IMPLEMENTATION OF A GRAPHICAL USER INTERFACE FOR THREAD PROTOCOL USING OPENTHREAD PLATFORM

by

Jitendra Gopaluni, B.Tech.

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Computer Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

DECEMBER, 2018

IMPLEMENTATION OF A GRAPHICAL USER INTERFACE FOR THREAD PROTOCOL USING OPENTHREAD PLATFORM

by

Jitendra Gopaluni

APPROVED BY

_____
Ishaq Unwala, Ph.D., Chair

_____
Jiang Lu, Ph.D., Committee Member

_____
Xiaokun Yang, Ph.D., Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

_____
Said Bettayeb, Ph.D., Associate Dean

_____
Ju H. Kim, Ph.D., Dean

Dedication

I dedicate this Thesis to my Mother, **Sarada Mani Gopaluni**, for her belief in me and Grand Parents **Venu Gopal Rao Kalapatapu** and **Lakshmi Kalapatapu** for their encouragement**.**

Acknowledgements

I express my sincere and deepest gratitude to my supervisor, Dr. Ishaq Unwala, Assistant Professor, Department of Computer Engineering. His expertise, invaluable guidance, constant encouragement, affectionate attitude, understanding, and patience added considerably to my experience. Without his continual inspiration, it would not have been possible to complete this study.

I owe my special thanks to Dr. Jiang Lu for being part of my thesis committee.

I owe my special thanks to Dr. Xiaokun Yang for being part of my thesis committee

Special thanks to Prof. Dr. Hakduran Koc, program chair and associate professor of Computer Engineering.

I extend my thanks to the Dean's Office, Librarians, Writing Centre and UHCL staff. The financial support from the Financial Aid Department in the form of a teaching Assistantship is gratefully acknowledged

I am also very thankful to my friends Shiva, Harold, Ngyuen Ly, April, the Francisco brothers from the Microprocessor Interfacing lab at the UHCL for their support.

I thank Harshini, Prathyusha, Jissmol, Teja, Nandini, Bhaskar, Johanna, Divya and Mounika, my roommates Muhammad, Stuart and Andrea for their support.

Above all, I would like to thank God, my mother, Grand-parents and my family members who with their love and encouragement have made all this possible.

ABSTRACT

IMPLEMENTATION OF A GRAPHICAL USER INTERFACE FOR THREAD PROTOCOL USING OPENTHREAD PLATFORM

Jitendra Gopaluni
University of Houston-Clear Lake, 2018

Thesis Chair: Ishaq Unwala Ph.D.

Thread is an IPv6-based protocol introduced in July 2014. Thread is a networking protocol for Internet of Things (IoT) for “smart" home automation devices to communicate on a local wireless mesh network. Thread uses 6LoWPAN, IEEE 802.15.4 wireless standard with mesh topology. Thread is IP-addressable, with AES encryption and supports up to 250 devices in one local network mesh.

OPENTHREAD

OpenThread is an open source implementation of the Thread Protocol. OpenThread implements all the features defined in the THREAD 1.1.1 specification (IPv6, 6LoWPAN, IEEE 802.15.4 with MAC security, Mesh Link Establishment, Mesh Routing).

GUI using TCL/tk

OpenThread is currently uses a command line Interface. The Thesis research is focused on adding a Graphical User Interface (GUI) based on Tool Control Language. (TCL “tickle”) to OpenThread. TCL/Tk enables building a GUI natively in TCL. OpenThread having a GUI will make working with OpenThread much easier, the use of basic commands will not need to be memorized and additionally the user does not need to know any programming language. GUI will make it easier for developers working on OpenThread to visualize the Thread network and its operations.

TABLE OF CONTENTS

List of Tables ix

List of Figures x

CHAPTER I: INTRODUCTION 1

CHAPTER II: INTERNET OF THINGS (IOT) 4

CHAPTER III: THREAD PROTOCOL 8

3. 1. Thread network Architecture 10
3.1.1 Border Routers 11
3.1.2 Leader 11
3.1.3 Router-eligible end devices 12
3.1.4 Sleepy end devices 12
3. 2. Thread stack Fundamentals 13
3.2.1 Addressing 13
3.2.2 6Lowpan 13
3.2.3 UDP 14
3. 3. Network Address and Devices 14
3. 3. 1. Mesh Network 14
3. 4. Mesh Link Establishment 16
3.4.1 MLE Messages 16
3.4.2. Path Identification and Route Updating 17
3.5. Join the network 19
3.5.1 Discovery 19
3.5.2 Commissioning 19
3.5.3 MLE data 20

CHAPTER IV: OPENTHREAD 21

4.1 Uses of OpenThread 21
4.2 Network Implementation 23
4.3 Routing Implementation 24
4.4 OpenThread CLI commands 25
4.5 Working of OpenThread 27
4.6 Problem Statement and Solution 30
4.6.1 GUI (Graphical User Interface) for OpenThread 30

CHAPTER V: GUI USING TCL/TK FOR OPENTHREAD 31

5.1 Building GUI for OpenThread 11
5.2 Working of GUI for OpenThread 12

5.2.1 Contents of the GUI. 32
5.2.2 Functioning of the GUI. 35

CHAPTER VI: CONCLUSION AND FUTURE WORKS 44

6.1 Conclusion 44
6.2 Future works 44

REFERENCES 46

LIST OF TABLES

Table 3.1 Overview of Thread Technical Specifications..... 10

LIST OF FIGURES

Figure 2.1 IOT 4
Figure 3.1 Thread standards with respective layers 8
Figure 3.2 Thread network Topology 11
Figure 3.3 Thread Topolgy roles 12
Figure 3.4 Mesh Link Establishment 16
Figure 3.5 Route Updating 17
Figure 4.1General Overview of OpenThread Module 23
Figure 4.2 IPV6 module in OpenThread 24
Figure 4.3 MLE module in OpenThread 24
Figure 4.4 Ipaddr output 25
Figure 4.5 Router table output 26
Figure 4.6 connecting Leader and Router 27
Figure 4.7 Commissioner function 28
Figure 4.8 Joiner function 29
Figure 5.1 OpenThread GUI 33
Figure 5.2 Add 34
Figure 5.3 Remove 34
Figure 5.4 Commissioner 35
Figure 5.5 Creating a node 36
Figure 5.6 State of Node 1 37
Figure 5.7 State of node 2 38
Figure 5.8 rloc16 of node 2 39
Figure 5.9 routing tble of node 1 39
Figure 5.10 Disabling node 1 40
Figure 5.11 State of node 1 41
Figure 5.12 State of node 2 41
Figure 5.13 Configuring the Commissioner 42

Figure 5.14 Configuring the Joiner 43

CHAPTER I:

INTRODUCTION

People always try to find new methods to increase their comfort. This includes ideas for making daily tasks easier or even automating some of the duties. Today people can install smart appliances inside their homes in order to automate some of the household tasks. Technology is becoming an increasingly important part of our everyday life. Using new technology, many at home can be automatically controlled. Thus, home automation has become very popular. Home automation is a term which refers to the automation of various home, housework or household activities. Home automation for the elderly and disabled can provide increased quality of life for user who might otherwise require caregivers or institutional care. The research and development in this field has continued over the years, but to actually automate a home is still a quite expensive job. These types of intelligent devices have the possibility of remote control, which eliminates the necessity of being near the device.

First, home automation brings convenience the temperature can be set to a certain value according to certain conditions, lighting can be turned on, off or may be dimmed based on daylight. Second, home automation implies remote access, such as monitoring the house using a laptop or even the own cell phone. Nowadays, one of the hottest topics in media is related to energy conservation. Automation systems can help the energy savings by, for example, turning off the electronic devices automatically when they are not in use. A house which is equipped with such a system offers much more comfort, flexibility, elegance, security, more importunately, reduced maintenance costs through the optimization of the consumption of electricity and heat.

For example, some of this smart house can include simple things like turning the sprinkler at some time during the day or detecting thieves in the middle of the night; others

are more advanced and employ sensors for detecting the presence of a person in a room, used to adjust the ambient light, to control the temperature or the music volume depending on various factors.

Home Automation System needs to collect data from various sensors and make things such as light and temperature to automatically adjust. Moreover, using these sensors, many tasks can be accomplished, such as, controlling curtains and windows without human intervention, opening, locking or unlocking the garage gate, controlling the climate inside the house, providing the corresponding light in each room, starting the sprinkler when the soil is too dry and so on. However, the concept of home automation is often heard on the market, being a relatively new concept, which draws attention to researchers. [1]

This leads to new technologies that can perform home automation functions. Since it is rapidly evolving and is present in many products, there is a need to understand its meaning and challenges. The term “IoT” refers to connecting devices that are not directly controlled by humans to the Internet. This type of devices is “things”. By connecting such devices, an intelligent and invisible network is created, which can be accessed through the cloud [1]. Besides the features which each device is capable of implementing, the network must offer an interface. The smart devices are controlled and programed remotely through an interface. All IoT [2] products benefit from embedded technology which allows them to communicate between each other or with the users through the Internet.

Before inception of Internet of Things (IoT), personal computers and laptop were used to handle daily tasks of individuals like Email, web surfing, access to bank portal, observing current temperature, among others. Smart homes have been widely accepted by individuals and organizations worldwide due to their many advantages. Many home security systems exist, but they have some challenging issues like: delay, is non-web enabled and difficult to handle during transfer of alerts to user in situation where any

unusual event occurred inside the home. If any unusual event encounters inside the home, cameras and other latest network technologies have enabled us to remotely monitor the home more effectively and efficiently from our smart phone.

It's hard to get devices to talk to one another. And once they do, the connection is often spotty and power hungry. Thread changes all that. It's a mesh network designed to securely and reliably connect hundreds of products around the home – without using major amount of battery. [3]

CHAPTER II:

INTERNET OF THINGS (IOT)

The Internet of Things (IoT) [1] is the biggest challenge and opportunity for the Internet today. An interesting example of application of the IoT is a home automation system. Using a home automation process in a household environment, we can give additional functionality through the integration of sensors and actuators to normally non-automated systems like lighting, heating, air conditioning and appliances. Recently, home automation systems have been challenged with the need for high interoperability between home devices and for accessing the system from different end points.

Figure 2.1. Internet of Things (IoT) [3]

The idea consists of IP-enabled embedded devices connected to the Internet using Internet Protocol Version-6 (IPv6) [4]. The Internet Engineering Task Force (IETF) added to this idea by defining 6LoWPAN [5] as a technique to apply IPv6 to IEEE 802.15.4 [5], a low-power wireless network standard, which adds the potential for transparent end-to-end communication, control and monitoring of home automation devices from anywhere on the globe. The use of 6LoWPAN technology also helps lowers expense and decreases complexity of home automation architecture. [5]

IoT can be termed as the connection among various kinds of analog and digital devices like personal computers, smart phones and tablets to the Internet that create connection between things and people. The IoT enables items to be detected or controlled remotely over existing system infrastructure, making open doors for more straightforward incorporation of the physical world into PC based frameworks, and bringing about enhanced effectiveness, exactness and financial advantage notwithstanding lessened human intervention.[2] When IoT is enlarged with sensors and actuators, the innovation turns into an occurrence of the broader class of digital physical frameworks, which likewise includes advancements, for example, keen lattices, virtual power plants, shrewd homes, insightful transportation and brilliant urban areas. [2]

"Things", in the IoT sense, can allude to a wide assortment of devices, for example, heart observing chips, biochip transponders on creatures, cameras spilling live sustains of wild creatures in waterfront waters, vehicles with sensors, Legal researchers propose regarding "things" as an "inseparable blend of equipment, programming, information and service".[1] These devices gather valuable information with the assistance of different existing innovations and afterward autonomous stream the information between different devices.

