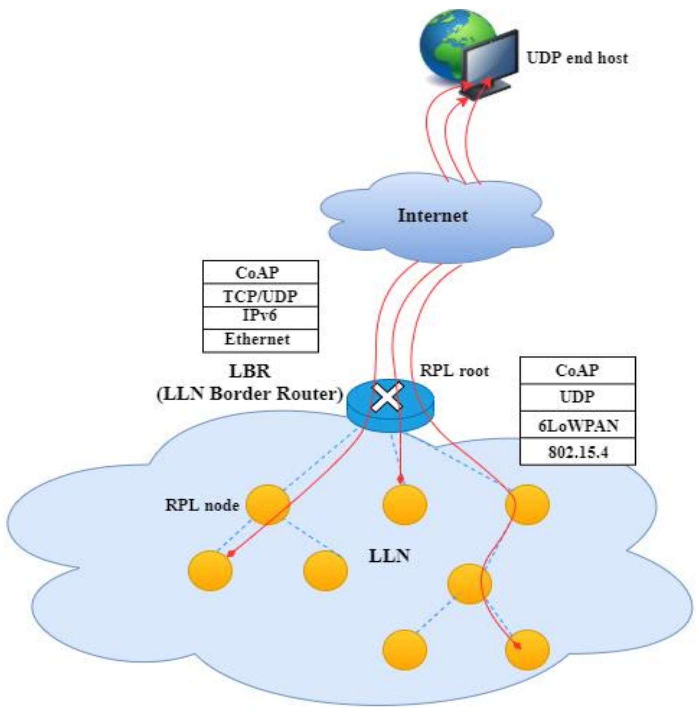


Figure 1 IoT Network Architecture **[24]**

2.4 Routing Protocol for Low-Power and Lossy Networks

RPL was introduced by Internet Engineering Task Force (IETF) as a suitable standard routing for low-power IoT devices. It has been standardized as RFC 6550 in 2012 **[28]**. It was designed to meet the requirements of several applications in the Wireless Sensor Network (WSN) and Internet of Things (IoT) domains.

2.4.1 RPL Architecture

RPL is proposed for LLN network and the overview of the RPL architecture is shown in Figure 2 **[9]**. In a LLN, there might be at least one border router or root node and many low-power IoT devices called non-root nodes. All sensor nodes are connected to the root as a tree structure and in RPL, it is called Destination Oriented Directed Acrylic Graph (DODAG). The border router or root node is also called DODAG root, and it has a unique ID and is represented by the IPv6 address. The border router or DODAG root is the main component of the LLN network and that can only connect to the internet.

According to the RPL network topology, there has another important one is RPL Instance that is a compound of one or more DODAGs. A RPL Instance also has a RPL Instance identifier called RPLInstanceID and DODAGs in same RPL Instance share same RPLInstanceID. Each RPL instance shares a specific Objective Function (OF) **[28]** . This OF is used to compute the position of the nodes in the DODAG and that is called as Rank. Based on the rank value, a node can choose it preferred parent that has lower rank value than child node. The less value of the rank means the closer to the root node and the root node has always the lowest rank value in the DODAG. The last important identifier of RPL is the DODAGVersionNumber that is a specific number of iterations of a DODAG. Sometime a DODAG is reconstructed from the root node to maintain the topology. Then the root node sets the increased version number for the new DAG. The combination of four RPL identifiers, RPLInstanceID, DODAGID, DODAGVersionNumber and Rank value, uniquely identifies a DODAG.

A node's rank is computed by the defined objective function **[21]** in RPL instance and this rank value is increased when the node is farther away from the DODAG root. A parent node that has lower rank value than its child node and nodes which have same rank are called siblings nodes. A node that has no incoming link or that is not a parent of any other nodes is called a leaf node. The rank is increased when route goes to downward, and it is decreased when it goes from the leaf nodes to the root node. And to determine the nodes which are roots or parents or child nodes, rank property is played in vital role in RPL instance. So, maintain the right rank value is the very important for effective routing operation of RPL network. each RPL instance.

According to RFC 6552, a node's rank is determined by the OF and the OF is an optimizing criterion based on different operating scenarios, applications, and network designs. It can be thought of as a set of rules in terms of link metrics and/or constraints to enhance the routing paths in a network based on different design considerations. These include distance, bandwidth, latency, energy, etc. The routing topology in a DODAG is formulated by the selection of the parent with best route link. For this parent selection and rout formation between parent and child nodes, RPL uses four types of control messages. They are DODAG Information Object (DIO), DODAG Information Solicitation (DIS), DODAG Advertisement Object (DAO) and DODAG Advertisement Object Acknowledgement (DAO-ACK). RPL uses trickle timer mechanism [11] to control the sending rate of above control messages especially for DIO messages which are main responsible for DODAG formation in RPL networks. The timer algorithm decides when DIO message is send and can adjust the transmission of DIO message by setting of higher or lower value of trickle timer results [12].

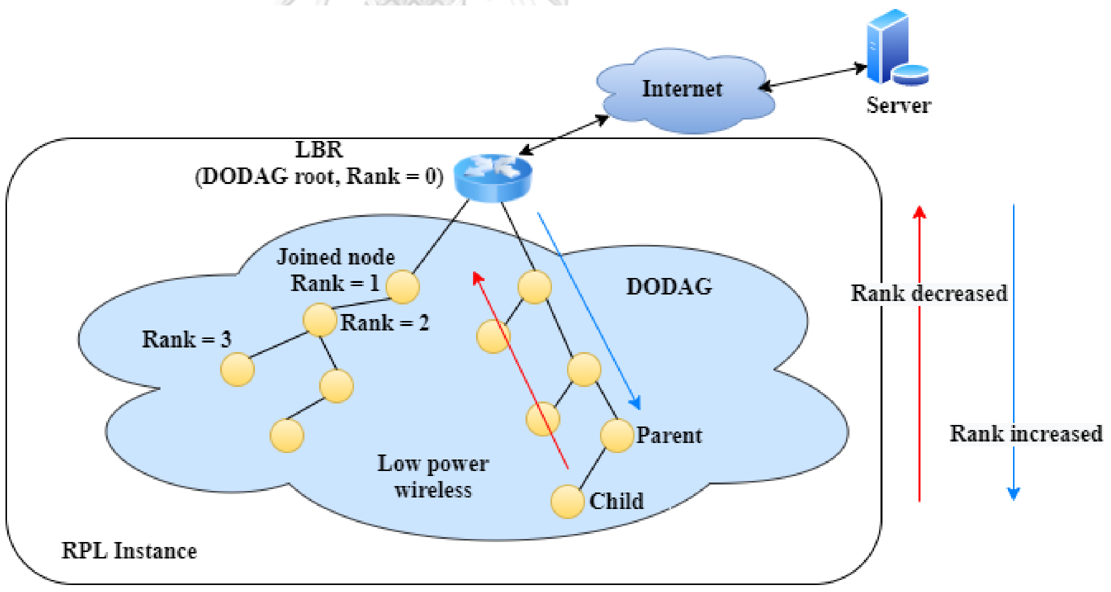


Figure 2. RPL Network Architecture [19]

2.4.2 RPL Control Message

Because RPL use Internet Protocol version 6 (IPv6) for communication, it controls messages are put in the IPv6 packet format called ICMPv6 messages. For the RPL control message, the type of value of ICMPv6 is 155 and next code field shows what kind of this message. The main RPL control message places in the base field of ICMP (Internet Control Message Protocol). Figure 3 shows the brief structure of all RPL control messages, their basic functions and where they have been put in the IPv6 message. The four basic control messages of RPL are

- 0x00: **DODAG Information Solicitation (DIS)** - Request message of a new node to the RPL node to solicit the routing information.
- 0x01: **DODAG Information Object (DIO)** - Carries information that allows a node to discover an RPL Instance, learn its configuration parameters and select DODAG parents.
- 0x02: **Destination Advertisement Object (DAO)** - Used to propagate destination information upwards along the DODAG.
- 0x03 **Destination Advertisement Object Acknowledgement (DAO-ACK)** – Response message from the DAO parent or DODAG root node.

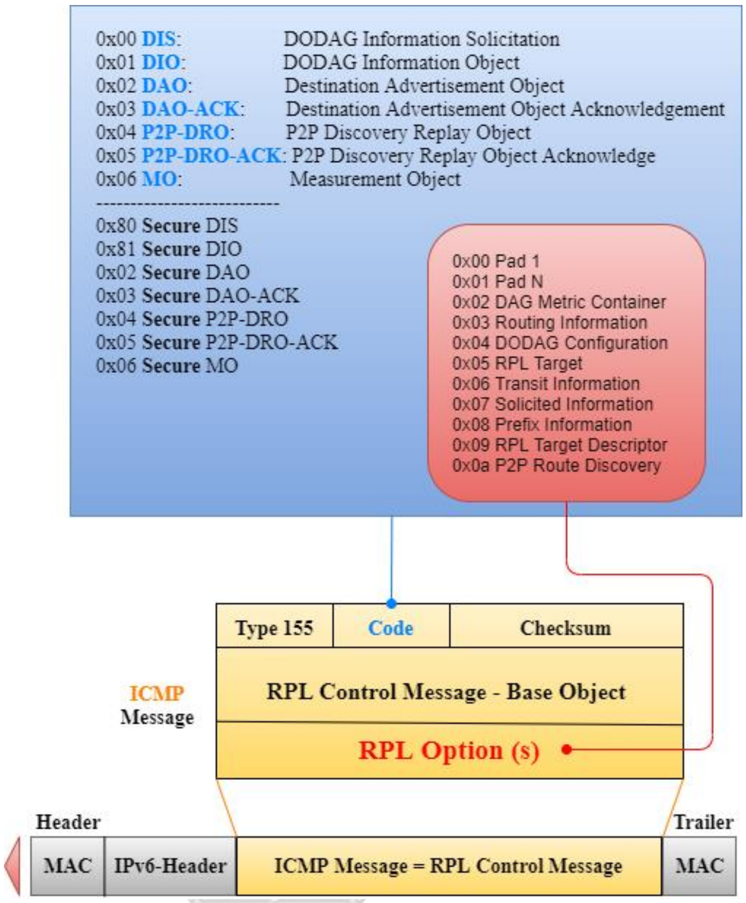

0x00 DIS: DODAG Information Solicitation
0x01 DIO: DODAG Information Object
0x02 DAO: Destination Advertisement Object
0x03 DAO-ACK: Destination Advertisement Object Acknowledgement
0x04 P2P-DRO: P2P Discovery Replay Object
0x05 P2P-DRO-ACK: P2P Discovery Replay Object Acknowledge
0x06 MO: Measurement Object
0x80 Secure DIS
0x81 Secure DIO
0x02 Secure DAO
0x03 Secure DAO-ACK
0x04 Secure P2P-DRO
0x05 Secure P2P-DRO-ACK
0x06 Secure MO
0x00 Pad 1
0x01 Pad N
0x02 DAG Metric Container
0x03 Routing Information
0x04 DODAG Configuration
0x05 RPL Target
0x06 Transit Information
0x07 Solicited Information
0x08 Prefix Information
0x09 RPL Target Descriptor
0x0a P2P Route Discovery
Type 155
Code
Checksum
ICMP Message
RPL Control Message - Base Object
RPL Option (s)
Header
Trailer
MAC
IPv6-Header
ICMP Message = RPL Control Message
MAC

Figure 3 RPL Control Message Structure [23]

2.4.2.1 DODAD Information Object (DIO)

When a DODAG in RPL network is created, there have two ways links for connection. Mostly IoT networks work as data collection networks in smart systems. For sending the collected data to the parent node and then forward these data packets to the root node or border router, the formation of upward link is very important in data collected networks. In RPL, the border router or root node uses DODAG Information Object (DIO) message to create an upward link in networks. The DIO only has the important information such RPLInstanceID, DODAGID, Rank value and Version Number, for DODAG formation in RPL network. The first 8 bits is RPLInstanceID and next 8 bits is Version Number of the current DODAG. Rank occupies 16 bits, and 1 bit

'G' flag defines the goal of the DODAG. Mode of Operation (MOP) is 3 bits to show what type of mode is used for DODAG operation and DODAGPreference (Prf) indicates the level of preferable of the DODAG root. Destination Advertisement Trigger Sequence Number (DTSN) is for the maintenance of the downward routes in DODAG. Flags and Reserved fields are 8 bits unused fields and already assigned with 0 numbers. 128 bits DODAGID is IPv6 address and that is set by the DODAG root or border router. The standard structure of DIO message is shown in Figure 4.

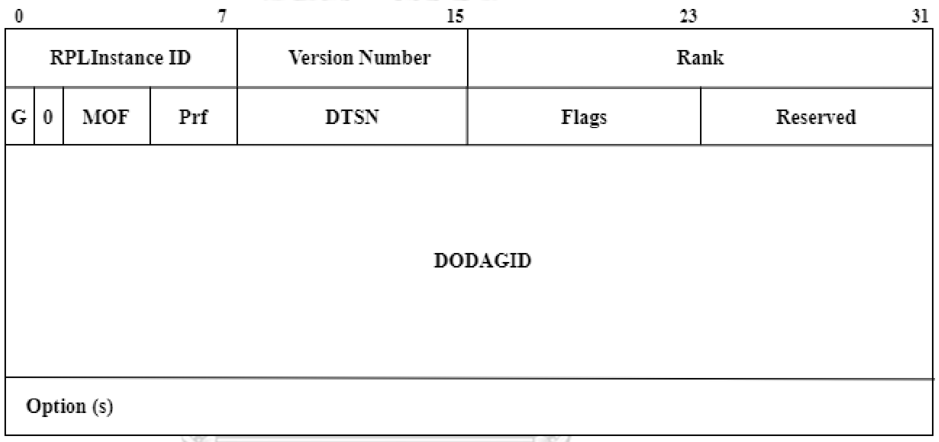


Figure 4 The DIO base Message **[28]**

2.4.3 DODAG Configuration

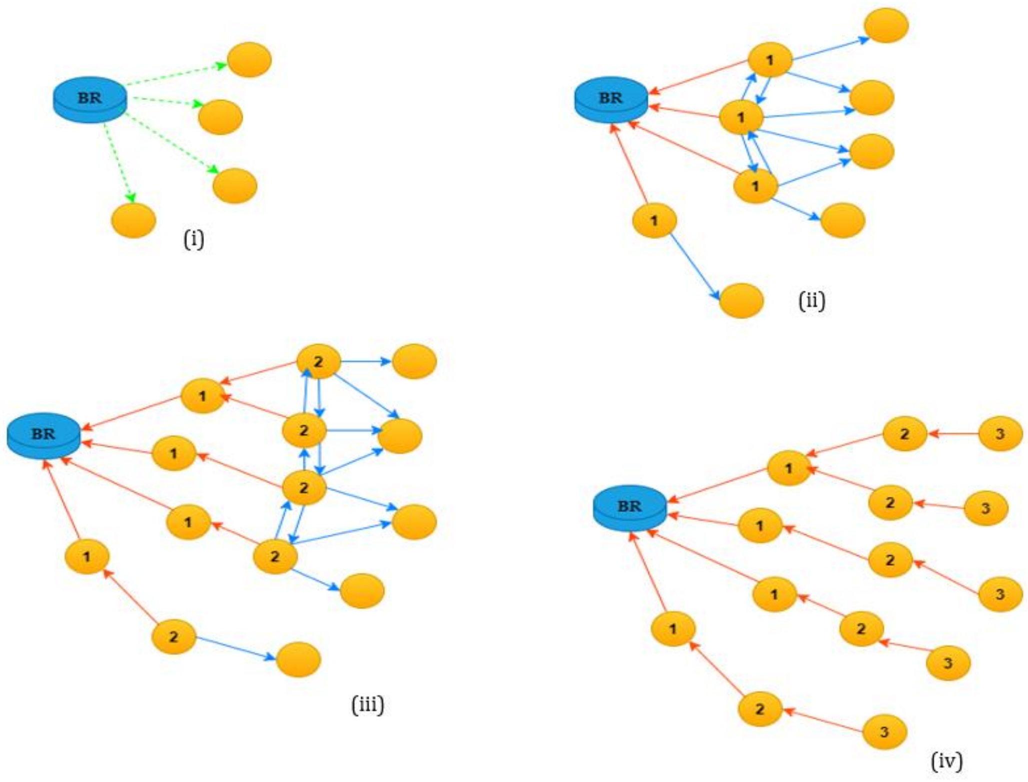


Figure 5.. DODAG Configuration Process [16]:*(i) a root node multicasts DIO messages to the nodes in its transmission range, (ii) by getting the DIO message, the neighbor node selects the root node as its parent, then calculate its rank value according to specific objective function and later multicasts another new DIO to its surroundings (iii) next level node does same procedure as its parent node and (iv) the configuration process is finished when all node in the RPL network have connected to the root node.*

The general task of RPL is to set up the optimal DODAG in an LLN. In RPL, every node constructs a tree-based topology network that has no loop, and it is called Destination Oriented Directed Acyclic Graph. The RPL DODAG configuration process is briefly shown in Figure 5. DODAG is formed by an iterative exchange of DIO and DAO messages between sets of parent and child nodes. Assuming that the OF is a minimum function, such as distance, we see that a lower rank indicates a distance closer to the source.

In the initial stage, the sink node or bolder router multicasts DIO messages to all its neighbours, indicating its presence. Upon receiving the message, the neighbour nodes, yellow ones, which are essentially the child nodes in this case, calculate their associated ranks based on the rank of the sender and the distance to the source. This is followed by a DAO response to the sink node with corresponding advertisement on its route information. Upon accepting this information, the sink node then provides an acknowledgement with a DAO-ACK message.

This process repeats iteratively at the next tier with the parent nodes as Rank 2. In a separate scenario where there is a route update due to the change of rank, the process is similar. Multicast DIO messages are sent to all neighbours first, followed by the exchange of DAO and DAO-ACK messages to complete the route update process. Additionally, in the case when a new node enters the network, the only difference is that the new node first multicasts DIS messages to its neighbours. The neighbours then respond with multicast DIO messages, and the follow-on process is the same as what has been discussed.

