DEVELOPMENT OF AN IMPROVED SPOOFING ATTACK MITIGATION SCHEME IN IPV6 OVER LOW-POWER WIRELESS PERSONAL AREA NETWORK

BY

AMALIMEH Esther Ugonma

P17EGCP8058

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA.

SEPTEMBER, 2021

DEVELOPMENT OF AN IMPROVED SPOOFING ATTACK MITIGATION SCHEME IN IPV6 OVER LOW-POWER WIRELESS PERSONAL AREA NETWORK

BY

AMALIMEH Esther Ugonma

P17EGCP8058

estheramalimeh@gmail.com

A Dissertation Submitted to the School of Post Graduate Studies, Ahmadu Bello University Zaria, in Partial Fulfilment of the Requirement for the Award of Master of Science (M.Sc.) Degree in Computer Engineering.

SEPTEMBER, 2021

DECLARATION

I declare that this dissertation entitled "Development Of An Improved Spoofing Attack Mitigation Scheme In IPv6 Over Low-Power Wireless Personal Area Network" has been carried out by me in the Department of Computer Engineering, Ahmadu Bello University, Zaria as part of the requirements for the award of the degree of Master of Science in Computer Engineering. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other institution.

Amalimeh Esther	____________________	____________________
	Signature	Date

CERTIFICATION

This dissertation entitled **“**DEVELOPMENT OF AN IMPROVED SPOOFING ATTACK MITIGATION SCHEME IN IPV6 OVER LOW-POWER WIRELESS PERSONAL AREA NETWORK” by AMALIMEH Esther meets the regulations governing the award of the degree of Master of Science (M.Sc.) in Computer Engineering of the Ahmadu Bello University and is approved for its contribution to knowledge and literary presentation.

Dr E. A. Adedokun	___________________	___________________
(Chairman, Supervisory Committee)	Signature	Date
Prof. M. B. Mu’azu	___________________	___________________
(Member, Supervisory Committee)	Signature	Date
Prof. M. B. Mu’azu	___________________	___________________
(Head of Department)	Signature	Date
Prof. Sani. A. Abdullahi	___________________	___________________
(Dean, School of Postgraduate Studies)	Signature	Date

DEDICATION

This research is dedicated to the God-head and also to my children ‘Tega and ‘Mine.

ACKNOWLEDGEMENT

My Profound gratitude goes first and foremost to my Heavenly Father, the author and creator of all things, my Lord Jesus Christ, without whom I am nothing and to the Holy spirit, my inspiration.

My sincere appreciation goes to my supervisors, Dr. E. A. Adedokun and Prof. M. B. Mu'azu for their patience, guidiance, motivation, vast knowledge and constant drive to make this work a reality. I am so privileged and honoured to have such a wonderful combination of supervisory committee. God bless you Sirs.

My heartfelt thanks also goes to Dr. B.O. Sadiq, and Engr. Abdul for their excellent contributions and counselling. Their immeasurable guidance helped me shape the research problem and constantly provided me with the insight towards the research and constant reminder of the need to complete this research work on time.

I am thankful to all the lecturers of Computer Engineering Department, Ahmadu Bello University, namely; Dr. T. H. Sikiru, Dr. Y. A. Sha'aban, Dr. Emmanuel Okafor, Dr. I. J. Umoh, Dr. M.B. Abdulrazaq, Dr. B. Yahaya, Dr. A. T. Salawudeen, Dr. H. Bello-Salau, Engr. A. Umar, Engr. O. Ajayi, Engr. R. Adebiyi, Dr. Z. M Abubakar, Dr Y. Ibrahim, Engr. Z. Haruna and others not mentioned for all their contributions, motivation and assitance.

I am also grateful to all my friends and associates who have supported me in one form or another, to all my fellow researchers and P17 Computer/Control Engineering colleagues, thank you for being a part of this.

My deepest regard and appreciation goes to my parents Pastor E.O.Oji and Elder Mrs C.N Oji, thank you for your prayers and support.

Last but not the least, my special thanks and deepest gratitude to my backbone, my husband Dr. B.E.Amalimeh your endless love and unconditional support is beyond expression and to my children Oghenetega and Oghenemine Amalimeh for their understanding, inspiration, motivations and prayers throughout the course of this program.

Esther Amalimeh

April, 2021.

ABSTRACT

This research entails the development of an improved scheme (temporary private IPv6 address scheme based on node and access-router relationship) for mitigating spoofing attack in IPv6 over low-power wireless personal area network (6LoWPAN). Existing schemes for mitigating spoofing attack in a 6LoWPAN still suffers from spoofing attacks and also encounters the problem of high computational overhead resulting from frequent changes in IP addresses in the network. This usually happens when the neighbor solicitation (NS) and neighbor advertisement (NA) control messages containing a victim's MAC address is spoofed. The IP address and lifetime information of a legitimate node can then be deregistered by an attacker, thereby disconnecting legitimate nodes from the network, disrupting the nodes from actively participating in the network and corrupting the routing table with incorrect routing information. Therefore, this research is carried out to address these problems and is executed in two stages; first, a temporary-private addressing scheme utilizing node and access-router relationship is developed while in the second stage a node based mitigating scheme for spoofing attack is developed for the 6LoWPAN utilizing the temporary addressing scheme to periodically change and assign temporary addresses to communicating nodes in the 6LoWPAN. The developed improved spoofing mitigation scheme achieved an average marginal attack disruption window (ADW) of 13% above the previous scheme, while the percentage improvement for energy consumed was 66% and the address change failed rate(ACFR) was also significantly improved by 82% when compared with previous scheme.

TABLE OF CONTENT

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronyms	Definitions
6LoWPAN	IPv6 over Low power Wireless Personal Area Network
ADW	Address Disruption Window
DAD	Duplicate Address Detection
DAO	Destination Advertisement Object
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graph
DTLS	Datagram Transprot Layer Security
ICMPv6	Internet Control Message Protocol version 6
ID	Identification
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IID	Interface Identification
IoT	Internet of Things
IP	Internet Protocol

IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
MAC	Medium Access Control
MTU	Maximum Transmission Unit
NA	Neighbour Advertisement
ND	Neighbour Discovery
NDP	Neighbour Discovery Protocol
NS	Neighbour Solicitation
OSI	Open Systems Interconnection
PAC	Periodic Address Change
PAN	Personal Area Network
RPL	Routing Protocol for LowPower and Lossy
RFID	Radio Frequency Identification
TSCH	Time Synchronisation Channel Hopping
TTL	Time to Live
WBSN	Wireless Body Sensor Network
WSN	Wireless Sensor Network

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

With recent advancements in the Internet of Things (IoT) technology, many devices are getting connected to the world wide web. According to a report, there will be over 30 billion connected Internet of Things (IoT) devices by the end of the decade, which will create about $1.9 trillion economic value and comprises billions of intelligent communicating devices (Mathew, 2019). The existing internet protocol, Internet Protocol Version 4 (IPv4) cannot accommodate this expected considerable number of devices; therefore, the transition to Internet Protocol Version 6 (IPv6) becomes inevitable. This growth is predicted because wireless sensor networking described as one of the fastest-growing sectors in ubiquitous networking today, is expected to be a particular purpose vehicle for the spread of the IoT.

In order to transform Wireless Sensor Networks (WSN) from Personal Area Network (PAN) to Low-Power Personal Area Network (LoWPAN), IEEE standard 802.15.4 was introduced, which contains a wireless Medium Access Control (MAC) and a physical layer for low-rate wireless personal area network. Presently, in some Sensor networks there exist non-IP network layer protocol which connects sensor node to the internet. An example of such devices includes ZigBee, where TCP/IP protocol is not used.(Ee *et al.*, 2010).

Internet Protocol version 6 (IPv6) over Low Power Wireless Personal Area Network (6LoWPAN) is a technique designed with the capabilities of integrating sensor network node to the internet by applying TCP/IP to WSN. Hence based on this technology, WSN devices are provided with an IP-based communication abilities by integrating an adaptation layer on IEEE 802.15.4 link layer for packet reassembling and packet fragmentation (Hummen *et al.*, 2013). Furthermore, 6LoWPAN is a standard developed and maintained by Internet Engineering Tax Force (IETF) working group developed for devices that are compatible with the IEEE 802.15.4 standard (EEE 802.15.4 is a technical standard which defines the operation of low-rate wireless personal area networks (LR-WPANs). It specifies the physical layer and media access control for LR-WPANs, and is maintained by the IEEE 802.15 working group, which defined the standard in 2003). It is therefore, characterized by low computing power, low memory-capacity, lower bit-rate, short-range, and low cost. (This concept was initiated with the view that smaller and low-power devices with little processing power should apply to internet protocol and participate in the internet of things. (Zach & Carsten 2011; (Kumar & Tiwari, 2012)).

6LoWPAN has numerous application areas ranging from healthcare, smart homes, smart cities, agriculture, industrial and environmental monitoring and many others. With these diverse applications using 6LoWPAN protocol in nodes deployed to insecured environment or physically unattended area, such devices may be susceptible to malicious insider nodes as a result of lack of proper authentication mechanism for nodes protection (Ozturk & Nagarnaik, 2011). The identity of legitimate nodes within such an environment may be spoofed, spoofing by itself may be considered as an attack or a means for carrying out future attacks such as Denial-of-Service attack, Man-in-the-middle attack, impersonation attack, and several other forms of attack on the network can eventually be carried out (Rai & Asawa, 2017). Thus when spoofing is considered as an attack it may deprive a legitimate node from performing certain actions. For instance, when the identity of a node is spoofed, a message from an attacker may create a valid neighboring relationship with other nodes on the network thereby preventing the attacked node from developing such a relationship with other legitimate neighbors, thus there is a possibility that an attacker can disrupt the network when its presence is not detected (Mavani & Asawa, 2019). Hence this research is focused on developing a technique which is aimed at minimizing the attack disruption time thereby reducing the duration of attacks by malicious nodes as a result of spoofing in the network.

1.2 Significance of Research

The 6LoWPAN protocol is designed to act as a platform that provides a pathway for compatibility of devices and also for connecting devices with limited processing capabilities to the internet. Since its invention, this protocol has been discovered to serve as one of the critical requirements for connecting limited capability devices to the internet effectively and securely and needs to be explored. Network security is a major challenge in a constrained IoT environment, this is because traditional security solutions like firewalls cannot be applied in constrained environments. Thus, ensuring secure communication in 6LoWPAN is very important as the network is prone to different forms of attacks such as DoS attack, sinkhole attack, wormhole attack, man-in-the-middle attacks, impersonation attacks e.t.c. which can be launched to disrupt the network and corrupt routing information on the network. Most significantly among the numerous attacks is the spoofing attacks where legitimate node identity can be spoofed by a malicious node. Due to the peculiarity of the 6LoWPAN, the nodes can be deployed to physically unattended areas, insecured environments, and are mostly used for remote monitoring, therefore the nodes usually go into sleep mode whenever they are not transmitting or receiving data in order to conserve energy making them prone to such attacks. Attacker nodes can actually study the sleep and wake patterns of their targeted victim node and take advantage of the sleep period to spoof the victims address. The use of spoofed IP addresses of legitimate nodes compromises the identity of the nodes thereby corrupting the routing information on the network, disconnects legitimate nodes from the network and disrupts the network generally. This is possible because it is difficult to differentiate a legitimate node from a malicious node on a network. Hence in the absence of a mechanism to detect and differentiate spoofed IDs in a 6LoWPAN, it becomes necessary to overcome such security threats by developing a suitable spoofing attack mitigation scheme that is reliable, robust and offers lower communication overhead on the network to mitigate against spoofing.

1.3 Statement of Problem

Network security is a major problem in resource-constrained IoT environments because traditional security solutions such as firewalls typically can not be applied in such constrained environments. In a 6LoWPAN, smooth transmission of packets is required and this can only be achieved by ensuring the security of node/device identity. The 6LoWPAN usually suffers from an IP spoofing attack as a result of IP-MAC binding where a malicious node attaches its IP address to the MAC address of a legitimate node, directs traffic to itself, and disconnects the victim from the network. Popular approaches which propose constant changing of IPv6 addresses in 6LoWPAN to mitigate IP spoofing still suffers from spoofing attacks where the nodes are compromised when in sleep mode due to energy-conserving nature of the 6LoWPAN devices. This is because an attacker can use spoofed neighbor solicitation and neighbor advertisement control message which contains the MAC address, IP address and lifetime information of a legitimate node and can deregister/disconnect a legitimate node from the network, thereby disrupting communication in the network, and corrupting the routing table with incorrect routing information. Also, the computational cost of the network is increased as a result of the high frequency of address changes. Hence it is necessary to address these problems by ensuring a secure identity of nodes in a 6LoWPAN at a reduced computational cost.

1.4 Aim and Objectives

This work is aimed at developing an improved spoofing attack mitigation scheme in IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) using a dynamic periodic allocation of temporary IPV6 address, and also node and access-router relationship.

In achieving the aforementioned aim, the following objectives are set

1. Develop a temporary-private IPv6 addressing scheme based on node and access-router relationship.
2. Develop a node-based dynamic IP changing scheme for mitigating spoofing in 6LoWPAN.
3. Evaluate the performance of the developed improved scheme using metrics such as Address Disruption Window, Energy Consumption and Frequency of Address Change (PAC) /Address Change Failed Rate(ACFR) and compare with the work of Mavani & Asawa (2019).

1.5 Dissertation organization

The organization of this dissertation is reported as follows: Chapter One presents the general background of the study, significance of the research, statement of problem as well as the aim and objectives of the study. Chapter Two comprises of detailed review of similar works and the fundamental concepts relevant to the study. The materials and methods used to realize the aim and objectives of the research is presented in Chapter Three, Chapter Four reports and discusses the results obtained from the study. Chapter Five concludes the research and makes reccommendations for further work.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This section is made up of two sub-sections. The first section discusses fundamental concepts relevant to the subject matter, and the second section will review works similar to this research.

2.2 Review of Fundamental Concepts

In this section, fundamental concepts to the subject matter are discussed and presented here.

2.2.1 6LoWPAN and Standardization

6LoWPAN stands for Internet Protocol version 6 over Low Power Wireless Personal Area Network (6LoWPAN). The concept of 6LoWPAN was invented with the idea that internet protocol should and could apply to devices with limited capabilities, these will enable devices with minimal processing power to be a part of the Internet of things (IoT). Thus 6LoWPAN is a set of standard defined by the Internet Engineering Task Force (IETF) which creates and maintains core internet standard and architecture work, this standard enables IPv6 over low-power, low-rate wireless networks on simple embedded devices to be efficiently used through an adaptation layer and the maximum use of related protocols**.** (Mathew, 2019; Mulligan, 2007; Shelby & Bormann, 2011). The release of the IEEE 802.15.4 standard in 2003 was the major influencing factor that led to 6LoWPAN standardization, thus resulting in the availability of an extensively supported standard for low power wireless embedded communications. The widespread acceptance of this standard encouraged the Internet community to further standardize an IP adaptation for similar wireless embedded links (Shelby & Bormann, 2011).

2.2.1.1 Routing protocol for low-power and lossy networks (RPL)

The Low-Power and Lossy Networks routing protocol (RPL) is a wireless network routing-protocol majorly used by devices with low power consumption capacities like 6LoWPAN. It is a major routing protocol for a route-over routing in 6LoWPANs. RPL uses Destination Oriented Directed Acyclic Graph (DODAG), DODAG Information Solicitation (DIS), DODAG Information Object (DIO), and Destination Advertisement Object (DAO) messages for control operations in terms of routing in the 6LoWPAN networks. In this network, three major classes of nodes are defined in RPL: Leaf Nodes, Intermediate Nodes and Sink (Root) Nodes. RPL does not have appropriate security implementations therefore it is prone to numerous attacks such as overloading routable infomation, rank attack, DAG inconsistency attack, version number attack etc. RPL control messages mainly are used in performing spoofing attacks on a 6LoWPAN which is mostly aimed at neighboring node routing information corruption or the neighboring cache connected to the network (Kumar & Tiwari, 2012; Mathew, 2019).

2.2.1.2 6LoWPAN neighbor discovery

The 6LoWPAN Neighbor Discovery (6LoWPAN-ND) is a protocol in 6LoWPAN that specifies the node addresses, configuration methods, networking techniques, and neighboring route discovery processes (Mulligan, 2007). In IPv6, the Neighbor Discovery (ND) protocol represents the major part in bootstrapping an IPv6 network. Here a node utilizes a ND protocol to locate neighboring nodes using the same link, to discover their link-layer addresses, search for routers, and also to maintain reacheability information about pathways to neighbors where nodes are actively involved in a communication. Other protocols can be used alongside ND such as Dynamic Host Configuration Protocol version 6 (DHCPv6) to acquire extra information regarding the node configuration process for resource-limited nodes in a LoWPAN (Kumar & Tiwari, 2012; Luo *et* *al.*, 2015); (Shelby & Bormann 2016). The 6LoWPAN-ND uses two major protocols; Neighbor Solicitation (NS) and Neighbor Advertisement (NA) control messages to discover neighbours. When a node comes into a LoWPAN network, much router attention is paid to either the advertisements messages being broadcasted from the routers in the network or the Router Solicitation expecting a response from any local router in the network (Mavani & Asawa, 2019).