Users now may have access to a multitude of data from multiple sources that were unavailable before, such as physiological or biometric data gathered with a wristband device or a watch. [7] IoT opens up possibilities that were until recently only in the realm of sci-fi movies. Today, we are starting to build smart homes, cities and vehicles that manage and control themselves without needing user input. [2] Interfacing different sort of devices like PDAs, individual PC and tablets to web can be used to characterized IoT. Different things can be associated using IoT innovation. Without human effort, we can control things via various devices through home automation fully depending on seasons. The logic of home automation is significant for home appliances companies and researchers. An automated device has lowest error rated which works with versatility and diligence. [8] Different type of technologies is used in wireless-based home automation system. Multiple devices can be connected to one medium via Wireless Fidelity (Wi-Fi) technology which provides an excellent medium. It can operate over frequency band of 2.4GHZ which has been approved internationally.

IoT can be named as the association among different sorts of simple and computerized devices like PC, advanced cells and tablets to the web that make approach among things and people groups and thusly among things.

The main components of an IoT system usually are,

a. Embedded sensor/actuator devices (things);
b. A gateway (also called hub), which is required when the embedded devices do not have native IP connectivity;
c. An Internet service/database running on an online Web server;

The gateway has two main functions:

1. Translate the packets between the embedded device protocols (e.g., ZigBee [10] Z-Wave [9] and/or Bluetooth [8]) and the TCP/IP [11] protocol;

2. Provide access to the Internet. In many cases the gateway can be co-located with the local Internet. router, sharing the utilization with conventional (non IoT) traffic such as Web browsing.

The objectives of IoT are to,

1. Interface many devices in the home with best-in-class work organizing,
2. Use the organization's skill in low-control, obliged devices.
3. Give minimal effort connecting to existing Ethernet and Wi-Fi devices.
4. Empower cloud administrations and availability to cell phones and tablets that will advance convenience and a typical client encounter for clients.

CHAPTER III:

THREAD PROTOCOL

Thread [18] is a wireless mesh networking protocol. The main aim of thread protocol is to create the best way to connect and control products in the home. The Thread stack is an open standard that is assembled accumulation of existing Institute for Electrical and Electronics Engineers (IEEE) and Internet Engineering Task Force (IETF) standards, instead of a radical new standard. [24]

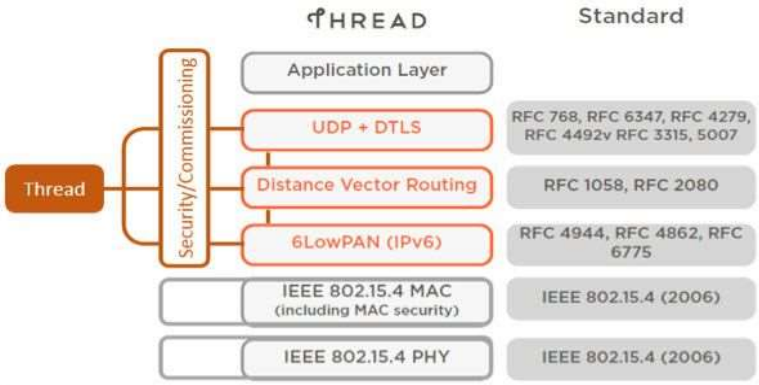

Figure 3.1 Thread Protocol Standards with respective layers [12]

Thread is an IPv6-based, closed-documentation networking protocol for Internet of Things (IoT) [12]. In July 2014, the "Thread Group" alliance was announced by the working group with the companies Nest Labs (a subsidiary of Alphabet/Google), Samsung, ARM Holdings, BigAss Fans, NXP Semiconductors/Freescale, Silicon Labs and Yale in an attempt to have Thread become the industry standard by providing Thread certification for products. [12] Thread is designed to connect products in and around the home into low-power, wireless mesh networks.

Simple network installation, start up and operation: The basic protocols for forming, joining, and maintaining Thread networks allow systems to self-configure and fix routing problems. Installation is simply using a smartphone, tablet, or PC.

Secure: Devices don't join the Thread network unless approved and all communications are encrypted and secure. Security is given at the network layer and can be at the application layer. All Thread networks are encrypted using AES encryption and smartphone authentication scheme.

Large and small networks: Home systems vary from several devices to hundreds of devices communicating flawlessly. The network layer is intended to advance the network operation based on the normal utilize.

Range: Typical devices using with mesh networking give adequate range to cover an ordinary home. Spread range innovation is utilized at the physical layer to give great resistance to impedance. [12]

No single point of failure: The thread network is intended to give secure and dependable operations even with the loss of individual devices.

Low power: Host devices can regularly work for quite a long while on AA size batteries utilizing appropriate obligation cycles. Devices efficiently productively impart to convey an upgraded client involvement with long life under typical battery conditions. [12]

Cost-effective: Compatible chipsets and programming stacks from numerous merchants are preferred for mass organization, and planned starting from the earliest stage to have to great degree low-control utilization. Run of the mill home items keep running in the Connected Home include: typically controlled (lighting, fans); fueled or battery-worked (indoor regulators, and CO2 finders); and ordinarily battery-worked (window sensors, movement sensors).

IEEE 802.15.4 Thread is based on the IEEE 802.15.4 [12] [13] Physical and MAC layers working at 250 kbps in the 2.4 GHz band. The IEEE 802.15.4-2006 version of the specification is utilized by the Thread stack.

The IEEE 802.15.4 MAC layer is utilized for essential message taking care of and blockage control. This MAC layer incorporates a Carrier Sense Multiple Access (CSMA) mechanism for devices to listen for tune in for a reasonable channel and in addition a link layer to deal with retries and affirmation of messages for dependable interchanges between neighboring devices. MAC layer encryption and integrity protection is utilized on

messages based on keys established and configured by the higher layers of the software stack. The network layer builds on these important mechanisms to provide reliable end-to-end communications in the network. [13]

Specification	Thread Data
Design Focus	Home connectivity to IoT
Network Type	Mesh
Network	Thread
Distance	Normally 20-30 meters
Max Nodes Connected	250
Operating Band	2.4GHz (The ISM unlicensed band)
Spread Spectrum	Radio uses direct sequence spread spectrum (DSSS)
Throughput	Radio operates at 250 kbps
Data	Monitoring and control data
Voice Capable	No
Security	Banking-class, public-key cryptography
Modulation	Radio specification is O-QPSK modulation

Table 3.1 Overview of Thread Technical Specifications. [29]

3. 1. Thread network Architecture

Users can communicate with Thread network from using their own device for example smartphone, tablet, or PC via Wi-Fi on their Home Area Network (HAN) or using a cloud-based application. There are different types of device in the Thread network architecture.

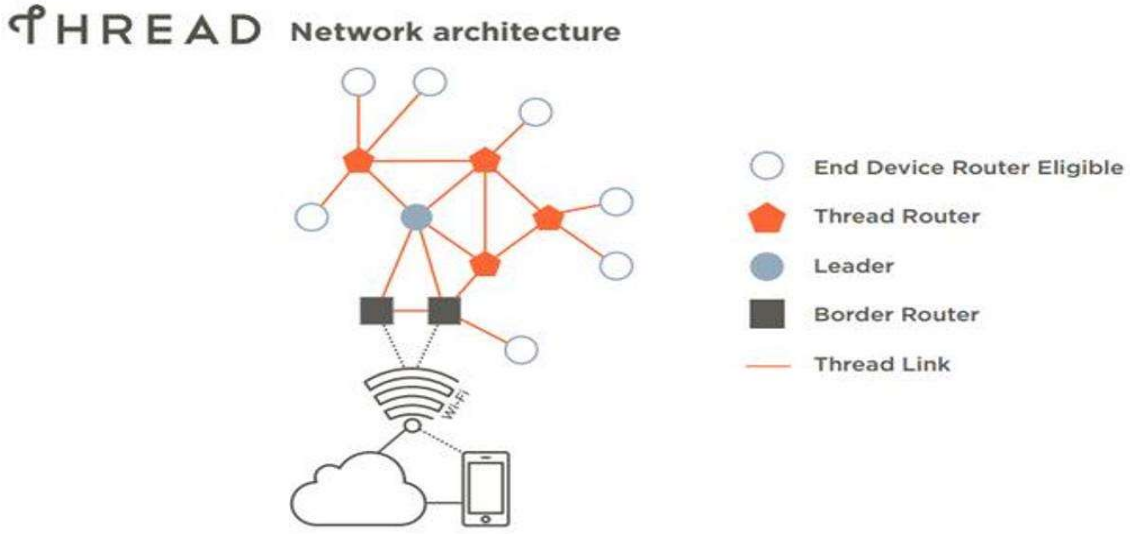

Figure 3.2 Thread network Topology. [25]

3.1.1 Border Routers:

A Border Router is sort of router that gives availability from the IEEE 802.15.4 system to neighboring systems on other physical layers, such as Wi-Fi, Ethernet, etc. Border Routers give services for devices within the IEEE 802.15.4 network, including routing services for off network operations. There will be least one or more Border Routers in a Thread network. [25]

3.1.2 Leader:

A Router or Border Router can turn into a Leader router for specific capacities in the Thread network. This Leader router is required to make decisions in the Home Area Network (HAN) network. It deals with a registry of current node IDs and acknowledges demands from node qualified devices to wind up nodes or the other way around. The Leader router allocates which ought to be nodes. It also chooses node addresses and allows new node requests. Likewise oversees and relegates node tends to utilizing DHCP. Notwithstanding, all data present in the Leader is available in the other Thread Routers. In this way, if the Leader fails or loses availability with the network, another Thread Router

is elected, and it takes over as Leader without user intervention. It is this independent operation that ensures that there is no single point of failure. [25]

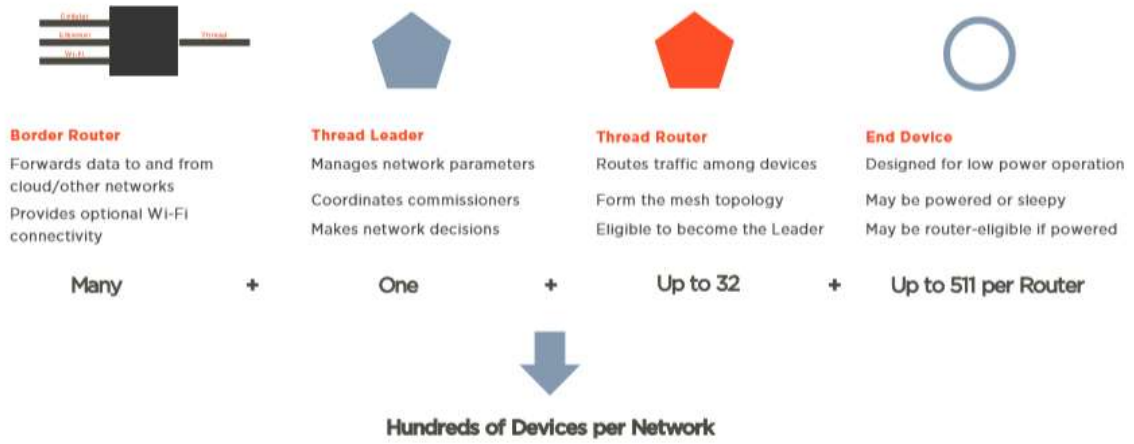

Figure 3.3 Thread Topology Roles. [32]