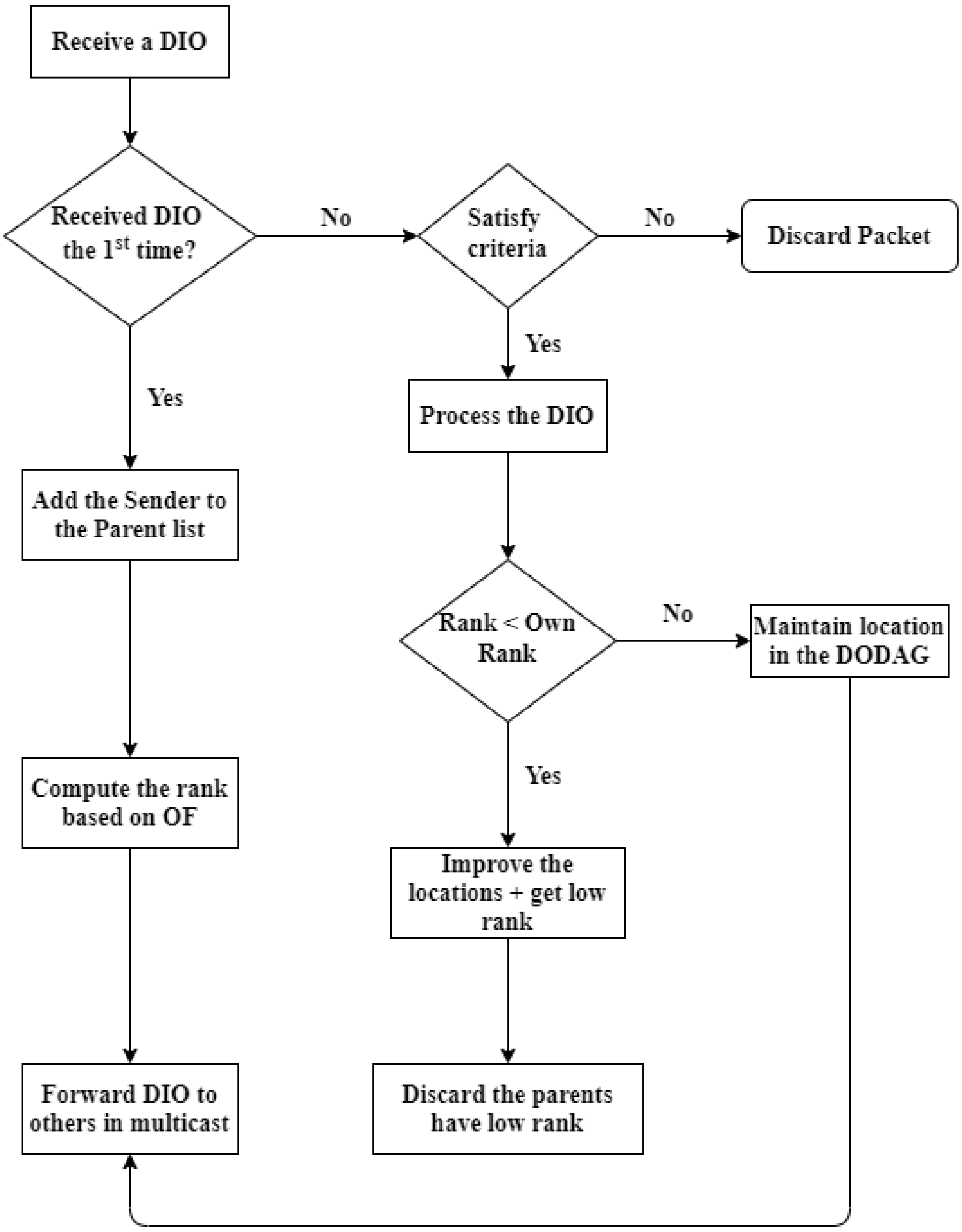

Receive a DIO
Received DIO the 1st time?
No
Satisfy criteria
No
Discard Packet
Yes
Yes
Process the DIO
Add the Sender to the Parent list
Rank < Own Rank
No
Maintain location in the DODAG
Yes
Compute the rank based on OF
Improve the locations + get low rank
Forward DIO to others in multicast
Discard the parents have low rank

Figure 6 Flowchart of Node Operation in DODAG **[14]**

2.4.4 RPL Vulnerabilities

RPL works on low power devices and in lossy networks. Like other networks, RPL networks also have the same security issues. They have the vulnerability of both passive and active attacks **[28]**. Because they have constrained storage and limited power and are low cost, RPL nodes are not very week and not always said good enough to defence a kind of attack.

Some special function nodes such as Low Power Border Router (LBR)can also vulnerable RPL networks **[10]**. They require the assurance in a security context of the availability of communication channels and the neighbour discovery process.

2.4.5 Attacks on RPL

Because of the constrained structures of LLN networks such as poor infrastructures, limited resources, lossy connection and lack of physical security, there might be many security attacks happened in the routing in a LLN network. Although many attacks can easily happen in RPL, it is very hard to detect or protect these attacks. But RPL provides some basic mechanisms like local and global repair mechanisms and to detect or avoid the loop formation in the network. Moreover, RPL supports three security modes for routing, but standard RPL works in unsecured mode to reduce the energy consumption because they are working on low power devices. Therefore, a standard RPL can assume a very week in routing protocol comparing with the others routing protocol like Open Shortest Path First (OSPF).

Because of RPL's lightweight structure, there have been many attacks on the RPL routing protocol. All these attacks are basically classified into two kinds: direct and indirect attack. In active attack, the adversary node cracks the network in order to modify, isolate, exhaust and obliterate the data. The passive attack sniffs the crucial information between the legitimate nodes or inside the network. Mostly attacks in RPL are active type and passive is a few. Both active and passive can harm part of the network or even the whole system.

For more details, the attacking types on RPL protocol can classify into three groups **[22]**. In Figure 6, a taxonomy of attacks on RPL routing protocol is clearly graphed based on three base groups: 1). Attacks on Network Resources: Attacks are targeting the exhaustion of network resources and they are very damaging for constrained networks because they have low power, memory, and energy, 2). Attacks on Network Topology: These attacks focus to disrupt the topology of RPL network and attackers target the isolating of some RPL nodes from the main network or the sub-optimization of the network topology. 3). Attacks on Network Traffic: This group of attacks harm the network traffics by eavesdropping or impersonation behaviours.

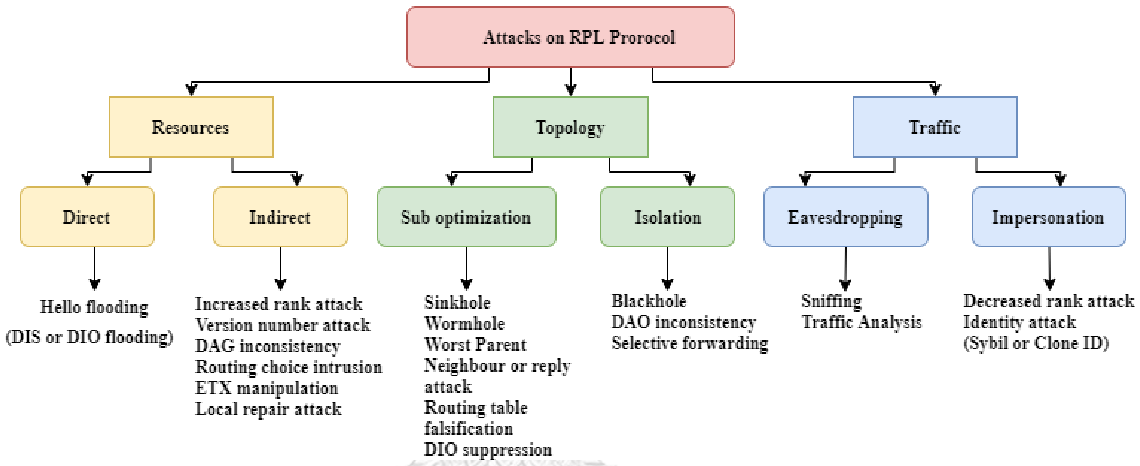


Figure 7. Taxonomy of Attacks on RPL [22]

2.4.5.1 Attacks on Resources

Some attacks on RPL are classified as resources attacks according to their attacking behaviors. The resources attacks on RPL are categorized again into two types: direct attack and indirect attack. In this kind of direct attack, the attacker node directly attacks to the node for exhaustion of the node resources such as power, memory, by sending many hello messages. These attacks can call Hello flooding attacks because of their attack type, and these can be divided in two like DIS flooding and DIO flooding attack in RPL. In indirect attack type, the adversary node causes the other legitimate nodes create an overwhelm for the network. These kind of attack on RPL are, ETX

(Expected Transmission Count) manipulation, Version number attack, Local repair attack, Increased rank attack, routing choice intrusion and DAG inconsistency attack.

2.4.5.2 Attacks on Traffic

Based on the effects of attacking structures, some attacks are named as attacks on the RPL routing traffic. In this kind, it also mainly subdivides into two forms: eavesdropping and impersonation or misappropriation attacks. The attacks act like eavesdroppers in the network to access the routing information. Later they can re-apply this information in unauthorized action or share to other to do more advance actions. Sniffing attack and Traffic analysis attacks are two kinds of eavesdropping attack on RPL. In this group of attack, the attacking node copies or clone the information of legitimate nodes and acts onto another node to get access the large part of network. There have two kinds of impersonation attack; Decreased rank attack and Identity attacks, the Identity attack is sub-divided into two, Clone ID and Sybil attack.

2.4.5.3 Attacks on Topology

On RPL routing protocol, some attacks disrupt the network topology, make false route and route disruption. These kinds of attacks are named topology attacks and are classified into two types on detail of their harming structure: sub optimization attack and isolation attack. The attacker node makes the network as sub-optimization from the main network by disrupting between two legitimate nodes. Sinkhole attack, Wormhole attack, Worst parent attack, Neighbour or replay attack, Routing table falsification attack and DIO suppression attack are under this kind of sub optimization attack on RPL routing protocol. In some case, a malicious nodes make some part of network isolate from the main network. Then this isolated part lost the connectivity of the whole network and far away from the current activity of the network. This isolation attack mainly includes Blackhole attack, DAO inconsistency attack and Selective forwarding attack.

2.4.6 Security of RPL

RPL supports three security modes for DODAG operation. They are Unsecured mode, Preinstalled mode, and Authentication mode.

Unsecured Mode: In this mode, the RPL control messages are sent freely that means no need to add any security protection when a DIO, DIS, DAO or DAO-ACK message is sent.

Preinstalled Mode: By compiling with cryptographic secure mechanism, RPL control messages are sent in this secure mode. To get the successful secure connection, every node must have a preinstalled key at boot time and use this key for joining RPL network.

Authentication Mode: In this secure mode, the node works same as preinstalled mode mechanism, but it is only for host node joining the network. For the node that works as a router in the network, it must need another key from a key authority and need to authenticate with him.

2.5 Traffic Aware Scheduling Algorithm

The Traffic Aware Scheduling Algorithm (TASA) was introduced as a centralized scheduling algorithm for Time Synchronized Channel Hopping (TSCH) that is highly reliable low-power Medium Access Control (MAC) protocols [29]. TASA provides a highly efficient plan for TSCH with the concepts of combination matching and coloring. In TASA, low power and lossy network routing protocols are considered network layer protocols over IEEE 802.15.4eMAC. TASA can provide for many smart wireless systems which need to work with low latency at low power consumption [30].

TASA works based on the network topology that must has a tree topology like a DODAG of the RPL network and network traffic load of sensor node of the network that is a constant integer number of packets. When running the TASA procedure in the RPL, TASA constructs time/frequency patterns for the nodes in the network. The root node knows the whole topology of the network and all the other nodes in the network are forwarded the traffic load to the root. Node in the network neither can do transmitting or receiving the packet to the other node at the same time nor receive the packet from many nodes at the same time. After that, the main two steps of *Matching* and *Colouring* are performed in the network and then forwards the traffic within a minimum timeslot.

Chapter 3
Sinkhole Attacks in Routing Protocol for Low Power and Lossy Network

In this chapter, we will study the impact of the Sinkhole attack on the RPL network. A sinkhole attack is one of the topology attacks and it can create attacking behaviour easily and make serious damage to the network. According to the network structure of RPL, the position of a node in the network is very important and the damage of the sinkhole node can be varied due to the different locations. We will create an RPL network and plot various effects of the Sinkhole attack in different ranks (level). We will evaluate the impact of the sinkhole node in our routing topology and measure and compare the throughput of every single node in the RPL network.

3.1 Sinkhole Attack

Among the many types of topology attacks in RPL routing, the sinkhole attack is an internal attack, and it is very easy to create by decreasing the actual rank value of the node **[15]**. To launch a simple Sinkhole attack in the RPL network, the adversary node makes its fake DIO messages with a false rank value that is lower than its actual rank value. Then it broadcast its fake DIO message to its neighbour to persuade them to become the preferred parent of all nodes in its coverage range. The adversary node will make its fake rank value as low as possible to induce the neighbours. The less the rank value, the more attracted to not only its child nodes and neighbour nodes that have the same rank value, but its parent node also will join back to its as it preferred parent.

So, adversary nodes are created the sinkhole attack by reducing their own rank value but rank 0, the rank of the root node, and rank 1 the closest level of the root node are hard to make it.

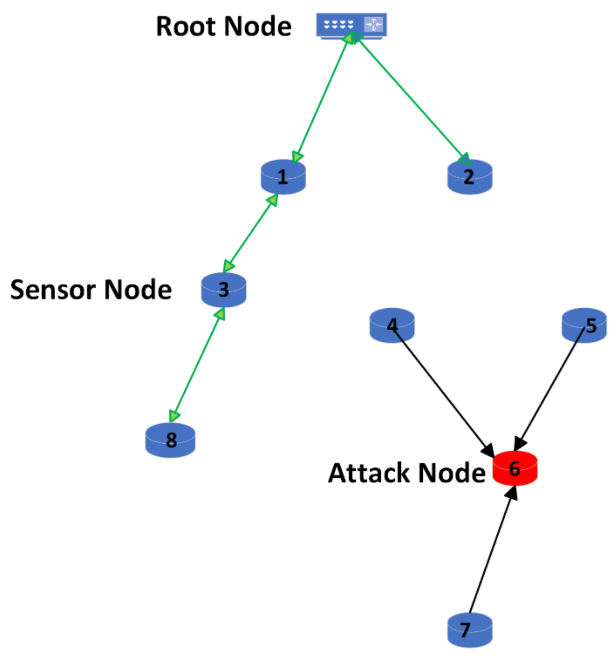


Figure 8 Example of Sinkhole Attack on RPL Network

3.1.1 Sinkhole Attack Behavior

In sinkhole attacks, the attacker node advertises a beneficial path to attract many nearby nodes to route traffic through it. This attack disrupts the network operation, and it can become very powerful when combined with another sinkhole attack, which becomes a Blackhole attack. RPL does not have the self-healing capacity against the sinkhole.

The attacks against the topology also serve as a support for isolating a node or a subset of nodes in the RPL network which means that those nodes are no longer able to communicate with their parents or with the root. In a Blackhole attack, a malicious intruder drops all the packets that it is supposed to forward. This attack can be very damaging when combined with a sinkhole attack causing the loss of a large part of the traffic. It can be seen as a type of denial-of-service attack. If the attacker is located at a strategic position in the graph, it can isolate several nodes from the network. There is also a variant of this attack called gray hole (or also selective forwarding attack) where the attacker only discards a specific subpart of the network traffic.

3.1.2 Sinkhole Attack Formation

An attacker can create a sinkhole attack easily by reducing the actual rank value of the node as low as possible. In the RPL network, every node in the DODAG is placed by its specific DODAG ID, Instance ID and its own Rank value. The lowest rank is the root node of the network, and the highest rank is the end nodes or child nodes which have no child nodes anymore in the DODAG. Sinkhole attacks can form at every level node except level 0 node (Rank 1 of the root node) and level 1 nodes (primary child nodes of the root node) which are very hard to change their rank values. Many attackers make the sinkhole attack at the nearest level of the root node because it has a big impact on the local network. Although the attacker node is not placed at the low level, if the position of the attacker node is very central or close to many other nodes, it may harm or make serious damage to the network. Therefore, the formation of the sinkhole node in the RPL network is very important to determine how much a malicious node can be harmful to the entire network.

In Figure 9 (i), the root node is at the topmost level, and it broadcasts the DIO message to its neighbour nodes, node 1 and node 2. Node 1 transmits the DIO message again to the neighbours, and node 3 has become its child node. At the same time, node 2 makes a DODAG formation by transmitting a DIO message to the last end node that has no child node anymore. In Figure 9 (ii), when creating an attack behaviour, node 6 is a malicious node and it broadcasts its DIO message again in the range of its transmission. Because of the fake lower rank value, not only its child node but also its parent node and same level nodes are chosen as their preferred parent. They forward their data packets to node 6 to make a shorter way for data transmission. But the malicious node, node 6 is a sinkhole attack node so it will not forward and discard all the packets which are coming to it. Then the sub DODAG that is forming at node 6 is isolated from the main network that can see in Figure 9 (iii).

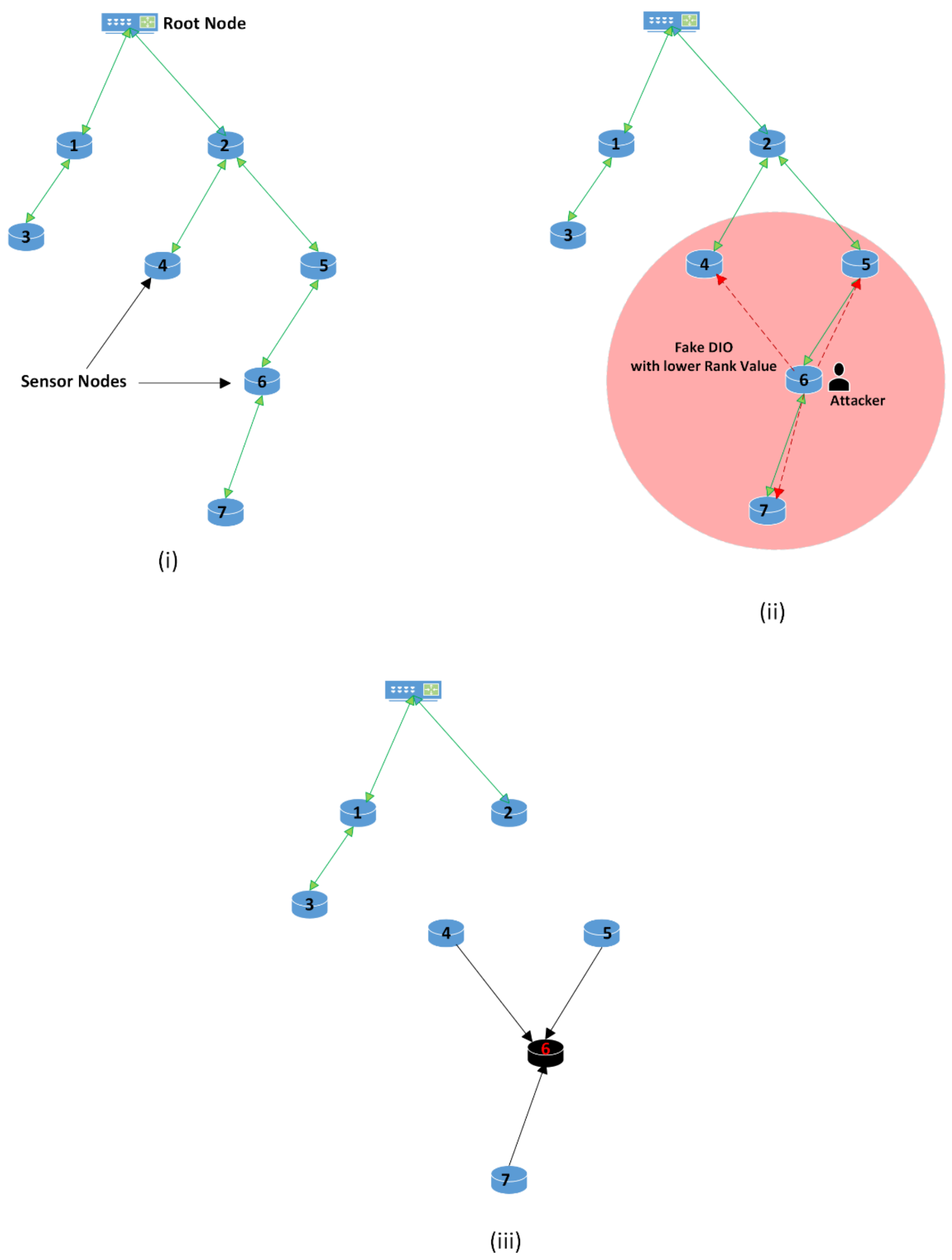


Figure 9 Sinkhole Attack formation in RPL network

3.2 Implementation of Sinkhole Attack in RPL Network

In this section, we will implement sinkhole attack formation in the RPL network. We will implement the sinkhole attack at every single node on different levels in the same topology. Our implementation is simulated with python programming and runs the topology with the different local traffic. In our implementation, the root node is Rank 1 (level 0) and other sources/child nodes are placed randomly by different levels. We will assume every child node has dual parents except Rank 1 (level 1) nodes because they have only the root node as their preferred parent. Because of the behaviour of the sinkhole node, attack formation will not make in level 1 nodes, but other remaining levels of nodes will make. The main parameters of the implementation are given in the following table.