2.2.2 Internet Protocol Version 6 (IPv6) Address

The IPv6 address is an alpha numerical description used in a network, to identify a network node or a computer that is part of an IPv6 network interface and to locate it on the network. These IP addresses are mostly sent out in the field of packet headers to show the origin and the destination of each packet in a network and the destination address is used for deciding packet routing to other networks (Mavani & Asawa, 2017b). The IPv6 has a large address scheme of 128bits, unlike the IPv4 which has only a 32bits scheme. Furthermore, the IPv6 address comprises two parts: the global routing prefix and the Interface Identifier (IID) part as presented in figure 2.1. The Global routing prefix is unique and identifies IPv6 subnet globally while the IID part is split into two parts namely: Node ID and Router ID parts, as also presented in figure 2.1 where the number of bits in the routing identification section is given as i while j is defined as the number of bits for Node identification part (Mavani & Asawa, 2017b, 2018).

Prefix ID	Router Id (i bits)	Node Id (j bits)
Global routing prefix	Interface ID (IID)	

Figure 2.1: *IPv6 Addressing (Mavani & Asawa, 2018)*

In IPv6 addressing, the unique global prefix and fixed MAC addresses, used by stateless address auto-configuration for interface identification provides a user a way to track nodes and other devices connected to the network using their IPv6 network prefix to minimize the likelihood of a user identity permanently tied to an IPv6 address format, a node may create temporary addresses with IIDs with respect to time-varying random bit strings and relatively short lifetimes, and subsequently replaced with new addresses. Thus these provisional addresses may be utilized as source addresses for originating links from nodes connected to the network (Mavani & Asawa, 2019; Wang & Mu, 2015).

2.2.3 IPv6 Address Auto Configuration

IPv6 Addressing is classified into stateless and stateful configuration types. In the stateless configuration, the process employs the use of every device. It automatically configures addresses using its Medium Access Control (MAC) address in Interface Identification (IID) part and router advertised global routing prefix. This process employs a Duplicate Address Detection (DAD) mechanism to ensure that the self-configured address is unique and not duplicated, however it attracts heavy communication overhead. While in the stateful address configuration, the addresses used by the device are formed using stateful protocols like Dynamic Host Configuration Protocol for IPv6 (DHCPv6). Using this method, the DHCPv6 server assigns a unique IPv6 address to every device located in the network, which can be configured in both ways, usually, the addresses are unchanged for a defined length of time as far as the nodes remain connected in the network. (Mavani & Asawa, 2018; Simpson *et al.*, 2007) (Tariq & Reyaz 2015) .

2.2.4 Approaches to Nodes and Service discovery in 6LoWPAN

There are basicially three (3) nodes and services supported by 6LoWPAN and used to discover and register a new node into the network.

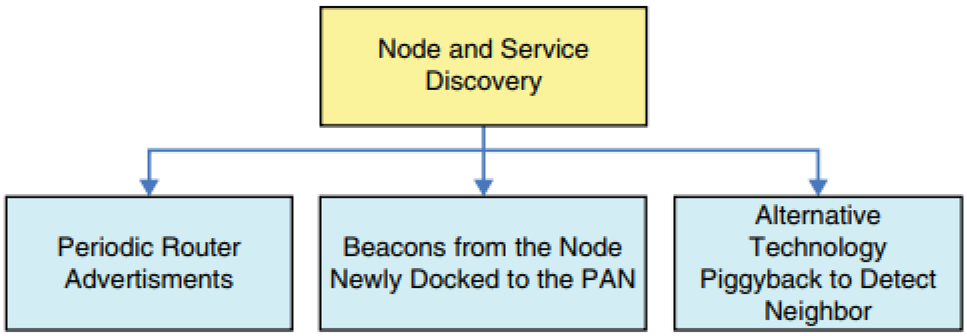

Figure 2.2:Node and Services in Neighbor Discovery (Misra & Goswami, 2017)

1. Periodic Advertisement: This neighbor discovery approach is also known as the You-Catch-Me method, a router receives a periodic advertisment from a node seeking an address via a stateless autoconfiguration mode and chooses its default router. The node will directly request an edge router for registeration using the node unicast message via an intermediate router for self registeration to the edge router, then an edge router feedback is provided by sending a confirmatory message directly to either the node or also via an intermediate router thus making an entry for the node. By this process, the node is registered to the network and can communicate with any device within and outside the LoWPAN (Mavani & Asawa, 2018; Misra & Goswami, 2017).
2. Router Solicitation: This neighbor discovery approach is also known as the I-Catch-You method, when a node comes into the Personal Area Network, it first of all sends a beacon to a sink node which would listen to the network and perform node registeration and docking of the node. It is the responsibility of the nodes to search out and discover the sink

nodes and after receiving its Time-To-Live (TTL) from the sink node, it becomes a part of the network. However, in this scenario, a sink node has all the needed information about the registered node in the network therefore it does not require any additional technique to remove a node as it uses TTL (Mavani & Asawa, 2018; Wang & Mu, 2015).

3. Alternative Technology: This approach also known as Some-1-Catch-Me uses an alternative technology such as Radio Frequency Identification (RFID). In this technique, node discovery is done by using an RFID tag and reader to discover sensor nodes, it saves all broadcast messages as it integrates RFID with 6LoWPAN.

These three categories of node and service discovery are commonly described as YouCatchMe, ICatchYou, and Some1CatchMe methods respectively (Silva, 2009); (Misra & Goswami, 2017) the node and service hierachy as presented above in Figure 2.2.

2.2.5. Neighbour Discovery Message Format

The following are classified among the five (5) neighbor discovery protocol messages (NDP); (i).Router Advertisement (Internet Control Message Protocol version 6(ICMPv6 type 134)), (ii).Router Solicitation (ICMPv6 type 133), (iii).Neighbor Advertisement (ICMPv6 type 136), (iv).Neighbor Solicitation (ICMPv6 type 135) and (v). Redirect (ICMPv6 type 137). In operating within this domain the ICMPv6 message structure, the network administrators have to format all NDP messages uniquely. Components like message headers, NDP messages and ICMPv6 header-specific data and zero or more NDP options are part of messaging in NDP. In order to implement certain functions, several alternatives are available in NDP messages. Extra information are also provided through these functions, for instance; mobility information, redirection data, specific routes, indicating IP addresses and MAC, on-link Maximum Transmission Unit (MTU) information, and on-link network prefixes. (Ahmed *et al.*, 2017). Figure 2.3 below depicts the message format of NDP.

Figure 2.3: Neighbor Discovery Message Structure (Ahmed *et al.*, 2017)

2.2.6. 6LoWPAN Application Areas

The 6LoWPAN has a very wide area of application which include but not limited to the following; healthcare, industrial monitoring, structural monitoring, agriculture, smart home network, childcare, vehicular telematics, situational awareness, and precision asset tracking for defense or firefighting (Mathew, 2019). Figure 2.4 gives an illustration of data communication between a smart soldier with a smart weapon equipped with different types of sensors in a 6LoWPAN network. The communication process in the network is a short-range communication that requires less data transmission. Thus, in this application scenario, the actual data are not transmitted rather it is the threshold information that is transmitted over the network hence the process is with the help of a 6LoWPAN. In the medical health care system, the 6LoWPAN is very helpful in monitoring patients by putting/embedding different types of the wireless sensor on their body forming what is a called wireless body sensor network (WBSN) to enable health personnel to track and monitor the health status of patients (Minoli, 2013; Misra & Goswami, 2017; Shelby & Bormann, 2011).

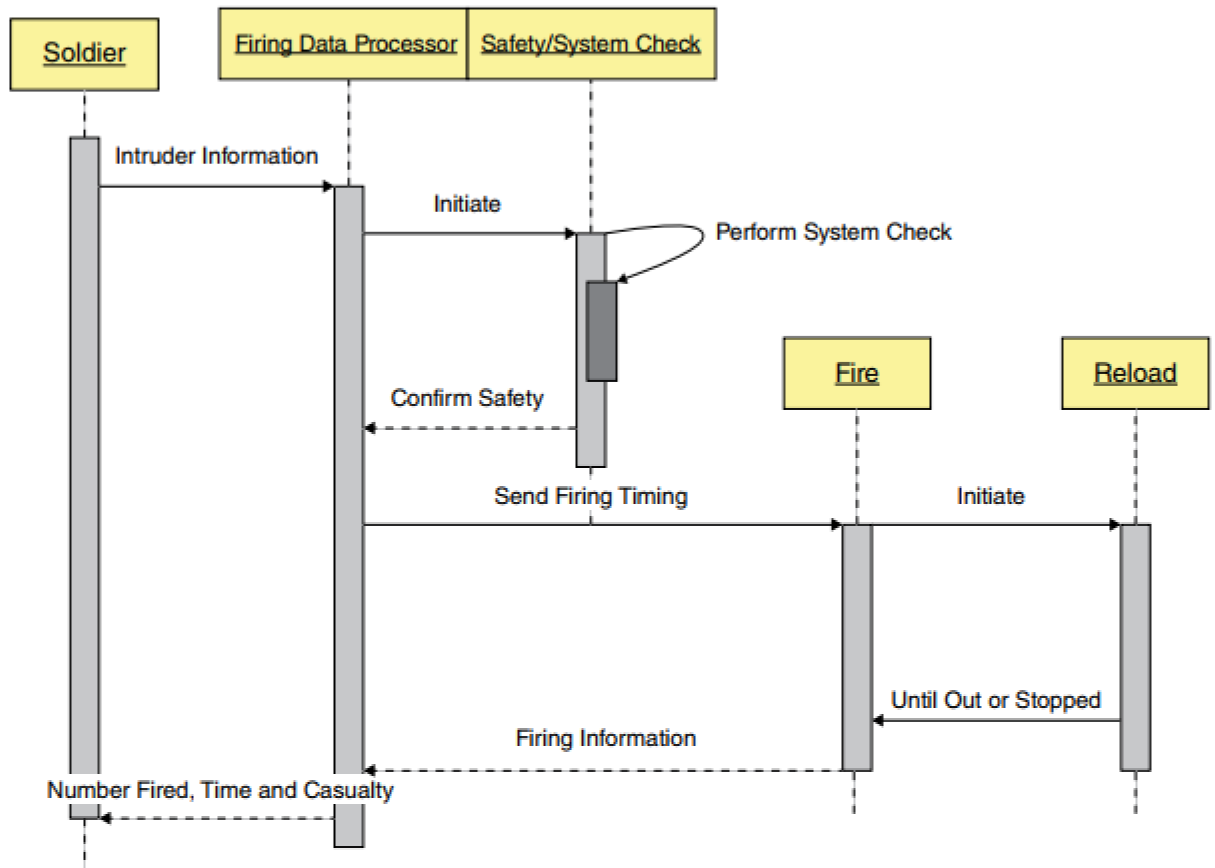

Figure 2.4: Illustration Of Data Communication Between A Smart Soldier And A Smart Weapon (Misra & Goswami, 2017).

2.2.7 6LoWPAN Architecture

The architecture for a 6LoWPAN constitutes of Low Power Wireless Area Networks (LoWPAN) which is regarded as IPv6 stub-networks. In a stub-network, an IP packet originates from or sent to but it does not act as a passage to another network. (Luo *et al.*, 2015) . Figure 2.5 shows the 6LoWPAN architecture. In the architecture, three different types of LoWPAN (a LoWPAN is a group of 6LoWPAN devices sharing the same IPv6 address prefix, this implies that irrespective of where a node is in a LoWPAN, it maintains the same address) has been defined namely.

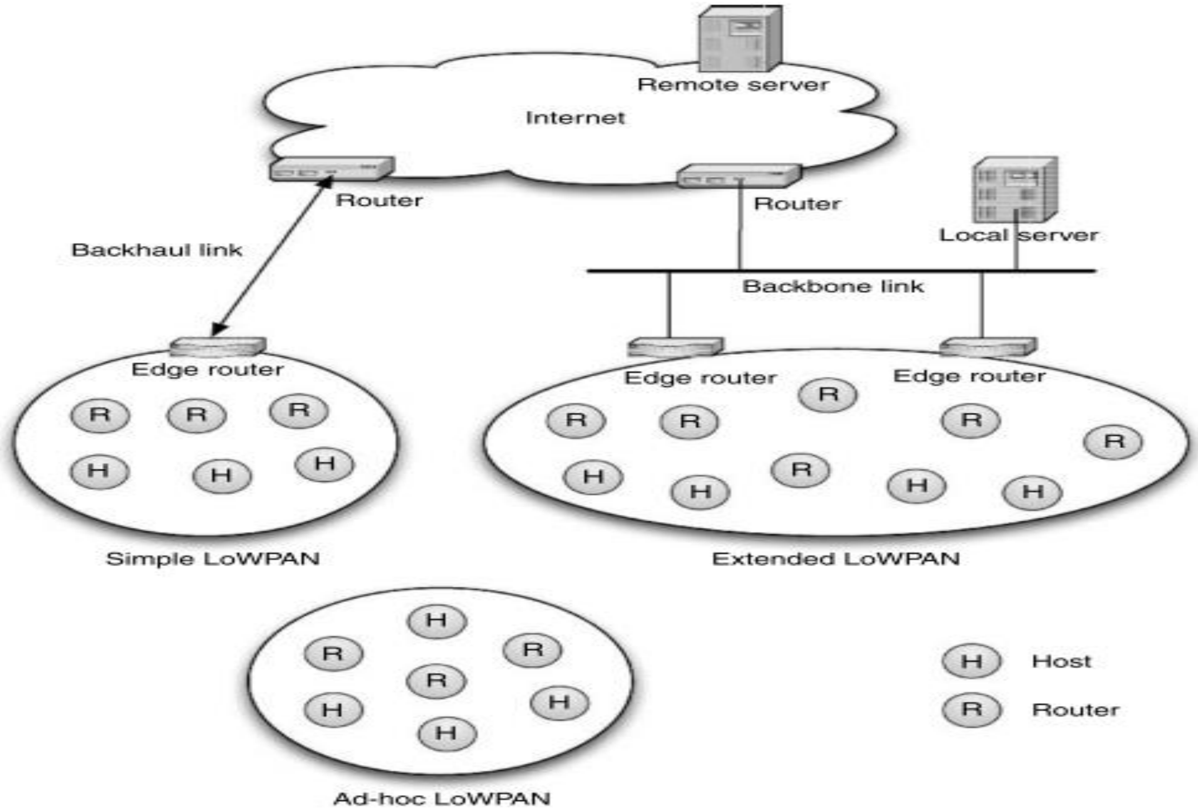

Figure 2.5: 6LoWPAN Architecture (Luo *et al.*, 2015)

1. Simple LoWPAN: This type of LoWPAN is connected through a single Edge Router to another IP network.
2. Ad hoc LoWPAN: This LoWPAN does not connect to the internet, it also operates without an infrastructure. In this LoWPAN, an edge router can be configured to perform two functions such as generating unique local unicast addresses and handling neighbor discovery registration.
3. Extended LoWPAN: This type of LoWPAN has multiple edge routers that belongs to different LoWPANs are interconnected with a backbone link (Shelby & Bormann, 2011).

2.2.8 IP address spoofing

This is the process of creating Internet Protocol (IP) packets using a false origin identity IP address to impersonate a node/device or another computing system on a network. The spoofed addresses can be used for numerous processes ranging from corrupting the routing information to denial of service by an intruder (Rai & Asawa, 2017). Examples of possible scenario where spoofed IP addresses are used in a trusted IP addressing system, the spoofed address can be employed by a network intruder in overcoming network security measures in a case where authentication based on IP addresses are used. This category of attack is very effective because of the trust and relationship existing among machines. This process may require the user to login into a network without providing user credential thus spoofing a device IP with a trusted connection, an intruder may gain uninterrupted access to the targeted machine or device without providing authentication (Wang & Mu, 2017) . Secondly, a spoofed IP address can also be used to start up a denial of service attack on a network by flooding and overwhelming the target system with a large volume of traffic. In a 6LoWPAN, spoofing attack can be an attack by itself or it can be used as a means to carry out an attack, it requires an illegitimate user to act as and be recognized as a legitimate user in order to attain an unlawful advantage. This spoofed address can be dishonestly used to launch other attacks like; Man-in-the-Middle attack, Denial-of-Service attack, impersonation attacks e.t.c spoofed IP address can be used in different manners in a 6LoWPAN. Instances of this kind of misuse include corruption of routing table, in this example, the routing information contained in the routing table is corrupted by using these spoofed RPL control message by an attacker via wrongly binding of IP-MAC address of the communicating devices on the network (Mavani & Asawa, 2019).

2.3 Review of Similar Works

In this section, literatures relevant to the subject matter are reviewed in order to gain a better perspective on the subject matter.