3.1.3 Router-eligible end devices:

The ability to wind up a Thread Router or a Leader, however because of the system topology or conditions these devices are not assigned as Routers. It cannot really a Border Router that has uncommon properties, for example, various interfaces. Router qualified device by and large forward messages or give joining or security administrations to different device in the Home Area Network (HAN). The Thread network oversees and elevates node qualified devices to Routers if vital without client association to Routers if vital without user association. [32]

3.1.4 Sleepy end devices:

Sleepy end devices convey just through their Thread Router and can't hand-off messages for others devices. They are also called as are host devices. [32]

3. 2. Thread stack Fundamentals

3.2.1 Addressing

Devices in the Thread stack advocates IPv6 tending to configuration demonstrated in [RFC 4291]. Devices uses something like one Unique Local Address (ULA) or Global Unicast Address (GUA). The device starting the framework picks a/64 prefix that is then used all through the Thread network. The prefix is a Locally Assigned Global ID, routinely known as a ULA prefix [RFC 4193], and can be insinuated as the work neighborhood ULA prefix. The Device in the Thread network uses its Extended Media Access control (MAC) convey to decide its interface identifier as portrayed in territory 6 of [RFC 4944] and from this plans an association adjacent IPv6 address with the striking neighborhood prefix FE80::0/64 as delineated in [RFC 4862] and [RFC 4944]. [14] The devices in like manner advocates fitting multicast addresses. Each Device joining the Thread network is dispensed a 16-bit short location as decided in IEEE802.15.4. For routers, this location is given using the high bits in the location field with the lower bits set to 0, exhibiting a router address. Child nodes are then disseminated a 16-bit short location using their Parent's high bits and the correct lower bits for their location. This allows some other Device in the Thread network to calculate the child's routing just by using the high bits of its location field

3.2.2 6LoWPAN:

All devices utilize 6LoWPAN as characterized in [RFC 4944] and [RFC 6282]. [15] Header compression is utilized inside the Thread network and devices transmitting messages pack the IPv6 header however much as could be expected to limit the measure of the transmitted parcel. The mesh header is upheld for more proficient messages inside the work and for interface layer. End devices and REEDs are allotted short locations by their Router Parent. This short deliver is then used to arrange the work neighborhood ULA that is utilized for intra-organize correspondences. Additionally, subtle elements on 6LoWPAN utilization and arrangement are contained in the "Thread Usage of 6LoWPAN" white paper. Part 3 of the Thread particular subtle elements the particular 6LoWPAN setup utilized. [16]

3.2.3 UDP:

The Thread stack advocates UDP (User Datagram Protocol) as characterized in [RFC 768] for informing between devices.

3. 3. Network Address and Devices

A network address is an identifier for a hub or system interface of a broadcast communications network. Network addresses are intended to be good identifiers over the system, albeit a few systems take into consideration local, private locations or privately regulated tends to that may not be globally unique. The Thread stack advocates full Mesh connectivity between all routers in the Thread network. The topology depends on the quantity of routers in the Thread network. If there is just a single router or Border Router, at that point an essential star topology with a solitary router is framed. On the off chance that there is more than one Router then a work topology is consequently framed. There are different types of topology available for 6LoWPAN. THREAD uses Mesh network. [12]

3. 3. 1. Mesh Network

A mesh-network is a local-network-topology in which the framework hubs (i.e. spans, nodes and other foundation devices) associate specifically, powerfully and non-progressively to whatever number different hubs as could reasonably be expected and collaborate with each other to effectively course information from/to customers. This absence of reliance on one hub considers each hub to take an interest in the transfer of data. Work arranges powerfully self-sort out and self-design, which can lessen establishment overhead. The capacity to self-arrange empowers dynamic conveyance of workloads, especially if a couple of hubs ought to fizzle.

Mesh-topology might be appeared differently in relation to regular star/tree local-network-topologies in which the scaffolds/nodes are straightforwardly connected to just a little subset of different extensions/nodes, and the connections between these foundation neighbors are various leveled. While star-and-tree topologies are exceptionally settled, exceedingly institutionalized and merchant unbiased, sellers of work arrange devices have

not yet all conceded to regular guidelines, and interoperability between devices from various sellers isn't yet guaranteed. [17]

Mesh networks make radio frameworks more solid by enabling radios to forward messages for different radios. For instance, if a hub can't communicate something specific straightforwardly to another hub, the work organize advances the message through at least one mediator hubs. There is normally a point of confinement of 32 dynamic routers in the Thread-network. Be that as it may, 64 router delivers are utilized to permit reusing of router addresses. In a work organize, the languid end devices or REEDs don't route for different devices. These devices send messages to a Parent router. This Parent router handles the directing operations for its Child-devices.

The Thread network regularly has up to 32 active routers that use next-hop routing for messages based on the device routing table. The routing table is maintained by the stack to guarantee all routers have availability and forward ways for some other router in the Thread network. The RIPng algorithm is utilized. All routers trade with other routers their cost of routing to different routers in the Thread network in a compacted arrange utilizing MLE (Mesh Link Establishment). From an IP viewpoint, the Thread network advocates for routers and hosts. Hosts are either languid end devices or REEDs.

3. 4. Mesh Link Establishment

3.4.1 MLE Messages

MLE messages are utilized for setting up and arranging secure radio connections, distinguishing neighboring devices, and keeping up directing expenses between devices in the Thread network. MLE messages are transported utilizing single-hop link local unicasts and multicasts between routers. MLE messages are utilized for distinguishing, arranging, and securing connects to neighboring devices as the topology and physical condition change. MLE is likewise used to disperse designs that are shared over the Thread network, for example, the channel and Personal Area Network (PAN) ID. These messages can be sent with basic flooding as determined by MPL. MLE messages likewise guarantee lopsided connection costs are considered while building up routing costs between two devices. Awry connection costs are regular in IEEE 802.15.4 systems. To guarantee two-way communications is solid, it is vital to consider the expenses of bidirectional connections.

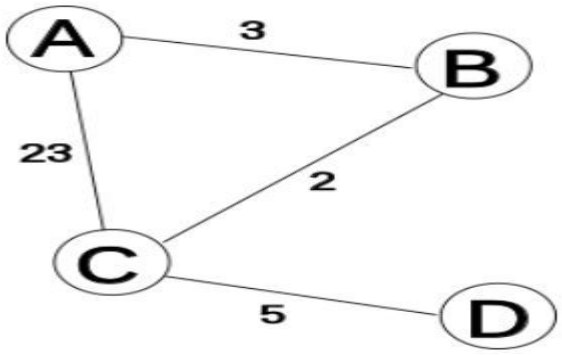

Figure 3.4 Mesh Link Establishment (MLE) and its routing cost. [32]

3.4.2. Path Identification and Route Updating

On-demand route disclosure is normally utilized as a part of low-power IEEE802.15.4 systems. Notwithstanding, on-demand route revelation is expensive regarding system overhead and data transfer capacity because of routing disclosure demands flooding the network. In a Thread network, all routers intermittently trade single-hop MLE advertisement packets containing join cost data to all neighbor routers, and way expenses to every single other router in the Thread network. Through these periodic, local updates, all routers have progressive way cost data to some other router in the Thread network; on-demand route discovery isn't required. If that route is not usable, routers can decide on the following most reasonable route to the goal. This self-mending steering component enables routers to rapidly distinguish when different routers have dropped off the Thread network, and compute the best way to keep up availability to every single other device in the Thread network.

Figure 3.5 Route Updating. [32]

The link quality toward every path depends on the connection cost on approaching messages from those neighboring devices. This approaching connection cost is mapped to a connection quality from 0 to 3. An estimation of 0 implies obscure cost. The connection cost is a measure of RSSI [35] of got messages over the get level achieves that hub. Nodes screen these costs, even as the radio connection quality or topology of the system changes, and spread the new costs through the Thread network utilizing the occasional MLE notice messages. Routing cost depends on bi-directional connection quality between two devices. To represent through an improved illustration, envision a pre-dispatched connect with shared security material where all devices are controlled on in the meantime.

Every router would occasionally send a notice to initialize just with expenses to single-jump neighbors. Inside, every router would store hop information that isn't sent in the advertisement. The initial couple of ads would have way taken a toll equivalent to interface cost, because the main routers that are known are quick neighbors. Be that as it may, as routers begin hearing promotions from their neighbors that contain expenses to different routers that are at least two jumps away, their tables populate with multi-hop way costs which at that point spread significantly more remote, until in the long run there is availability data between all routers in the system.

At the point when a router gets another MLE notice from a neighbor, it is possible that it as of now has a neighbor table section for the gadget or one is included. The MLE ad contains the approaching expense from the neighbor so this is refreshed in the router's neighbor table. The MLE promotion likewise contains refreshed routing information for different routers and this data is refreshed in the device routing table. routing to Child devices is finished by taking a gander at the high bits of the Child's deliver to decide the parent router address. Once the gadget knows the Parent router, it has the way cost data and next-hop routing information data for that device. The quantity of dynamic routers is constrained to the measure of steering and cost data that can be contained in a solitary IEEE 802.15.4 bundle. This breaking point is at present 32 Routers yet 64 dynamic router delivers are given to permit maturing out of router addresses.

3.5. Join the network

There are different stages a joining device needs to experience before it can take and participate in a Thread network:

1. Discovery
2. Commissioning
3. Attaching

All joining is user-initiated in Thread-Networks. Once joined, a device is of the Thread network and can trade application layer data with different devices and administrations inside and outside the Thread network.

3.5.1 Discovery

A joining device needs to find the Thread network and set up contact with a router for joining. The joining device filters all channels, issues a guide ask for on each channel, and waits for reference point signals. The signal contains a payload including the system SSID (Service Set Identifier) and an allow joining reference point showing if the Thread network is allowing new individuals. Once a device has found the Thread network, it utilizes MLE messages to set up a neighboring router through which it can perform dispatching.

Revelation isn't required if the device has just acquired charging data since it as of now has adequate data to specifically connect to the Thread network.

3.5.2 Commissioning

Thread gives two authorizing techniques: [12]

1. Configuring authorizing data straightforwardly onto a device utilizing an out-of-band technique. The dispatching data enables the joining device to connect to the best possible Thread network when it is acquainted with the network
2. Establishing a dispatching session between a joining device and a charging application on a cell phone, tablet, or the web. The appointing session safely conveys authorizing data to the joining device, enabling it to append to the correct Thread network in the wake of having finished the charging session.

The regularly utilized IEEE 802.15.4 technique for joining which allow joining banner in the guide payload isn't utilized as a part of Thread network. This strategy is most normally utilized for push catch sort joining where there is no User Interface (UI) or out-of-band channel to devices. In Thread networks, all joining is client started. In the wake of joining, a security verification is finished at the application level with an authorizing device.

Devices join a system as either an end device or a (REED) Router-Eligible End Device. Simply after a REED has joined and taken in the system design, would it be able to possibly demand to end up plainly a Thread router. After joining, a device is given a 16-bit short address in view of its parent. If a node qualified device turns into a Thread router, it is given a node address by the Leader. Copy address discovery for Thread routers is guaranteed by the unified node address circulation component which dwells on the Leader. The parent is in charge of maintaining a strategic distance from copy addresses for have devices since it finds out addresses to them after joining.

3.5.3 MLE data

Once a device has connected to a Thread network, there is an assortment of data required for it to keep up its investment in the system. MLE gives administrations to disseminate arrange information all through the system and trade connect expenses and security outline counters between neighbors.

The MLE messages convey or trade the accompanying data:

- The 16-bit short and 64-bit EUI 64 long address of neighboring devices.
- Device abilities data including if it is a tired end device and the rest cycle of the slow host device.
- Neighbor interface costs (if a router).
- Security material and casing counters between devices.
- Routing cost to every other router in the Thread network.
- Updates to route information such.