Table 2 Parameters for Simulation

Parameter	Value
Simulation Tool	Python
Routing Protocol	RPL
Simulation Coverage	120x120cm
Inference Range	70cm
Total Number of Nodes	50
Root Node	1
Malicious Nodes	1

Figures 10 and 11 show the implementation results of the RPL network. In Figure 10, the RPL network is simulated with random traffic value at each node. Root node in level 0 and all the other 50 nodes are placed randomly in the network to form the DODAG. When we create a sinkhole attack that is formed at the node ID 22 in the network, not only its child nodes of node ID 28 and 20, but also some child nodes of node ID 24, same level node, node 8, 14, 24, 36 and 49, and its preferred parent node 20 are reconnected back to it because of sinkhole attack behaviour. We can see the simulation result in Figure 11.

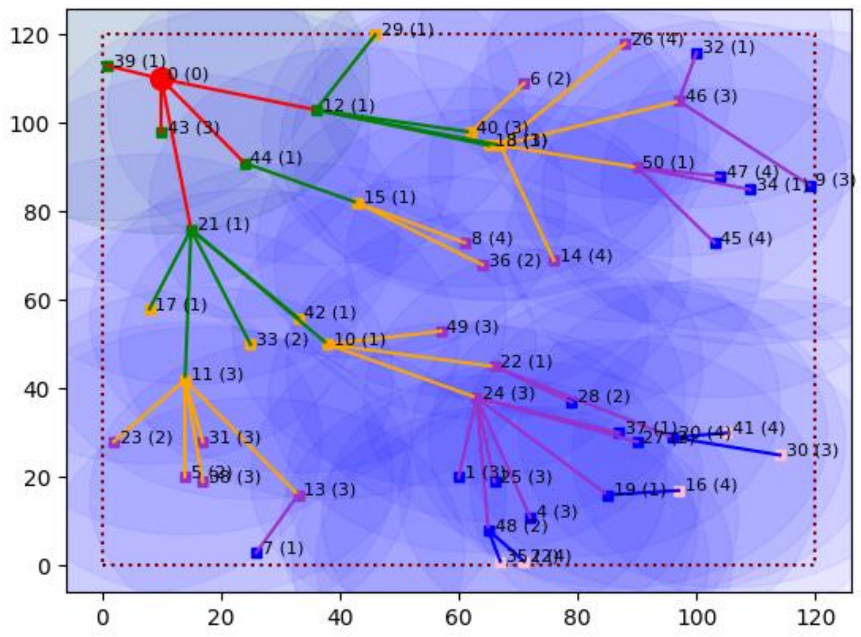


Figure 10. Example of RPL Network with Random Traffic Value

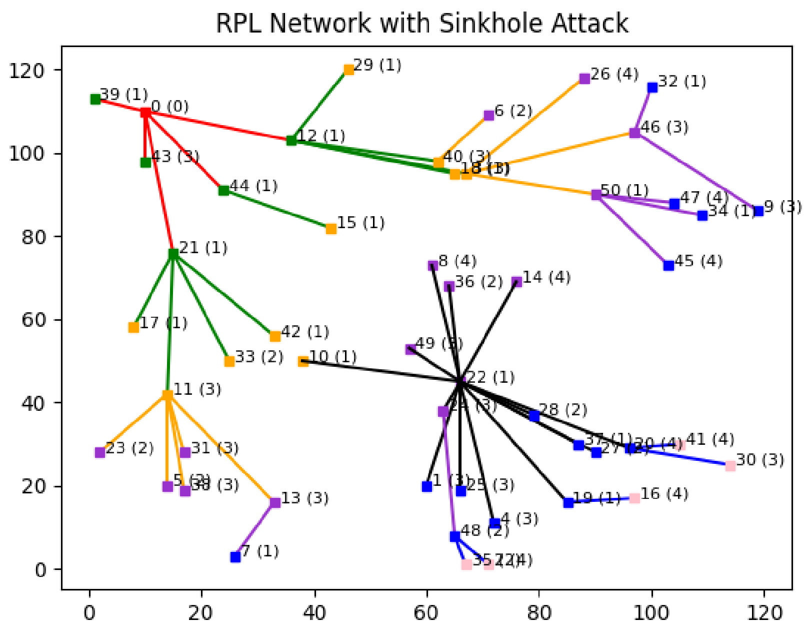
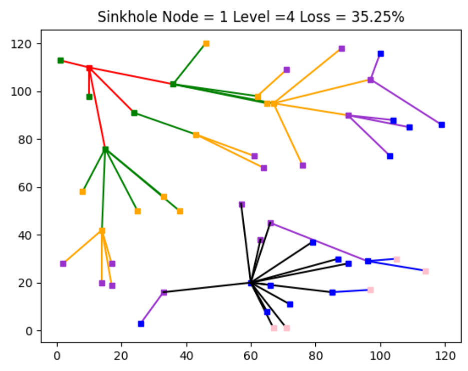


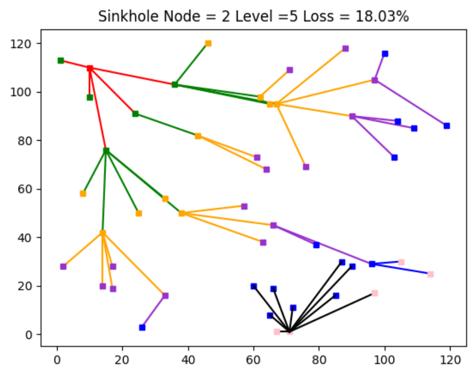
Figure 11. RPL Network with Sinkhole Attack with Traffic Loss

3.3 Results and Discussion

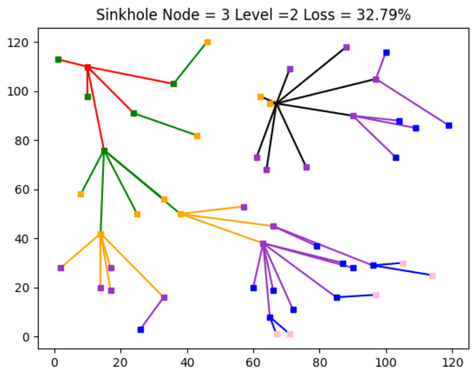
This section will present the effects of different sinkhole attacks at every node of levels 2, 4, 5, and 6 in the same topology. We simulate an RPL topology with fifty sensor nodes randomly placed at various levels. The node number is the root node (red colour in the plot) that has the lowest rank (Rank 1) and the other fifty nodes; Rank 2 nodes are 5 nodes, Rank 3 nodes are 10, Rank 4 nodes are 15, Rank 5 nodes are 15 and Rank 6 nodes are 5, are coloured with a different colour; green, yellow, purple, blue and pink. An attacking node is formed at every single node of each level, and we run our simulation with the same topology to get the obvious results which are needed to compare which one has the lowest traffic loss and which one has the highest impact on the network. In each figure, we plot the attack node ID, where the attack node has existed (the level of the node) and the percentage of traffic loss that is we calculated based on the total traffic of the topology. The traffic losses of each node are presented in the Table 3.



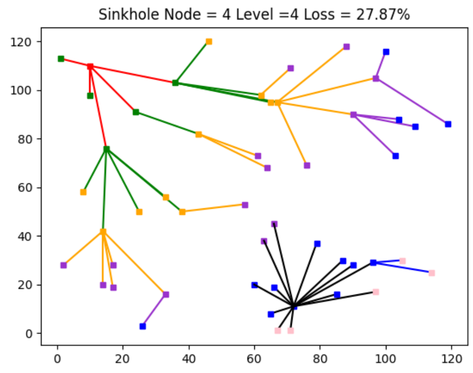
(i)



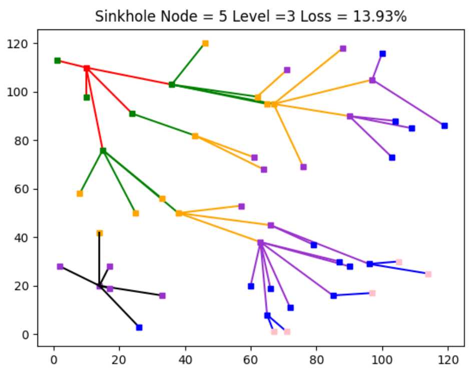
(ii)



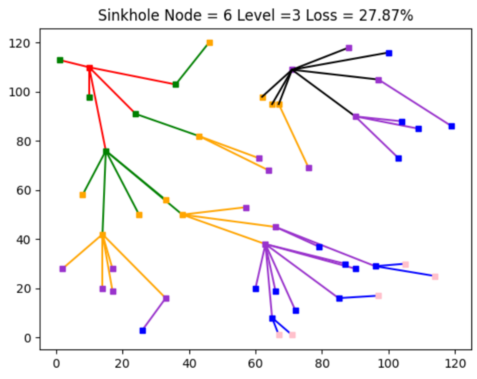
(iii)



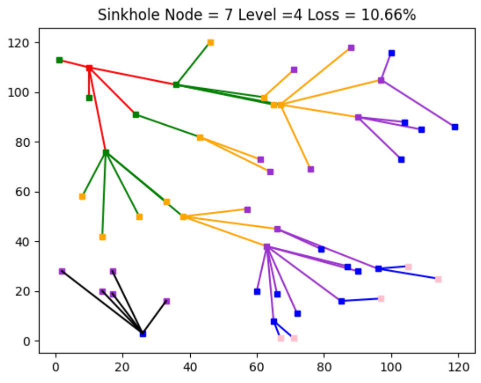
(iv)



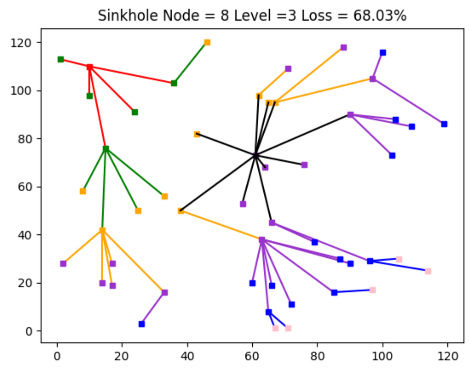
(v)



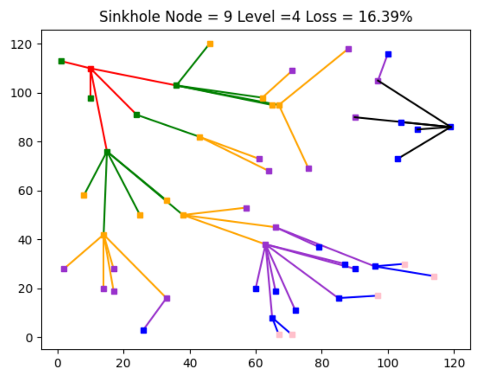
(vi)



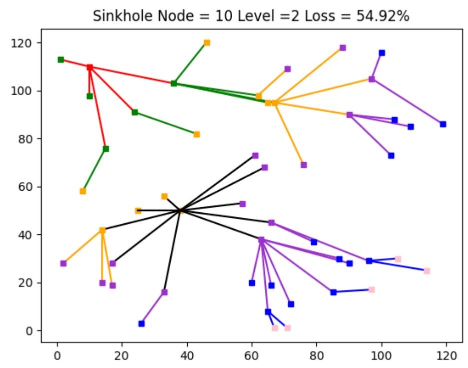
(vii)



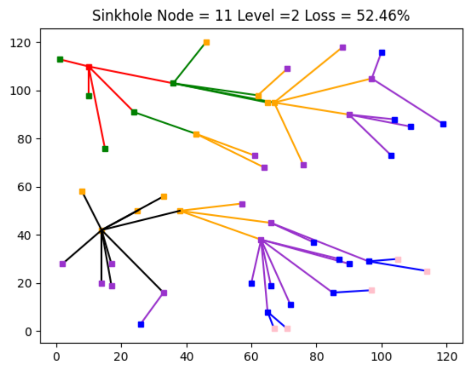
(viii)



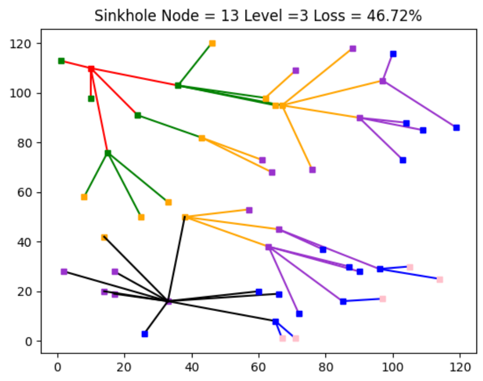
(ix)



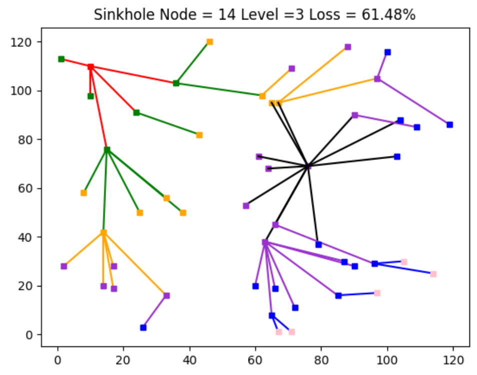
(x)



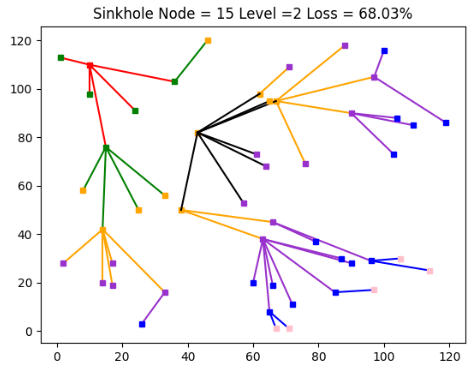
(xi)



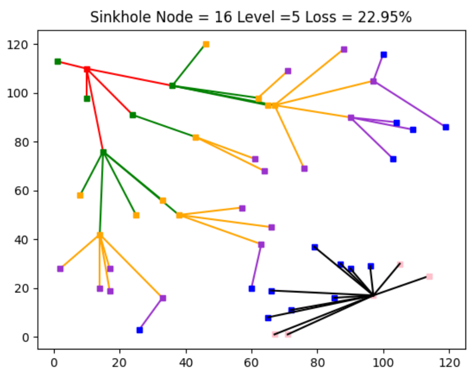
(xii)



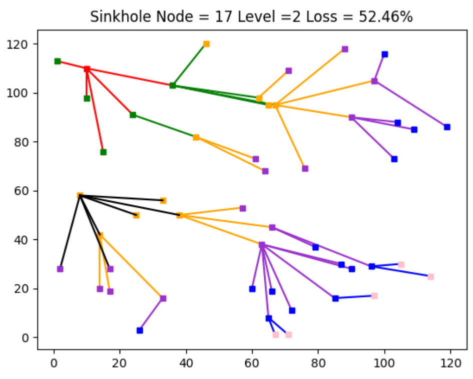
(xiii)



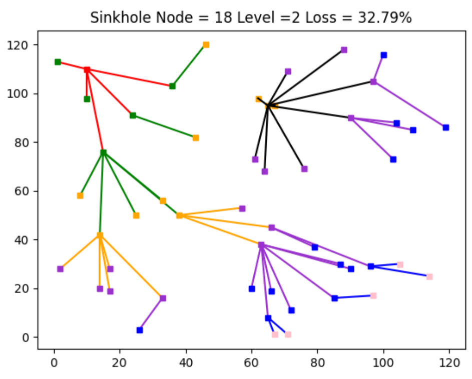
(xiv)



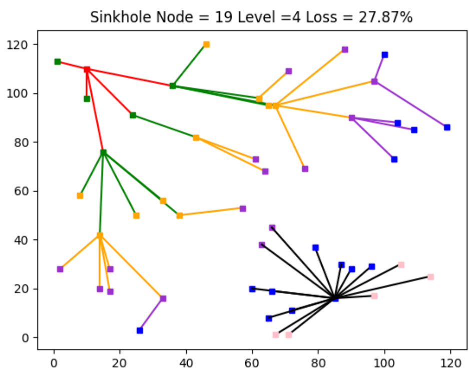
(xv)



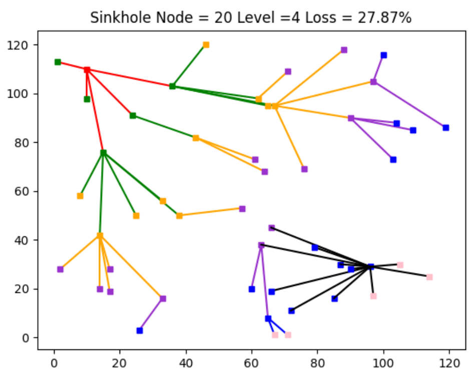
(xvi)



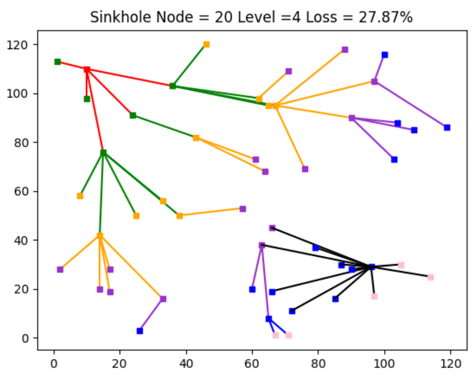
(xvii)



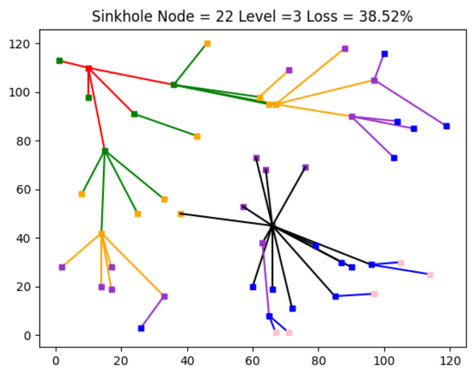
(xviii)



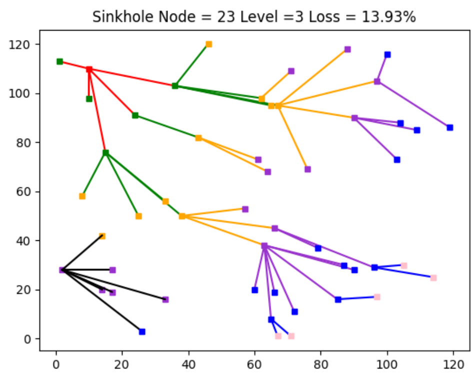
(xix)



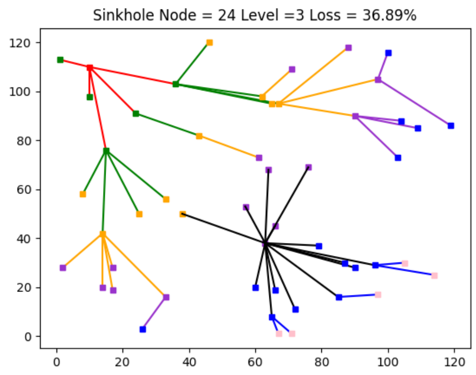
(xx)