Ozturk and Nagarnaik (2011) proposed a distributed address allocation protocol for mitigating spoofing attack and recovering for node and link failure. An analytical modeling was conducted and based on this it was suggested that in the local network domain, the address-allocation latency in it is bounded by 2s, furthermore, by modelling the system analytically, it was further observed that the latency for address allocation in creating a network completely depends on the number of hops (depth of the tree) to the root node and probability of packet loss in a single-hop communication. Hence, the latency in the local node ID allocation and network ID allocation process were equally measured and evaluated the time required for such operation to be performed. Thus, in measuring the address allocation time, a timer was set when the radio was activated and later queried the time passed when address allocation is completed. Therfore based on the Experimental evaluation, the assignment time for node address is observed to be in the range of 386ms, which is usually so in the absence of background traffic. Therefore, in the presence of background traffic, (that is when background traffic is injected into the network), the allocation time for node address is observed to fall in the range of about 2s. This evaluation is based and verified by means of analytical solution. However, when compared with existing works, the proposed scheme offers a very fast recovery time in addition to inline or lower address allocation latencies. In this scheme, the identity of a legitmate node is easily spoofed in the presence of background traffic on the network.

Hannebert and Santos (2014) proposed and described protocols and security solutions for constrained devices, the scheme proposed showed notable benefit in terms of security extension of the IEEE 802.15.4e in Time Synchronisation Channel Hopping(TSCH) mode. The IPsec protocol suites have been compressed and adapted to the LoWPAN, this provides features to ensure the source authentication and data confidentiality even though it has an additional message overhead cost. Datagram Transport Layer Security (DTLS) – embedded at different levels of the OSI model into the 6LoWPAN stack is scalable, compatible with the constrained environment but authenticates only a few part of the message and does not protect privacy. Conclusively, DTLS is quite heavy for constrained nodes.

Wang and Mu (2015) proposed a 6LoWPAN addressing scheme with privacy support for protecting against IP address spoofing by constantly changing addresses to maintain address privacy. The addresses are valid only for a time until valid nodes change their addresses, in this case when an address is successfully spoofed the default IP becomes invalid after a given time. The proposed scheme is based on hierarchal addresses given by cluster-heads, they do not use EUI-64 based IPv6 addresses. It has a high communication overhead. In this scheme after establishing the network, a cluster-head is chosen and is assigned the responsibility of periodically assigning member node temporary IP addresses, this serves as a means of addressing IP spoofing attack in a 6LoWPAN. Furthermore, an address reclamation was designed for the regularly changing process, thus using this address reclamation technique a cluster-head is required to send a signal to a member of the network and when such signal is not received after a period, the process is initiated by any member of the network. Finally, the performance of the scheme was evaluated, thus the result obtained from the scheme demonstrates that they achieved good address privacy and exhibited an improved performance in fighting against spoofing without much extra communication overhead in the network. However, the proposed scheme still suffers from IP spoofing attacks most especially during alloction of temporary address to new member node, the identity of the new node gets spoofed during the process of IP assignment from the cluster-head. This degrades the network performance as legitimate nodes are disconnected from the network.

Oliveira *et al.* (2016) An advanced encryption standard-based node authentication and authorization solution are being proposed for granting access to only legitimate nodes in joining the network. In the proposed scheme a pre-shared key system with tamper-proof hardware was used to provide security from node to device. However, the proposed scheme requires each node in the network to maintain an approved list of a legitimate node in the network. Thus, the routers and gateways are susceptible to DOS attacks. When there is a growth in the network, it is equally expected that there is a growth in node list thus, there would be the need for storage requirement constrained devices with limited memory. Also, when the network constitutes either a mobile node or dynamic node, frequent refreshing and re-communicating becomes a necessity by the approved node list to all nodes this becomes a constant requirement which leads to high computational cost.

Mavani and Asawa (2017a) proposed and performed an indepth experimental analysis of IPv6 spoofing attacks in 6LoWPAN by modelling an attack tree. In the proposed method, the attack tree modelling provides an opportunity for analyzing spoofing attack using different parameters, such as time required to perform the attack, attack feasibility, network disruption attack and others. Secondly, the method proposed that spoofing attack is modelled based on IP-MAC bindings, this required finding attack time complexity and deriving attack disruption window(ADW) and Time-To-Live(TTL). However, based on the proposed method results obtained showed high level of IP spoofing attack in a 6LoWPAN by using IP-MAC binding, therefore finding the temporary characteristic of the attack was difficult leading to high energy consumption.

Wang and Mu (2017) introduced a communication scheme that ensured nodes security with privacy support for 6LoWPAN (CSP) aimed at providing end-to-end communication security for devices participating in a 6LoWPAN. In the proposed scheme, a node is first configured with its permanent node identity which is valid and kept invariable during its entire life time, when a session is launched by a node, a new identity is created to identify the node then using the generated ID communication can be established. When the session elapses, this new ID becomes invalid. Hence, in the proposed method, an identity for the node is created and randomly generated without routing information and network prefix, based on this process the origin of the source network segement and destination network segement message and the information flow can not be determine by an attacker. Secondly in the proposed scheme an address acquisition scheme is designed using beacon and usually broadcast periodically after every one hop count for new nodes joining the network. However when there is a session delay an increase in consumption in a routing path from source to destination a spoofing might occur and disrupt or corrupt the routing table of the access router.

Mavani and Asawa (2017b) proposed a technique for modeling and analyzing IP spoofing attack in 6LoWPAN, to identify the feasibility of carrying out IPv6 spoofing attack in the 6LoWPAN with respect to memory and energy consumption. The proposed method identified two different attack paths associated with either the wrong IPv6 address and the wrong MAC address for the node. Hence, these identified paths point out that spoofed RPL and 6LoWPAN-ND messages were used to perform the IPv6 spoofing attack in the network. Furthermore, attack success rate probability was analyzed using radio propagation environment and attack tree as parameters, hence it affects correct signal reception. Therefore, based on the analyses carried out, the attack success rate was dependent on the signal path loss. Secondly, based on the experimental analysis it was also observed that path loss grows exponentially as the distance with respect to loss path in the network it affects, the probability of attack success and the code utilized for the attack can be found contained in the node memory, and utilizing little energy the attack can be performed this is manifested in the feasibility study reported.

Hossain *et al.* (2018) proposed a technique called Secu-PAN for mitigating fragmentation-based network attacks in 6LoWPAN and its devices. In order to fend off fabrication and duplication attacks, they incorporated a Message Authentication Code-based scheme for per-fragment integrity and authenticity verification in their proposed scheme. This aids in verifying the autenticity of the node in the network. In order to prevent aganist spoofing attacks in the proposed method, a cryptographic technique for generating datagram-tag for IPv6 address was intergrated in the scheme to enhance the security of the node. Hence the performance of the proposed technique was evaluated and it was observed based on the results obtained that the proposed scheme reduces end-to-end delay and energy consumption when a 6LoWPAN is under attacks as long as the required keys are uncompromised and CGAs remained certified. However, in the proposed scheme under the condition of the ranking property in the 6LoWPAN using the neighbor advertisement control message of legitimate node IP can be spoofed if not detected on time it degrades the network performance in terms of packet delay and corrupt the routing information of neighboring nodes in the network.

Mavani and Asawa (2018) Proposed a privacy-enabled disjointed and dynamic addressing scheme with an auto-configuration protocol for 6LoWPAN to ensure the privacy of node and conflict-free IPv6 addressing in a LoWPAN. In the proposed scheme a three-level hierarchical addressing space was designed for each node to dynamically generate IP addresses based on congruence classes thus by using the congruence classes in the design along with the hierarchical addressing, the process facilitates the generation process of disjointed and non fragmented addresses for each node in the proposed scheme hence resulting in the generation of conflict-free addresses. Secondly in the proposed scheme, the node auto-configuration function was also designed, the process uses congruence seed shared by the access router to independently configure their address thereby reducing extra computational complexity on an access router. Finally, in ensuring address privacy in the protocol, the MAC address is periodically changed when the IP address changes. The privacy is derived from the interface identification part of the IPv6 address, the performance of the scheme was also evaluated and the result obtained as reported exhibits a better performance with a lower latency when compared with some existing schemes. However, the computational complexity is high and IP address spoofing still occurred in the network.

Mavani and Asawa (2019) proposed a technique for periodically changing private IPv6 addresses in a 6LoWPAN in order to mitigate spoofing attacks. The process incorporates the use of time to live (TTL) parameter and further derived an attack disruption window, thus by this process the disruption time of the attack caused by spoofing in 6LoWPAN can be reduced using a private-temporary address, the process allowed for self-healing of the network to take place thereby recovering from spoofing attack when detected on time. The node's IPv6 addresses are periodically changed to disassociate a node from its permanent identity, thus the border router information corrupted are repaired using the periodically changing temporary-private address. This scheme reduces the attack disruption time, however, in the proposed scheme when a new node is joining the network for the first time a spoofing attack window is created. This process enables a malicious node to successfully bind a spoofed ID to a node to transmit false information to a neighboring node. Also, high communication cost is incurred in the proposed scheme as a result of frequent address change this is quite important for energy consideration.

2.4 Gaps from literature

In summary, it is noted that according to literature, most previous works carried out in this research area were not able to fully eradicate spoofing of legitimate nodes in the network and in some cases incurred high computational complexity as well as compromising of nodes privacy. Thus, there exists a need to develop an improved technique which has a good resilience to spoofing and without incurring a high computational cost.

CHAPTER THREE

MATERIAL AND METHODOLOGY

3.1 Introduction

In this section, the material and methodology for developing the spoofing attack mitigation scheme are presented.

3.2 Material

The following hardware and software resources were used to carry out this research:

1. Computing platform, a PC with the following hardware requirement 2.40 GHz CPU, 8G RAM, and 500 GB hard drive.
2. Cooja Simulator running on Contiki Operating System (Instant Contiki 3.0) was used for developing the project codes with a simulation area of 100 by 100 square meters.
3. A virtual machine (VM-ware workstation 14 player) was also used to run the Contiki Operating system and also to run the Cooja simulator.

3.3 Methodology

The following method was adopted for developing the proposed scheme;

1. **Development of a temporary-private IP address scheme based on node and access-router relation as follows:**

i. Initializing the 6LoWPAN network

ii. Adding nodes into the network

iii. Create a neighbor table for each node

iv. Node A broadcast a beacon to join the network

v. Building of neighbor routing table as follows;

a. Node A request for a unique ID from the access-router

b. Access-router R receives and sends unique ID to node A

c. Randomly generate disjoint congruent integer for node A

d. Generate a non-repeatable unique ID for node and router using the congruent relationship between node U_ID and router U_ID

vi. Combine a node unique ID (U_ID) with an access-router global routing prefix to construct a temporary address

vii. Use the new IP address to communicate with other nodes

2. Development of an improved temporary IPV6 address changing scheme for mitigating spoofing attack on 6LoWPAN network

i. Compute the active time for the current address

ii. Compute the lower and upper bound time required to perform a successful attack

iii. Compare the current active address generation time, current active time, TTL and ADW length to determine attack success rate

iv. Compute the total time and the ACFR

v. From ii, iii and iv select an address change time

vi. Change address periodically when a node recieves a new member identification.

3. Performance Evaluation of the Developed Improved Scheme

The following parameter was used to evaluated the performance of the developed scheme

i. Energy Consumption,

ii. Attack Disruption Window

iii. Address Change Failed Rate

3.4 Development of a temporary-private IP address scheme based on node and access-router relation.

The essence of temporary addressing is to dynamically generate multiple unique and global routable IPv6 addresses for each node participating in the network. The idea of multiple addresses is to enable node privacy and the address generation does not require address space management.

3.4.1 The Network Model

The network model utilized in this work is a multi-hop network 6LoWPAN model on an indoor setting which could either be smart homes or offices. The network constitutes of multiple interconnected single-hop 6LoWPAN and its device type includes node denoted as N. N is a resource constraint device deployed for specfic task, this usually depends on the application. A node could typically be a battery with limited storage capacity. A router denoted as R, represents a node with forwarding abilities. In the router, the congruence seed utilized by all node used in the single-hop 6LoWPAN are generated by the router and equally broadcast to all other nodes. Finally, the edge router is denoted as ER, the edge router in this network serves as a gateway interface between multiple 6LoWPANs and the broadcasting all congurence seed that is utilized by all other router in the multi-hop 6LoWPAN. Here, the ER also supplies the global routing prefix that uniquely identifies the multi-hop 6LoWPAN.

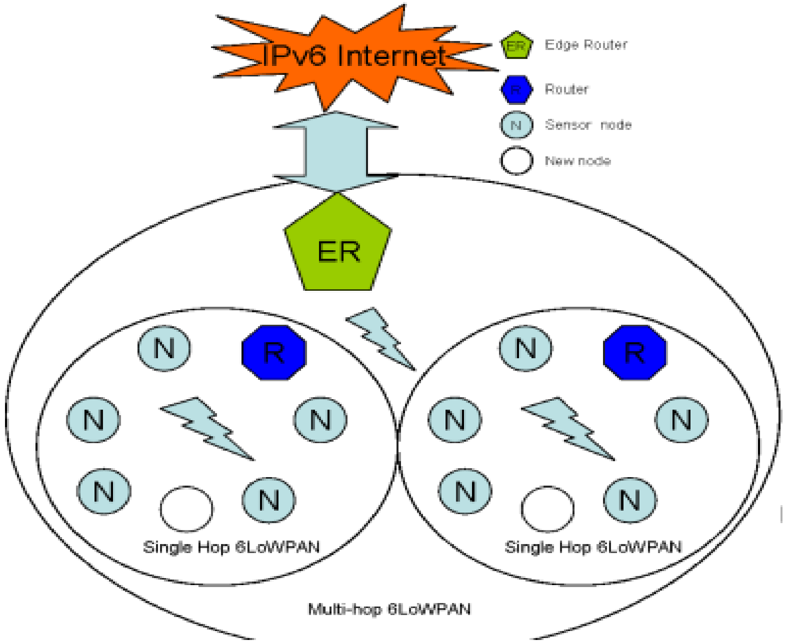

Figure 3.1: Network Model of the 6LoWPAN

Figure 3.1 shows the layout of the proposed network. Here the nodes are assumed to be mobile and static occasionally, using the node and their access router relationship, ER global routing prefix congruence seed are passed from R to N for construction of node routable IPv6 address.

3.4.2 Address Structure

In this work, each participating node in a single-hop 6LoWPAN adopts the hierarchical address structure.

The IPv6 address structure consists of two parts, namely; the global routing prefix and the interface ID. The first part which is the global routing prefix uniquely identifies the 6LoWPAN, and all nodes that belongs to the same 6LoWPAN have an identical prefix. Interface ID is the second part of the address structure, the interface ID is further divided into two part which is called router-ID and node-ID. Router-ID uniquely identifies the routers, all routers in a 6LoWPAN operates as a cluster hence all members of the LoWPAN have identical router-ID with their value equal to the router-ID of the single-hop 6LoWPAN to which they belong while the second part is the node-ID. This ID uniquely specifies nodes connected to the LoWPAN.

Table 3.1: The Restructured Address Structure of the 6LoWPAN.

Bit 128-ij	Router_ID	Node-ID
Global routing prefix	i-bit	j-bit

3.4.3 Interface Identification Generation (IID)

The Interface identification generation process consists of two processes: First, disjointed congruent integer sequences representing the Router-ID and Node-ID in the address sections are generated based on the congruence relationship between the node and access router. Secondly, the ID sequence part is randomly concatenated individually to dynamically generate unique multiple IIDs. Each router (R) at the initialization phase is assigned a unique identity called U_R-ID (m bit where k < i) which it uses to communicate with the edge router. The edge router generates and broadcasts a-bit congruence seed to all routers $(\boldsymbol{a \geq m+1})$. The congruence seed generated is larger than the largest U_R-ID contained in a multi-hop 6LoWPAN. Then, R relates the U_R-ID to the congruence seed to generate a set of disjointed congruent integer for Router_ID. The generated integers are randomly used as Router-ID. Also, each node (N) is given a unique node identity called N-U_ID (n-bit where n < j), this is used to communicate with R. R then reserve some N-U_ID to itself, this is a number greater than the N-U_ID received. R then generates and broadcasts b-bits called router congruence seed to all nodes in the single-hop 6LoWPAN, $(\boldsymbol{b \geq k+1})$.

All sequence generated by R are larger than the N-U_ID contained in a single-hop 6LoWPAN. N now relates it N-U_ID to the routers congruence seed and generates a set of disjoint congruent integer sequence as Node-ID. The router equally broadcast it U_R-ID with congruence seed generated by edge router to enable nodes generate a set of router ID. Hence, the integer sequence are randomly used as Router-ID and Node-ID to generated unique interface identification.