CHAPTER IV:

OPENTHREAD

Nest Labs, Inc. (acquired by Google in the beginning of 2014) released OpenThread in May 2016, OpenThread is an open source implementation based on the draft Thread 1.0 specification of the Thread networking protocol. With OpenThread, Nest wants to make the technology used in Nest products more broadly available to accelerate the development of products for the connected home. The idea is as more silicon providers adopt Thread, manufacturers will have the option of using a proven networking technology rather than creating their own, and consumers will have a growing selection of secure and reliable connected products which to choose from.

4.1 Uses of OpenThread

Deploy a Thread network: Determine the hardware and platform design the user wish to use to build and deploy their own Thread network. Add a Border Router to connect your Thread network to other network layers, such as Wi-Fi or Ethernet. [36]

Develop applications on top of a Thread network: Use the API Reference as a guide to all application development. IPv6, UDP, CoAP, DHCPv6, DNSv6. [36]

Port OpenThread to a new hardware platform: Porting Guide, which goes through all the steps necessary to port OpenThread to a new hardware platform. [36]

Get Thread Certification for your OpenThread product: OpenThread can be used for certification by the Thread Group. As a Thread reference stack, OpenThread makes certification easy. [36]

OpenThread implements all Thread networking layers including: [36]

- IEEE 802.15.4 with MAC security.
- IPv6 and 6LoWPAN.
- Mesh Link Establishment and Mesh Routing.
- Key management.
- Definitions in code of specific roles in Thread including:
    - Leader.
    - Router.
    - End Device.
    - The Border router.
- UDP packet compression.
- A CoAP implementation.

OpenThread is highly portable: OS and platform agnostic with a radio abstraction layer. Is written mostly in C++. The implementation depends on a platform layer, basically a Hardware Abstraction Layer (HAL), and if that layer is implemented, it can potentially run on most microcontroller or 802.15.4 SoCs (essentially microcontrollers with an integrated 802.15.4 radio) with the advantage of small memory footprint.

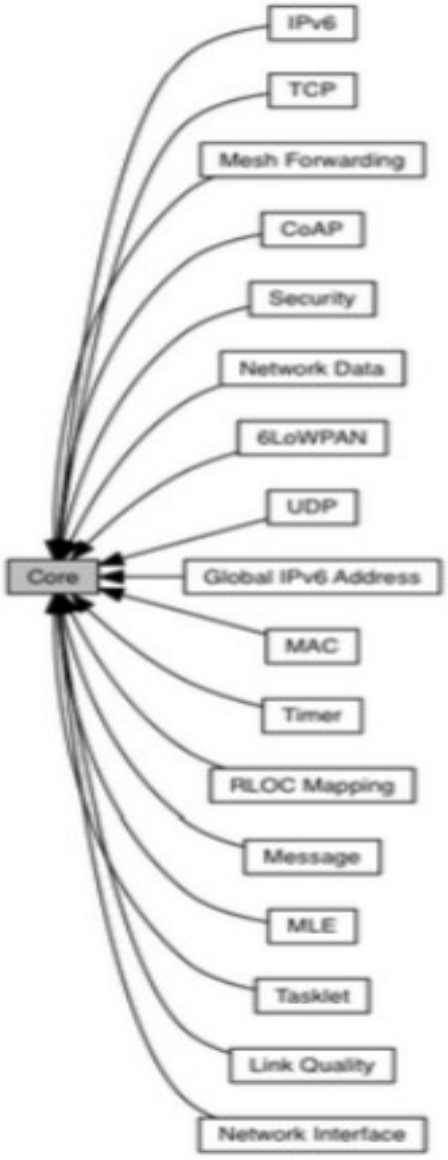

Figure 4.1 General Overview of OpenThread Modules. [32]

4.2 Network Implementation

IPv6 includes definitions for the IPv6 network layer. IPV6 module defines the ICMPv6 implementation the network interfaces, the multicast protocol and the IPv6 implementation itself.

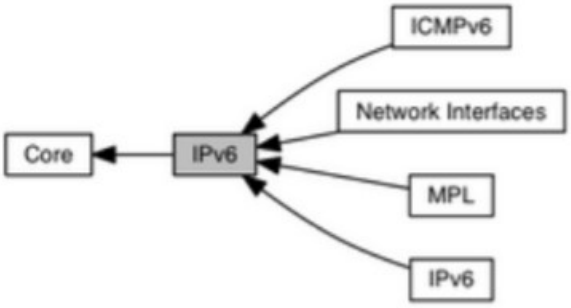

Figure 4.2 IPV6 Module in OpenThread. [32]

4.3 Routing Implementation

OpenThread implements MLE to propagate the routing table information and RIPng to process information and maintain routing tables. The implemented MLE module (depends on Core) and implements MLE functionality required for the Thread router and Leader roles. The Type Length Value (TLV) module includes definitions for generating and processing MLE TLVs.

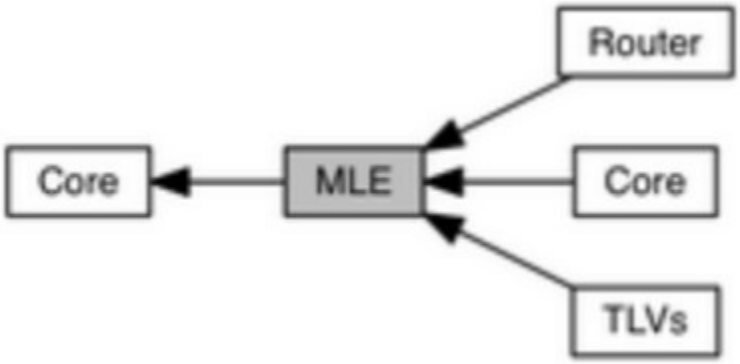

Figure 4.3 MLE Module in OpenThread. [32]

4.4 OpenThread CLI commands

The OpenThread Command Line Interface (CLI) exposes configuration and management APIs via a command line interface. Use the CLI to play with OpenThread, which can also be used with additional application code.

Some of the CLI available commands are:

- autostart
    - o autostart
    - o autostart true
    - o autostart false
- eui64
    - o eui64
- factoryreset
    - o factoryreset.
- ipaddr
    - o ipaddr.

```
> ipaddr
fdde:ad00:beef:0:0:ff:fe00:0
fdde:ad00:beef:0:558:f56b:d688:799
fe80:0:0:0:f3d9:2a82:c8d8:fe43
Done
```

Figure 4.4 Ipaddr Output.

- Joiner
    - o Joiner start.
    - o Joiner stop.
- panid
    - o panid.

- o panid

- ping
  - o ping
- reset
  - o reset.
- rloc16
  - o rloc16
- router
  - o router list
  - o router
  - o router table

```
> router table
| ID | RLOC16 | Next Hop | Path Cost | LQI In | LQI Out | Age | Extended MAC     |
+----+--------+----------+-----------+--------+---------+-----+------------------+
| 21 | 0x5400 |       21 |         0 |      3 |       3 |   5 | d28d7f875888fccb |
| 56 | 0xe000 |       56 |         0 |      0 |       0 | 182 | f2d92a82c8d8fe43 |
Done
```

Figure 4.5 Router Table output.

- state
  - o state
    - detached
    - child
    - router
    - Leader.
- thread
  - o thread start.

o thread stop.

4.5 Working of OpenThread

```
vagrant@vagrant-ubuntu-trusty-64: ~/src/openthread
> panid 0x1234
Done
> thread start
Error 13: InvalidState
> ifconfig up
Done
> thread start
Done
> state
detached
Done
> state
leader
Done
> ipaddr
fdde:ad00:beef:0:0:ff:fe00:fc00
fdde:ad00:beef:0:0:ff:fe00:e800
fe80:0:0:0:f4c3:12d6:569d:e78b
fdde:ad00:beef:0:b987:d96:2955:42c3
Done
> rroutertable
Error 6: Parse
> router table
| ID | RLOC16 | Next Hop | Path Cost | LQ In | LQ Out | Age | Extended MA
+----+--------+----------+-----------+-------+--------+-----+------------
| 58 | 0xe800 |       63 |         0 |     0 |      0 |   0 | f6c312d6569
| 62 | 0xf800 |       63 |         0 |     3 |      3 |   9 | c2b7142b06b

Done
```

a) Leader

```
vagrant@vagrant-ubuntu-trusty-64: ~/src/openthread

New release '16.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Thu Nov 29 06:40:44 2018 from 10.0.2.2
vagrant@vagrant-ubuntu-trusty-64:~$ cd ~/src/openthread
vagrant@vagrant-ubuntu-trusty-64:~/src/openthread$ ./output/x86_64-unknown-l
inux-gnu/bin/ot-cli-ftd 2

> panid 0x1234
Done
> ifconfig up
Done
> thread start
Done
> rloc16
f800
Done
> ping fe80:0:0:0:f4c3:12d6:569d:e78b
> 16 bytes from fe80:0:0:0:f4c3:12d6:569d:e78b: icmp_seq=1 hlim=64 time=1ms

> state
router
Done
```

b) Router

Figure 4.6 Connecting Leader and Router.

One of the features of OpenThread is configuring a Thread network using the CLI (Command Line Interface). In Figure 4.7, the command prompt on the top is configured to be a Leader node (node 1) and the command prompt on the bottom is configured to be a router (node 2). By calling the functions, “panid”, “ifconfig up” and “thread start” we have created a Leader node. The state of the nodes can be found by calling the “state” function. To verify the thread connection, we called “rloc16” function, which gives the 16 bits End Point Identifier (EID) of node 2. And we call “router table” function for node 1, which gives the routing table of node 1. It is shown that the rloc16 value of node 2 is in the routing table of the node1, proving the connection between the nodes.

We also called functions, “ipaddr” and “ping” to verify the mesh link establishments between the nodes. The “ipaddr” functions gives the IPV6 values of the nodes. “ipaddr” function for node 1 and called “ping” function from node 2 using the IPV6 address of node 1 which shows the output, thus proving the mesh link establishment between the nodes.

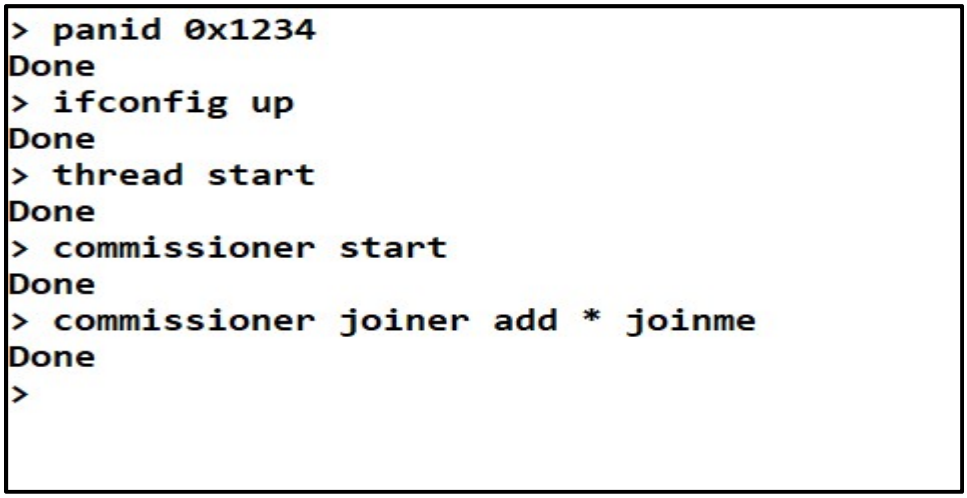

Figure 4.7 Commissioner function.