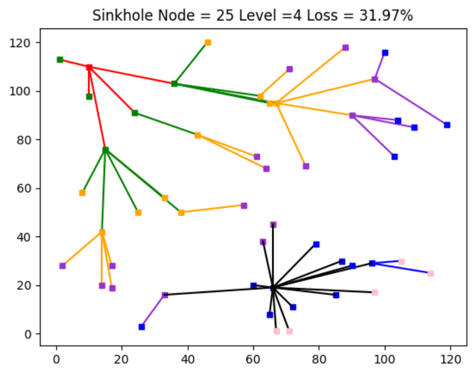
(xxi)



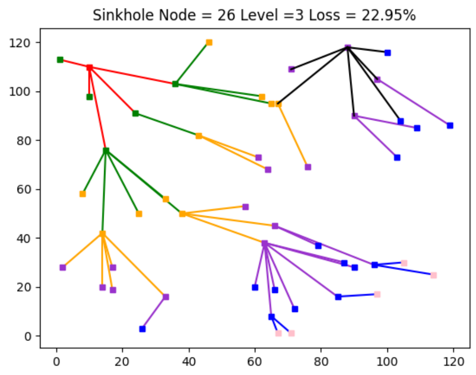
(xxii)



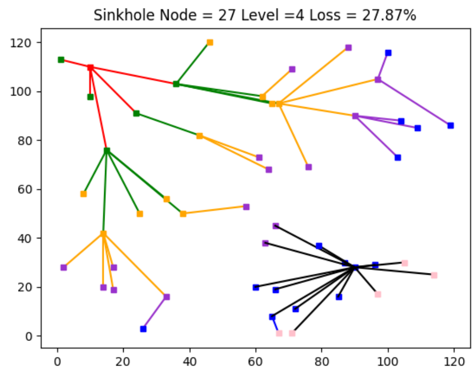
(xxiii)



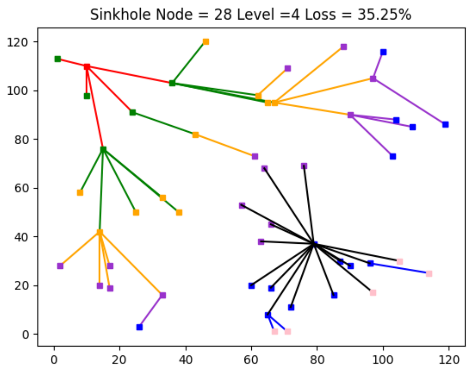
(xxiv)



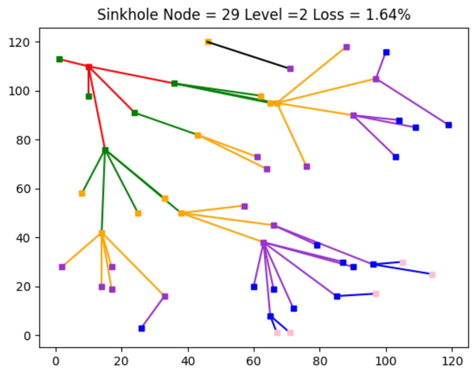
(xxv)



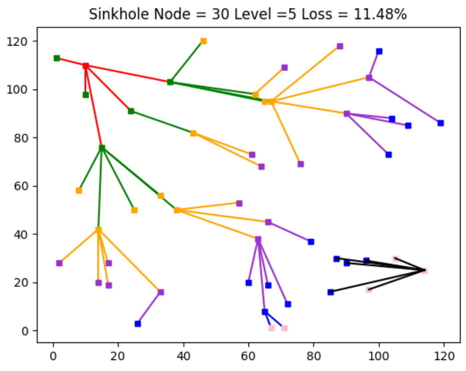
(xxvi)



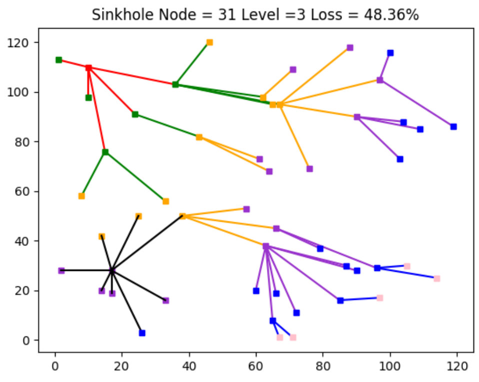
(xxvii)



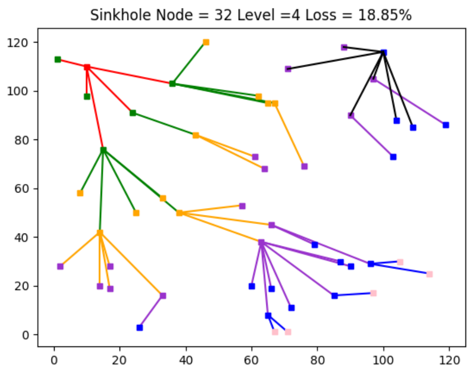
(xxviii)



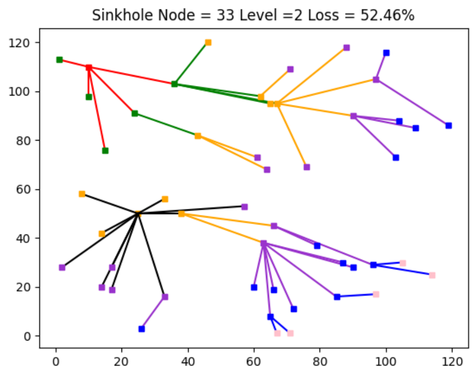
(xxix)



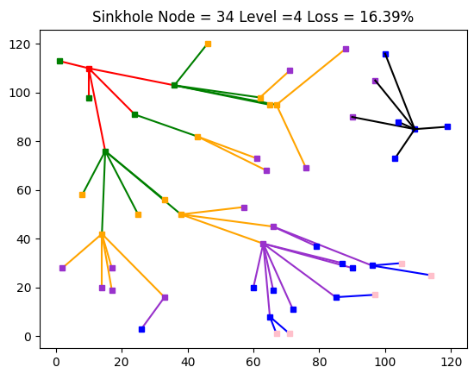
(xxx)



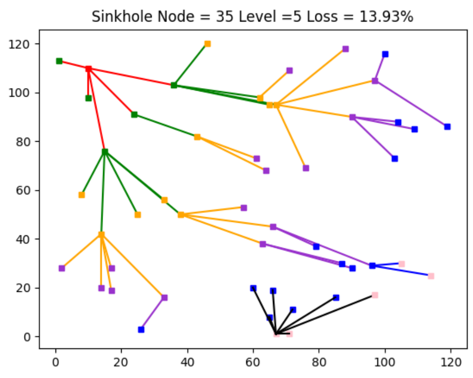
(xxxi)



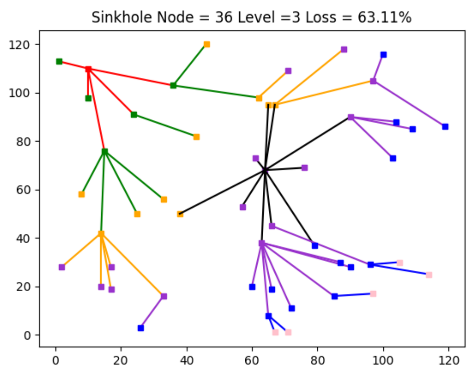
(xxxii)



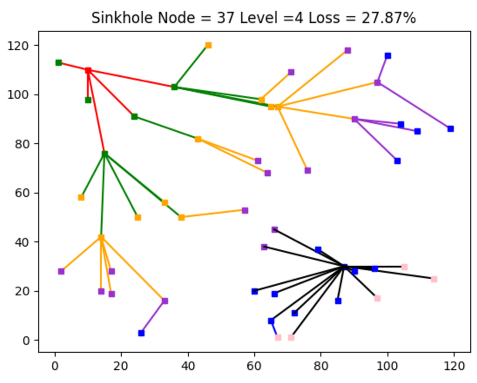
(xxxiii)



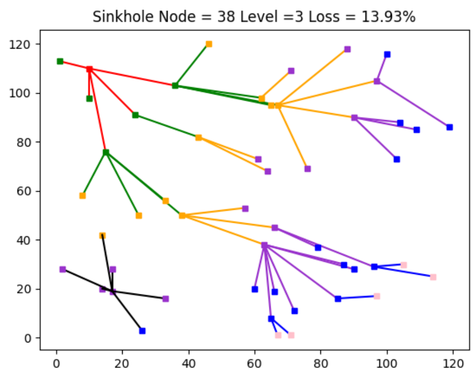
(xxxiv)



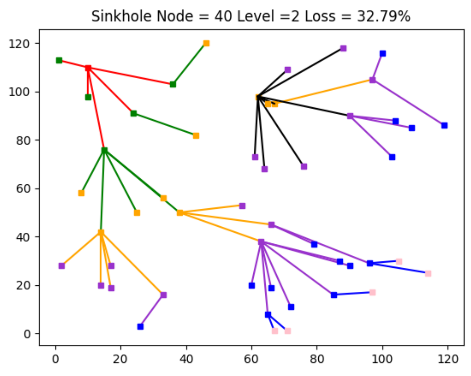
(xxxv)



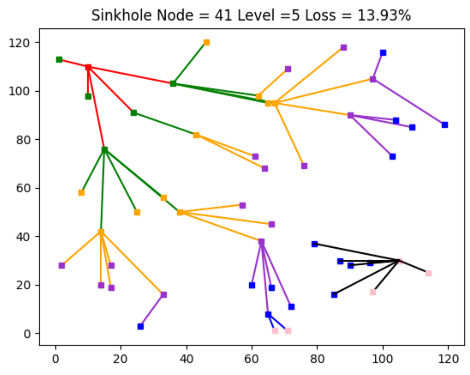
(xxxvi)



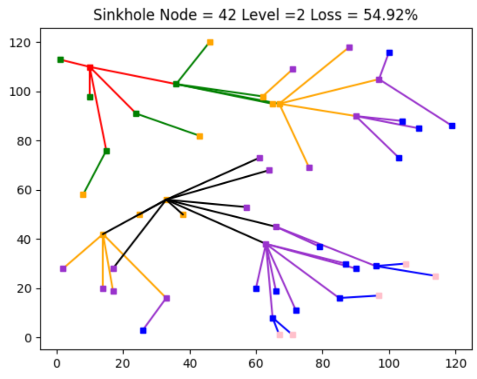
(xxxvii)



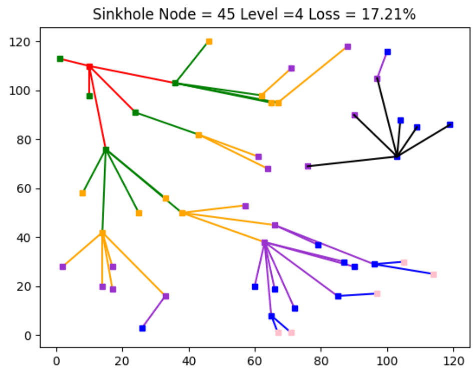
(xxxviii)



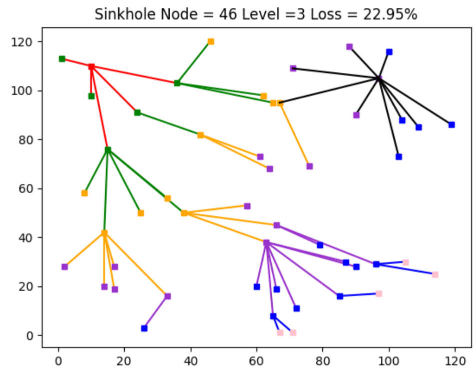
(xxxix)



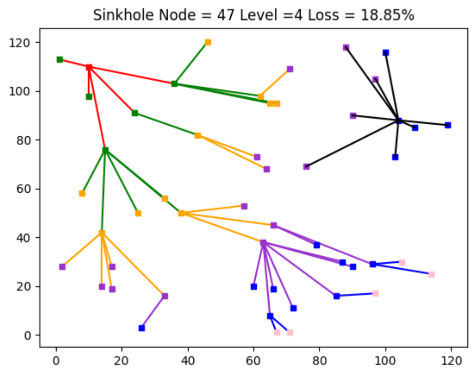
(xl)



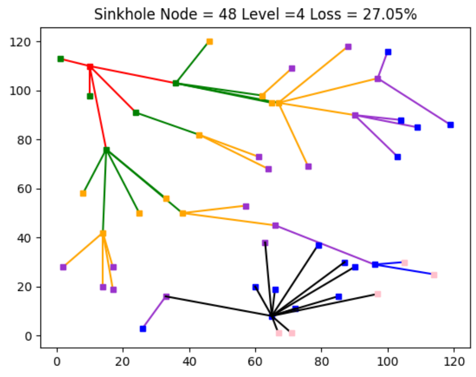
(xli)



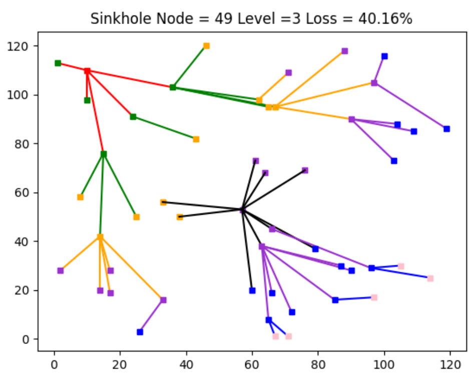
(xlii)



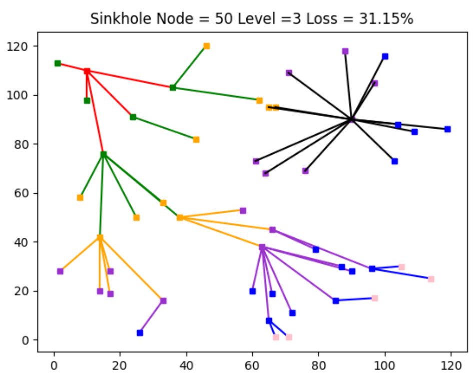
(xliii)



(xliv)



(xlv)



(xlvi)

Figure 12 Sinkhole Attack in Specific Node at Different Levels in RPL network

Table 3 Traffic Loss of the Network when the Sinkhole Attack is located at Each Specific Node ID

Node ID	Level	Traffic Loss (%)
1	4	35.25
2	5	18.03
3	2	32.79
4	4	27.87
5	3	13.93
6	3	27.87
7	4	10.66
8	3	68.03
9	4	16.39
10	2	54.92
11	2	52.46
13	3	46.72
14	3	61.48
15	2	68.03
16	5	22.95
17	2	52.46

18	2	32.79
19	4	27.87
20	4	27.87
22	3	38.52
23	3	13.93
24	3	36.89
25	4	31.97
26	3	22.95
27	4	27.87
28	4	35.25
29	2	1.64
30	5	11.48
31	3	48.36
32	4	18.85
33	2	52.46
34	4	16.39
35	5	13.93
36	3	63.11
37	4	27.87
38	3	13.93
40	2	32.79
41	5	13.93
42	2	54.92
45	4	17.21
46	3	22.95
47	4	18.85
48	4	27.05
49	3	40.16
0	3	31.15

Table 4 Traffic loss of the Network when the Attacker is located at Nodes of Different Node Levels

Level	Number of Nodes	Minimum Traffic Loss (%)	Maximum Traffic Loss (%)	Average Traffic Loss (%)
2	10	1.64	68.03	43.03
3	15	13.93	68.03	36.67
4	15	10.66	35.25	24.48
5	5	11.48	22.95	16.06

Table 3 discusses the traffic loss of the whole network when the sinkhole attack is located at each specific node ID of in every level except in level 1. We make every single node at a different level to know each impact of the sinkhole attack on our same RPL topology. Even though the node is a bit far away from the root node, it can make big isolation and the percentage of traffic loss is high, see traffic loss of node number 16 in Table 3. In our topology, the effect of node number 29 is the lowest, and node number 15 in level 2 and node number 8 in level 3 have the highest effect. We can check what is the minimum and maximum traffic losses of each level of our testing in the Table 4. According to that table, the minimum traffic loss of the whole topology is at level 2, and levels 2 and 3 have maximum traffic loss. When we do the average traffic loss of each level, the nearest level has the highest percentage and the furthest level from the root node has the lowest value as see in Table 4.

Figure 13 shows the comparison of the traffic loss of each attack node in the RPL network. We plot the bar chart with assorted colours for each level, but we arrange every single bar according to the ascending order of the node ID. In Figure 13, the x-axis represents the node ID of our topology, and the y-axis represents the percentage of traffic loss of the attack node. According to our simulation, there have 4 levels of attacking nodes and these levels are represented in colour, blue, orange, green and red. Because nodes are placed randomly, the colour is diffused in the figure and a little hard to classify which node has the height traffic loss and which level has the highest or lowest traffic loss too. Therefore, we plot another figure, Figure 14, to show the traffic loss at each level of the network. In Figure 14, nodes are gathered by their same level and arranged the bars to follow the level.

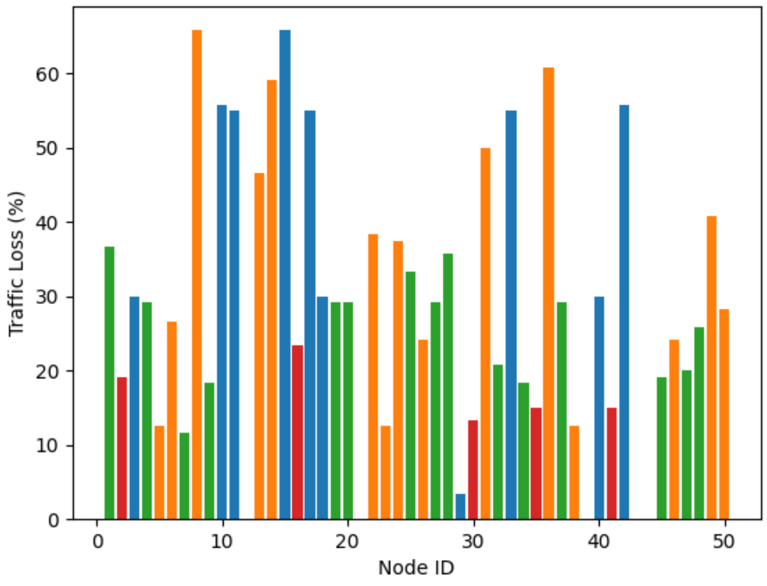


Figure 13 Comparison of the traffic loss of the network by the sinkhole attack at each specific node ID

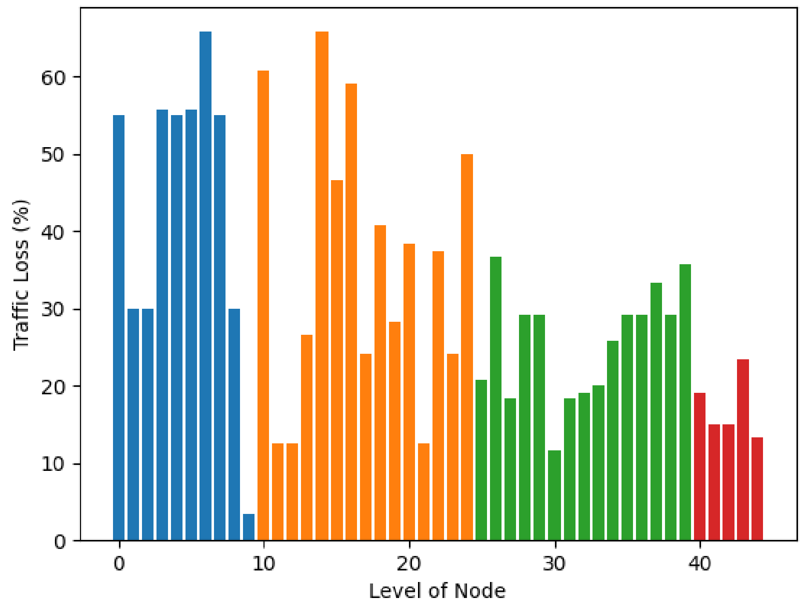


Figure 14. Comparison of Traffic Loss of the Network when the Attacker is located at Nodes of Different Node Levels

Chapter 4
DODAG Implementation in RPL with Dual-Parents Topology to mitigate Sinkhole Attack and Load balancing Approach with TASA Algorithm

4.1: Routing Protocol for Low Power and Lossy Network (RPL) with Dual-Parent Topology

Implementing the RPL network with many random nodes is hardly assigned to get dual parents for each sensor node. And it is hard to implement Traffic Aware Scheduling Algorithm (TASA) in other RPL network simulators. So, we use python language to implement the simulation of an RPL network that has one root node (router) and other sensor nodes which have dual parents at every level (Rank).