3.4.4 Congruent generation process for temporary address

Congruent class generation for temporary address is an un-repeatable arithmetic sequence called a residue class. Supposing, R_m is the congruence relational modulus *m* on a set of all positive integer $b \in Z$ and $m \in Z$ with the class equivalance b represent as b_m. This process is defined as given:

$$\boldsymbol{Z_m} = \{[\boldsymbol{0}]_{\boldsymbol{m}}, [\boldsymbol{1}]_{\boldsymbol{m}}, [\boldsymbol{2}]_{\boldsymbol{m}}, \dots \dots, [\boldsymbol{m-1}]_{\boldsymbol{m}}\} \tag{3.1}$$

the generation of non-repeatable sequence is governed by equation (3.1)

Where b_m is the congruence class called residue class of b mod m, while the congruence modulus quotient set Zm is defined as shown in equation (3.1)

According to equation (3.1), the modulus congruence set quotient Z_m from a partition of Z; where Z represents each LoWPAN address space and M is the modulus for a single-hop 6LoWPAN with a congruence seed by a router. The disjointed address set for each node in a 6LoWPAN is based on the relationship between 6LoWPAN address space and the Z partition which are disjointed. Based on equation (3.1), a non zero postive integer value is taken at different time interval by the node, this is used to generate a set of addresses for the nodes in a single-hop 6LoWPAN. This is done by first considering the unique identity (U_ID) router congruence sequence seed generated by the router and is the larger than all node U_ID present in a single-hop 6LoWPAN.

In a single-hop 6LoWPAN, the congruence sequence set congruence classes for each node unique identity, a maximum of 2^n - 2 node can be contained in a single-hop 6LoWPAN, this is described as defined in equations below:

$$\boldsymbol{Max_a = floor\ \left(\frac{(2^j - 1)}{minRCs}\right)} \tag{3.2}$$

$$\boldsymbol{Min_a = floor\ \left(\frac{(2^j - 1)}{maxRCs}\right)} \tag{3.3}$$

From equation (3.2), a is defined to be U_ID for node (zcsq), this equation is used to set all the addresses of the node in single-hop 6LoWPAN. Supposing that the maximum and the minimum address limit for a node is Max_a and Min_b and the routers congruence seed generated by the router is RCs Max_a and Min_b and are given as equations 3.3 and 3.4 repectively. Hence, the Maximum limit is calculated when the minimum value of RCs is choosen as:

$$\boldsymbol{MinRCs = ln - U\_ID + 2}, \tag{3.4}$$

Where $\boldsymbol{ln}$ is the largest node and U-ID is the unqiue identification number of the node. The minimum limit is calculated when the maximum RCs is choosen as:

$$\boldsymbol{Max}\_\mathrm{RCs} = \boldsymbol{2^j - 1} \tag{3.5}$$

The address generation process is defined in equation (3.2) and (3.3) respectively as sequence unique for generating address in a single-hop 6LoWPAN.

3.5 Development of an improved temporary IPv6 address changing scheme for mitigating spoofing attack on 6LoWPAN network

The improved address changing scheme for mitigating spoofing attack on a 6LoWPAN network focuses majorly on determining the Time-to-live and the Attack disruption window for a successful attack to take place in a 6LoWPAN network, with these, the lower and upper bound time for a successful attack can be reduced using the current address time of the legitmate node and the definite time required for periodic address change when a 6LoWPAN member node obtains a new set of unique identity.

3.5.1 Network Time-To-live

The time-to-live in a network is the time required for an attacker in a given network to successfully carry out and finish an attack action and eventually achieve the objective. This operation utilizes received DIO message, sending Echo-Request message, waiting for Echo-Response message and transmitting spoofed DIO message as 6LoWPAN events to efficiently provide resilent to spoofing in a 6LoWPAN. Using each of these events, the time-to-live is obtained and by analysing the individual event, spoofing attack can be determined, thus the time an attacker node will need to learn a victim's IPv6 address and perform an attack is shorted. The analysis of each event is based on equation (3.6 to 3.9)

Event 1: The time taken for a DIO message to be received

TTL for received DIO message

$$\boldsymbol{TTL_{DIO} = T_{tx} = \frac{s_{dio}}{250kbps}} \tag{3.6}$$

Event 2 : The time taken to send one Echo Request packet

TTL for sending $\mathbf{Echo_{Request}}$message

$$\boldsymbol{TTL_{ERQ} = (noEchorq * t_{min}) + turnTime} \tag{3.7a}$$

$$\boldsymbol{= (noEchorq * t_{max}) + turnTime} \tag{3.7b}$$

Event 3: The time taken to wait for an Echo Response packet

Waiting $\mathbf{Echo_{Response}}$message

$$TTL_{ERP} = (noEchors * \ t_{min}) + turnTime \quad (3.8a)$$

$$= (noEchors * \ t_{max}) + turnTime \quad (3.8b)$$

Event 4: The time for transmitting a spoofed DIO message

Transmitting spoofed DIO message

$$TTL_S = (noDio * \ t_{min}) + turnTime \quad (3.9a)$$

$$= (noDio * \ t_{max}) + turnTime \quad (3.9b)$$

3.5.2 Attack Disruption Window

The attack disruption window (ADW) is the time in which a successful spoofing attempt can disrupt the network. During this period an attacker node can successfully spoof the IPv6 address of a legitimate node and use the address to launch a series of attack in the likes of DoS, Man-in-the-middle attack, monitoring of victim activities to gain valuable sensitive information by learning and tracking the IPv6 address of the victim node in the network. This often occurs when the defence mechansim against spoofing fails. Spoofing attempts on failure of the defence mechansim can cause enormous damage to the entire network until such threats are detected and corrected. This is considering the fact that the IoT network is an insecure wireless medium where any node can sneak in and out of the network and gain considerable access to the node IPv6 address for futher attack. In the work, spoofing is mitigated in the network by minimizing the ADW length, this process is ensured by utilizing the total time established for a node to successfully change its address in the address change cycle.

$$\text{ADW} = \text{Total time} - \text{TTL} \tag{3.10}$$

This is done with the aid of equation (3.9) where the total time is considered to be the time taken until spoofing attempt is detected and corrected while the TTL is the time required for a node to live in a particular event.

In making the network resilient to spoofing the required ADW time length is decreased, thereby utilizing a periodic change technique where the IP addresses of each node are deprecated and new addresses are periodically assigned to the nodes. When the nodes assume a new address after some time before the next periodic cycle for address change, the spoofed address becomes invalid for further communication, then the ADW length becomes the current time the IPv6 address of the legitimate node is active. Once the address deprecation occurs, no disruption can be caused by the attacker node using the spoofed address.

$$\text{ADW} = \text{C\_addr\_active\_time} - \text{TTL} \tag{3.11}$$

Hence equation (3.10) is redefined as given in equation (3.11), where the ADW becomes approximately equal to the active time of the current address. Thus as the current address active time reduces, the ADW also reduces, therefore, the lesser time it takes for the C_addr_active_time to change, the more frequent address change is required. Hence considering the resource constraint nature of the network, managing frequent address changes add significant overhead on the node. In dealing with this there was a trade-off between address change rate and ADW length, the address change rate is increased and the ADW length was reduced in order to manage node resources and maintain efficient network performance.

3.5.3 Performance evaluation of the developed scheme

This section accesses and evaluates the performance of the developed improved scheme using different measures. In analyzing the developed improved spoofing mitigation scheme based on temporary address scheme against spoofing of IPv6 addresses in a 6LoWPAN network, performance evaluation metrics such as Attack Disruption Window, Energy Consumption, and Periodic Address Change were utilized.

i. Attack disruption window: This is the time during which a successful spoofing attempt can disrupt the network, it is define as shown in equation 3.11 where C_addr_active_time is the current address active time in the network and TTL is the time to live of a given temporary address generated by the developed scheme.

ii. Energy consumption: The energy consumption measures the amount of energy consumed by each node in the network, that is the total amount of energy consumed over the total simulation time. This includes all possible energy spent in address change, transmission, reception, idle and sleep state. This is defined as shown

$$\boldsymbol{e_r} = \sum_{s=1}^{Smax} \boldsymbol{node\_time} \times \boldsymbol{e_s} \qquad (3.12)$$

Where $\boldsymbol{e_s}$ is the energy spent by a node in a state per second state, s is one of CPU, low power mode (LPM) or deep low power mode (DLPM). While $\boldsymbol{node\_time}$ is the number of seconds a node spent at state (s).

iii. Periodic address change fail: This is number of temporary address fail change in the 6LoWPAN network used to reduce the attack disruption effect on the network.

These parameters (metrics) measures the resilient ability of the spoofing mitigation scheme in resisting spoofing attack on 6LoWPAN network and the energy consumption rate at different time interval considering that it is an energy constrained network.

Performance of the developed spoofing mitigation scheme is compared with state-of-the-art scheme using the stated metrics used to evaluate the performance of the scheme, comparison report is given in detail in the result and discussion section chapter 4.

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 Introduction

In this section, the results obtained from the developed improved spoofing mitigation scheme for 6LoWPAN are presented and discussed. In this research, the performance of the developed improved scheme was tested and evaluated to determine the resilience ability of the scheme to spoofing attempt of IPv6 address in 6LoWPAN. A number of scenerios were considered namely: a normal simulation scenario without attack, simulation scenario with spoofing attempt and simulation scenario showing resilience to spoofing attempt. In the simulation scenario, an attack is performed in the simulated network using Contiki operating system and Cooja simulator. In the simulation, nodes are deployed in a random manner using the 100 by 100 square meter area. The initialisation of each node defers from each other by a small amount of time, (some fractions of seconds) before starting the communication to avoid collision.

4.2 Simulated Network Scenario without Attack

Presented in this section is the simulated network scenarios without attack, as shown in Figure 4.1. The simulated network is based on a single-hop network. This choice is because, attackers often try to spoof 6LoWPAN node IP addresses that fall within their communication range that is its radio range by listening to radio messages. However, attacks initiated remotely are easily mitigated because the solution implemented in the developed scheme is host initiated as each node in the network becomes a part of DODAG, whose root is the edge router. Furthermore, in the simulated network each node contains a global routable address whose prefix is distributed using RPL border router running on node 1. In demonstrating the resilience against spoofing attempts the developed scheme uses the developed temporary addressing scheme, where a number of nodes are operating using the addressing mechanism to generate their temporary addresses with the services of the edge routers and other routers. Nodes connected to the network obtain their temporary addresses from the edge router when the router has initiated the process, these addresses are used for communication and change periodically whenever the edge router initiate the periodic change in order to effectively mitigate IP address spoofing attempt on the leaf nodes/normal node (local devices) connected to the network and are within the communication range of an attacker's nodes. The developed temporary privacy addressing scheme requires an edge router (border router) for disseminating routable prefix and other routers in each single-hop 6LoWPAN used for distribution of address configuration parameter to other nodes in the network.

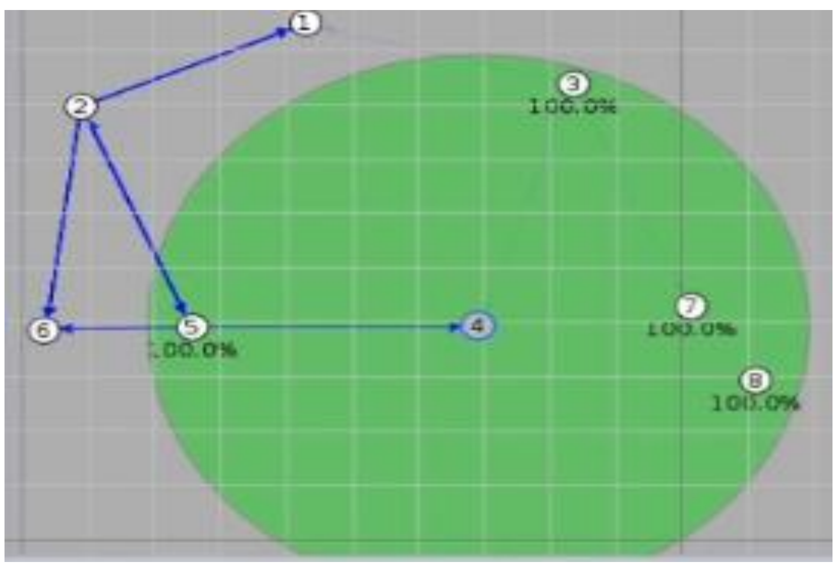

Figure 4.1: Simulated Network Scenario without Attack

The simulated network in Figure 4.1 contains seven legitimate nodes and one attacker node. In this Figure, node 1 runs as Contiki's border router/edge router (modified to print layer two addresses along with IP addresses of the neighbours), it runs addressing protocol and distribute prefix for a global routable permanent address for the temporary addresses and is also a root of DODAG. Node 2 and Node 3 run as normal routers for broadcasting and disseminating the remaining address configuration parameters whereas the remaining nodes 4, 5, 6 and 7 are normal nodes which can be victim nodes and node 8 is the attacker node which runs attacker codes and forward/sends spoofed RPL control messages.

4.3 Simulated Network Scenario Under Attack

In the simulated network presented in Figure 4.2 consisting of 8 nodes each, represented as 1-to-8. Node 1 is acting in the capacity of a border router/edge router, node 2 and 3 are acting as routers, the remaining node 4,5,6 and 7 are normal node while node 8 is a malicious insider node (attacker node).

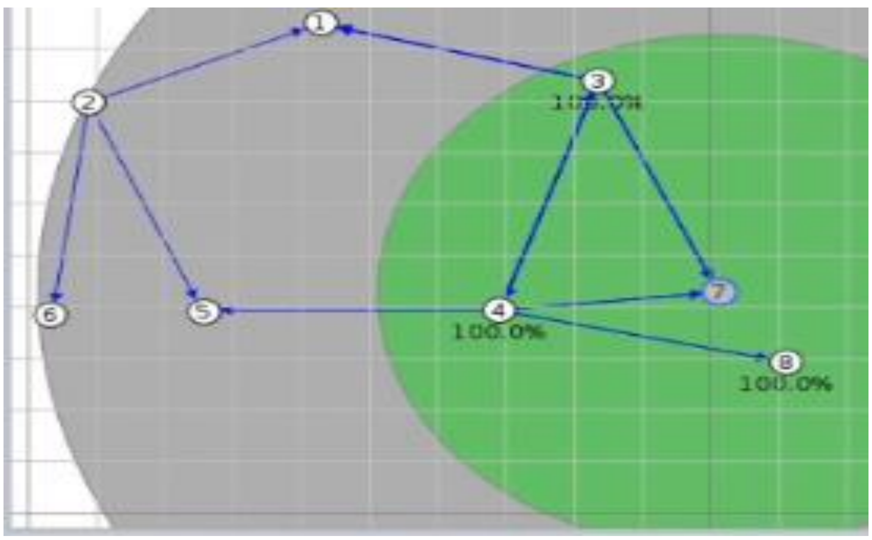

Figure 4.2: Simulated Network Scenario under Attack

In figure 4.2, the green region indicates a single-hop 6LoWPAN communication range with node 3 serving as a router. Also, as can be seen in figure 4.2, node 8 can successfully spoof the IPv6 addresses of node 4 and 7 because they are within the same communication radio and register its entry with the host router node 3. Furthermore, in the simulated network of Figure 4.2 node 8 successfully spoofed the identity of node 4 and binds its own MAC address to the spoofed IPv6 address of the victim node. Thus, as a result of the binding, the victim node is unable to register itself within the edge router leading to denial of service and breach of confidentiality because all network traffic meant for node 4 are redirected to node 8.

4.4 Simulated Network Scenario with Resilient to Spoofing

Presented in the simulated network in Figure 4.3 is the introduction of the resilience scheme based on temporary addressing.

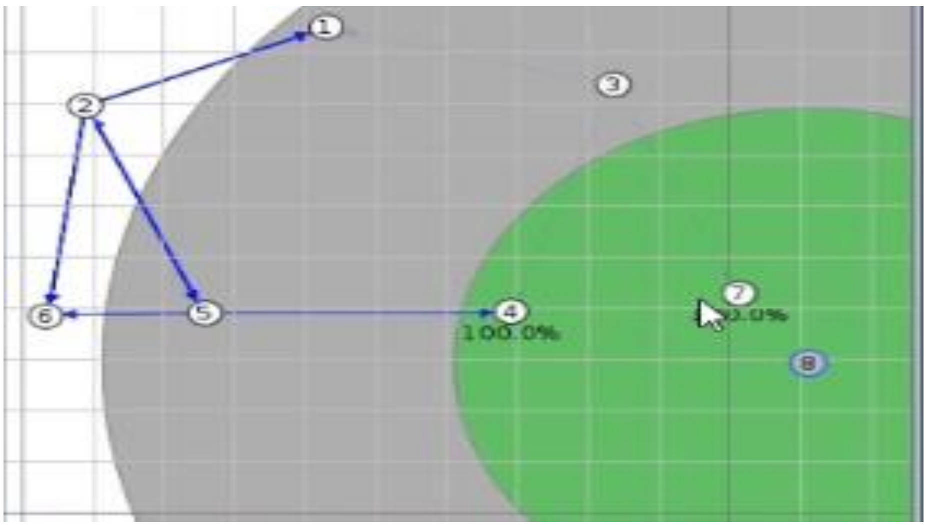

Figure 4.3: Simulated Network Scenario showing Resilience to Spoofing

In Figure 4.3 when the temporary addresses are used, and periodic address change has been initiated by the border router, the victim node which is node 4 now assumes a new IPv6 address and denounces the old address. Hence the process allows node 4 to register itself with the edge router and resumes normal communication. The new IPv6 address of node 4 is bound to its own MAC address while the MAC address of node 8 (attacker node) is bound to node 4 old address which is now denounced leading to the removal of node 8 (attacker node) from the network. Hence, the routing information of node 4 is being repaired as shown in Figure 4.3 as node 4 has started communicating using the new changed address. The address which the attacker has spoofed is no longer valid therefore reducing the effectiveness of network disruption in the network.