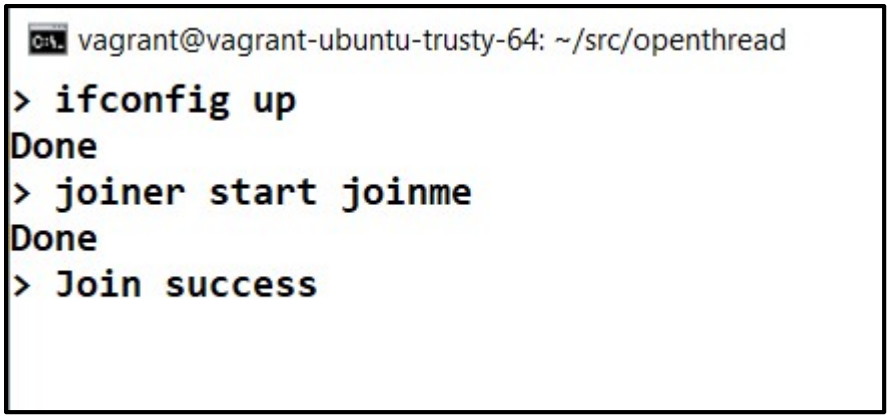

Figure 4.8 Joiner function.

Thread protocol's biggest advantage is the security it offers. A new device cannot join a Thread network unless it has the user defined key. The key is given to the Commissioner. A Commissioner is a Leader router that is responsible for the commissioning of new devices that wants to join a Thread network. Figure 5.8 shows the operation of a Commissioner and Joiner.

In Figure 4.8, the command prompt on the top is configured to be a Leader node (node 1) and the command prompt on the bottom is configured to be a router (node 2). By calling the functions "ifconfig up" and "thread start" we have created a Leader node. The state of the nodes can be found by calling the "state" function. Using the "Commissioner start" function we configured Leader node i.e. node 1 to be Commissioner and calling the "Joiner" function we have configured node 2 to be a Joiner device.

By calling the function "commissioner Joiner add * user-key" we assigned a Joiner key to the Commissioner. If a new device wants to join the network, it needs the same key to join. The Commissioner verifies the key and lets the new device to join the network. For node 2, we called "joiner start user-key" function to give the key to the new device. The Commissioner then checks the key and if it matches, lets the Joiner join the network by saying "join success."

OpenThread, has a very strict time restraint. The user defined key given to the Commissioner is only valid for 2 minutes. After 2 minutes, the key is expired and the Commissioner process starts from the beginning. The user gives a new key to the Commissioner and the new devices needs the new key to join the network. Also, Thread network provides, no single point of failure which means that, when one of the Leader router is disabled, the next router becomes a Leader and a REED (Router Eligible End Device) assumes the role of the router thus removing any point of failures.

4.6 Problem Statement and Solution.

As seen, for node to be configured, user need a new command prompt. Hence, when working with a larger network, having a command prompt for each node will complicate and confuses the user. Since, the only way to know the state of the node is by calling “state” function, it becomes really difficult for the user to remember the state of the node, if there are about 20 nodes, the user cannot keep a track of 20 command prompts and remember the state of each node.

4.6.1 GUI (Graphical User Interface) for OpenThread

Having a Graphical User Interface (GUI) for the OpenThread would make the life of the user easier. On a single screen, user can create any number of nodes and with just a click can get the information on the node. With a GUI, the user also need not remember the functions that are needed to configure a node. With just the inputs that are user defined, nodes are created and tested using the GUI.

CHAPTER V:

GUI USING TCL/TK FOR OPENTHREAD

This Thesis research is focused on adding a Graphical User Interface (GUI) based on Tool Control Language (TCL "tickle") to OpenThread. TCL is a high-level, general-purpose, interpreted, dynamic programming language. The popular combination of TCL with a graphical component, Tk, is referred to as TCL/Tk. TCL/Tk enables building a GUI natively in TCL. OpenThread having a GUI will make working with OpenThread much easier, the use of basic commands need not be memorized and additionally the user do not need to know any programming languages. GUI will make it easier for developers working on OpenThread to visualize the Thread network and its operations.

The GUI uses the OpenThread CLI commands and creates the nodes, connections, Leaders, routers, End Devices. Supports Commissioning and ping functions. With just buttons, functions like "thread start", "thread stop", "ifconfig up", "panid", "Commissioner Joiner start", "Joiner start" nodes are created, and a network is created based on thread protocol.

5.1 Building GUI for OpenThread

GUI for OpenThread is built using TCL/TK. GUI is purely written in TCL using TK libraries for building the GUI. OpenThread is accessed using Vagrant shell on any OS. But Ubuntu 18.04.1 LTS OS is used for building the GUI. Path for accessing OpenThread are given below,

```
$ cd ~/OpenThread/etc/vagrant
$ vagrant ssh
```

Once the shell is up and running, nodes are created using following path,

```
$ cd ~/src/openthread
$. /output/x86_64-unknown-linux-gnu/bin/ot-cli-ftd 1
```

For building the GUI, ot-cli-ftd file is used. Hence when OpenThread is accessed using vagrant shell, ot-cli-ftd file is copied and saved at the TCL file location.

5.2 Working of GUI for OpenThread

OpenThread GUI consists of “Add”, “remove”,” Commissioner” in the menu. Add consists of node id, and panid if the user wants to create a Leader. Remove consists of stop which disables the selected node, and remove that removes the node. The Commissioner consists of panid and the key assigned. The panid is used to make a Leader assume the role of Commissioner. The left click on the node gives “state”,” rloc16”,” router table” and “ipaddr” functions. In the bottom, we have a “result box” that displays the result after the function is called.

5.2.1 Contents of the GUI

The GUI consists of Add, Remove and Commissioner options in the menu. It also consists of a result box at the end, where results are viewed.

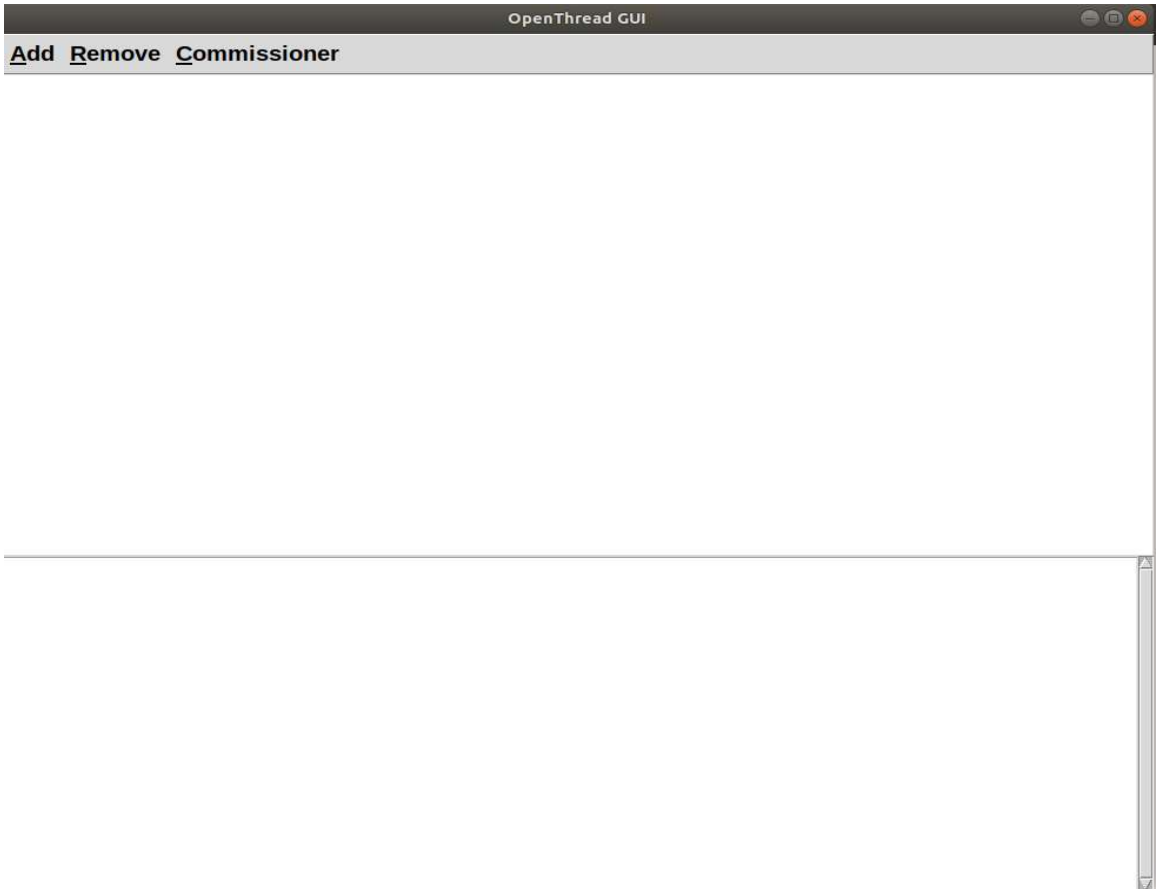

Figure 5.1 OpenThread GUI.

Figure 5.2 ADD.

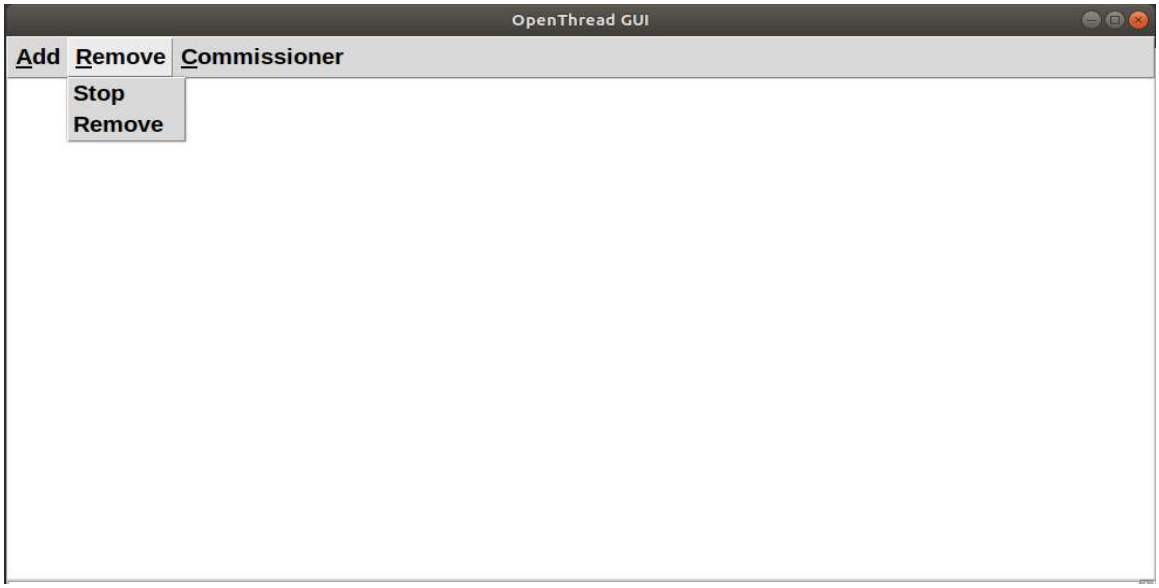

Figure 5.3 Remove.

Figure 5.4 Commissioner.