We created an experimental simulation with Python Programming to simulate the RPL network for testing. The router (root node) is assigned first at the topmost level of the network. To form the suitable network topology for our testing, the root node has the lowest rank (Rank 0) and calculates the rank value of its neighbour nodes by the help of Objective Function Zero (OF0). Moreover, we use Minimum Rank with Hysteresis Objective Function (MRHOF) which is based on the Expected Transmission Count (ETX) to create the DODAG and that MRHOF helps us to find the preferred parent that is chosen with the minimum rank value among the candidate neighbour nodes [31].

The rood node broadcasts its very first DIO message to its neighbour nodes which are in the cover range of its wireless transmission. That DIO message carries all the information to create a DODAG in the RPL network. The first neighbour nodes receive the DIO message, and they calculate their rank value based on the objective function. After finished calculating their rank value, they broadcast their DIO message to the neighbours, the next-level nodes. The next-level neighbour nodes can receive more than one DIO message from the above-level nodes. After receiving a DIO message from the above node, a node calculates its own rank value and creates a routing table for connection. To create a dual-parent network topology, we assume that every single node has at least two parents. So, a child node in our simulated network will get another DIO message from another above-level node. A child node chooses its preferred parent among the above-level nodes which have the same rank value based on the minimum cost path. And the remaining node is put in the routing table as a second parent to make a connection if the preferred parent node is down or disconnected from the root node. The process will take at every single node of each level until the end of DODAG where all the child nodes in that level have no child node which means they are not the preferred parent node of other child nodes of this network.

4.2: Parent Selection in RPL

In Figure 14, the root node broadcasts the rank value in a DIO message to the neighbour nodes. The neighbour nodes, P_1 to P_n, receive the DIO message and calculate their and retransmit the DIO message with their own rank. The rank of the node is controlled by hop rank increase value (Δ). When the child node, receives the rank value from its upper-level nodes, it calculates its own rank on it. According to Figure 14, after child node X_k has received the DIO message from parent node P_i, and then it calculates its own rank based on Rank (P_i) + $\Delta_{i,k}$.

$$\Delta_{i,k} = ETX_{(k,i)} = \frac{1}{D_f \times D_r} \qquad (1)$$

where ETX matrix is the number of transmissions a node expects to make to a destination in order to successfully deliver a packet which may not be integer [32] and D_f is the measured probability that a packet is received by the neighbor, and D_r is the measured probability that the acknowledgment packet is successfully received.

X_k gets its rank value, Rank (X_k| P_i), that is the path cost between parent P_i and child node X_k , and X_k chooses P_i as its preferred parent and puts the name in the routing table. Later it receives another DIO message from P_m, and then it calculates it new rank value, new path cost, again based on the Rank (X_k| P_m). When a child node receives the path cost more than one, it compares them according to the objective function.

$$\mathrm{Rank}_{old}(X_k) = \mathrm{OldPathCost} = \mathrm{Rank}(X_k|P_i) = \mathrm{Rank}(P_i) + \Delta_{i,k} \qquad (2)$$

$$\mathrm{Rank}_{new}(X_k) = \mathrm{NewPathCost} = \mathrm{Rank}(X_k|P_m) = \mathrm{Rank}(P_m) + \Delta_{m,k} \qquad (3)$$

$$\text{New Path Cost} < \text{Old Path Cost} - \partial, \qquad (4)$$

where ∂ is the parent switch threshold in DODAG. If the old cast cost is higher than new path cost, the child node X_k changes its preferred parent list, and it makes the parent list and puts the name second parent name in it.

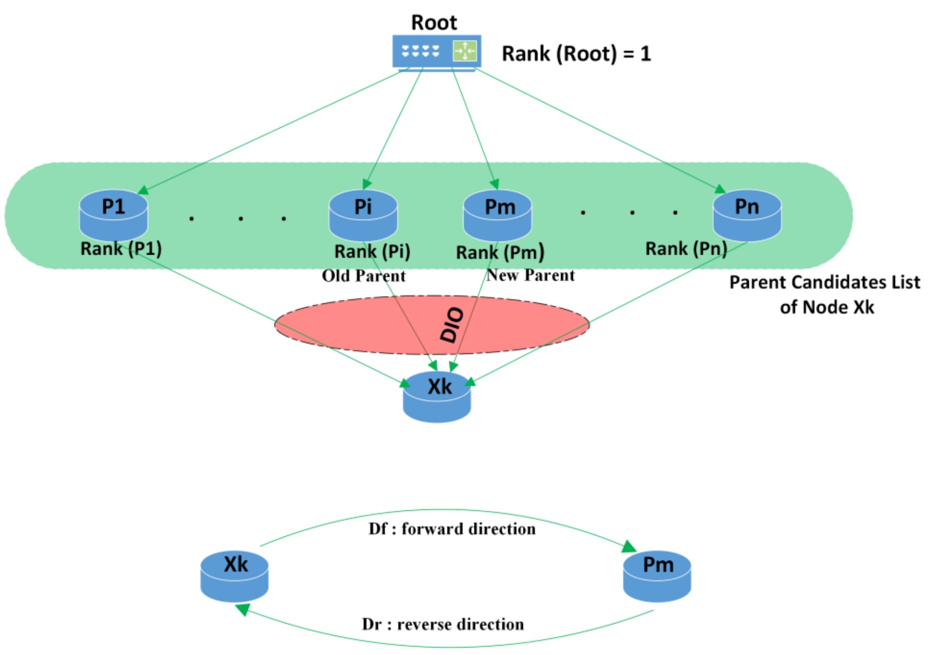


Figure 15 Parent Selection Process in RPL Network

4.3: RPL with Traffic Aware Scheduling Algorithm (TASA)

TASA assumes that the root node of the network is aware of the network topology, single-slot frame traffic load, one-hop neighbor node, and parent node. Two types of queue lengths are defined to represent local and global traffic load information. Each node's local queue specifies the number of packets in its own internal queue, while each node's global queue contains all packets from its own stream node and its own packets. TASA is designed for wireless sensor networks where sensor nodes are configured as directed acyclic graphs, DODAGs of the RPL network. The destination oriented directed acyclic graph (DODAG) is represented as G. G = (V, E), where V = {n_0,n_1,n_2,…,n_{N-1}} is the set of nodes, and |V| = N is the total number of nodes in the network.

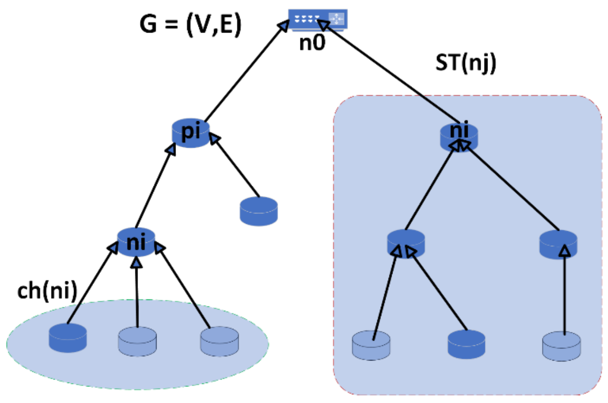


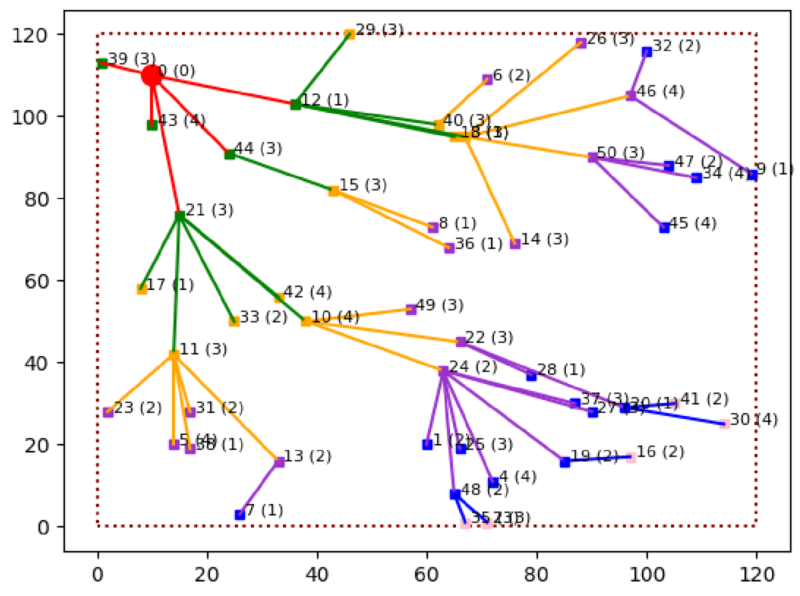
Figure 16 Graph G = (V, E) Modeling A Network with A Tree Topology

It works in two steps: Matching and Coloring. The comparison identifies a set of duplex node pairs (links) with no sender and receiver conflicts. Coloring, on the other hand, assigns the correct frequency channel to all links and ensures that there is no interference between them.

$$Q_i(k) = \max \{Q_j(k) \mid n_j \in ch(p_i) \wedge q_j(k) \neq 0\} \qquad (5)$$

4.4 Implementation of RPL network with TASA without Sinkhole Attack

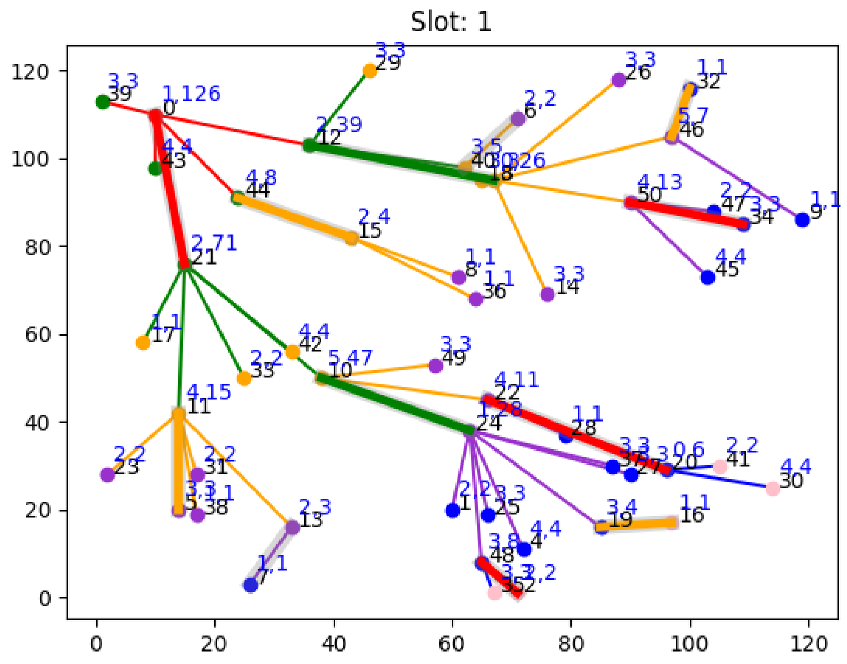
To create an experimental network topology, we build up the same topology that has been simulated as in Chapter 3. All the nodes are placed randomly and set up the various levels position according to their rank values. Our topology has different traffic loads at each node, and all these traffic loads are forwarded to the root node. By applying the TASA algorithm in the RPL network, we forward all data packs in a minimum timeslot. Our topology has one root node, other 50 sensor nodes, 5 levels and total of 126 traffic loads. All these traffic loads are delivered to root node with a minimum time slot by TASA. Figure 16 is the first step of RPL network formation with TASA algorithm.



```
q:      [0, 2, 3, 1, 4, 4, 2, 1, 1, 1, 4, 3, 1, 2, 3, 3, 2, 1, 3, 2, 1, 3, 3, 2
, 2, 3, 3, 3, 1, 3, 4, 2, 2, 2, 4, 3, 1, 3, 1, 3, 3, 2, 4, 4, 3, 4, 4, 2, 2, 3,
3]
 Q0:     126
Number of levels =  5
-------------------------------------------------------
```

Figure 17. RPL Network with TASA Algorithm

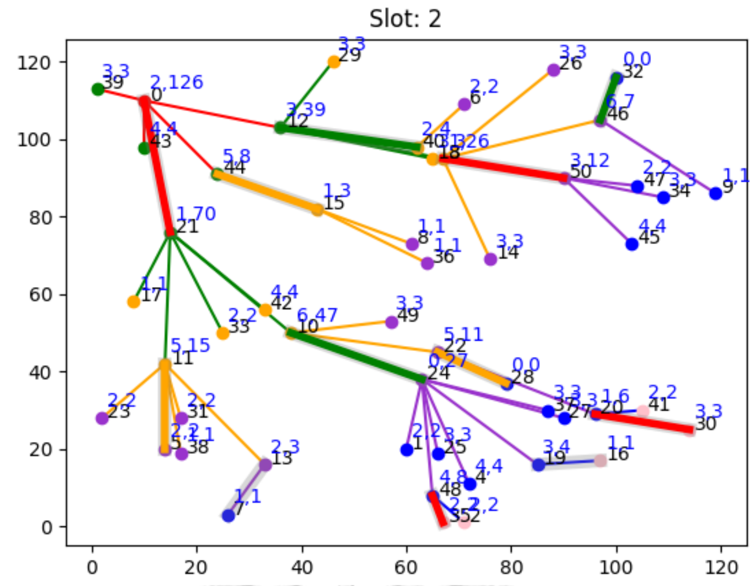
Figure 17 is the first time slot of our topology which has no sinkhole attack. In Figure 17, we make the pair for our matching and do for colouring by TASA algorithm. The vertexes pairs are [(21,0) (20,22) (34,50) (2,48) | (24,20) (3,12) | (15,44) (5,11) (32,46) (16,19)] and they are coloured in assorted colours.



```
 ---------------------------------------------------------------
 21,0 20,22 34,50 2,48   |24,10 3,12    |15,44 5,11 32,46 16,19        |1
 ---------------------------------------------------------------
```

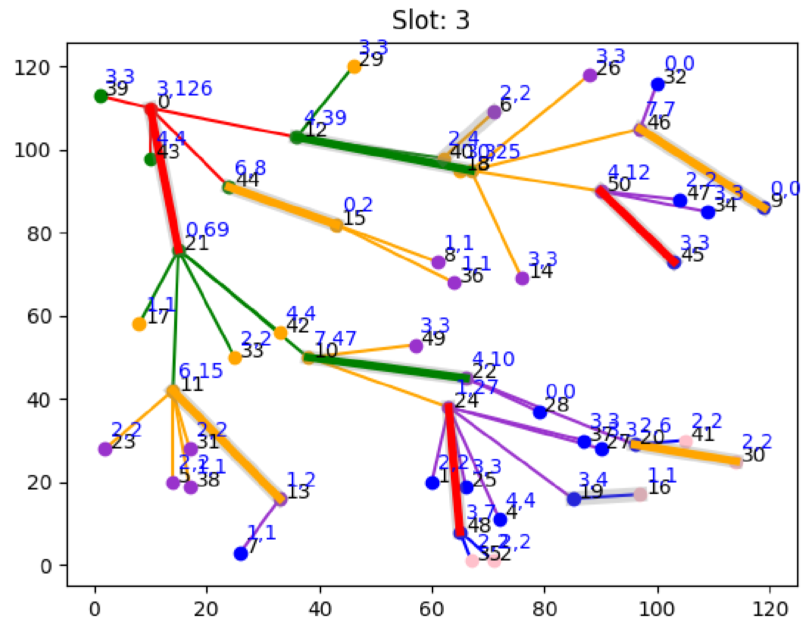
Figure 18. The Connection of RPL Network at the First Time Slot with TASA Algorithm

Figures 18,19, and 20 are next 3 time slots of our topology and they show the pairs for matching and colored the pairs step by step.



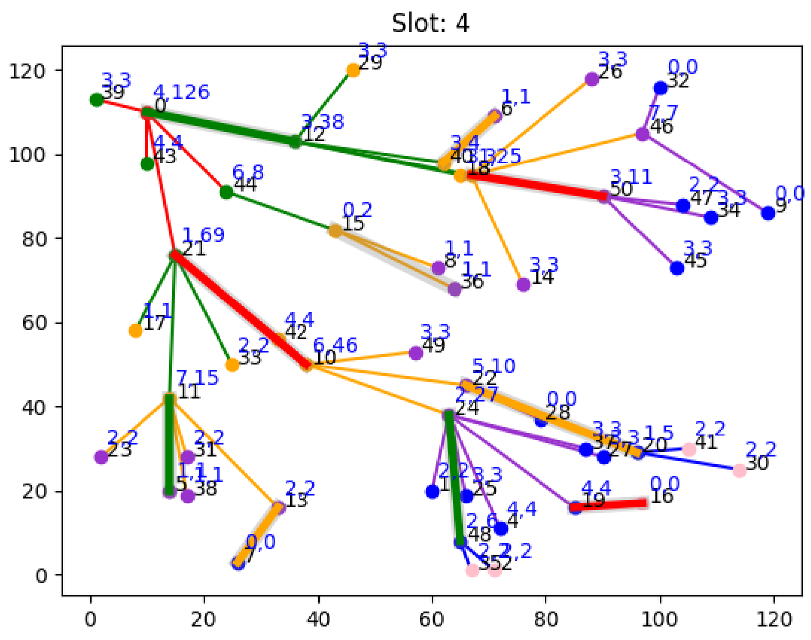
```
-------------------------------------------------------------------
21,0 50,3 30,20 35,48    |24,10 40,12 32,46       |15,44 5,11 28,22        |2
-------------------------------------------------------------------
```

Figure 19 The Connection of RPL Network at the Second Time Slot with TASA Algorithm



```
-------------------------------------------------------------------
21,0 48,24 45,50         |3,12 22,10        |13,11 15,44 30,20 9,46         |3
-------------------------------------------------------------------
```

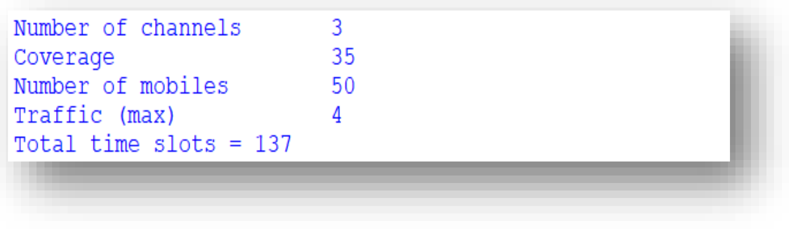
Figure 20 The Connection of RPL Network at the Third Time Slot with TASA Algorithm



```
----------------------------------------------------------------------
10,21 50,3 16,19          |12,0 48,24 5,11          |20,22 6,40 7,13          |4
----------------------------------------------------------------------
```

Figure 21 The Connection of RPL Network at the Fourth Time Slot with TASA Algorithm

After the last data packet from the node 21 to root node is delivered, the total time slot of transmitting all traffic load is 137. Figure 21 shows the result of our simulation.