4.5 Periodic address change evaluation

The periodicity of address change rate in a 6LoWPAN is a measure used to reduce the attack disruption effect on the network. When nodes frequently change their addresses, it gives little amount of time for an attacker node to successfully launch a spoofing attack and cause disruption to the network. Also, when an attack has been successfully created and launched, the periodic address change reduces the effect due to nodes frequent periodic change of address, using this, the victim node would change and assume a new identity and their old identity are denounced rendering them invalid thereby reducing the attack disruption time. Presented in Table 4.1 and Figure 4.5 is the evaluation results of the periodic address change for different time interval ranging from 30s to 300s for each individual node used during the simulation.

Table 4.1 Periodic Address Change Time in Seconds (s)

Timer_Row (s)	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7
30	166	119	119	54	54	51	51
60	77	75	75	47	46	49	59
90	56	55	55	45	45	45	45
120	41	41	41	37	37	39	39
150	32	32	32	32	32	31	31
180	29	28	28	28	28	28	28
210	26	26	26	26	26	25	25
240	24	24	24	23	23	24	24
270	21	21	21	21	21	21	21
300	20	20	20	20	20	20	20

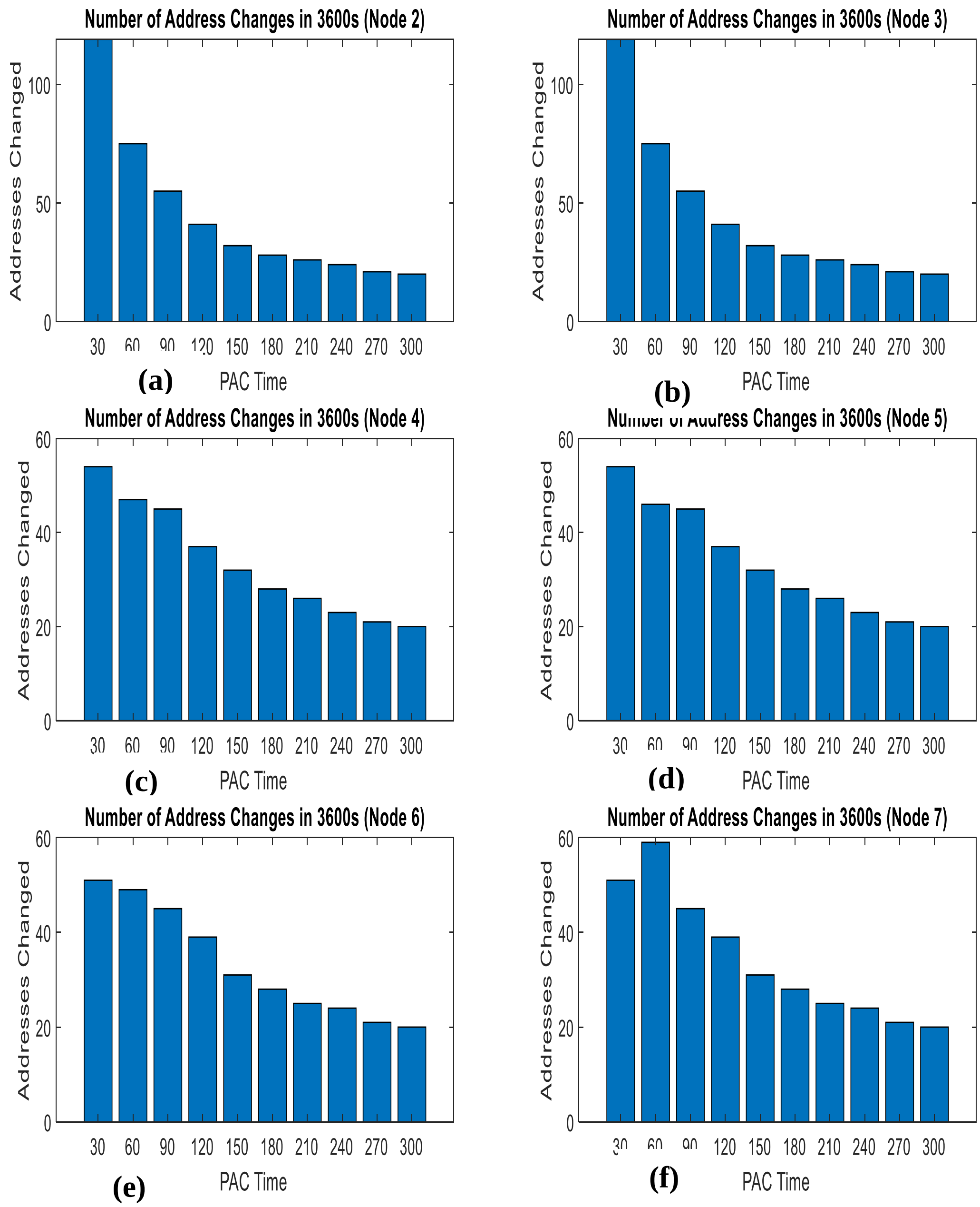

Figure 4.4(a-f) Shows Evaluation of Address Changed Rate

As shown in Figure 4.4, (a-f) at 30s the frequency at which nodes changes their address is high, while at 300s the frequency of address change rate is low. Therefore, at higher address change the possibility of spoofing attack is very low but it often results to high energy consumption and the possibility of nodes not being able to transmit packet. As the frequency of address change reduces, the possibility of nodes being able to communicate is increased for different nodes and different time intervals. However, it can be deduced that between the period of 150s to 180s the periodic address change is minimal. Nodes can effectively change their addresses without much burden on the network and at the same time reducing the disruption effect of attack on the networks.

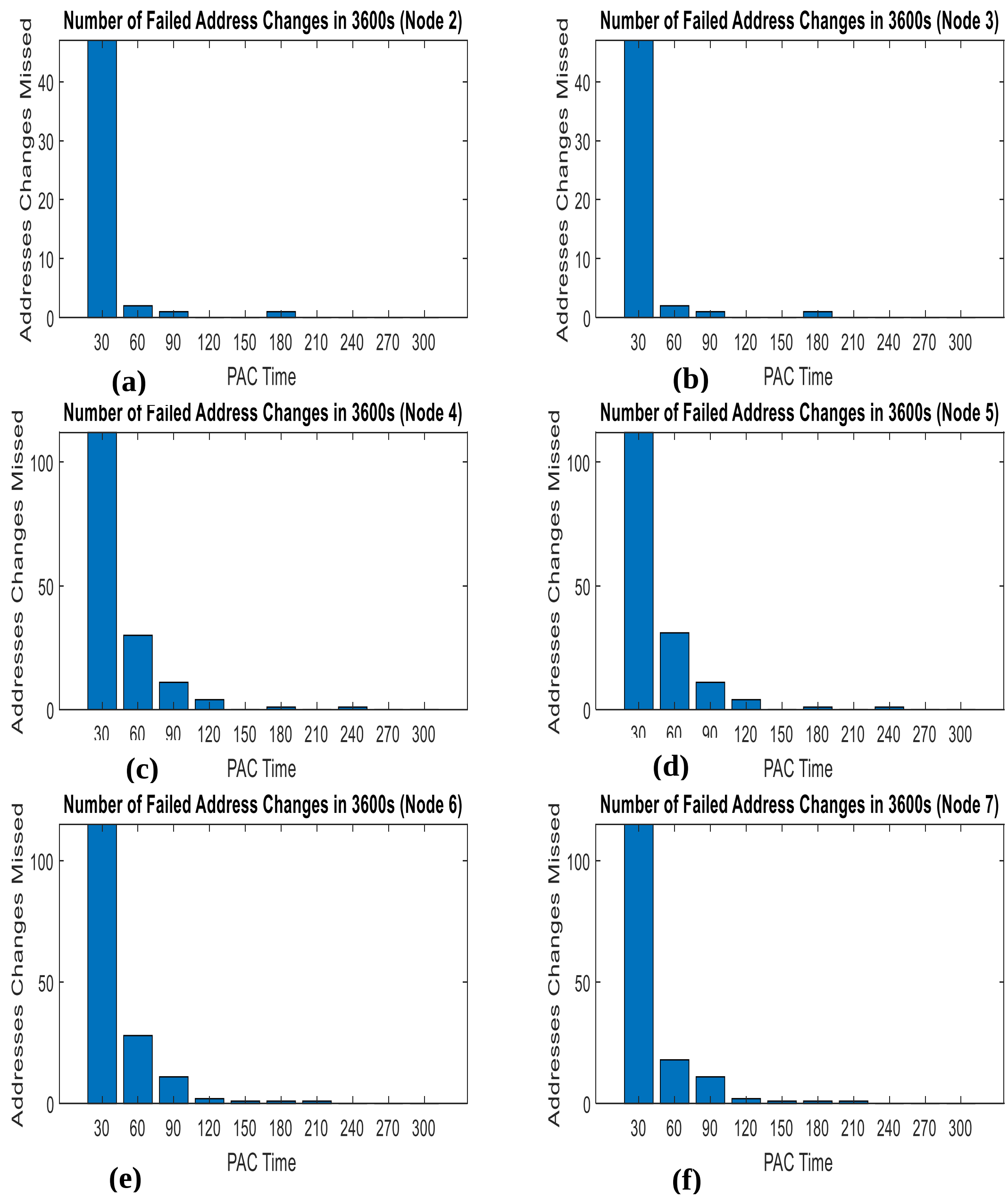

Figure 4.5 (a-f) Evaluation of Failed Address Changed Rate

In Figure 4.5(a-f) the address change failed rate is reported, as can be seen above. At 30s for each individual node in the network there is a high level of address change failed rate, this failure in nodes changing their address is as a result of the fast time interval required by the nodes to change their addresses. Whenever the border router initiates a periodic address change, most nodes connected to the network are unable to change their address before the next round of address change but as the address change time increases the window of address change failed decreases linearly as presented in Figure 4.5 giving the nodes enough time to change their addresses and start sending and receiving packets on the network. Furthermore, it is observed that from 150s to 300s most nodes can efficiently change their addresses without much failure while in some cases there is minimal address change failed rate. Therefore, this time range is considered more suitable for periodic address change given that there exists a small window for address change failure.

4.6 Attack Disruption Window Evaluation

The attack disruption window (ADW) evaluation is performed to determine the time-to-live window period that an attacker node can successfully cause disruption to the network before it is detected and removed from the network based on the frequency of periodic address and the current address active time of each node. As given in chapter three section 3.5.3 the ADW is obtained by substracting the current address active time with the total time to live (TTL) defined as equation (3.14). Presented in Table 4.2 is the current address time while Figure 4.6 is the ADW results obtained for the edge router and a normal node connected to the network. For normal node ADWs from 30s to about 120s gives a mimiunm attack disruption window while for edge routers from 150s and above, the ADW increases linearly as the frequency of address changes. However, at a lower periodic change interval ADW showed minimum attack disruption while a longer disruption is shown at a longer periodic address change interval in the network. Therefore it can be said that normal nodes achieved a minimum ADW at PAC interval between 30s to 120s while the edge router achieved mininum ADW between 30s to 90s respectively.

Table 4.2: Attack Disruption Window Evaluation in seconds(s)

Timer/Row(s)	**Node 2**	**Node 3**	**Node 4**	**Node 5**	**Node 6**	**Node 7**
30	10.972	10.972	8.532	8.868	8.868	8.532
60	32.132	32.132	21.059	18.28	18.28	21.059
90	49.445	49.445	33.546	33.202	33.202	33.546
120	72.213	72.213	47.651	54.724	54.724	47.651
150	92.939	92.939	75.264	73.752	73.752	75.264
180	109.019	109.019	79.541	89.813	89.813	79.541
210	124.894	124.894	103.383	91.997	91.997	103.383
240	128.794	128.794	109.967	105.373	105.373	109.967
270	152.596	152.596	132.572	130.357	130.357	132.572
300	161.423	161.423	133.629	140.467	140.467	133.629

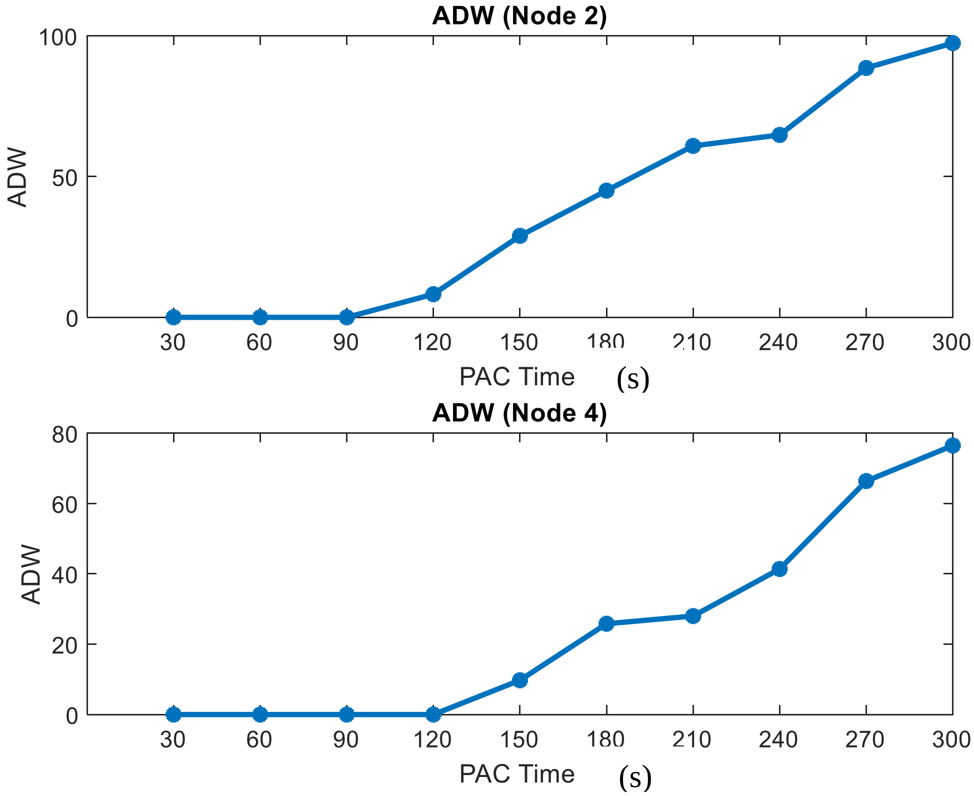

Figure 4.6: Attack Disruption Evaluation

4.7 Energy Consumption Evaluation

The energy consumption evaluation measures the amount of energy consumed by each node in the network, that is the total amount of energy consumed over the total simulation time. This includes all possible energy spent in address change, transmission, reception, idle and sleep state. As shown in Table 4.3 and Figure 4.7 the energy consumption for the routers and normal nodes are presented here, as can be seen from in Figures 4.7(a, b), where (a) represent ennergy consumption for normal node and (b) represents energy consumption for edge routers. The amount of energy consumed by the routers at 30s, is extremely high and starts decreasing up to about 120s from about 150s, minimal amount of energy is consumed. Similarly for normal nodes, the energy consumption is equally high up to about 90s before a decrease in energy consumption seen at about 120s the energy consumption rate of each node is minimal. This is because the nodes are more involved in frequent address changing leading to the consumption of much energy. The high rate of energy consumption at this point is not efficient for an energy constraint network however, at this interval the developed improved scheme exhibit better performance in terms of periodic address change and ADW at the expense of the network life span. In order to maintain an improved network peformance and extend the network life span considering the resource constraint nature of the network, the signifcant load/overhead caused when the developed scheme achieved minimum attack disruption both at the normal node level and the routers, the energy is always higher which is not good for a resource constraint network to this end, a tradeoff between PAC and ADW length is established this is due to the fact that as PAC increases ADW length is reduced.

