الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

BACHELOR OF SCIENCE IN ELECTRONIC AND TELECOMMUNICATION ENGINEERING

A Network infrastructure for Smart Home

Supervised By

Mohammad Woli Ullah
Lecturer
Department of Electronic and Telecommunication Engineering, IIUC

Submitted By

Talha Misbah Mohiuddin
Matric ID: T151032
Program: B. Sc. in ETE

Md. Kamrul Hassan Riyad
Matric ID: T151038
Program: B. Sc. in ETE

Department of Electronic and Telecommunication Engineering
International Islamic University Chittagong
Kumira, Sitakunda, Chittagong – 4318
September 2019

A Network infrastructure for Smart Home

(THIS THESIS REPORT IS SUBMITTED FOR THE PARTIAL FULFILMENT OF THE BACHELOR OF SCIENCE IN ELECTRONIC AND TELECOMMUNICATION ENGINEERING)

Submitted By

Talha Misbah Mohiuddin
Matric ID: T151032
Program: B. Sc. in ETE

Md. Kamrul Hassan Riyad
Matric ID: T151038
Program: B. Sc. in ETE

Supervised By

Mohammad Woli Ullah
Lecturer
Department of Electronic and Telecommunication Engineering, IIUC

Department of Electronic and Telecommunication Engineering
International Islamic University Chittagong
Kumira, Sitakunda, Chittagong – 4318

DEDICATION

This thesis work is dedicated to all of our honorable teachers and parents.

CERTIFICATE OF APPROVAL

The paper entitled as "A Network infrastructure for Smart Home" submitted by Talha Misbah Mohiuddin, bearing ID No: T-151032 and Md. Kamrul Hassan Riyad, bearing ID No: T-151038, to the Department of Electronic and Telecommunication Engineering (ETE) of International Islamic University Chittagong (IIUC) has been accepted as satisfactory for the partial fulfillment of the requirements for the Degree of Bachelor in Electronic and Telecommunication Engineering and approved as to its style and contents for the examination held on 14th September 2019.

Approved by

Mohammad Woli Ullah

Supervisor

Lecturer

Department of Electronics and Telecommunication Engineering

Faculty of Science and Engineering

International Islamic University Chittagong

Kumira, Sitakunda, Chittagong.

CANDIDATES DECLARATION

It is hereby declared that the work presented in this thesis has not been submitted elsewhere for the award of any degree or diploma, does not contain any unlawful statement.

_____

Talha Misbah Mohiuddin

T-151032

_____

Kamrul Hassan Riyad

T-151038

Acknowledgements

In the name of Allah, the Entirely Merciful, the Especially Merciful. All praises and glory is to Allah (SWT) for blessing us with opportunities abound and showering upon us His mercy and guidance all through the life. And may peace and blessings of Allah be upon Prophet Muhammad (pbuh), a guidance and inspiration to our lives. We express our sincere gratitude and indebtedness to our thesis supervisor Mohammad Woli Ullah, for his initiative in this field of research, valuable guidance, encouragement throughout the time of research. We express our thankfulness to Md. Jashim Uddin, Chairman of the Department of Electronic and Telecommunication Engineering, IIUC for providing us with best facilities in the department and his timely suggestions. We are also thankful to Engr. Syed Zahidur Rashid, Advisor of 12th batch for his dedication and sacrifice. We also express our sincere gratitude to all our teacher for giving us the best effort throughout our entire academic years. We also remember our parents for their support in our life till now. Finally, we also express gratitude to our friends, well-wishers for their directly or indirectly involvement in the completion of this thesis work.

Abstract

There are uncertainties with regard to which network technologies will remain current and adaptable to emerging changes in Smart Home requirements. This research is based on using the existing and available network technologies for Smart Homes. In particular, it used CSMA network and Wifi Network based on IEEE 802.11ac standard. It is expected that a usual home has to upgrade the infrastructural network to fully upgrade into Smart home. So, the focus is to make a building design, based on Bangladesh's prospective and use CSMA and WiFi network individually in simulation.

Table of Contents

Table of Figure

Table of Table

Chapter 1

Introduction

1.1. Smart Home

This the era of the combination of technology and smartness. Technology is advancing day by day. In the past, smart homes were considered to be part of a luxurious life style but today It becomes a necessity to upgrade our homes into smart home.

Figure 1. 1 Smart Home

A home that uses both technology and process to create an environment that is safe, responsive, adaptive and comfortable for its occupants is said to be smart home. [1] In short, if a residence have a number of devices that automate tasks normally handled by humans and controlled by humans then it's a smart home. [2] Sometimes technologies are built into the structure itself and sometimes added later.

1.2. Why Smart Home?

Smart home has the potential to make your home safer, better, efficient. Some of the basic reasons discussed below.