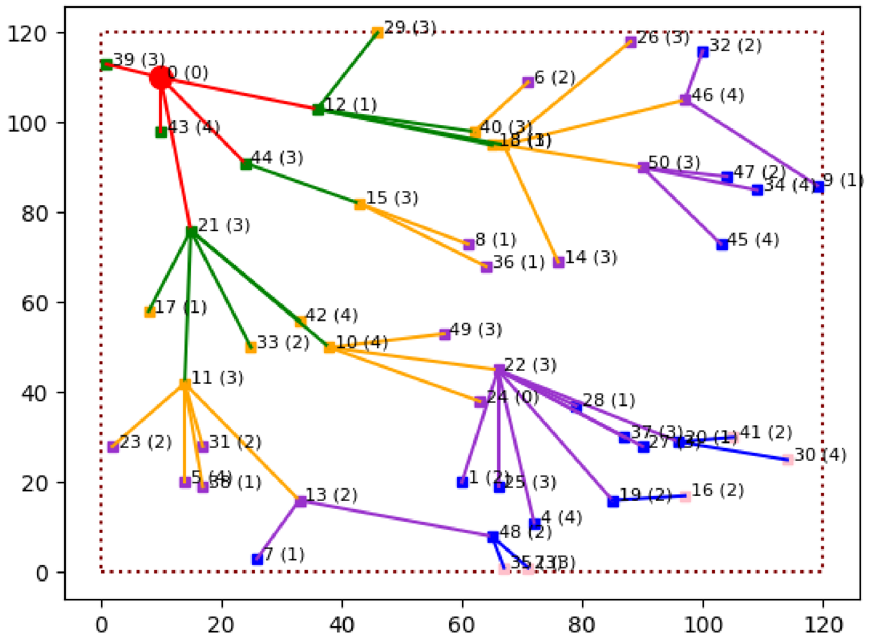


```
Number of channels        3
Coverage                  35
Number of mobiles         50
Traffic (max)             4
Total time slots = 137
```

Figure 22 Final Simulated Result with TASA algorithm

4.5 Implementation of RPL with TASA under Sinkhole Attack

This session presents the simulation results of RPL network with TASA algorithm under the running of a sinkhole attack. Our topology is same as the previous session in Figure 16. Numbers of root node, other sensor nodes and channel are same as previous one, but the total traffic load are different. We make a sinkhole attack formation at node ID 24. Therefore, the traffic load of node 24 is 0, the total traffic is changed from 126 to 124. Because node 24 is a malicious node, the child nodes of node 24 are changed their preferred parent. Node 1,4,19,25,27 and 37 connect to node 22 and node 48 connects to node 13 instead of their preferred parent node. The rerouting or rescheduling of our RPL network is seen in Figure 22.



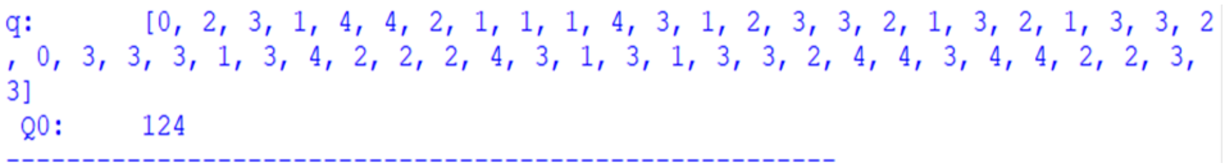
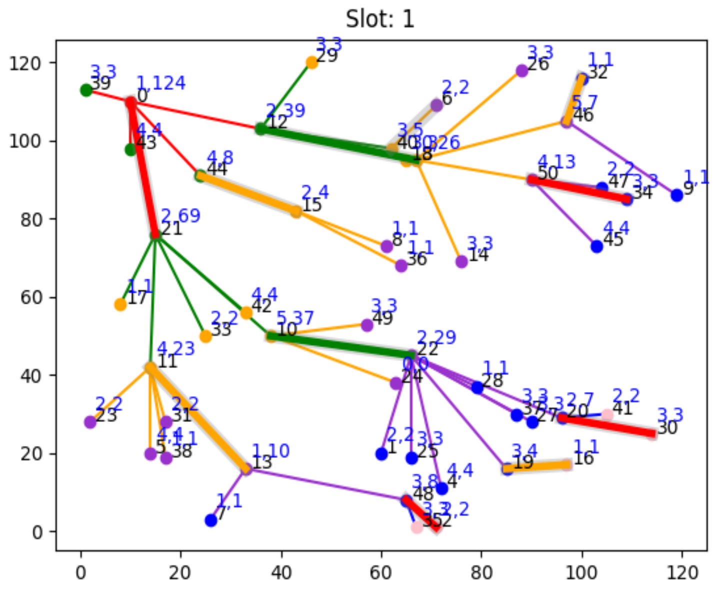
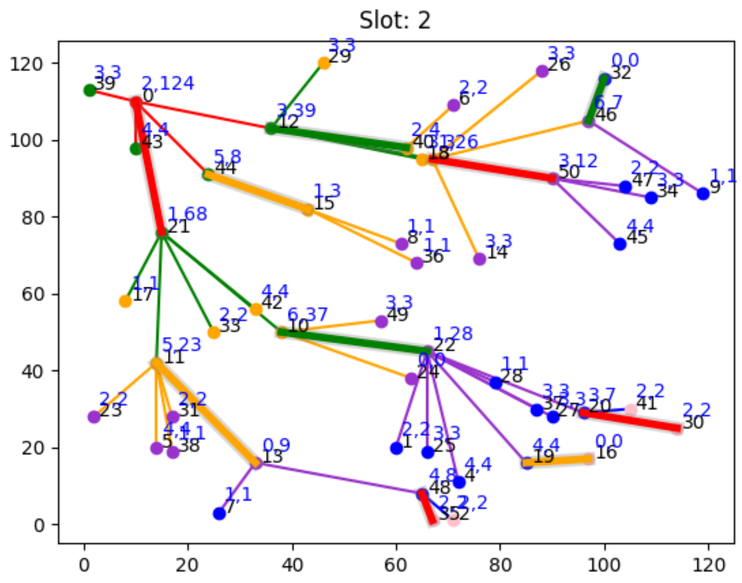

```
q:       [0, 2, 3, 1, 4, 4, 2, 1, 1, 1, 4, 3, 1, 2, 3, 3, 2, 1, 3, 2, 1, 3, 3, 2
, 0, 3, 3, 3, 1, 3, 4, 2, 2, 2, 4, 3, 1, 3, 1, 3, 3, 2, 4, 4, 3, 4, 4, 2, 2, 3,
3]
 Q0:     124
-----------------------------------------------------
```

Figure 23 RPL Network with TASA Algorithm Under Sinkhole Attack



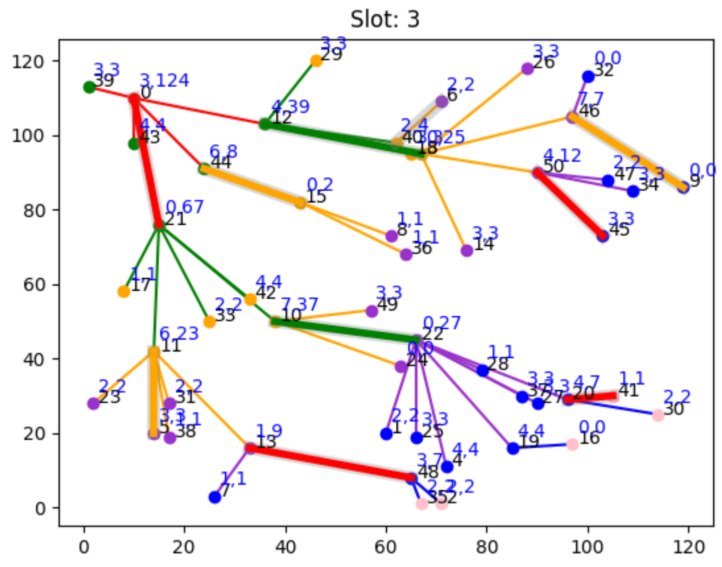
```
------------------------------------------------------------
21,0 34,50 30,20 2,48    |22,10 3,12     |13,11 15,44 32,46 16,19        |1
------------------------------------------------------------
```

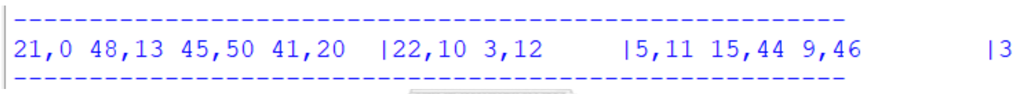
Figure 24 The Connection of RPL Network at the First Time Slot with TASA Algorithm Under Sinkhole Attack



```
------------------------------------------------------------
21,0 50,3 35,48 30,20   |22,10 40,12 32,46      |13,11 15,44 16,19      |2
------------------------------------------------------------
```

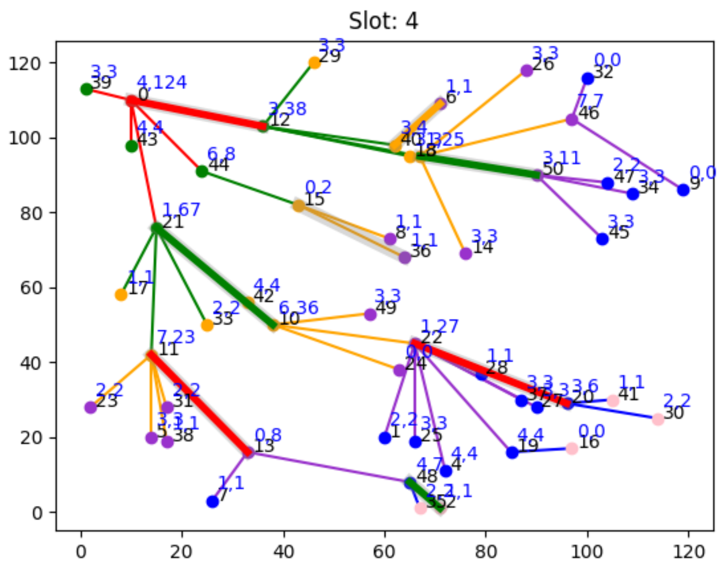
Figure 25 The Connection of RPL Network at the Second Time Slot with TASA Algorithm Under Sinkhole Attack





```
------------------------------------------------------------
21,0 48,13 45,50 41,20  |22,10 3,12     |5,11 15,44 9,46        |3
------------------------------------------------------------
```

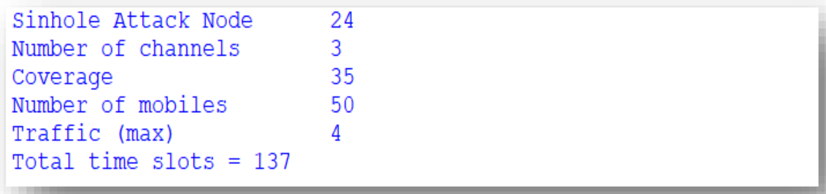
Figure 26 The Connection of RPL Network at the Third Time Slot with TASA Algorithm Under Sinkhole Attack



```
------------------------------------------------------------
12,0 13,11 20,22          |10,21 50,3 2,48          |6,40   |4
------------------------------------------------------------
```

Figure 27 The Connection of RPL Network at the Fourth Time Slot with TASA Algorithm Under Sinkhole Attack

In Figure 25, we can the total time slot number of our topology that is running under the sinkhole attack. The number of total time slots is the same as the number of time slots that was given in Figure 21. Therefore, dual-parent formation in RPL network can avoid the sinkhole attack effectively and we can transmit the traffic load of each node within in minimum time slot.



```
Sinhole Attack Node      24
Number of channels       3
Coverage                 35
Number of mobiles        50
Traffic (max)            4
Total time slots = 137
```

Figure 28 Final Simulated result with TASA algorithm Under Sinkhole Attack

4.6 Results and Discussion

To check the process of our simulation how it is work well or not, we run the simulation with attack and without attack at every single node at level 2,3,4 and 5. We collect the data of every simulation process and tabulate the data. Table 5 shows the results of minimum time slots of the network with sinkhole attack and without sinkhole attack.

Table 5 Time Slot Comparison in Different Scenarios

Level	Sinkhole Node ID	Number of Time Slots (Without Attack)	Number of Time Slots (Under Attack)
2	3	141	141
2	10	133	133
2	11	135	135
2	15	141	141
2	17	139	139
2	18	141	141
2	29	141	141
2	33	137	137
2	40	141	141
2	42	133	133
3	5	133	133
3	6	141	141
3	8	141	141
3	13	137	137
3	14	141	141
3	22	135	135
3	23	137	137
3	24	137	137

3	26	141	141
3	31	137	137
3	36	141	141
3	38	139	139
3	46	141	141
3	49	135	135
3	50	141	141
4	1	137	137
4	4	133	133
4	7	139	139
4	9	141	141
4	19	137	137
4	20	139	139
4	25	135	135
4	27	135	135
4	28	139	139
4	32	141	141
4	34	141	141
4	37	135	135
4	45	141	141
4	47	141	141
4	48	137	137
5	2	135	135
5	16	137	137
5	30	133	133
5	35	135	135
5	41	137	137

In Figure 26, we can check the comparison of number of time slots of each level where attack is having. There is no change in each level before and after the sinkhole attack. Therefore, our comparison results are same at every level.

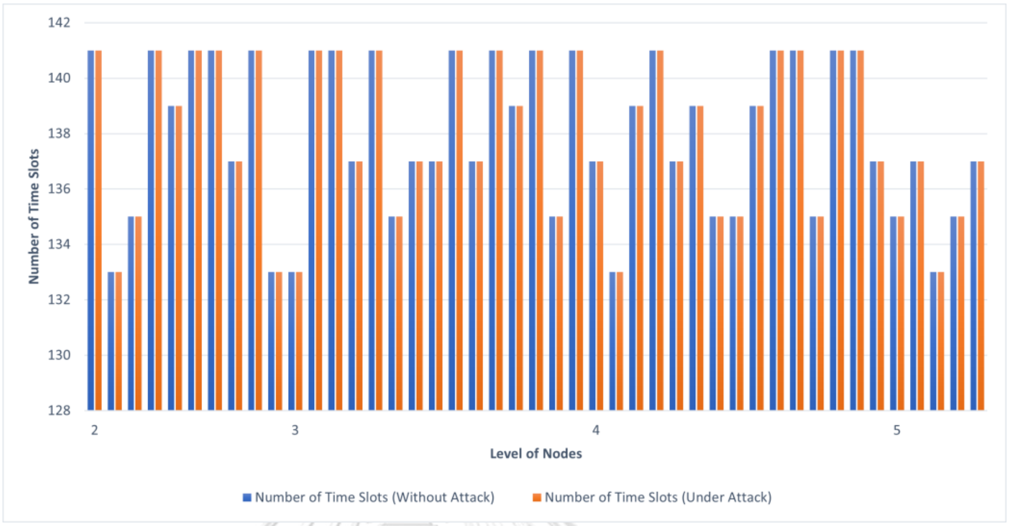


Figure 29. Comparison of Time Slots between with Attack and without Attack

After we have simulated and compared the results of number of time slots in both situations (with and without attack) in Table 5, we make other simulations which have no dual parents under sinkhole attacking node at different levels. Table 6 discusses the comparison of number of time slots and packet loss under without and with sinkhole attack. Because there are no dual parent nodes under sinkhole attack, the child nodes of sinkhole attack node have no chance to switch its preferred parents. There is no parent to deliver the packets, the number of packet loss is high, and the number of time slots is lower than the number of time slots of without sinkhole attack condition.

In Table. 6, we can see one of the sinkhole attack nodes in level 2 is node ID 10 and the total number of time slots is 133 under dual parent mechanisms. If there is no dual parent situation under sinkhole attack node 3, at first the total traffic of the network is changed from 126 to 122 because node 3 has 4 local traffic. After we run the simulation with sinkhole attack under no dual parent mechanisms, we deliver total

79 packets out 122 and we loss 43 packets. Because of increasing packets loss, the number of time slots is decrease and it changes from 133 to 79. But in some attack nodes which have no child nodes, the packet loss and number of time slots are not change. We can see the comparison results of time slots of with and without dual parent mechanisms under sinkhole attack in Figure 30 and the total packets to delivered packets in different levels is seen in Figure 31.