Table 4.3 Energy Consumption Evaluation

Timer_Row (s)	**Node's Energy Charge in (mAh)**					
	Node 2	**Node 3**	**Node 4**	**Node 5**	**Node 6**	**Node 7**
30	0.162	0.162	0.059	0.059	0.062	0.062
60	0.151	0.159	0.059	0.059	0.062	0.062
90	0.154	0.159	0.059	0.059	0.062	0.062
120	0.145	0.159	0.056	0.056	0.062	0.059
150	0.140	0.156	0.056	0.056	0.059	0.059
180	0.140	0.156	0.056	0.056	0.062	0.059
210	0.134	0.156	0.056	0.056	0.059	0.059
240	0.134	0.156	0.056	0.053	0.059	0.059
270	0.134	0.156	0.056	0.053	0.057	0.059
300	0.134	0.154	0.056	0.053	0.057	0.059

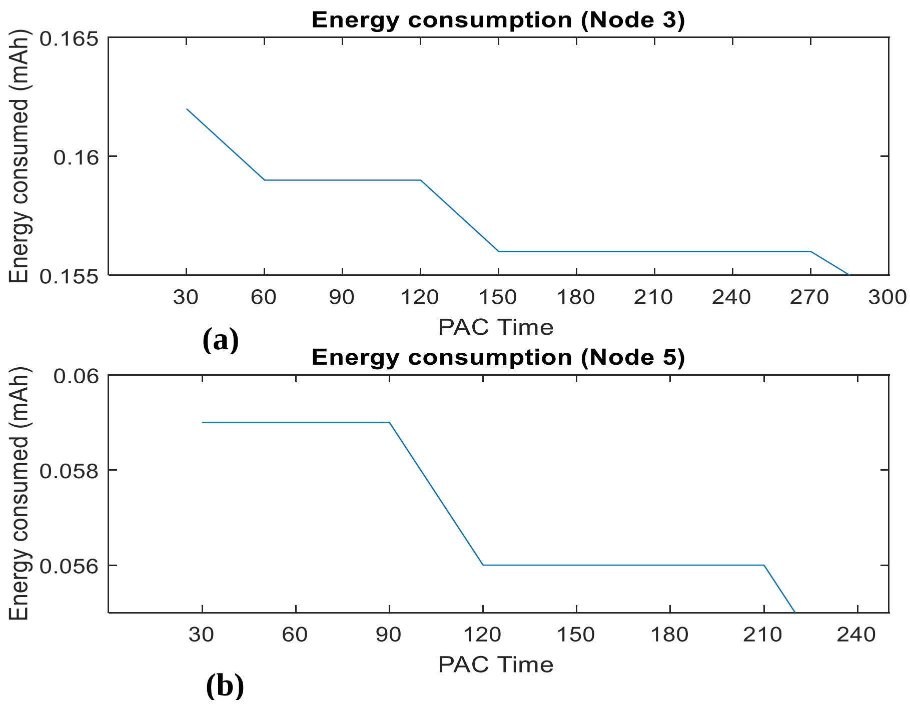

Figure 4.7: Energy Consumption For Normal Node and Edge Router

4.8 Performance Comparison

Performance evaluation results of the developed improved scheme is presented in Table 4.5 (a) and (b), using the following parameters; Address Disruption Window (ADW), Energy Consumption and Address Fail Change Rate (AFCR). The evaluation of both scheme performance was carried out under the same computing enviroment. The obtained results as shown in Table 4.5, showed that the developed scheme achieved significant improvement as compared to the scheme of Mavani and Asawa 2019, with an average attack disruption period of 33.35, 51.18 and 74.50 obtained for 90s, 120s and 150s respectively for normal nodes, while for routers, the developed scheme achieved an average ADW of 49.44, 72.21 and 92.93 for 90s, 120s, and 150s, while providing resilience to spoofing attack. Also an averge energy of 0.158, 0.152 and 0.014 is consumed under by router at 90s 120s and 150s respectively while for normal node, the average energy consumed are 0.060, 0.058 and 0.057. Finally, the number for address fail change rate was also significantly improved as compared to the scheme of Mavani and Asawa 2019 specifically at 120s and 150s respectively both the normal nodes and the routers were able to successfully change their addresses and forward/establish communication with their neighboring nodes. Therefore this time is considered in this work as the best time interval because all participating node changed addresses successful and communicated before the next periodic address cycle therefore, it can be concluded that the developed improved spoofing mitigation scheme **achieved an average marginal attack disruption window of 13% above the previous scheme, while the percentage improvement for energy was 66% and the address failure change rate was also significantly improved by 82% when compared with previous scheme and** can sucessfully mitigate spoofing attack on a 6LoWPAN network.

Table 4.5 (a) Performance Comparsion for Normal Nodes

Scheme	**Mavani 2019**			**Developed scheme**		
	Normal node time interval(s)			**Normal node time interval(s)**		
	90	**120**	**150**	**90**	**120**	**150**
ADW	34.38	53.12	74.93	33.35	51.18	74.50
Energy consumption	0.158	0.152	0.014	0.060	0.058	0.057
AFCR	14	8	6	10	0	0

Table 4.5 (b) Performance Comparson for Routers

Scheme	Mavani 2019			Developed scheme		
	Router time interval(s)			Router time interval(s)		
	90	120	150	90	120	150
ADW	50.44	73.44	92.96	49.44	72.21	92.93
Energy consumption	0.162	0.159	0.151	0.158	0.152	0.014
AFCR	10	8	6	2	0	0

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Summary

This research has developed an improved spoofing mitigation scheme for a 6LoWPAN network. Significant improvement was achieved by the developed improved scheme as a result of making use of the dynamic periodic address changing process and access-router relationship in generating the temporary addressing used by the nodes. The dynamic periodic address change ensured unpredictability of address changing period by an attacker evesdropping the network for periodic address changes. Hence a tradeoff was established between periodic address change (PAC) and the attack disruption window (ADW) because as PAC increases, ADW decreases and at a lower disruption window with a frequent address change rate the energy consumption for normal nodes and the routers tend to be high which is not appropriate in a resource constraint environment. Hence, the developed scheme can efficiently provide resilience to the 6LoWPAN using the IP address changing scheme based on node and edge router relationship.

5.2 Conclusion

This research has developed an improved spoofing mitigation scheme for 6LoWPAN based on a temporary-private IPv6 addresss and access-router relationship with an efficient periodic address changing scheme. This approach provides a spoofing resilence on a 6LoWPAN network by nodes periodically changing their addresses to assume a new identity and denounce the old identity in the event of a successful spoofing attempt. Furthermore, the developed scheme performance was evaluated and compared with the scheme of Mavani and Asawa 2019, using frequency of periodic address change, Attack disruption window and energy consumption. The results obtained showed significant improvement in terms of ADW, periodic address change rate and energy consumption compared to the scheme of Mavani and Asawa 2019 under the same working conditions.

5.3 Significant Contribution to Knowledge

The significant contributions of this research work are as follows:

1. The developed improved spoofing mitigation scheme based on node and access router relationship achieved a marginal attack disruption window of about 13% as compared with the scheme of Mavani & Asawa 2019, therefore providing resilience to spoofing attack.
2. The proposed improved spoofing mitigation scheme equally achived signicant improvement with regard to reducing the number for address fail change rate at 120s and 150s respectively. Both normal node and router were able to successfully change their address and forward/establish communication with their neigboring nodes.
3. An improved spoofing mitigation scheme was developed with about 66% improvement in energy consumption. Were the averge energy consumped for routers was recorded as 0.158, 0.152 and 0.014 all measeured in (mAh) at 90s 120s and 150s respectively while for normal node, the average energy consumed was 0.060, 0.058 and 0.057 in (mAh). While the address failure change rate was also significantly improved by about 82% as compared to Mavani & Asawa 2019 under the same working condition.

5.4 Recommendation

The following areas are recommended for futherwork

1. A session break should be integrated to the network such that when there is address change at node level all incoming packet to the node will assume the new address to avoid spoofing.

2. Address reclamation should be integrated to the proposed scheme to enhance the security of the network

REFERENCE

Ahmed, A. S. A. M. S., Hassan, R., & Othman, N. E. (2017). IPv6 neighbor discovery protocol specifications, threats and countermeasures: a survey. *IEEE Access, 5*, 18187-18210.

Ee, G. K., Ng, C. K., Noordin, N. K., & Ali, B. M. (2010). A review of 6LoWPAN routing protocols. *Proceedings of the Asia-Pacific Advanced Network, 30*, 71-81.

Hossain, M., Karim, Y., & Hasan, R. (2018, March). Secupan: A security scheme to mitigate fragmentation-based network attacks in 6lowpan. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy* (pp. 307-318).

Hummen, R., Hiller, J., Wirtz, H., Henze, M., Shafagh, H., & Wehrle, K. (2013, April). 6LoWPAN fragmentation attacks and mitigation mechanisms. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks* (pp. 55-66).

Kumar, V., & Tiwari, S. (2012). Routing in IPv6 over low-power wireless personal area networks (6LoWPAN): A survey. *Journal of Computer Networks and Communications,(pp. 2-12)*.

Luo, B., Tang, S., & Sun, Z. (2015, August). Research of neighbor discovery for IPv6 over low-power wireless personal area networks. In *2015 11th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE)* (pp. 233-238). IEEE.

Mathew, R. (2019). Literature Survey on IPv6 over low power personal area networks.

Mavani, M., & Asawa, K. (2017, a). Experimental study of IP spoofing attack in 6LoWPAN network. In *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence* (pp. 445-449). IEEE.

Mavani, M., & Asawa, K. (2017b). Modeling and analyses of IP spoofing attack in 6LoWPAN network. *computers & security, 70*, 95-110.

Mavani, M., & Asawa, K. (2018). Privacy enabled disjoint and dynamic address auto-configuration protocol for 6Lowpan. *Ad Hoc Networks*, *79*, 72-86.

Mavani, M., & Asawa, K. (2020). Resilient against spoofing in 6LoWPAN networks by temporary-private IPv6 addresses. *Peer-to-Peer Networking and Applications, 13*(1), 333-347.

Minoli, D. (2013). *Building the internet of things with IPv6 and MIPv6: the evolving world of M2m communications*: John Wiley & Sons.

Misra, S., & Goswami, S. (2017). *Network Routing*: Wiley Online Library.

Mulligan, G. (2007, June). The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors* (pp. 78-82).

Oliveira, L. M. L., Rodrigues, J. J., de Sousa, A. F., & Denisov, V. M. (2016). Network admission control solution for 6LoWPAN networks based on symmetric key mechanisms. *IEEE Transactions on Industrial Informatics, 12*(6), 2186-2195.

Ozturk, Y., & Nagarnaik, V. (2011). A scalable distributed dynamic address allocation protocol for ad-hoc networks. *Wireless Networks, 17*(2), 357-370.

Rai, K. K., & Asawa, K. (2017, August). Impact analysis of rank attack with spoofed IP on routing in 6LoWPAN network. In *2017 Tenth International Conference on Contemporary Computing (IC3)* (pp. 1-5). IEEE.

Shelby, Z., & Bormann, C. (2011). *6LoWPAN: The wireless embedded Internet* (Vol. 43): John Wiley & Sons.

Simpson, W., Narten, D. T., Nordmark, E., & Soliman, H. (2007). Neighbor discovery for IP version 6 (IPv6). *Internet Engineering Task Force, RFC2461 (pp.1-98)*.

Wang, X., & Mu, Y. (2015). Addressing and privacy support for 6LoWPAN. *IEEE Sensors Journal*, *15*(9), 5193-5201.

Wang, X., & Mu, Y. (2017). Communication security and privacy support in 6LoWPAN. *Journal of Information Security and Applications*, *34*, 108-119.

Narten, T., Nordmark, E., Simpson, W., Soliman, H.: Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard) (2007). http://www.ietf.org/rfc/rfc4861.txt retrived on 20 march 2019

R. Mendão Silva. Wireless sensor networks – node discovery and mobility. http://rtcm. inescn.pt/fileadmin/rtcm/Workshop_23_Jun_09/s2p2.pdf, retrived on 18 June 2019.