- For many people, the thought behind creating a smart home is the potential to save energy and money. Automate the lights, fan, air-conditions and appliances of a home usually reduces the owners' electric bill. The cost of electricity is not constant through the whole day. Smart home allows users to use electricity according to their budget.
- In Bangladesh, Fire and theft and other small crimes are now very common and in this era. Smart homes offer an added security so that users don't have to buy security system. Connected lights, cameras, doorbells and even heat sensor can help make a home that much safer. Monitoring the security of home through smart phones became very easy.
- It made easy to handle regular household by using automatic devices. Like it can detect the temperature state it is in and make adjustments to itself. Automatic door control, lights control all of them could be done more easily.
- Smart homes make life comfortable. As an example, turning on off lights, fans from laying in the bed.
- A smart home also gives a peace of mind. User don't have to worry to check the doors, windows, water tank fill up etc.
- Smart homes also allows to have electronic things the way user like. Shades drawn automatically at a certain time, adjust the brightness of lighting, customize every single electronic device can be set as user preference.

1.3. Communication System

There are a bunch of communication system which can be use as smart homes communication system. Most interesting thing is that there is no fixed communication system to use in smart homes. So there are researches about which will be the best among different types of communication systems. Here a CSMA and a WiFi based communication system is researched in prospective of Bangladesh.

1.3.1 CSMA

Carrier Sense Multiple Access is implemented in Ethernet networks with more than one computer or network device attached to it. CSMA is part of the Media Access Control (MAC) protocol. [3] CSMA works when a device needs to initiate or transfer data over the network. The CSMA device models a simple bus network in the spirit of Ethernet. Although it does not model any real physical network but it does provide some very useful functionality.

Typically, when one thinks of a bus network Ethernet or IEEE 802.3 comes to mind, CSMA/CD (Carrier Sense Multiple Access with Collision Detection). In this paper CSMA device models only a portion of this process, using the nature of the globally available channel to provide instantaneous carrier sense and priority-based collision.

In layered communications architectures the CsmaNetDevice and CsmaChannel pair occupies the physical layer, and data link layer. Another reference model that required for Internet Hosts occupies the communication layers. From the IEEE 802 standards which speak of LLC (Logical Link Control), MAC (Media Access Control), MII (Media Independent Interface) and PHY (Physical Layer) layering. In this case the *LLC* and *MAC* are sublayers of the OSI data link layer and the *MII* and *PHY* are sublayers of the OSI physical layer. The "top" of the CSMA device defines the transition from the network layer to the data link layer. [4]

1.3.2 WiFi

A WiFi Network is a wireless network that connects to Internet router and wireless-enabled devices using a wireless radio signal. [5] A WiFi network gives the convenience and mobility to access shared files, Internet connection, and other wireless devices from any location within your network's coverage area.

The term stands for Wireless Fidelity and means that a device is compatible with the 802.11 standard, the established standard for wireless networks. [6]

1.4. Network Simulator 3

Network simulator is a tool used for simulating the real world network on one computer by writing scripts in C++ or Python. Normally if we want to perform experiments, to see how our network works using various parameters. We don't have required number

of computers and routers for making different topologies. Even if we have these resources it is very expensive to build such a network for experiment purposes.

NS3 is a discrete event network simulator for Internet. It helps to create various virtual nodes and with the help of various Helper classes it allows us to install devices, internet stacks, application, etc to those nodes.

NS3 gives some special features which can be used for real life integrations. Some of these features are:

- Tracing of the nodes**:** NS3 allows to trace the routes of the nodes which helps to know how much data is send or received.
- NetAnim: It stands for Network Animator. It is an animated version of how network will look in real and how data will be transferred from one node to other.
- Pcap file: NS3 helps to generate pcap file which can be used to get all information of the packets These pcaps can be seen using a software tool known as wireshark.
- GnuPlot: GnuPlot is used to plot graphs from the data which we get from trace file of NS3. Gnuplot gives more accurate graph compare to other graph making tools and also it is less complex than other tools. [7]
- NS-3 is an free and open source simulation environment suitable for networking research.
- The ns-3 simulation core supports research on both IP and non-IP based networks.
- NS3 reuse the real application and kernel code. The Direct Code Execution framework allows users to run C++ or Python based applications or the Linux kernel networking stack within ns-3. [8]

1.5. Motivation

There are uncertainties with regard to which technologies will remain current and adaptable to emerging changes in digital home requirements, energy dynamics, as well

as smart grid infrastructure deployment. There are a lot of paper proposed an infrastructure using different types on network like PLC, Zigbee, P2P, VPN etc.

This paper proposed to build a network infrastructure with CSMA (Carrier sense multiple Access) and Wifi which is based on IEEE 802.11ac for smart home using P2P as a gateway. So that it could be possible to use these two networks as an infrastructural network of future smart homes of Bangladesh.

1.6. Thesis outline

This thesis is structured to provide a logical progression of the research conducted by the author. It shows the design, simulation and analysis of two individual network, CSMA and Wifi. To communicate this investigation, this thesis contains Five chapters and one list of references.

The rest of this thesis is organized as follows:

- **Chapter 1** introduces the topic's overview of this thesis.
- **Chapter 2** introduces a brief review of exiting Network technologies and discussed about literature reviews.
- **Chapter 3** provides details of the Methodology of this paper.
- **Chapter 4** represents the details of Design of the module, simulation results and analysis.
- **Chapter 5** presents the conclusion of the work, and the limitations. Also some future works are discussed.