Table 6 Comparison of Number of Time Slots and Packets Loss Under Sinkhole Attack with and without Dual Parent Mechanisms

Level	Sinkhole Node ID	Number of Time Slots (Dual parent)	Number of Delivered / Total Packets (Dual Parent)	Packet Loss (%) (Dual Parent)	Number of Time Slots (No Dual Parent)	Number of Delivered / Total Packets (No Dual Parent)	Packet Loss (%) (No Dual Parent)
2	3	141	125/125	0	141	99/125	20.8
2	10	133	122/122	0	79	79/122	35.25
2	11	135	123/123	0	111	111/123	9.76
2	15	141	123/123	0	141	121/123	1.63
2	17	139	125/125	0	139	125/125	0
2	18	141	123/123	0	141	123/123	0
2	29	141	123/123	0	141	123/123	0
2	33	137	124/124	0	137	124/124	0
2	40	141	123/123	0	141	121/123	1.63
2	42	133	122/122	0	133	122/122	0
3	5	133	122/122	0	133	122/122	0
3	6	141	124/124	0	141	124/124	0

3	8	141	125/125	0	141	125/125	0
3	13	137	124/124	0	135	123/124	0.8
3	14	141	123/123	0	141	123/123	0
3	22	135	123/123	0	119	115/123	6.5
3	23	137	124/124	0	137	124/124	0
3	24	137	124/124	0	97	97/124	21.77
3	26	141	123/123	0	141	123/123	0
3	31	137	124/124	0	137	124/124	0
3	36	141	125/125	0	141	125/125	0
3	38	139	125/125	0	139	125/125	0
3	46	141	122/122	0	141	119/122	2.46
3	49	135	123/123	0	135	123/123	0
3	50	141	123/123	0	141	113/123	8.13
4	1	137	124/124	0	137	124/124	0
4	4	133	122/122	0	133	122/122	0
4	7	139	125/125	0	139	125/125	0
4	9	141	125/125	0	141	125/125	0
4	19	137	124/124	0	133	122/124	1.61
4	20	139	125/125	0	127	119/125	4.8
4	25	135	123/123	0	135	123/123	0
4	27	135	123/123	0	135	123/123	0
4	28	139	125/125	0	139	125/125	0
4	32	141	124/124	0	141	124/124	0
4	34	141	122/122	0	141	122/122	0
4	37	135	123/123	0	135	123/123	0
4	45	141	122/122	0	141	122/122	0
4	47	141	124/124	0	141	124/124	0
4	48	137	124/124	0	125	118/124	4.84
5	2	135	123/123	0	135	123/123	0
5	16	137	124/124	0	137	124/124	0

5	30	133	122/122	0	133	122/122	0
5	35	135	123/123	0	135	123/123	0
5	41	137	124/124	0	137	124/124	0

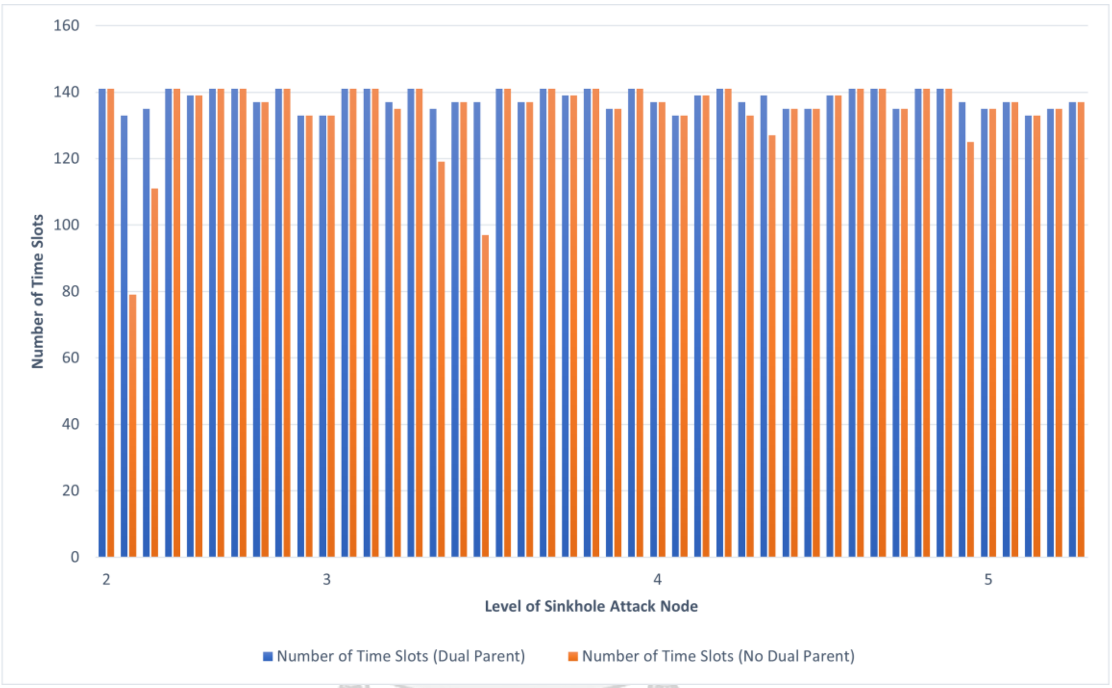


Figure 30 Comparison of Time Slots between with and without Dual Parent under Attack

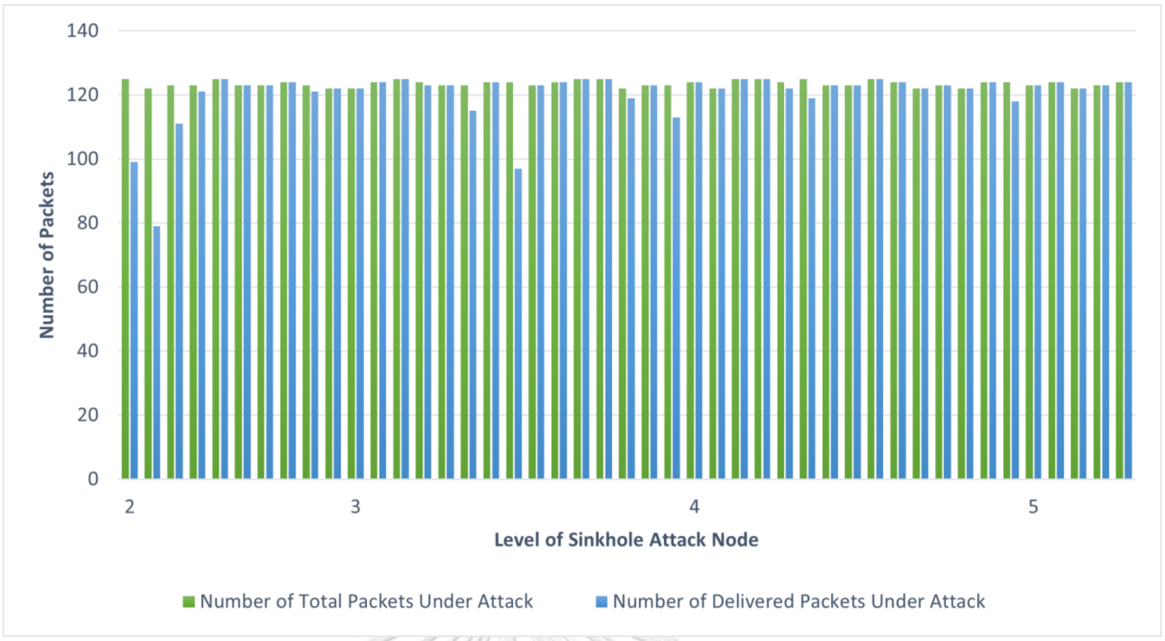


Figure 31 Comparison of Total Packets to Delivered Packets Without Dual Parent Mechanisms Under Sinkhole Attack

To check the correctness of our simulation, we run the simulator twice with the dual parent mechanism and no dual parents' formation in child node under the attack. The sinkhole attack node is node ID 10 and topology is same as the Figure 10 in Chapter 3. By applying the TASA algorithm in our simulation, we clearly know the pairs of nodes and how many channels are used to deliver the packets per slot. The comparison results are shown in Table 7 and 8. After plotting the channels and time slot of each topology, number of the time slots results are same as time slots numbers of sinkhole attack ID 10 in Table 6.

Table 7. Details of Time Slots Formation of the Network with Dual Parent Mechanism under the Attack

Channel 1	Channel 2	Channel 3	Slot
21,0	24,10	15,44	1
20,22	3,12	5,11	
34,50		32,46	
2,48		16,19	

Table 8. Details of Time Slots Formation of the Network with No Dual Parent Mechanism under the Attack

Channel 1	Channel 2	Channel 3	Slot
12,0	11,21	15,44	1
48,24	50,3	16,19	
32,46	20,22	7,13	

21,0 50,3 30,20 35,48	24,10 40,12 32,46	15,44 5,11 28,22	2
21,0 48,24 45,50	3,12 22,10	13,11 1 5,44 30,20 9,46	3
10,21 50,3 16,19	12,0 48,24 5,11	20,22 6,40 7,13	4
21,0 20,22 34,50 2,48	24,10 3,12	13,11 6,40	5
10,21 50,3 41,20	12,0 48,24 23,11	36,15	6
21,0 20,22 45,50 35,48	24,10 3,12	31,11 15,44	7
10,21 50,3 30,20	12,0 48,24 5,11	8,15	8
21,0 20,22 34,50 2,48	24,10 3,12	38,11 15,44	9
10,21 50,3 35,48 41,20	12,0 4,24 13,11		10
21,0 20,22 45,50	24,10 3,12	23,11	11
10,21 50,3 30,20	12,0 48,24 31,11		12
21,0 47,50	24,10 3,12		13

3,12 **48,24** **5,11**	21,0 34,50 30,20	15,44 28,22 32,46	2
12,0 **20,22** **2,48** **9,46**	11,21 50,3 4,24	15,44 16,19	3
3,12 **48,24** **5,11**	21,0 45,50 30,20	6,40	4
12,0 **20,22** **35,48**	11,21 50,3 19,24	6,40	5
3,12 **48,24**	21,0 34,50 41,20	13,11 36,15	6
12,0 **20,22** **2,48**	11,21 50,3 4,24	15,44	7
3,12 **48,24** **5,11**	11,21 50,3 4,24	15,44	8
3,12 **48,24** **5,11**	21,0 45,50 30,20	8,15	9
12,0 **37,24**	11,21 50,3 20,22 35,48	15,44	10
3,12 **48,24**	21,0 34,50 41,20	13,11	11
12,0 **19,24**	11,21 50,3 20,22 2,48		12
3,12 **25,24** **23,11**	21,0 45,50 35,48 30,20		13
12,0	11,21		14

20,22			
10,21 46,3	12,0 19,24 34,50		14
21,0 45,50	24,10 3,12		15
10,21 50,3	12,0 4,24		16
21,0 47,50	24,10 3,12		17
10,21 46,3	12,0 37,24		18
21,0	24,10 3,12		19
10,21 50,3	12,0 48,24		20
21,0	24,10 3,12		21
10,21 46,3	12,0 19,24		22
21,0	24,10 3,12		23
10,21 50,3	12,0 25,24		24
21,0	24,10 3,12		25
10,21 46,3	12,0 27,24		26
21,0	24,10 3,12		27
10,21 50,3	12,0 1,24		28
21,0	24,10 3,12		29
10,21 14,3	12,0 4,24		30
21,0	24,10 3,12		31
10,21 46,3	12,0 37,24		32

27,24 **47,50**	46,3 20,22		
3,12 **1,24** **31,11**	21,0 34,50		15
12,0 **4,24**	11,21 50,3		16
3,12 **37,24** **5,11**	21,0 45,50		17
12,0 **48,24** **47,50**	11,21 46,3		18
3,12 **19,24** **38,11**	21,0		19
12,0 **25,24**	50,3 11,21		20
3,12 **27,24** **13,11**	3,12 27,24 13,11		21
12,0 **1,24**	46,3 11,21		22
3,12 **4,24** **23,11**	3,12 4,24 23,11		
12,0 **37,24**	50,3 11,21		23
3,12 **48,24** **31,11**	21,0		24
12,0 **19,24**	12,0 19,24		25
3,12 **25,24**	21,0		26
12,0 **27,24**	50,3 11,21		27
3,12	21,0		28
12,0	14,3 42,21		29

21,0	24,10 3,12		33
10,21 50,3	12,0 48,24		34
21,0	24,10 3,12		35
10,21 26,3	12,0 19,24		36
21,0	24,10 3,12		37
10,21 14,3	12,0 25,24		38
21,0 27,24	22,10 3,12		39
10,21 46,3	12,0 1,24		40
21,0	24,10 3,12		41
10,21 50,3	12,0 4,24		42
21,0 37,24	22,10 3,12		43
10,21 26,3	12,0 48,24		44
21,0	24,10 3,12		45
10,21 14,3	12,0 19,24		46
21,0 25,24	22,10 3,12		47
10,21 46,3	12,0 27,24		48
21,0 50,3	24,10 40,12		49
10,21 26,3	12,0		50
21,0	22,10 3,12		51
10,21	12,0		52
21,0	24,10		53

3,12	21,0		30
12,0	46,3 33,21		31
3,12	21,0		32
12,0	50,3 11,21		33
3,12	21,0		34
12,0	26,3 42,21		35
3,12	44,0	33,21	36
12,0	14,3 11,21		37
3,12	21,0		38
12,0	46,3 42,21		39
3,12	44,0	17,21	40
12,0	50,3		41
21,0	3,12		42
12,0	26,3		43
44,0	3,12		44
12,0	14,3		45
21,0	3,12		46
12,0	46,3		47
44,0 50,3	40,12		48
12,0	26,3		49
21,0	3,12		50
12,0			51
43,0	40,12		52
12,0			53
44,0	3,12		54
12,0			55
21,0	40,12		56
12,0			57
39,0	18,12		58
12,0			59
43,0	29,12		60
12,0			61
44,0	3,12		62
12,0			63

	40,12		
10,21	12,0		54
21,0	22,10 18,12		55
10,21	12,0		56
21,0	24,10 29,12		57
10,21	12,0		58
21,0	22,10 3,12		59
10,21	12,0		60
21,0	24,10 3,12		61
11,21	12,0 22,10		62
21,0	24,10 18,12		63
10,21	12,0		64
21,0	49,10 29,12		65
11,21	12,0 22,10		66
21,0	24,10 3,12		67
10,21	44,0	40,12	68
21,0	49,10 18,12		69
11,21	12,0 22,10		70
21,0	24,10 29,12		71
10,21	44,0		72
21,0	49,10		73
11,21	12,0 22,10		74
21,0	24,10		75
10,21	44,0		76

21,0	40,12		64
12,0			65
39,0	18,12		66
12,0			67
43,0	29,12		68
12,0			69
44,0	3,12		70
12,0			71
21,0	40,12		72
12,0			73
39,0	18,12		74
12,0			75
43,0	29,12		76
12,0			77
44,0			78
21,0			79

21,0			77
11,21	12,0		78
21,0			79
10,21	44,0		80
21,0			81
11,21	43,0		82
21,0			83
10,21	12,0		84
21,0			85
11,21	44,0		86
21,0			87
10,21	39,0		88
21,0			89
11,21	43,0		90
21,0			91
10,21	12,0		92
21,0			93
11,21	44,0		94
21,0			95
10,21	39,0		96
21,0			97
11,21	43,0		98
21,0			99
10,21	12,0		100
21,0			101
11,21	44,0		102
21,0			103
10,21	39,0		104
21,0			105
11,21	43,0		106

21,0			107
42,21	12,0		108
21,0			109
10,21	44,0		110
21,0			111
11,21			112
21,0			113
42,21			114
21,0			115
33,21			116
21,0			117
10,21			118
21,0			119
11,21			120
21,0			121
42,21			122
21,0			123
33,21			124
21,0			125
10,21			126
21,0			127
11,21			128
21,0			129
42,21			130
21,0			131
17,21			132
21,0			133

Chapter 5
Conclusion

5.1 Conclusion

This thesis studied the damage of a Sinkhole Attack in the RPL network and proposed the simple and very effective way of defense mechanism to mitigate that sinkhole attack. According to our simulated results, we can conclude the damage of Sinkhole Attack in the RPL network is very big and it makes a high impact to the main network. Position of the attack node is very important in the network, and even thought the attack node is far away from the root node, but it close to the many sensor nodes, it can make a huge network isolation and loss the high percentage of traffic loss. We can aware the impact of the Sinkhole attacking behaviour very dangerous especially for low-power and lossy networks. Our proposed method, making a dual-parent formation for each child node in the network when the topology is set up, is the effective way to defense the Sinkhole Attack. It can mitigate the attack when the attack is happening in the network by switching their preferred parent in their parent list. The DODAG formation in the RPL network is changed at every period, but we will update and change only the sub-DODAG of the network where attack is happening when we do not receive the Acknowledge message from that attack parent node. Our proposed system does not need to set up and install another extra protection mechanisms to defense the Sinkhole Attack.

This thesis also implements the traffic load balancing of the network by applying Traffic Aware Scheduling Algorithm (TASA). To save the time consuming of low-power sensor nodes, we need to forward the total traffic load of the whole network to the root node within a minimum time slots. Applying the TASA in the RPL network topology is a good way to concern the total traffic load of our data acquisition network, avoid the collision between child nodes and parent node transmission and reduce the time and delay of the network. And then we also implement and compare the results of running the RPL network without Sinkhole Attack and with Attack applying the TASA. We can conclude that our dual-parent Sinkhole Attack defense mechanism is worked well by checking the comparisons of time slot because there is no change the number for every single node and level. We can avoid the Sinkhole Attack formation in the network effectively by our proposed dual parent mechanisms.

5.2 Future Work

If there is no dual-parent node in the network, how to avoid the Sinkhole Attack is the main problem of our research work. If we have time to implement to solve that issue, we are considering the concept that the powerful child node expands its cover range of transmission and can help for rescheduling as a preferred parent.

Appendix 1

TASA Algorithm:

Procedure SCHEDULING (G, P, **q**, **Q**, n_{ch}) > G : Graph, P : PHY Connectivity Graph, **q**: q_i (k) vector, **Q**: Q_i (k) vector, n_{ch}: number of available channels

```
k ⟵ 0                                  > slot initial value
pattern ⟵ [ ]                          > time/frequency pattern initial value
while $q_0$ (k) ≠ Q do                 > The procedure is run until a pattern that allows to delivery to the PAN Coordinator all the packets generated in one time slot frame, is built

DCFL (k) ⟵ MATCHING (G, q, Q, $n_0$, k)
I (k) ⟵ FINDINTERFGRAPG ((DCFL (k), P) > Building the Interference Conflict graph for DCFL(k)
colored ⟵ COLORING (I (k), Q)
selected ⟵ [ ]
for n ⟵ 1 to $n_{ch}$ do
    selected ⟵ GETFIRST (colored)
end for
UPDATE (q,Q, selected)                 > Updating q,Q according to links scheduled at k
pattern ⟵ pattern + [(k, selected)]
k ⟵ k + 1
    end while
    return pattern                     > returning time/ frequency pattern
end procedure
```

Appendix 2

*************************** This is importing the modules to run the simulation ***************************

```
import matplotlib.pyplot as plt
import math
import random

********************************* This is parameters for Network Setting ******************************

RoomWidth = 120
RoomHeight = 120
CoverageRadius = 35
NumMobile = 50
channels = 3
X = 4

method = 'Heuristic TASA algorithm'
seedval = 55555555555151010198924011996
random.seed(seedval)

mycolor = ['red','green','orange','darkorchid','blue','pink','magenta','skyblue']
channel_color = ['blue','orange','pink','magenta','skyblue']


****************************** This is generated traffic q for each node ******************************

seedval_q = 4
random.seed(seedval_q)
q = [0]
for k in range(1,NumMobile+1):
    traff = random.randint(1,X)
    q.append(traff)
Q0 = sum(q)
print("q:\t",q)
print(" Q0:\t", Q0)



*************** This is only the q of sink_hole_node is set to zero. Set the Sinkhole node ************

sink_hole_node = 49
seedval_q2 = seedval_q
random.seed(seedval_q2)
```

```
q = [0]

**************************** This is the effect caused by the sink hole node *************************
# Not only the q of sink_hole_node is set to zero, but also other neighoring nodes are affected.

for k in range(1,NumMobile+1):
    traff = random.randint(1,X)
    q.append(traff)
q[sink_hole_node] = 0
#q[sink_hole_node1] = 0
Q0 = sum(q)
print("q:\t",q)
print(" Q0:\t", Q0)


**************************** This is Random location (x,y) for each node  *************************

All_Nodes_loc = []
All_Nodes_loc.append((10,110))
for k in range(1,NumMobile+1):
    xloc = random.randint(0,RoomWidth)
    yloc = random.randint(0,RoomHeight)
    All_Nodes_loc.append((xloc,yloc))

*********************** This is manual set up of locations to make Dual parent ************************

All_Nodes_loc[1] = (60,20)
All_Nodes_loc[41] = (105,30)
All_Nodes_loc[18] = (65,95)
All_Nodes_loc[33] = (25,50)
All_Nodes_loc[24] = (63,38)
All_Nodes_loc[10] = (38,50)
All_Nodes_loc[50] = (90,90)
**************************** This is all nodes are shown with traffic ******************************

fig, ax = plt.subplots()
plt.plot([0,RoomWidth,RoomWidth,0,0],[0,0,RoomHeight,RoomHeight,0],
        color='maroon',linestyle=':',linewidth=1.5)
```

```
for k in range(NumMobile+1):
    xloc = All_Nodes_loc[k][0]
    yloc = All_Nodes_loc[k][1]
    if k==0:
        plt.plot([xloc],[yloc],marker='o',markersize=10, color='red')
            else:
        plt.plot([xloc],[yloc],marker='o',markersize=4, color='gray')
            plt.text(xloc+1,yloc,str(k)+' ('+str(q[k])+')',fontsize=8)
plt.title('RPL with Sinkhole Attack')
plt.pause(0.1)
```