5.2.2 Functioning of the GUI

In Figure 5.5, we create a node using the ID, panid. With these inputs from the user, a node is created with an ID as shown in the Figure. Similarly, with the user defined parameters a second node is created with an ID '2' in Figure 5.6.

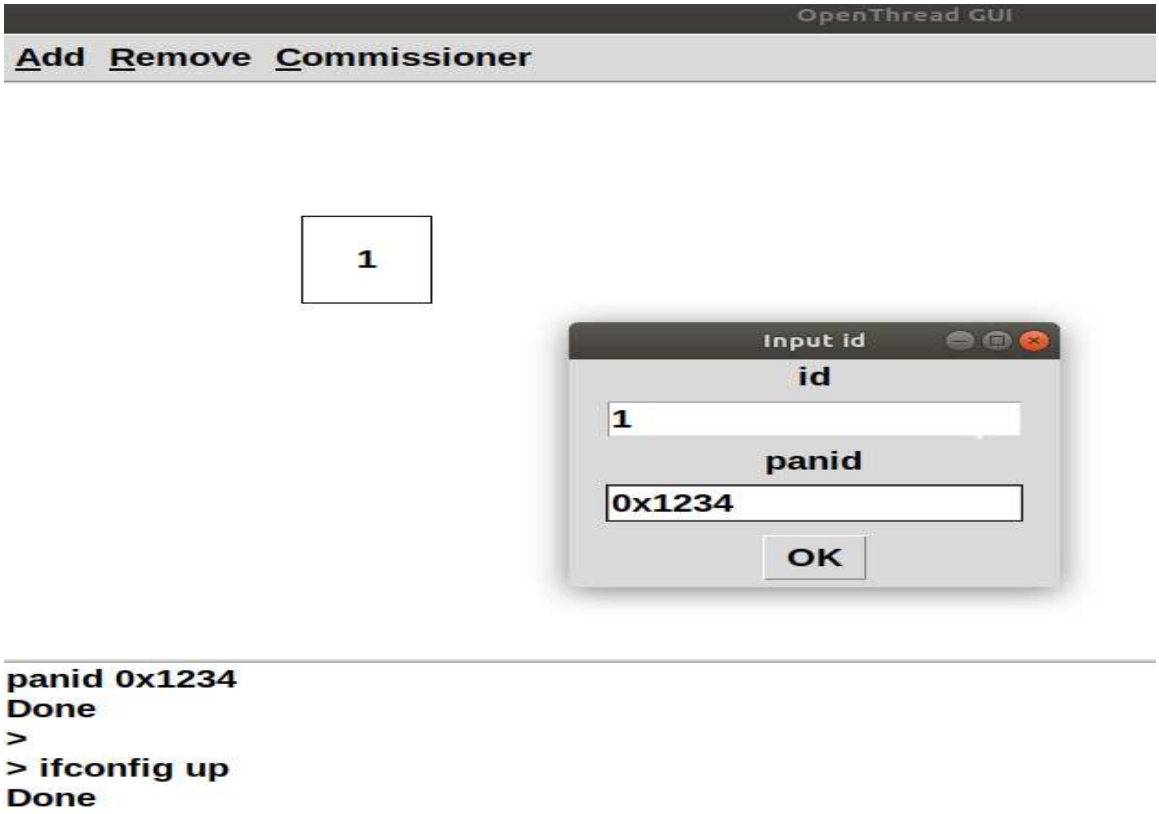

Figure 5.5 Creating a Node.

The node with id '1' is configured to be a Leader with the panid 0x1234. And Figure 5.6 gives the state of the router. It is seen that, the node 1 is Leader. Similarly, node 2 is created with similar panid as that of the node 1. Since, having the same id, node 2 is connected to node 1.

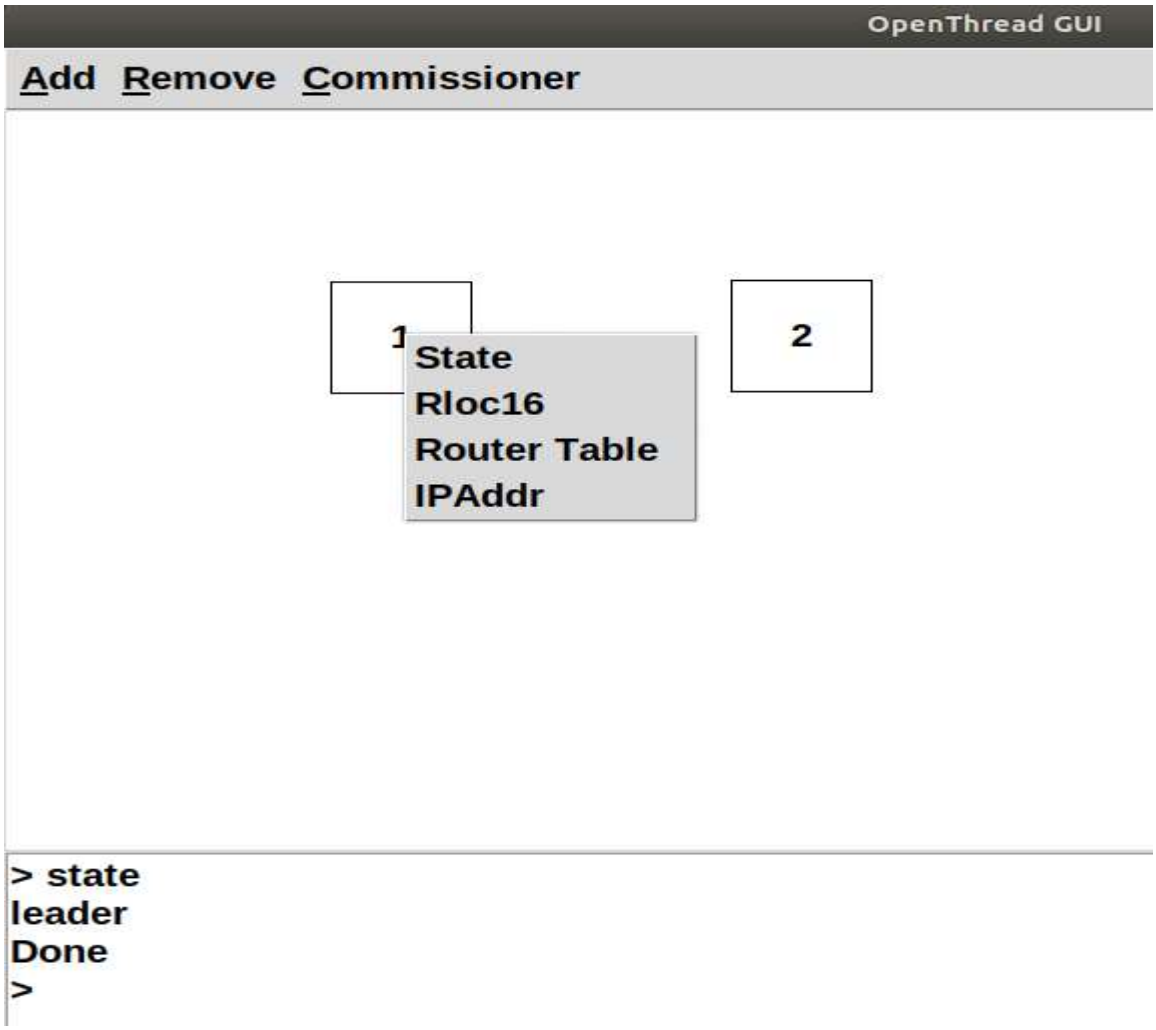

Figure 5.6 State of Node 1.

In Figure 5.7, the state of node 2 is shown. Once connected, the state of the node 2 is showed as router. Hence, we have configured node 1 to be a Leader and node 2 is the router connected to the Leader.

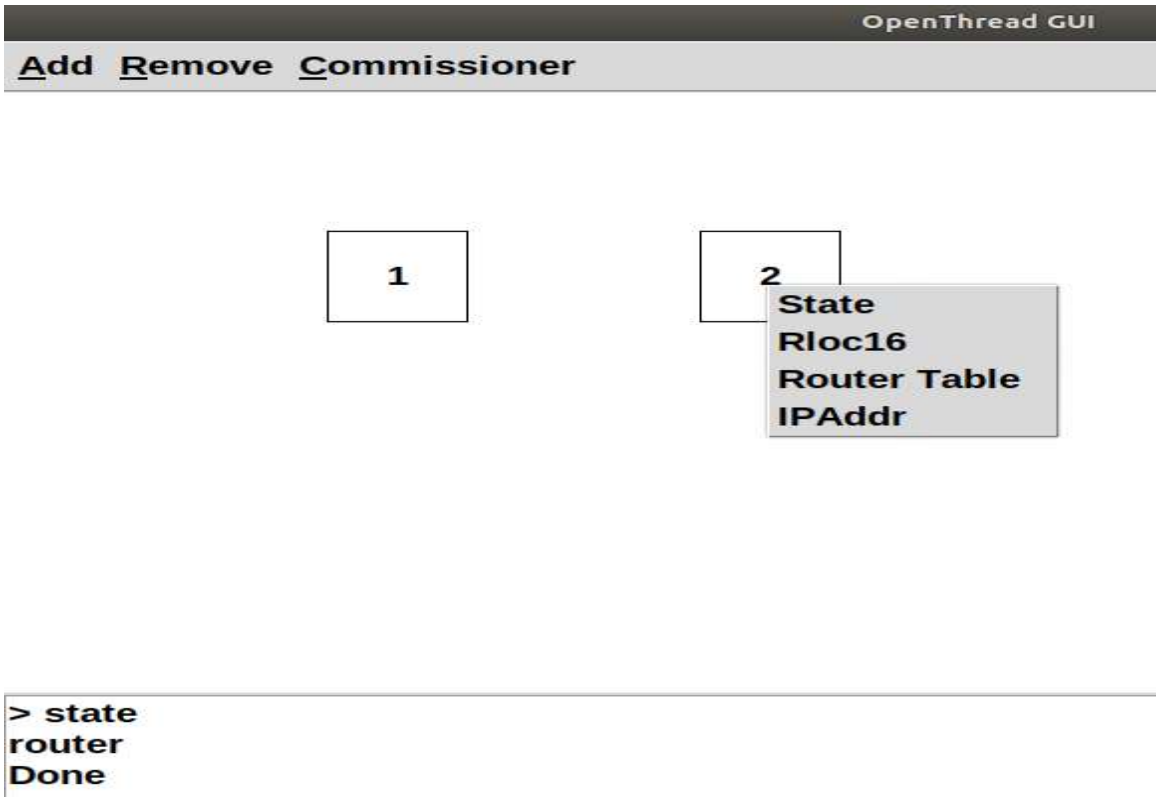

Figure 5.7 State of Node 2.

Figure 5.8 and Figure 5.9 proves the connection between the Leader and the router. In Figure 5.9 we get the Rloc16 which is the 16 bites End Point Identifier (EID) of the router node. And in the Figure 5.9, we get the router table of node 1 i.e. Leader node.