ss

Appendice

```
###################### pednet-routing.c ######################
#include "contiki.h"
#include "net/packetbuf.h"
#include "net/netstack.h"
#include "net/pednet/pednet.h"
#include "net/pednet/pednet-conf.h"
#include "net/pednet/pednet-const.h"
#include "net/pednet/pednet-types.h"
#include "net/pednet/pednet-timers.h"
#include "net/pednet/pednet-routing.h"
//#include "net/pednet/pednet-er.h"
#include "net/ipv6/uip.h"
#include "net/ipv6/uip-ds6.h"
#include "net/ipv6/uipbuf.h"
#include "lib/random.h"
#include "sys/node-id.h"
//#include "services/deployment/deployment.h"
/* Log configuration */
#include "sys/log.h"
#define LOG_MODULE "PED"
#define LOG_LEVEL LOG_LEVEL_PED

//static uint8_t *packetbuf_ptr;

static uint8_t ped_as_er = PED_ER;
static uint8_t ped_as_r = PED_R;
static uint8_t ped_as_leaf = PED_LEAF;
uint8_t ped_rand_num;


LIST(input_handler_list);

void ped_input_callback(const void *data, uint16_t len,
  const linkaddr_t *src, const linkaddr_t *dest);
/*---------------------------------------------------------------------------*/
static pednet_input_handler_t *
input_handler_lookup(uint8_t type, uint8_t icode)
{
  pednet_input_handler_t *handler = NULL;
  for(handler = list_head(input_handler_list);
      handler != NULL;
      handler = list_item_next(handler)) {
    if(handler->type == type &&
       (handler->icode == icode ||
        handler->icode == PED_HANDLER_CODE_ANY)) {
      return handler;
    }
  }
  return NULL;
}
void update_random_num()
{
  ped_rand_num = random_rand() % 255;
}
```

```
void
set_ip_from_prefix(uip_ipaddr_t *ipaddr, uip_ds6_prefix_t *prefix)
{
  memset(ipaddr, 0, sizeof(uip_ipaddr_t));
  memcpy(ipaddr, &prefix->ipaddr, (prefix->length + 7) / 8);
  uip_ds6_set_addr_iid(ipaddr, &uip_lladdr);
}

int
ped_node_is_er(void){

  return ped_as_er;

}

//static
int
ped_node_is_r(void){
  //return PED_R ? 1 : 0;

  return ped_as_r;
}

//static
int
ped_node_is_leaf(void){
  return ped_as_leaf;
}

int
ped_get_er_ipaddr(uip_ipaddr_t *ipaddr)
{
  return 1;
}

const uip_ipaddr_t *
ped_get_global_address(void)
{
  LOG_DBG("gET GLOBAL ADDR");


  return NULL;
}

/*****************************************************************************/
uint8_t
ped_input(uint8_t type, uint8_t icode)
{
  pednet_input_handler_t *handler = input_handler_lookup(type, icode);
  LOG_DBG("PED input reached \n");

  if(handler == NULL) {
    return PED_INPUT_ERROR;
  }

  if(handler->handler == NULL) {
```

```
    return PED_INPUT_ERROR;
  }


  handler->handler();
  return PED_INPUT_MSG;
}
/*---------------------------------------------------------------------------*/

void test_send(){
  // LOG_INFO("OOOKKK..... \n");
  ped_send(NULL, PED_INPUT_MSG, PED_CODE_NODE_NEW, 0);
}

uint8_t ped_addr_in_class(const linkaddr_t *linkaddr)
{
  // linkaddr_t *addr_to_ckeck;
  // addr_to_ckeck = packetbuf_addr(PACKETBUF_ADDR_RECEIVER);
  // uint8_t ped_node_id = linkaddr_node_addr.u8[LINKADDR_SIZE - 1]
  //          + (linkaddr_node_addr.u8[LINKADDR_SIZE - 2] << 8);

  return 0;


}

void
ped_send(const uip_ipaddr_t *dest, int type, int code, int payload_len)
{
  uip_lladdr_t *linkaddr;

  if(payload_len == 0){
    LOG_DBG("No Payload. Exit \n");
    return;
  }else if(payload_len > PED_PAYLOAD_SIZE)
  {
    LOG_DBG("Too large payload. Exit \n");
  }

  if(uip_is_addr_mcast(&PED_PACKET_BUF->destipaddr)) {
    linkaddr = NULL;
  }
  else
  {
    uip_lladdr_t lladdr;
    uip_ds6_set_lladdr_from_iid(&lladdr, &PED_PACKET_BUF->destipaddr);
    linkaddr = &lladdr;
  }
  if(node_id == 1){
    uint16_t tested = 0;

    memcpy(&tested, PED_PACKET_PAYLOAD, sizeof(uint16_t));

    LOG_INFO("CHECK CSER %u", tested);
  }
```

```
  pednet_buf = (uint8_t *)PED_PACKET_BUF;
  pednet_len = PED_HDR_LEN + payload_len;


  NETSTACK_NETWORK1.output((const linkaddr_t *)linkaddr);


}

void
generate_cser(){
  uint8_t max = curr_pednet_instance.max_node_num;
  LOG_DBG("MAX ROUTER: %u ", max);

  curr_pednet_instance.cser = max + (random_rand() % (255 - max));
  LOG_DBG("TIMER CSER: %u \n", curr_pednet_instance.cser);

}

void
generate_csr(){
  uint8_t max = curr_pednet_instance.max_node_num + 2;
  LOG_DBG("MAX NODE: %u ", max);

  curr_pednet_instance.csr = max + (random_rand() % (255 - max));
  LOG_DBG("TIMER CSR: %u \n", curr_pednet_instance.csr);

}

/*---------------------------------------------------------------------------*/
void
ped_set_global_address(uip_ipaddr_t *prefix, uip_ipaddr_t *iid)
{
  static uip_ipaddr_t root_ipaddr;
  const uip_ipaddr_t *default_prefix;
  int i;
  uint8_t state;

  default_prefix = uip_ds6_default_prefix();

  /* Assign a unique local address (RFC4193,
     */
  if(prefix == NULL) {
    uip_ip6addr_copy(&root_ipaddr, default_prefix);
  } else {
    memcpy(&root_ipaddr, prefix, 8);
  }
  if(iid == NULL) {
    uip_ds6_set_addr_iid(&root_ipaddr, &uip_lladdr);
  } else {
    memcpy(((uint8_t*)&root_ipaddr) + 8, ((uint8_t*)iid) + 8, 8);
  }

  uip_ds6_addr_add(&root_ipaddr, 0, ADDR_AUTOCONF);
  curr_pednet_instance.prefix = *prefix;
```

```
  LOG_INFO("IPv6 addresses:\n");
  for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
    state = uip_ds6_if.addr_list[i].state;
    if(uip_ds6_if.addr_list[i].isused &&
       (state == ADDR_TENTATIVE || state == ADDR_PREFERRED)) {
      LOG_INFO("-- ");
      LOG_INFO_6ADDR(&uip_ds6_if.addr_list[i].ipaddr);
      LOG_INFO_("\n");
    }
  }
}
/*---------------------------------------------------------------------------*/
void
ped_link_callback(const linkaddr_t *addr, int status, int numtx)
{

      /* Link stats were updated, and we need to update our internal state.
      Updating from here is unsafe; postpone */
      LOG_INFO("packet sent to ");
      LOG_INFO_LLADDR(addr);
      LOG_INFO("\n");



}
/*---------------------------------------------------------------------------*/

int
ped_node_has_joined(void)
{
  return 0;
}
/*---------------------------------------------------------------------------*/
int
ped_node_is_reachable(void)
{
  return 0;
}
void
ped_leave_network(void){
}
//static
void
neighbor_state_changed(uip_ds6_nbr_t *nbr)
{
  /* Nothing needs be done in non-storing mode */
}
//static
void
drop_route(uip_ds6_route_t *route)
{
  /* Do nothing. RPL-lite only supports non-storing mode, i.e. no routes */
}
void
ped_router_unreachable_output(uip_ipaddr_t *addr)
```

```
{
}
void
ped_register_input_handler(pednet_input_handler_t *handler)
{
  list_add(input_handler_list, handler);
}
#################### pednet-timers.c ############################
#include "contiki.h"
#include "net/ipv6/uip-sr.h"
#include "net/link-stats.h"
#include "lib/random.h"
#include "sys/ctimer.h"
#include "net/pednet/pednet-types.h"
#include "net/pednet/pednet-routing.h"
#include "net/pednet/pednet-er.h"
#include "net/pednet/pednet-r.h"
#include "net/pednet/pednet-leaf.h"
#include "uip-ds6-nbr.h"
#include "nbr-table.h"
/* Log configuration */
#include "sys/log.h"
#define LOG_MODULE "PED"
#define LOG_LEVEL LOG_LEVEL_PED
#define PERIODIC_DELAY_SECONDS     60
#define DEFAULT_DELAY              30
/* change this to the number of seconds for eap and rap to be sent */
#define DEFAULT_EAP_DELAY          DEFAULT_DELAY
#define DEFAULT_RAP_DELAY          DEFAULT_DELAY
/*
 Continue sending node new and router new with this until acknowledge message is received
 */
#define DEFAULT_NODE_NEW_DELAY     5
#define DEFAULT_ROUTER_NEW_DELAY   5
#define PERIODIC_DELAY             ((PERIODIC_DELAY_SECONDS) * CLOCK_SECOND)
#define R_INIT_DELAY               (5 * CLOCK_SECOND) /* Number of seconds * clock_second */
#define ER_INIT_DELAY              (10 * CLOCK_SECOND) /* Number of seconds * clock_second */
#define N_MAX_DELAY                200 /* This is going to be in milliseconds */
/*
* Node new and router new messages will be sent this number of seconds in order to inform the upstream
* that they are alive
*/
#define NODE_NEW_DELAY             (20 * CLOCK_SECOND) /* Number of seconds * clock_second */
#define ROUTER_NEW_DELAY           (20 * CLOCK_SECOND) /* Number of seconds * clock_second */

#define ADDR_CHANGE_PERIOD         (240 * CLOCK_SECOND) // periodic addr change in seconds
void handle_node_new_timer(void *ptr);
void handle_router_new_timer(void *ptr);
void handle_router_new_ack_timer(void *ptr);
void handle_node_new_ack_timer(void *ptr);
void handle_eap_timer(void *ptr);
void handle_rap_timer(void *ptr);
void handle_pac_timer(void *ptr);
static void handle_periodic_timer(void *ptr);
static struct ctimer periodic_timer; /* Not part of a DAG because used for general state maintenance */
static struct ctimer ER_init_timer = {};
```

```
static struct ctimer R_init_timer = {};
static struct ctimer N_init_timer = {};
pednet_leaf_instance_t curr_pednet_leaf_instance;
pednet_r_instance_t curr_pednet_r_instance;
pednet_er_instance_t curr_pednet_er_instance;
void
ped_timers_schedule_periodic_node_new(void)
{
  LOG_DBG("SCHEDULE Node New \n");
  if(ctimer_expired(&curr_pednet_leaf_instance.node_new_timer)) {
    uint8_t delay = curr_pednet_leaf_instance.node_new_delay;
    clock_time_t expiration_time = delay * CLOCK_SECOND / 2 + (random_rand() % (delay *
CLOCK_SECOND));
    ctimer_set(&curr_pednet_leaf_instance.node_new_timer, expiration_time, handle_node_new_timer, NULL);
  }
}
void
ped_timers_schedule_periodic_router_new(void)
{
  LOG_DBG("SCHEDULE Router New \n");
  if(ctimer_expired(&curr_pednet_r_instance.router_new_timer)) {
    uint8_t delay = curr_pednet_r_instance.router_new_delay;
    clock_time_t expiration_time = delay * CLOCK_SECOND / 2 + (random_rand() % (delay *
CLOCK_SECOND));
    ctimer_set(&curr_pednet_r_instance.router_new_timer, expiration_time, handle_router_new_timer, NULL);
  }
}
void
ped_timers_schedule_periodic_eap(){
  LOG_DBG("SCHEDULE eap \n");
  if(ctimer_expired(&curr_pednet_er_instance.ER_addr_config_timer)) {
    uint8_t delay = curr_pednet_er_instance.eap_delay;
    clock_time_t expiration_time = delay * CLOCK_SECOND / 2 + (random_rand() % (delay *
CLOCK_SECOND));
    ctimer_set(&curr_pednet_er_instance.ER_addr_config_timer, expiration_time, handle_eap_timer, NULL);
  }
}
void
ped_timers_schedule_pac(){
  LOG_DBG("SCHEDULE periodic addr change \n");
  if(ctimer_expired(&curr_pednet_er_instance.periodic_addr_change_timer)) {
    clock_time_t expiration_time = ADDR_CHANGE_PERIOD / 2 + (random_rand() %
(ADDR_CHANGE_PERIOD / 2));
    ctimer_set(&curr_pednet_er_instance.periodic_addr_change_timer, expiration_time, handle_pac_timer, NULL);
  }
}
void
ped_timers_schedule_periodic_rap(){
  LOG_DBG("SCHEDULE rap \n");
  if(ctimer_expired(&curr_pednet_r_instance.R_addr_config_timer)) {
    uint8_t delay = curr_pednet_r_instance.rap_delay;
    clock_time_t expiration_time = delay * CLOCK_SECOND / 2 + (random_rand() % (delay *
CLOCK_SECOND));
    ctimer_set(&curr_pednet_r_instance.R_addr_config_timer, expiration_time, handle_rap_timer, NULL);
  }
}
```

```
void
handle_er_timer()
{
  LOG_DBG("handle_Er_timer \n");
  if(!curr_pednet_er_instance.initialized){
    generate_cser();
    ped_er_addr_param_output(NULL);
    curr_pednet_er_instance.initialized = 1;
    ped_timers_schedule_periodic_eap();
    LOG_DBG("GENERATED CSER \n");
  }
void
handle_n_timer()
{
  LOG_DBG("N INIT \n");
  if(!curr_pednet_leaf_instance.initialized){
    ped_node_new_output(NULL);
    curr_pednet_leaf_instance.initialized = 1;
  }
  ped_timers_schedule_periodic_node_new();
}
void node_new_delay_change(uint8_t reduce_time)
{
  if(reduce_time == 0){
    curr_pednet_leaf_instance.node_new_delay = NODE_NEW_DELAY / CLOCK_SECOND;
  }else
  {
    curr_pednet_leaf_instance.node_new_delay = DEFAULT_NODE_NEW_DELAY;
  }
}
void router_new_delay_change(uint8_t reduce_time)
{
  if(reduce_time == 0){
    curr_pednet_r_instance.router_new_delay = ROUTER_NEW_DELAY / CLOCK_SECOND;
  }else
  {
    curr_pednet_r_instance.router_new_delay = DEFAULT_ROUTER_NEW_DELAY;
  }
}
void
ped_timers_init(void)
{
  LOG_INFO("INIT TIMERS \n");
  ctimer_set(&periodic_timer, PERIODIC_DELAY, handle_periodic_timer, NULL);
  if(ped_node_is_er()){
    ctimer_set(&ER_init_timer, ER_INIT_DELAY, handle_er_timer, NULL);
    curr_pednet_er_instance.eap_delay = DEFAULT_EAP_DELAY;
    ped_timers_schedule_pac();
  }
  else if (ped_node_is_r())
  {
    ctimer_set(&R_init_timer, R_INIT_DELAY, handle_r_timer, NULL);
    curr_pednet_r_instance.rap_delay = DEFAULT_RAP_DELAY;
    curr_pednet_r_instance.router_new_delay = DEFAULT_ROUTER_NEW_DELAY;
    ped_timers_schedule_periodic_router_new();
  }
```

```
  else
  {
    uint8_t max_delay = (N_MAX_DELAY / 2) + random_rand() % N_MAX_DELAY;
    ctimer_set(&N_init_timer, max_delay, handle_n_timer, NULL);
    curr_pednet_leaf_instance.node_new_delay = DEFAULT_NODE_NEW_DELAY;
    ped_timers_schedule_periodic_node_new();
  }
}
/** @}*/
#################### pednet-types.h ##########################
#ifndef PEDNET_TYPES_H
#define PEDNET_TYPES_H
#include "pednet-conf.h"
 /********** Macros **********/
#define PED_PACKET_BUF                  ((struct ped_packet_hdr *)ped_buf)
#define PED_PACKET_PAYLOAD              ((unsigned char *)ped_buf + PED_HDR_LEN)

#define uip_is_addr_same(a, b)                   \
  ((((a)->u16[0]) == ((b)->u16[0])) &&           \
   (((a)->u16[1]) == ((b)->u16[0])) &&           \
   (((a)->u16[2]) == ((b)->u16[0])) &&           \
   (((a)->u16[3]) == ((b)->u16[0])) &&           \
   (((a)->u16[4]) == ((b)->u16[0])) &&           \
   (((a)->u16[5]) == ((b)->u16[0])) &&           \
   (((a)->u16[6]) == ((b)->u16[0])) &&           \
   (((a)->u16[7]) == ((b)->u16[0])))
/* Multicast address: create and compare */
/** \brief Set IP address addr to the link-local, all-ped-nodes
   multicast address. */
/* #define uip_create_linklocal_pednodes_mcast(addr)        \
  uip_ip6addr((addr), 0xff02, 0, 0, 0, 0, 0, 0, 0x001a)
 */
/** \brief Is IPv6 address addr the link-local, all-PED-nodes
   multicast address? */
/* #define uip_is_addr_linklocal_pednodes_mcast(addr)        \
  ((addr)->u8[0] == 0xff) &&                              \
  ((addr)->u8[1] == 0x02) &&                              \
  ((addr)->u16[1] == 0) &&                                \
  ((addr)->u16[2] == 0) &&                                \
  ((addr)->u16[3] == 0) &&                                \
  ((addr)->u16[4] == 0) &&                                \
  ((addr)->u16[5] == 0) &&                                \
  ((addr)->u16[6] == 0) &&                                \
  ((addr)->u8[14] == 0) &&                                \
  ((addr)->u8[15] == 0x1a))
 */
typedef struct ped_iid_addr {
  uint16_t router_id;
  uint16_t node_id;
  uint8_t icode;
} ped_iid_addr_t;
/* handle input messages */
typedef struct pednet_input_handler {
  struct pednet_input_handler *next;
  uint8_t type;
  uint8_t icode;
```

```
  void (*handler)(void);
} pednet_input_handler_t;
/* */
struct pednet_instance {
  uint8_t num_routers;
  uint8_t max_node_num;
  struct ctimer state_update;
  struct ctimer leave;
  uint8_t csr;
  uint8_t cser;
  uip_ipaddr_t prefix;
};
typedef struct pednet_instance pednet_instance_t;
struct ped_packet_hdr {
  uint8_t type, icode;
  uint8_t len[2];
  uint8_t reserved[4];
  uip_ip6addr_t srcipaddr, destipaddr;
};
typedef union {
  uint32_t u32[(PED_PAYLOAD_SIZE + 3) / 4];
  uint8_t u8[PED_PAYLOAD_SIZE];
} ped_buf_t;
ped_buf_t ped_aligned_buf;
#define ped_buf (ped_aligned_buf.u8)
struct er_addr_param
{
  uint16_t cser;
  uip_ipaddr_t prefix;
};
typedef struct er_addr_param er_addr_param_t;