****************************** This is assigning appropriate level to each node ***************************

```
NodesAtLevel = {}
NodesAtLevel[0] = [0]
NodesTobeAssignedLevel = list(range(1,NumMobile+1))
NodeInPrevLevel = [0]
level = 1
Net_Creation_Success = True
while len(NodesTobeAssignedLevel)>0:

    nodesInCurrLevel = []
    for n in NodeInPrevLevel:
        for m in NodesTobeAssignedLevel:
            dist = math.sqrt((All_Nodes_loc[m][0]-All_Nodes_loc[n][0])**2
                 + (All_Nodes_loc[m][1]-All_Nodes_loc[n][1])**2)
            if dist <= CoverageRadius:
                nodesInCurrLevel.append(m)

    nodesInCurrLevel = list(set(nodesInCurrLevel))
    if len(nodesInCurrLevel)>0:
        NodesAtLevel[level] = [k for k in nodesInCurrLevel]
        for k in nodesInCurrLevel:
            NodesTobeAssignedLevel.remove(k)
        level = level + 1
        NodeInPrevLevel = [k for k in nodesInCurrLevel]
    else:
        print('Network unconnected........................')
        print('Nodes',NodesTobeAssignedLevel,'are isolated')
        Net_Creation_Success = False
        break
maxLevel = level – 1
```

************** This is making nodes in square shape are colored differently for each levels. ***********

```
if Net_Creation_Success:

    for level in range(maxLevel+1):
        for node_id in NodesAtLevel[level]:
            xloc = All_Nodes_loc[node_id][0]
            yloc = All_Nodes_loc[node_id][1]

            if level==0:
                pass
            else:
                plt.plot([xloc],[yloc],marker='s',markersize=5, color=mycolor[level%len(mycolor)])
                if level in [ ]:
                    pass
      plt.pause(0.1)
```

************* This is finding the parent and children relationship based on nearest parent ***************

```
    chosen_parents = {0:-1}
    for i in range(1,maxLevel+1):
        parents = NodesAtLevel[i-1]
        for k in NodesAtLevel[i]:
            min_dist = float('Inf')
            min_parent = -1
            loc_x = All_Nodes_loc[k][0]
            loc_y = All_Nodes_loc[k][1]
            for p in parents:
                if p!= sink_hole_node:
                    dist = (All_Nodes_loc[p][0]-loc_x)**2 +(All_Nodes_loc[p][1]-loc_y)**2
                if dist<min_dist:
                    min_dist = dist
                    min_parent = p
            chosen_parents[k] = min_parent

    chosen_children = {}
    for j in range (NumMobile+1):
        chosen_children[j]= [key for key,value in chosen_parents.items() if value == j]
```

**************************** This is finding all possible parents in Network *************************

```
    possible_parents = {0:[-1]}
    for i in range(1,maxLevel+1):
        parents = NodesAtLevel[i-1]
        for k in NodesAtLevel[i]:
            possible_parents[k] = []
            loc_x = All_Nodes_loc[k][0]
            loc_y = All_Nodes_loc[k][1]
            for p in parents:
                if p!= sink_hole_node:
                    dist = ((All_Nodes_loc[p][0]-loc_x)**2 +(All_Nodes_loc[p][1]-loc_y)**2)**0.5
                    if dist<CoverageRadius:
                        possible_parents[k].append(p)

    success = True
    for j in range(1, NumMobile+1):
        if j not in NodesAtLevel[1]:
            if j not in NodesAtLevel[2]:
                nParents = len(possible_parents[j])
                if nParents<1:
                    print("Some nodes have only NOOOO parent.........................")
                    print('Node ',j,':', possible_parents[j])
    Q = [k for k in q]
    for i in range(maxLevel,0,-1):
        nodes = NodesAtLevel[i]
        for n in nodes:
            parent = chosen_parents[n]
            Q[parent] += Q[n]
    Q_ = Q[0]
```

*************************** **This is plotting parents and children relationship** ************************

```
for i in range(maxLevel):
    nodes = NodesAtLevel[i]

    for n in nodes:
        xloc = All_Nodes_loc[n][0]
        yloc = All_Nodes_loc[n][1]

        for c in chosen_children[n]:

            child_xloc = All_Nodes_loc[c][0]
            child_yloc = All_Nodes_loc[c][1]
            plt.plot([xloc,child_xloc],[yloc,child_yloc],color=mycolor[i%len(mycolor)])

plt.pause(0.1)
```

**************************** **This is finding neighboring nodes in Network** *************************

```
neighbor_nodes = {}
for k in range(NumMobile+1):
    loc_x = All_Nodes_loc[k][0]
    loc_y = All_Nodes_loc[k][1]
    my_neighbors = []
    for n in range(NumMobile+1):
        if n!=k:
            dist = ((All_Nodes_loc[n][0]-loc_x)**2
                    +(All_Nodes_loc[n][1]-loc_y)**2)**0.5
            if dist<=CoverageRadius:
                my_neighbors.append(n)

    neighbor_nodes[k] = my_neighbors
```

*********************************** **This is TASA algorithm begins** ***********************************

```
slot = 0
while q[0] != Q_:
    slot += 1

    chosen_children_temp = {}
    for j in range (NumMobile+1):
        chosen_children_temp[j] = [key for key,value in chosen_parents.items() if value == j]

    parent_candidates = list(range(NumMobile+1))
    for k in range(NumMobile+1):
        if chosen_children_temp[k]==[]:
            parent_candidates.remove(k)

    DFCLk = []
    selectednodes_DFCL = []

    while len(parent_candidates)>0:

        Qparent_candidates = [Q[k] for k in parent_candidates]
        maxQ = max(Qparent_candidates)

        no_maxQ_nodes = 0
        indexval_maxQ = []
        it = 0
        for valQ in Qparent_candidates:
          if valQ == maxQ:
              no_maxQ_nodes += 1
              indexval_maxQ.append(it)
          it += 1

        level_Parent = {}
        if no_maxQ_nodes > 1:
            for i in range(level):
                level_Parent[i] = []
                nodes = NodesAtLevel[i]
                for n in nodes:
                    for index_v in indexval_maxQ:
                        if n == parent_candidates[index_v]:
                            level_Parent[i].append(parent_candidates[index_v])

            for k,val in level_Parent.items():
                if len(val) > 0:
                    if len(val) == 1:
                        select_parent = val[0]
                        break
                    elif len(val) > 1:
                        select_parent = random.choice(val)
                        break

        else:
            idx = Qparent_candidates.index(maxQ)
            select_parent = parent_candidates[idx]

         children = chosen_children_temp[select_parent]
        Qchildren_candidates = [Q[k] for k in children]
        qchildren_candidates = [q[k] for k in children]

        maxQ = max(Qchildren_candidates)
        idx = Qchildren_candidates.index(maxQ)
        selectedChild = children[idx]
        selecchild_q = qchildren_candidates[idx]

        inthelist = 1
        while inthelist == 1:
            if selectedChild in selectednodes_DFCL or selecchild_q == 0:
                #print("In DFCL or q = 0")
                children.remove(children[idx])
                if children == []:
                    parent_candidates.remove(select_parent)
                    break
                Qchildren_candidates = [Q[k] for k in children]
                qchildren_candidates = [q[k] for k in children]
                maxQ = max(Qchildren_candidates)
                idx = Qchildren_candidates.index(maxQ)
                selectedChild = children[idx]
                selecchild_q = qchildren_candidates[idx]

            else:
                inthelist = 0
                selectednodes_DFCL.append(select_parent)
                selectednodes_DFCL.append(selectedChild)
                DFCLk.append((select_parent,selectedChild))
                parent_candidates.remove(select_parent)
                if selectedChild in parent_candidates:
                    parent_candidates.remove(selectedChild)
```

************************ This is making Interference Conflict graph in Network ***********************

```
DCFL_nodes = []
DFCL_Parents = []
DFCL_Children = []
for k in DFCLk:
    DCFL_nodes.append(k[0])
    DCFL_nodes.append(k[1])
    DFCL_Parents.append(k[0])
    DFCL_Children.append(k[1])

IK = {}
for j in DCFL_nodes:
    neighbor_array = []
    neighborsofDCFL = neighbor_nodes[j]
    for neighbor in neighborsofDCFL:
        if neighbor in DCFL_nodes:
                neighbor_array.append(neighbor)
    IK[j] = neighbor_array
```

*********************************** This is coloring for each node ************************************

```
remaining_Parents = []
remaining_Children = []
for k in DFCLk:
    remaining_Parents.append(k[0])
    remaining_Children.append(k[1])

Channel_pattern = {}
Ch_assigned_nodes = []
for ch in range(channels):
    Channel_pattern[ch] = []

    while remaining_Children != []:
        if Channel_pattern[ch] == []:
            Qchildren_candidates = [Q[k] for k in remaining_Children]
            maxQ = max(Qchildren_candidates)
            idx = Qchildren_candidates.index(maxQ)
            select_child = remaining_Children[idx]
            select_parent = remaining_Parents[idx]

            Channel_pattern[ch].append((select_parent,select_child))
            remaining_Children.remove(select_child)
            remaining_Parents.remove(select_parent)

        else:
            Qchildren_candidates = [Q[k] for k in remaining_Children]
            maxQ = max(Qchildren_candidates)
            idx = Qchildren_candidates.index(maxQ)
            select_child = remaining_Children[idx]
            select_parent = remaining_Parents[idx]

             intherange = 0
            for val in Channel_pattern[ch]:
                for k,v in IK.items():
                    if k == select_child:
                        if val[0] in v or val[1] in v :
                            intherange = 1
                    elif k == select_parent:
```

```
                    if val[0] in v or val[1] in v :
                        intherange = 1
                if intherange == 0:
                    Channel_pattern[ch].append((select_parent,select_child))
                remaining_Children.remove(select_child)
                remaining_Parents.remove(select_parent)

        for v in Channel_pattern[ch]:
            DFCL_Children.remove(v[1])
            DFCL_Parents.remove(v[0])
        remaining_Parents = list(DFCL_Parents)
        remaining_Children = list(DFCL_Children)
    txnode = [ ]
    rxnode = [ ]

    for key,coloredLinks in Channel_pattern.items()
        for nodepair in coloredLinks:
            q[nodepair[0]] += 1
            q[nodepair[1]] -= 1
    Q = [k for k in q]
    for i in range(maxLevel,0,-1):
        nodes = NodesAtLevel[i]
        for n in nodes:
            parent = chosen_parents[n]
            Q[parent] += Q[n]
    string = ""
    for ch in Channel_pattern:
        for n in Channel_pattern[ch]:
            string += str(n[1])+","+str(n[0])+ " "
        string += "\t|"
    string += str(slot)
    print(string)
```

********************************** This is drawing nodes in the network ******************************

```
    if slot in [1,2,3]:
        fig2, ax2 = plt.subplots()
```

```
plt.title('Slot: '+str(slot))
for i in range(maxLevel+1):
    nodes = NodesAtLevel[i]
    for n in nodes:
        xloc = All_Nodes_loc[n][0]
        yloc = All_Nodes_loc[n][1]
        plt.plot([xloc],[yloc],marker='o',markersize=6, color=mycolor[i])
        plt.text(xloc+1,yloc,str(n),fontsize=10)
        plt.text(xloc+1,yloc+3,str(q[n])+','+str(Q[n]),fontsize=10,color='blue')
        for c in chosen_children[n]:
            child_xloc = All_Nodes_loc[c][0]
            child_yloc = All_Nodes_loc[c][1]
            plt.plot([xloc,child_xloc],[yloc,child_yloc],color=mycolor[i])

for k in DFCLk:
    rxnode = k[0]
    txnode = k[1]
    xloc_tx = All_Nodes_loc[txnode][0]
    yloc_tx = All_Nodes_loc[txnode][1]
    xloc_rx = All_Nodes_loc[rxnode][0]
    yloc_rx = All_Nodes_loc[rxnode][1]
    plt.plot([xloc_tx,xloc_rx],[yloc_tx,yloc_rx],color='gray',linewidth=7,alpha=0.3)
    for ch in range(channels):
        if (rxnode ,txnode) in Channel_pattern[ch]:
            link_col = mycolor[ch%len(mycolor)]
            plt.plot([xloc_tx,xloc_rx],[yloc_tx,yloc_rx],color= link_col,linewidth=4)
plt.pause(0.1)
```

**************************** This is printing the results *************************

```
print('Sinhole Attack Node \t', sink_hole_node )
print('Number of channels\t', channels)
print('Coverage          \t',CoverageRadius)
print('Number of mobiles \t',NumMobile)
print('Traffic (max)     \t',X)
print('Total time slots =',slot)
```

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Comput. Networks, vol. 54, no. 15, pp. 2787–2805, 2010, doi: 10.1016/j.comnet.2010.05.010.

[2] B. T. Mi, X. Liang, and S. Sen Zhang, “A Survey on Social Internet of Things,” Jisuanji Xuebao/Chinese J. Comput., vol. 41, no. 7, pp. 1448–1475, 2018, doi: 10.11897/SP.J.1016.2018.01448.

[3] Cisco Systems Inc., “Routing Protocol for LLN (RPL) Configuration Guide, Cisco IOS Release 15M&T,” p. 20, 2015.

[4] G. A. Da Costa and J. H. Kleinschmidt, “Implementation of a wireless sensor network using standardized IoT protocols,” Proc. Int. Symp. Consum. Electron. ISCE, pp. 17–18, 2016, doi: 10.1109/ISCE.2016.7797327.

[5] N. F. Andhini, “済無 No Title No Title,” J. Chem. Inf. Model., vol. 53, no. 9, pp. 1689–1699, 2017.

[6] A. Dhumane, A. Bagul, and P. Kulkarni, “A review on routing protocol for low power and lossy networks in IoT,” Int. J. Adv. Eng. Glob. Technol., vol. 3, no. 12, pp. 1440–1444, 2015.

[7] 2011 Citra Kunia putri dan trisna insan Noor, “済無 No Title No Title,” Anal. pendapatan dan tingkat Kesejaht. rumah tangga petani, vol. 53, no. 9, pp. 1689–1699, 2013.

[8] lucia maria aversa Villela, “済無 No Title No Title,” J. Chem. Inf. Model., vol. 53, no. 9, pp. 1689–1699, 2013.

[9] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” 1st Symp. Networked Syst. Des. Implementation, NSDI 2004, 2004.

[10] H. S. Kim, J. Ko, D. E. Culler, and J. Paek, “Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey,” IEEE Commun. Surv. Tutorials, vol. 19, no. 4, pp. 2502–2525, 2017, doi: 10.1109/COMST.2017.2751617.

[11] L. Networks and J. Hui, “RPL : IPv6 Routing Protocol for Low,” no. October, 2009.

[12] F. Medjek, D. Tandjaoui, M. R. Abdmeziem, and N. Djedjig, “Analytical evaluation of the impacts of Sybil attacks against RPL under mobility,” 12th Int. Symp. Program. Syst. ISPS 2015, pp. 13–21, 2015, doi: 10.1109/ISPS.2015.7244960.

[13] A. J. H. Witwit and A. K. Idrees, “A comprehensive review for rpl routing protocol in low power and lossy networks,” Commun. Comput. Inf. Sci., vol. 938, no. September, pp. 50–66, 2018, doi: 10.1007/978-3-030-01653-1_4.

[14] Tsao T, Alexander R, Dohler M, Daza V, Lozano A, Richardson M. A security threat analysis for the routing protocol for low-power and lossy networks (RPLs). RFC7416. 2015 Jan:131.

[15] A. Mayzaud, R. Badonnel, and I. Chrisment, “A taxonomy of attacks in RPL-based internet of things,” Int. J. Netw. Secur., vol. 18, no. 3, pp. 459–473, 2016.

[16] A. Raoof, A. Matrawy, and C. H. Lung, “Routing Attacks and Mitigation Methods for RPL-Based Internet of Things,” IEEE Commun. Surv. Tutorials, vol. 21, no. 2, pp. 1582–1606, 2019, doi: 10.1109/COMST.2018.2885894.

[17] M. Alzubaidi, M. Anbar, S. Al-Saleem, S. Al-Sarawi, and K. Alieyan, “Review on mechanisms for detecting sinkhole attacks on RPLs,” ICIT 2017 - 8th Int. Conf. Inf. Technol. Proc., pp. 369–374, 2017, doi: 10.1109/ICITECH.2017.8080028.

[18] A. Le, J. Loo, K. K. Chai, and M. Aiash, “A specification-based IDS for detecting attacks on RPL-based network topology,” Inf., vol. 7, no. 2, 2016, doi: 10.3390/info7020025.

[19] S. Raza, L. Wallgren, and T. Voigt, “SVELTE: Real-time intrusion detection in the Internet of Things,” Ad Hoc Networks, vol. 11, no. 8, pp. 2661–2674, 2013, doi: 10.1016/j.adhoc.2013.04.014.

[20] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, “Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things,” Proc. 2015 IFIP/IEEE Int. Symp. Integr. Netw. Manag. IM 2015, pp. 606–611, 2015, doi: 10.1109/INM.2015.7140344.

[21] A. Dvir, T. Holczer, and L. Buttyan, “VeRA - Version number and rank authentication in RPL,” Proc. - 8th IEEE Int. Conf. Mob. Ad-hoc Sens. Syst. MASS 2011, pp. 709–714, 2011, doi: 10.1109/MASS.2011.76.

[22] K. Weekly and K. Pister, “Evaluating sinkhole defense techniques in RPL networks,” Proc. - Int. Conf. Netw. Protoc. ICNP, 2012, doi: 10.1109/ICNP.2012.6459948.

[23] H. Perrey, M. Landsmann, O. Ugus, T. C. Schmidt, and M. Wählisch, “TRAIL: Topology Authentication in RPL,” 2013, [Online]. Available: http://arxiv.org/abs/1312.0984.

[24] K. Iuchi, T. Matsunaga, K. Toyoda, and I. Sasase, “Secure parent node selection scheme in route construction to exclude attacking nodes from RPL network,” 2015 21st Asia-Pacific Conf. Commun. APCC 2015, pp. 299–303, 2016, doi: 10.1109/APCC.2015.7412530.

[25] M. Zaminkar and R. Fotohi, “SoS-RPL: Securing Internet of Things Against Sinkhole Attack Using RPL Protocol-Based Node Rating and Ranking Mechanism,” Wirel. Pers. Commun., vol. 114, no. 2, pp. 1287–1312, 2020, doi: 10.1007/s11277-020-07421-z.

[26] R. Damasevicius, G. Ziberkas, V. Stuikys, and J. Toldinas, “Energy consumption of hash functions,” Elektron. ir Elektrotechnika, vol. 18, no. 10, pp. 81–84, 2012, doi: 10.5755/j01.eee.18.10.3069.

[27] Palattella MR, Accettura N, Dohler M, Grieco LA, Boggia G. Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks. In2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC) 2012 Sep 9 (pp. 327-332). IEEE.

[29] L. Paradis and Q. Han, “TIGRA: TImely sensor data collection usingdistributed GRAph coloring,” in Proc. of 6th IEEE Int. Conf. on Pervasive Computing and Communications (PerCom), Hong Kong, Mar. 2008

[30] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler and T. Engel, “On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH," in IEEE Sensors Journal, vol.13, no. 10, pp. 3655-3666, Oct. 2013, doi: 10.1109/JSEN.2013.2266417.

[31] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco and G. Boggia, “Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15.4e networks,"IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC), 2012, pp. 327-332, doi: 10.1109/PIMRC.2012.6362805.

REFERENCES

VITA

NAME	Tay Zar Bhone Maung
DATE OF BIRTH	15 Oct 1989
PLACE OF BIRTH	Myanmar
INSTITUTIONS ATTENDED	Computer University (Taungoo) University of Computer Studies Yangon, Myanmar
PUBLICATION	14th RC-EEE 2021 International Conference in Thailand "A Comprehensive Survey of Sinkhole Attack in Routing Protocol for Low-Power and Lossy Networks for IoT Devices"
AWARD RECEIVED	Student Poster Award in 15th ISST 2021