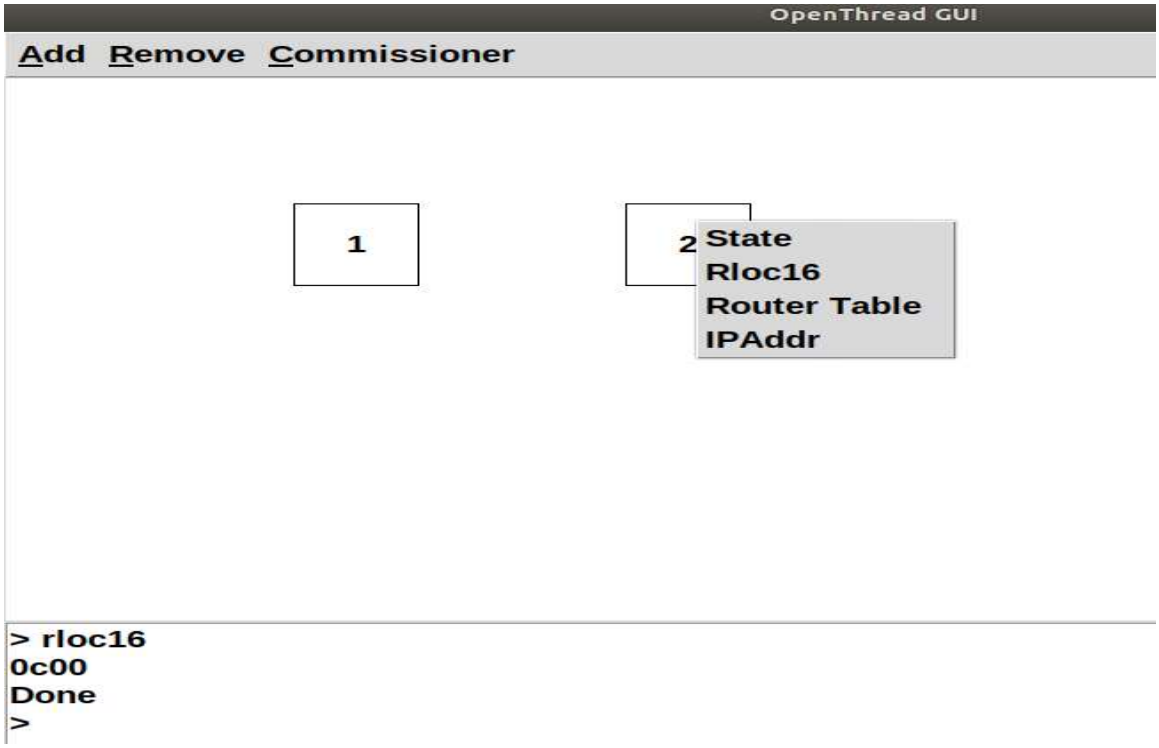

Figure 5.8 rloc16 of node 2.

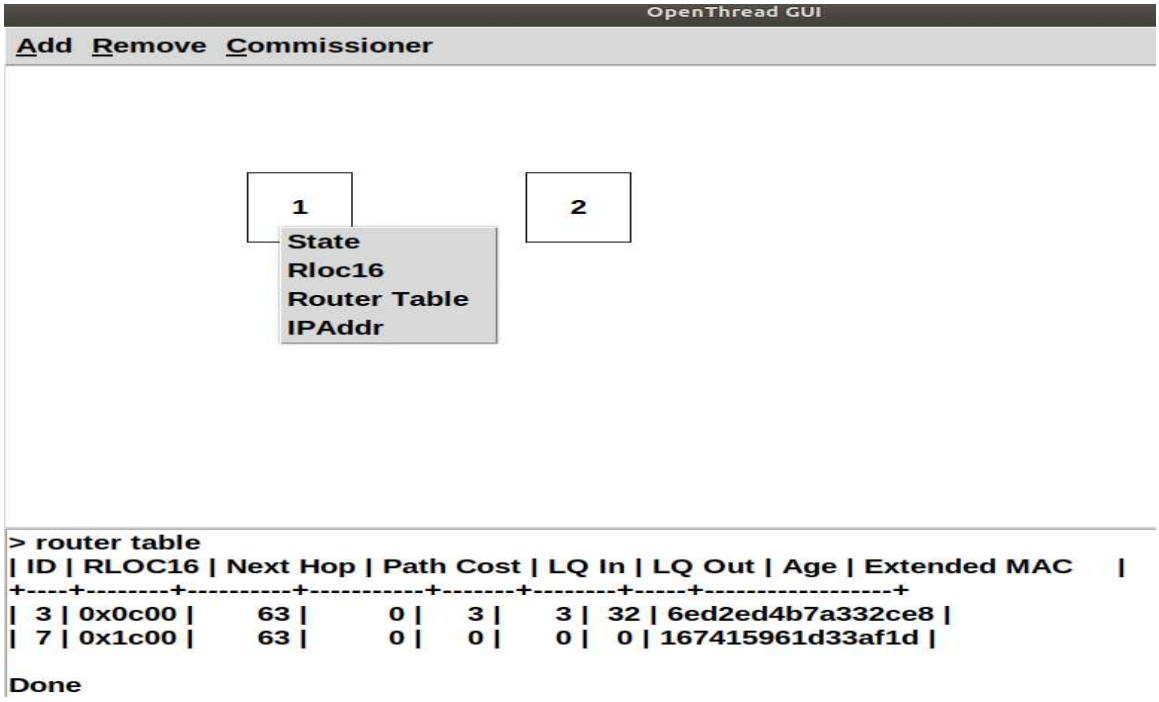

Figure 5.9 Routing table of Node 1.

Hence, we prove that the Leader and the router nodes are connected to each other

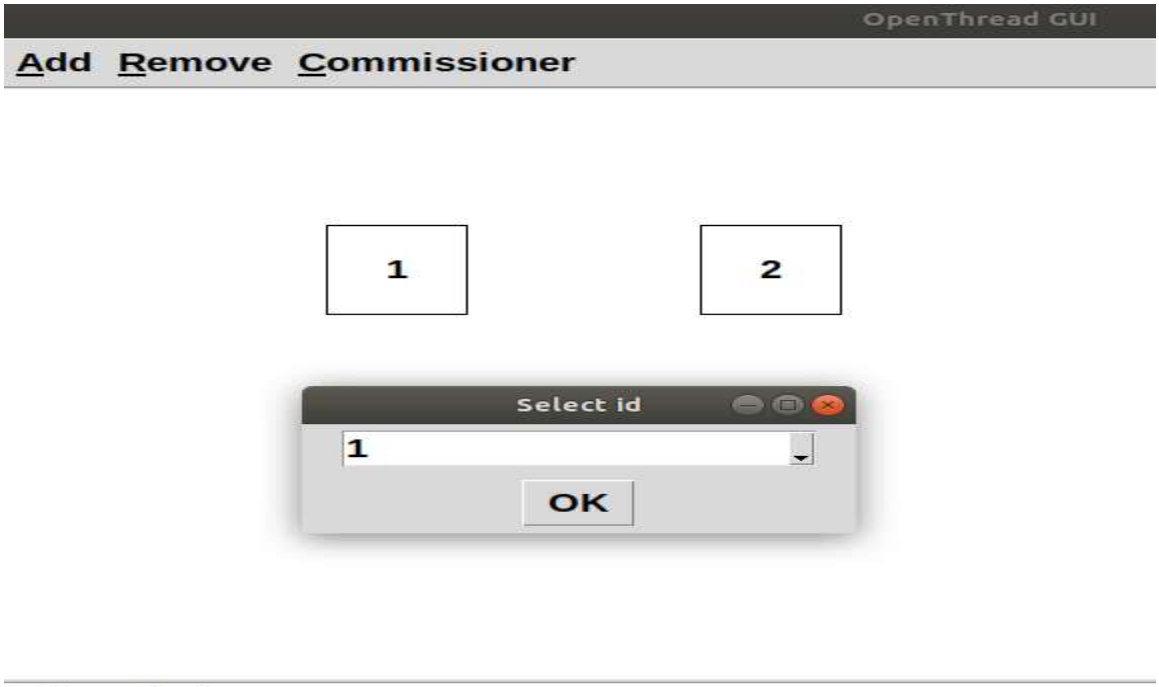

Figure 5.10 Disabling Node 1.

In Figure 5.10, we have disabled the Leader router using “stop” from “remove”. The router table of the Leader has the has the 16 bites End Point Identifier (EID) of the router node. It is seen from Figure 5.9; the EID value is 0x0c00 and the same value is in the router table of the Leader router as show in Figure 5.9.

In Figure 5.11, the state of the Leader, i.e. node 1 is disabled and according to Thread protocol, when a Leader is disabled, the next available router becomes Leader and a REED, becomes a router, this ensuring no point of failure.

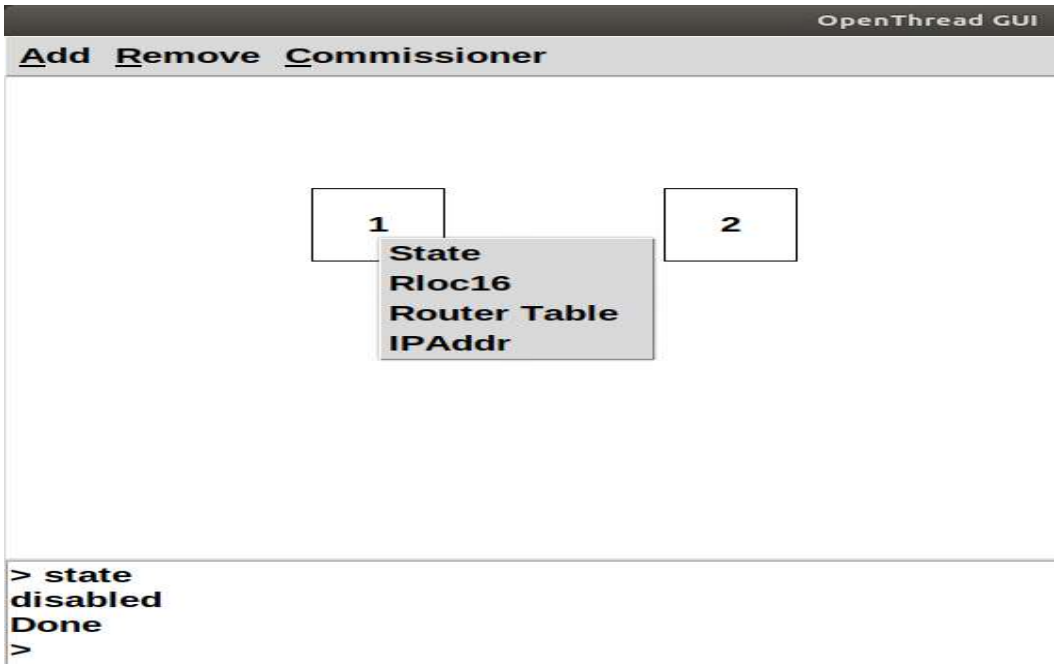

Figure 5.11 State of Node 1.

In Figure 5.12, it is observed that the node 2, which was a router when the Leader was active became a Leader when the Leader was in a disabled state.