struct r_addr_param
{
  uint16_t router_uid;
  uip_ipaddr_t prefix;
  uint16_t cser;
  uint16_t csr;
};
NULL;
  else if (len == PED_HDR_LEN)
  {
    LOG_ERR("No data only header... Drop. \n");
  }

  else{
    LOG_ERR("Pkt length < hdr length... Drop. \n");
  }
static uint8_t
output(const linkaddr_t *dest)
{
  packetbuf_clear();
  packetbuf_copyfrom(pednet_buf, pednet_len);
  if(dest != NULL) {
    packetbuf_set_addr(PACKETBUF_ADDR_RECEIVER, dest);
  } else {
    packetbuf_set_addr(PACKETBUF_ADDR_RECEIVER, &linkaddr_null);
```

```
  }
  packetbuf_set_addr(PACKETBUF_ADDR_SENDER, &linkaddr_node_addr);
  LOG_DBG("pedNet: sending %u bytes to ", packetbuf_datalen());
  LOG_DBG_LLADDR(packetbuf_addr(PACKETBUF_ADDR_RECEIVER));
  LOG_DBG("\n");
  NETSTACK_MAC.send(&packet_sent, NULL);
  return 1;
}
/*---------------------------------------------------------------------------*/
const struct network_driver pednet_driver = {
  "PEDNET",
  init,
  input,
  output
};
#include "contiki.h"
#include "contiki-net.h"
#include "net/pednet/pednet.h"
#include "net/pednet/pednet-conf.h"
#include "net/pednet/pednet-const.h"
#include "net/pednet/pednet-types.h"
#include "net/pednet/pednet-timers.h"
#include "net/pednet/pednet-er.h"
#include "net/pednet/pednet-routing.h"
#include "net/pednet/pednet-in-out.h"
#include "net/ipv6/uip-ds6-route.h"
#include "net/ipv6/uip-sr.h"
#include "net/ipv6/uip.h"
#include "sys/node-id.h"
/* Log configuration */
#include "sys/log.h"
#define LOG_MODULE "PED"
#define LOG_LEVEL LOG_LEVEL_PED

void ped_router_new_ack_output(uip_ipaddr_t *addr)
{
  LOG_DBG("Router new ack output \n");
  uip_ds6_addr_t *lladdr = uip_ds6_get_link_local(-1);
  uip_ipaddr_t* local_addr = &lladdr->ipaddr;
  unsigned char *buffer;
  PED_PACKET_BUF -> type = PED_MESSAGES;
  PED_PACKET_BUF -> icode = PED_CODE_ROUTER_NEW_ACK;
  memset(PED_PACKET_BUF -> reserved, 0, sizeof(PED_PACKET_BUF -> reserved));
  PED_PACKET_BUF -> destipaddr = *addr;
  PED_PACKET_BUF -> srcipaddr = *local_addr;
  buffer = PED_PACKET_PAYLOAD;
  buffer[0] = buffer[1] = 0;
  ped_send(addr, PED_MESSAGES, PED_CODE_ROUTER_NEW_ACK, 2);
}
static void
router_new_input()
{
  LOG_DBG("Router new input \n");
  uint8_t *buffer = PED_PACKET_PAYLOAD;
  uint16_t tmp = 0;
  memcpy(&tmp, buffer, sizeof(uint16_t));
```

```
  if(!curr_pednet_er_instance.initialized){
    // LOG_INFO("Data: %u", tmp);
    if(tmp > curr_pednet_instance.max_node_num){
      LOG_DBG("TMP: %u  max_num: %u \n", tmp, curr_pednet_instance.max_node_num);
      curr_pednet_instance.max_node_num = tmp;
    }
    LOG_DBG("TMP: %u  max_num: %u \n", tmp, curr_pednet_instance.max_node_num);
  }
  else
  {
    if(tmp > curr_pednet_instance.max_node_num){
      LOG_DBG("TMP: %u  max_num: %u \n", tmp, curr_pednet_instance.max_node_num);
      curr_pednet_instance.max_node_num = tmp;
      generate_cser();
      ped_timers_schedule_periodic_eap();
    }
    LOG_DBG("Initialized ER already \n");
  }
  // ped_router_new_ack_output(&PED_PACKET_BUF->srcipaddr);
}
void
node_failed_input(){
  LOG_DBG("node_failed_input \n");
}
void
node_mobile_input(){
  LOG_DBG("node_failed_input \n");
}
void
node_unreachable_input(){
  LOG_DBG("node_unreachable_input \n");
}
void
router_alive_input(){
  LOG_DBG("router_alive_input \n");
}
#include "contiki.h"
#include "contiki-net.h"
#include "net/pednet/pednet.h"
#include "net/pednet/pednet-conf.h"
#include "net/pednet/pednet-const.h"
#include "net/pednet/pednet-types.h"
#include "net/pednet/pednet-timers.h"
#include "net/pednet/pednet-r.h"
#include "net/pednet/pednet-routing.h"
#include "net/ipv6/uip-nd6.h"
#include "net/ipv6/uip-ds6-nbr.h"
#include "net/ipv6/uip-ds6.h"
#include "net/ipv6/uip-ds6-route.h"
#include "net/ipv6/uip-sr.h"
#include "net/ipv6/uip.h"
#include "sys/node-id.h"
/* Log configuration */
#include "sys/log.h"
#define LOG_MODULE "PED"
#define LOG_LEVEL LOG_LEVEL_PED
```

```
// hhhhh
pednet_instance_t curr_pednet_instance;
pednet_r_instance_t curr_pednet_r_instance;
er_addr_param_t er_addr_param_structure;
uip_lladdr_t ped_node_linkaddr;
uip_ipaddr_t ped_node_global_addr;
//uip_ds6_addr_t *lladdr;
static void node_new_input(void);
static void router_new_ack_input(void);
static void er_addr_param_input(void);
static void node_alive_input(void);
static void node_unreachable_input(void);
static void node_mobile_input(void);
static void router_unreachable_input(void);
void
change_r_node_address();
PED_HANDLER(node_new_handler, PED_MESSAGES, PED_CODE_NODE_NEW, node_new_input);
PED_HANDLER(router_new_ack_handler, PED_MESSAGES, PED_CODE_ROUTER_NEW_ACK,
router_new_ack_input);
PED_HANDLER(er_addr_param_handler, PED_MESSAGES, PED_CODE_ER_ADDR_PARAM,
er_addr_param_input);
PED_HANDLER(node_alive_handler, PED_FAIL_DETECT, PED_FAIL_NODE_ALIVE, node_alive_input);
PED_HANDLER(node_unreachable_handler, PED_FAIL_DETECT, PED_FAIL_NODE_UNREACHABLE,
node_unreachable_input);
PED_HANDLER(node_mobile_handler, PED_FAIL_DETECT, PED_FAIL_NODE_MOBILE,
node_mobile_input);
PED_HANDLER(router_unreachable_handler, PED_FAIL_DETECT, PED_FAIL_ROUTER_UNREACHABLE,
router_unreachable_input);
void
ped_r_init(void){
  LOG_INFO("R INIT REACHED \n");
  ped_timers_init();
  //curr_pednet_instance = NULL;
  ped_register_input_handler(&node_new_handler);
  ped_register_input_handler(&router_new_ack_handler);
void
node_alive_input(){}
static void
node_new_input(){
  uint8_t *buffer = PED_PACKET_PAYLOAD;
  uint16_t test = 0;
  memcpy(&test, buffer, sizeof(uint16_t));
  LOG_DBG("Node new received %u\n", test);
  if(!curr_pednet_r_instance.initialized){
    if(test > curr_pednet_instance.max_node_num){
      curr_pednet_instance.max_node_num = test;
    }
  }
  else
  {
    if(test > (curr_pednet_instance.max_node_num)){
      curr_pednet_instance.max_node_num = test;
      update_random_num();
      generate_csr();
      change_r_node_address();
    }
```

```
    ped_node_new_ack_output(&PED_PACKET_BUF->srcipaddr);
  }
  ped_node_new_ack_output(&PED_PACKET_BUF->srcipaddr);
}
uint8_t
ped_list_loop(uip_ds6_element_t *list, uint8_t size,
                  uint16_t elementsize, uip_ipaddr_t *ipaddr,
                  uint8_t ipaddrlen, uip_ds6_element_t **out_element)
{
  uip_ds6_element_t *element;
  if(list == NULL || ipaddr == NULL || out_element == NULL) {
    return NOSPACE;
  }
  *out_element = NULL;
  for(element = list;
      element <
      (uip_ds6_element_t *)((uint8_t *)list + (size * elementsize));
      element = (uip_ds6_element_t *)((uint8_t *)element + elementsize)) {
    if(element->isused) {
      if(uip_ipaddr_prefixcmp(&element->ipaddr, ipaddr, ipaddrlen)) {
        *out_element = element;
        LOG_INFO_6ADDR(&element->ipaddr);
        LOG_INFO(",,");
        return FOUND;
      }
      LOG_INFO_6ADDR(&element->ipaddr);
      LOG_INFO(",,");
    } else {
      LOG_INFO_6ADDR(&element->ipaddr);
      LOG_INFO(",,");
      *out_element = element;
    }
  }
  return *out_element != NULL ? FREESPACE : NOSPACE;
}

void
change_r_node_address(){
  uip_ipaddr_t last_ipaddr = ped_node_ipaddr;
  ped_curr_router_id = node_id + curr_pednet_instance.cser * ped_rand_num;
  // LOG_INFO("id: %u, cser: %u, rand: %u \n", node_id, curr_pednet_instance.cser, ped_rand_num);
  uint16_t dummy_node_id = curr_pednet_instance.csr - 1;
  ped_curr_node_id = dummy_node_id + curr_pednet_instance.csr * ped_rand_num;
  memcpy(&ped_node_ipaddr, &uip_ds6_get_link_local(-1)->ipaddr, 8);
  uip_ds6_addr_t *test_ds6 = (uip_ds6_get_global(-1));
  uip_ipaddr_t ipaddr;
  uip_create_unspecified(&ipaddr);
  LOG_INFO("Showing global: ");
  if(ped_list_loop
     ((uip_ds6_element_t *)uip_ds6_if.addr_list, UIP_DS6_ADDR_NB,
      sizeof(uip_ds6_addr_t), &ipaddr, 128,
      (uip_ds6_element_t **)&test_ds6) == FOUND) {
    LOG_INFO_6ADDR(&test_ds6->ipaddr);
  }

  memcpy(&er_addr_param_structure, buffer, sizeof(er_addr_param_t));
```

```
  LOG_DBG("PREFIX: ");
  LOG_DBG_6ADDR(&er_addr_param_structure.prefix);
  LOG_DBG("\n");
  if(!curr_pednet_r_instance.configured){
    // LOG_INFO("!configured \n");
    curr_pednet_instance.cser = er_addr_param_structure.cser;
    curr_pednet_instance.prefix = er_addr_param_structure.prefix;
    ped_node_ipaddr = uip_ds6_get_link_local(-1)->ipaddr;
    LOG_INFO("INIT NODEIPADDR: ");
    LOG_INFO_6ADDR(&ped_node_ipaddr);
    LOG_INFO("\n");
    ped_curr_node_id = node_id;
    update_random_num();
    change_r_node_address();
    curr_pednet_r_instance.configured = 1;
  }else{
    if(!(curr_pednet_instance.cser == er_addr_param_structure.cser)){
      update_random_num();
      ped_node_ipaddr = uip_ds6_get_link_local(-1)->ipaddr;
      LOG_INFO("INIT NODEIPADDR: ");
      LOG_INFO_6ADDR(&ped_node_ipaddr);
      LOG_INFO("\n");
      LOG_DBG("curr: %u struct: %u \n", curr_pednet_instance.cser , er_addr_param_structure.cser);
      curr_pednet_instance.cser = er_addr_param_structure.cser;
      curr_pednet_instance.prefix = er_addr_param_structure.prefix;
      change_r_node_address();
      ped_timers_schedule_periodic_rap();
    }
  }
}
void
node_mobile_input(){
}
void
node_unreachable_input(){}
void
router_unreachable_input(){}
/*---------------------------------------------------------------------------*/
void
ped_r_set_prefix(uip_ipaddr_t *prefix, uip_ipaddr_t *iid)
{
  static uint8_t initialized = 0;
  if(!initialized) {
    ped_set_global_address(prefix, iid);
    initialized = 1;
  }
}
#include "contiki.h"
#include "contiki-net.h"
#include "net/pednet/pednet.h"
#include "net/pednet/pednet-conf.h"
#include "net/pednet/pednet-const.h"
#include "net/pednet/pednet-types.h"
#include "net/pednet/pednet-timers.h"
#include "net/pednet/pednet-leaf.h"
#include "net/pednet/pednet-routing.h"
```

```
#include "net/ipv6/uip-nd6.h"
#include "net/ipv6/uip-ds6-nbr.h"
#include "net/ipv6/uip-ds6.h"
#include "net/ipv6/uip-ds6-route.h"
#include "net/ipv6/uip-sr.h"
#include "net/ipv6/uip.h"
#include "sys/node-id.h"
/* Log configuration */
#include "sys/log.h"
#define LOG_MODULE "PED"
#define LOG_LEVEL LOG_LEVEL_PED
// hhhhh
pednet_instance_t curr_pednet_instance;
pednet_leaf_instance_t curr_pednet_leaf_instance;
r_addr_param_t r_addr_param_structure;
uip_ipaddr_t ped_node_ipaddr;
uip_ipaddr_t ped_node_global_addr;
uip_lladdr_t ped_node_linkaddr;
uint16_t ped_curr_node_id;
uint16_t ped_curr_router_id;
static void r_addr_param_input(void);
static void node_new_ack_input(void);
static void node_unreachable_input(void);
PED_HANDLER(r_addr_param_handler, PED_MESSAGES, PED_CODE_ROUTER_ADDR_PARAM,
r_addr_param_input);
PED_HANDLER(node_new_ack_handler, PED_MESSAGES, PED_CODE_NODE_NEW_ACK,
node_new_ack_input);
PED_HANDLER(node_unreachable_handler, PED_FAIL_DETECT, PED_FAIL_NODE_UNREACHABLE,
node_unreachable_input);
void
ped_leaf_init(void){
  LOG_INFO("LEAF INIT REACHED \n");
  ped_timers_init();
  update_random_num();
  ped_curr_node_id = node_id;
  //curr_pednet_instance = NULL;
  ped_register_input_handler(&r_addr_param_handler);
  ped_register_input_handler(&node_new_ack_handler);
  ped_register_input_handler(&node_unreachable_handler);
  //uip_create_unspecified(&ped_node_ipaddr);
}
void
change_leaf_node_address(){/*
  uint16_t cser = curr_pednet_instance.cser;
  uint16_t csr = curr_pednet_instance.csr; */
  LOG_DBG("change_leaf_node_address \n");
  uip_ipaddr_t last_ipaddr = ped_node_ipaddr;
  ped_curr_router_id = r_addr_param_structure.router_uid + r_addr_param_structure.cser * ped_rand_num;
  ped_curr_node_id = node_id + r_addr_param_structure.csr * ped_rand_num;
  memcpy(&ped_node_ipaddr, &uip_ds6_get_link_local(-1)->ipaddr, 8);
  ped_node_ipaddr.u16[6] = UIP_HTONS(ped_curr_router_id);
  ped_node_ipaddr.u16[7] = UIP_HTONS(ped_curr_node_id);
  uip_ds6_set_lladdr_from_iid(&ped_node_linkaddr, &ped_node_ipaddr);
  LOG_DBG("r_id: %u, node_id: %u \n", ped_curr_router_id, ped_curr_node_id);
  linkaddr_copy(&linkaddr_node_addr, (linkaddr_t *)&ped_node_linkaddr);
  if(!uip_is_addr_unspecified(&ped_node_ipaddr)){
```

```
    if(uip_ds6_addr_lookup(&last_ipaddr)){
      LOG_DBG("REMOVING LAST ADDR :");
      LOG_DBG_6ADDR(&last_ipaddr);
      LOG_DBG("\n");
      uip_ds6_addr_rm(uip_ds6_addr_lookup(&last_ipaddr));
    }
    if(uip_ds6_addr_lookup(&ped_node_ipaddr) == NULL) {
      LOG_DBG("adding link local address ");
      LOG_DBG_6ADDR(&ped_node_ipaddr);
      LOG_DBG("\n");
      uip_ds6_addr_add(&ped_node_ipaddr, 0, ADDR_AUTOCONF);
    }
    if(!(uip_is_addr_unspecified(&curr_pednet_instance.prefix) ||
          uip_is_addr_linklocal(&curr_pednet_instance.prefix)) ){
      memcpy(&ped_node_global_addr, &ped_node_ipaddr, sizeof(uip_ipaddr_t));
      memcpy(&ped_node_global_addr, &curr_pednet_instance.prefix, 8);
      uip_ds6_addr_t *ipaddr = uip_ds6_get_global(-1);
      if(ipaddr){
        uip_ds6_addr_rm(ipaddr);
      }
      LOG_INFO("adding global address ");
      LOG_INFO_6ADDR(&ped_node_global_addr);
      LOG_INFO_("\n");
      uip_ds6_addr_add(&ped_node_global_addr, 0, ADDR_AUTOCONF);
    }
  }
  // uip_ds6_send_rs();
  LOG_INFO("PAC: Address changed! \n");
}
void
r_addr_param_input()
{
  LOG_DBG("r_addr_param_input \n");
  uint8_t *buffer;
  buffer = PED_PACKET_PAYLOAD;
  memcpy(&r_addr_param_structure, buffer, sizeof(r_addr_param_t));
  LOG_INFO("prefix: ");
  LOG_INFO_6ADDR(&curr_pednet_instance.prefix);
  LOG_INFO("\n");
  // LOG_INFO("ID1: %u  ID2: %u \n", r_addr_param_structure.router_uid, curr_pednet_leaf_instance.router_uid);
  if(!curr_pednet_leaf_instance.configured){
    // LOG_INFO("r_addr_param_structure.cser: %u", r_addr_param_structure.cser);
    if(r_addr_param_structure.cser > 1){
      curr_pednet_leaf_instance.configured = 1;
      curr_pednet_instance.cser = r_addr_param_structure.cser;
      curr_pednet_instance.csr = r_addr_param_structure.csr;
      curr_pednet_leaf_instance.router_uid = r_addr_param_structure.router_uid;
      ped_node_ipaddr = uip_ds6_get_link_local(-1)->ipaddr;
      LOG_INFO("INIT NODEIPADDR: ");
      LOG_INFO_6ADDR(&ped_node_ipaddr);
      LOG_INFO("\n");
      change_leaf_node_address();
      curr_pednet_leaf_instance.configured = 1;
    }
  }else{
```

```
    // LOG_INFO("addr_struct_uid: %u curr_uid: %u \n", r_addr_param_structure.router_uid,
curr_pednet_leaf_instance.router_uid);
    if(r_addr_param_structure.router_uid == curr_pednet_leaf_instance.router_uid){
      if(!(curr_pednet_instance.cser == r_addr_param_structure.cser &&
          curr_pednet_instance.csr == r_addr_param_structure.csr) ){
        curr_pednet_instance.cser = r_addr_param_structure.cser;
        curr_pednet_instance.csr = r_addr_param_structure.csr;
        if(!(uip_is_addr_unspecified(&r_addr_param_structure.prefix) ||
            uip_is_addr_linklocal(&r_addr_param_structure.prefix)) ){
              curr_pednet_instance.prefix = r_addr_param_structure.prefix;
        }
        change_leaf_node_address();
      }
    }
  }
}
void
node_new_ack_input()
{
  LOG_DBG("node_new_ack_input \n");
  node_new_delay_change(0);
  // LOG_DBG("ADDR: ");
  // LOG_DBG_6ADDR(&PED_PACKET_BUF->destipaddr);
  // LOG_DBG("\n");
}
void
node_unreachable_input()
{
}
void
ped_leaf_set_prefix(uip_ipaddr_t *prefix, uip_ipaddr_t *iid)
{
  static uint8_t initialized = 0;
  if(!initialized) {
    ped_set_global_address(prefix, iid);
    initialized = 1;
  }
}
/*---------------------------------------------------------------------------*/
int
ped_leaf_start(void)
{
  //struct uip_ds6_addr *root_if;
  //int i;
  //uint8_t state;
  //uip_ipaddr_t *ipaddr = NULL;
  //ped_leaf_set_prefix(NULL, NULL);
  return 0;
}
```