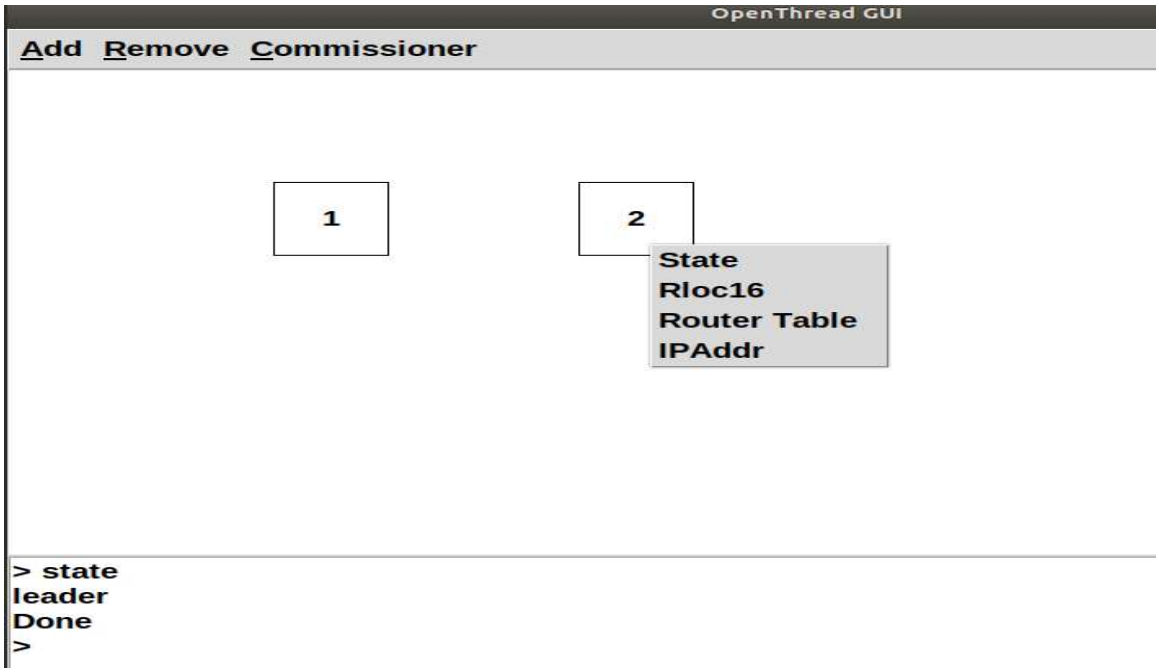

Figure 5.12 State of Node 2.

In Figure 5.13, we demonstrate the Commissioner and Joiner functions in the thread protocol.

Figure 5.13 Configuring the Commissioner

In Figure 5.14, we configured the node 2 to be Joiner by selecting the node id and giving the Joiner key i.e. “joinme”. And in the result box, there is a confirmation which shows that the connection was success and the Joiner joined the network. We created 2 nodes, where node 1 is the Leader and node 2 is the Joiner. Node 1 is configured to be a Commissioner. In Figure 5.13, we assigned the Leader with a panid and a user defined key i.e. “joinme”.

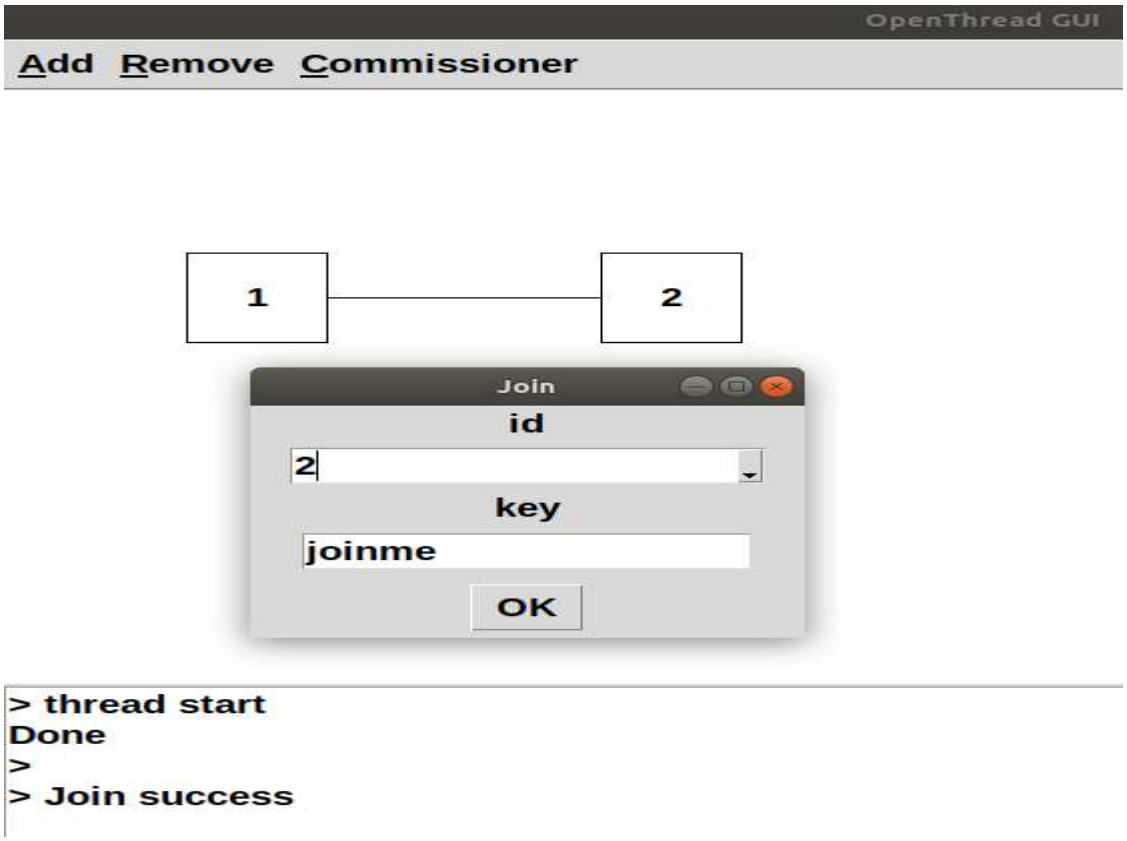

Figure 5.14 Configuring the Joiner

This shows how OpenThread works on the CLI command Line Interface level. But instead of having a command prompt for each node, this GUI will make the research involving OpenThread much easier to work with.

CHAPTER VI:

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

Thread Group Inc. began from an arrangement of necessities to create Thread. They built up a vigorous open standard which is seller impartial. One particular confinement of Thread is that it isn't really interoperable with other IoT structures like AllJoyn, ZigBee, Z-Wave, and so on.

The layers characterized by Thread depend on either a current standard or an RFC. No new norms or RFCs have been proposed by Thread. One of the studies of current circumstance in IoT commercial center is that, IoT devices from various sellers and conventions don't work with one another. Each IoT convention needs a different center and application to control it. This is because of various legitimacy frameworks in the market today.

OpenThread is an open source implementation of the thread networking protocol. Major set back for OpenThread is, Command Line Interface (CLI) is used for configuring nodes and each node requires a separate command prompt which will be very hard to keep track of when the user is using large number of nodes. Hence, a GUI will not only makes things easier for a developer but also the developer doesn’t need to remember the commands used to configure a node using OpenThread.

6.2 Future works

This GUI gives the flexibility of adding any new commands used on Command Line Interface using OpenThread. OpenThread has a problem of timing restraint. It takes about 2 minutes for the node to change the state from router to the Leader. The user given key for commissioning also expires under 2 minutes and is invalid. [31]. Hence, in order to join a new device into the network, the user have to give a new key.

These problems in the OpenThread can be a starting point for research and for obtaining better results on the GUI. User can create nodes using OpenThread on Command Line Interface (CLI), but cannot change the values like “next hop, channel, Link Quality in, Link Quality out.” Adding functions that can show simulation to the OpenThread like checking the quality of the transmission when noise is introduced, can add a great value and increases the scope of OpenThread. Any function added to the OpenThread can be added to the GUI hence, using the GUI as a simulator.

It also adds the flexibility of changing the simulation in the GUI according to the requirements. OpenThread is also used in hardware, and still needs the Command Line Interface (CLI) to operate. GUI again can make the life of the developers and tester easier when used to operate OpenThread dumped hardware.

Since the GUI is purely written in TCL, adding TCL scripts to the OpenThread can be more useful for extending the functions of the GUI. Increasing the functionality of the GUI will make its operations easier.

REFERENCES

1. Aston, Kevin. "That 'Internet of Things' Thing", Available: http://www.rfidjournal.com/articles/view?4986, accessed on June 2, 2017. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
2. https://en.wikipedia.org/wiki/Internet_of_things.
3. https://www.google.com/search?q=iot&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiVscD6qILfAhUp8IMKHfFXA9QQ_AUIDygC&biw=1536&bih=723#imgrc=pGwtYW9WXQO6WM:
4. https://en.wikipedia.org/wiki/IPv6.
5. https://en.wikipedia.org/wiki/6LoWPAN
6. https://en.wikipedia.org/wiki/IEEE_802.15.4
7. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038811/
8. Smart home automation system using Bluetooth technology, 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT), DOI: 10.1109/ICIEECT.2017.7916544, 04 May 2017.
9. https://en.wikipedia.org/wiki/Z-Wave
10. https://en.wikipedia.org/wiki/Zigbee
11. https://en.wikipedia.org/wiki/Internet_protocol_suite
12. https://www.threadgroup.org/
13. https://en.wikipedia.org/wiki/IEEE_802.15.4
14. https://tools.ietf.org/html/rfc4944
15. https://www.threadgroup.org/Portals/0/documents/support/6LoWPANUsage_632_2.pdf
16. https://en.wikipedia.org/wiki/Mesh_networking

17. Gartner, Inc. “Gartner Says 8.4 Billion Connected ‘Things’ Will Be in Use in 2017, Up 31 Percent From 2016”, http://www.gartner.com/newsroom/id/3598917, accessed on June 2, 2017. K. Elissa, “Title of paper if known,” unpublished.
18. Y. Dvorakian and S. Garg, "IoT-enabled distributed cyberattacks on transmission and distribution grids," 2017 North American Power Symposium (NAPS), Morgantown, WV, 2017, pp. 1-6. doi: 10.1109/NAPS.2017.8107363.
19. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
20. K. Sonar H. Upadhyay "A Survey: DDOS Attack on Internet of Things Intl", Journal of Engineering Research and Development, vol. 10 no. 11 pp.58-63.
21. S. A. P. Kumar, B. Bhargava, R. Macêdo and G. Mani, "Securing IoT-Based Cyber-Physical Human Systems against Collaborative Attacks," 2017 IEEE International Congress on Internet of Things (ICIOT), Honolulu, HI, 2017, pp. 9-16. doi: 10.1109/IEEE.ICIOT.2017.11
22. Ishaq Unwala, Jiang Lu, “IoT Protocols: Z-Wave and Thread”, November 17 Volume 3 Issue 11, International Journal on Future Revolution in Computer Science & Communication Engineering (IJFRSCE), PP: 355 – 359
23. http://www.gatedepot.com/landing-category-small/accessradio-remotes/, accessed on Feb 2, 2018
24. Lorex Technology, https://www.lorextechnology.com/
25. IEEE Standards, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs) IEEE Std 802.15.4 ™ -2006

26. Thread group Inc. website: http://threadgroup.org/Whatis-Thread/Connected-Home
27. Ishaq Unwala, Zafar Taqvi, Jiang Lu Thread: An IoT Protocol, 2018 IEEE Green Technologies Conference, **ISSN:** 2166-5478, **DOI:** 10.1109/GreenTech.2018.00037, **Accession Number:** 17823919.
28. RFC 4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks, "https://tools.ietf.org/html/rfc4944
29. RFC 6282, "Compression Format for IPv6 Datagrams over IEEE 02.15.4-Based Networks," https://tools.ietf.org/html/rfc6282
30. Humberto Gonzalez, Study of the protocol for home automation Thread. Telecommunications Engineering, February, 22nd 2017
31. https://codelabs.developers.google.com/codelabs/openthread-simulation/#3
32. Ishaq Unwala, Zafar Taqvi, Jiang Lu, IoT security: Z-wave and Thread. 2018 IEEE Green Technologies Conference, ISSN**:** 2166-5478, **DOI:** 10.1109/GreenTech.2018.00040
33. https://en.wikipedia.org/wiki/Received_signal_strength_indication
34. https://openthread.io/