Chapter 2

Literature review & Background

2.1. Communication Technologies

There are two kind of communication technology available. Wired and Wireless. Both kinds are used based on their prospective.

2.1.1. Wired Communication

The technologies are discussed below.

Fiber-Optic Communication

This communication technology is used for WAN (Wide Area Network). There are several forms of optical fiber.

- PON (Passive Optical Network)
  It is a point-to-multipoint network architecture, which utilizes optical splitters to enable a single optical fiber to serve multiple customers.
- WDM (Wavelength Division Multiplexing)
  It is an effective technology to exploit bandwidth capacity available in optical fiber. Multiple wavelengths are used to carry several data streams simultaneously over the same fiber in WDM networks.
- SONET/SDH (Synchronous Optical Networking / Synchronous Digital Hierarchy)
  It is a time-division multiplexing (TDM) architecture that was designed to carry high capacity traffic. SONET and SDH are essentially the same standard. [9]

Digital Subscriber Line (DSL)

This is a digital data transmission technology on ordinary telephone lines. There are three DSL systems:

- ADSL (Asymmetric DSL)
  It provides different data rates in two directions.

- HDSL (High speed DSL)

  This system supports data rate of up to 2.048 Mbps over a distance of 3.6 km.
- VDSL (Very high data rate DSL)

  It is a DSL technology provides upto 100 Mbps data rate. [9]

Coaxial Cable

This is similar to DSL. It provides high-speed data network-edge connectivity. Coaxial cable mainly designed for broadcast services, i.e., television and radio channels. Data Over Cable Service Interface Specification (DOCSIS) is an international coaxial cable communication technology that provides high-speed data transfer. In smart environment, It could share the bandwidth between home devices and also among the user.

Carrier-sense multiple access (CSMA)

This is a media access control (MAC) protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus or a band of the electromagnetic spectrum. Basically there is no physical appearance of CSMA, but using its collision detection it will use as a bus connection.

Power line communication (PLC)

This is a wired communication technology that enables data transmission over existing power lines. This technique injects a high frequency carrier into power lines and modulates the carrier with the data to be transmitted. It is helpful for the smart devices. However, it has significant technical issues such as it cant transmit signals cross a transformer, power line channel distortion, noise, interference and security concerns.

2.1.2. Wireless Communication

The technologies are discussed below.

ZigBee

This is a wireless personal area network protocol based on the IEEE 802.15.4 standard. It operates on the unlicensed ISM (industrial, scientific and medical) bands: 868 MHz in Europe, 915 MHz in the U.S., and 2.4 GHz worldwide. Its data rates range from 20kbps to 250kbps, that is: 250kbps at 2.4GHz, 40kbps at 915MHz, and 20kbps at

868MHz. ZigBee provides coverage distance of up to 100 meters, while ZigBee Pro provides coverage distance of up to 1,600 meters. ZigBee is widely used for home/building automation, energy monitoring, industrial plant management, as well as AMI applications . It supports various network topologies such as star, tree and mesh topologies, as well as a robust security layer with 128-bit AES encryption. [9]

Wireless Local Area Network (WLAN)

This is a high-speed wireless Internet and network communication technology, which is commonly known as Wi-Fi. It is based on the IEEE 802.11 series of standards including 802.11, 802.11a, 802.11b, 802.11g and 802.11n. WLAN operates in 2.4 GHz, 3.6 GHz and 5 GHz unlicensed ISM frequency bands. The IEEE 802.11x standards specify data rates from 2 Mbps to 600 Mbps, and the coverage range of up to 100 meters. WLAN communication is another promising alternative for HANs and NANs. [9]

Wireless Mesh

This is a cost-effective, robust and flexible wireless network which consists of many nodes including mesh clients and routers. In a wireless mesh network, each node can act as a signal repeater and automatically route messages from one node to another. This is known as dynamic routing. Mesh networks can be implemented with various wireless technologies, i.e., 802.11, 802.15 and 802.16.

Z-Wave

This is a reliable, low-power, low-cost proprietary wireless technology that is suitable for short-range communications. It operates in the 900 MHz ISM. It supports data rate of up to 40 kbps and the coverage distance of up to 30 meters. Z-Wave is designed specifically for remote control applications in residential and light commercial environments. It can be applied to smart grid applications in HANs. Z-Wave supports mesh networks, which makes it a good candidate for HANs. However, it provides short-range communications and has low data transmission rate.

WiMAX

This is a 4G wireless technology based on the IEEE 802.16 series of standards, i.e., IEEE 802.16-2004, 802.16e, for Metropolitan Area Networks (MAN). WiMAX

operates in the 2.3, 2.5, 3.3, and 3.5 GHz frequency bands, as well as the unlicensed 5.8 GHz band. It provides data rates of up to 75 Mbps with a coverage distance of 50 km, and has low latency (10-50 ms). [9]

Cellular

This is a radio network that uses a large number of transmitters to create cells. Cellular systems allow reusing frequencies to increase both coverage and capacity. Telecommunications industry divides cellular technologies into four generations that are labeled 1G, 2G (GSM), 3G (UMTS), and 4G (WiMAX and LTE) with intermediate versions labeled 2.5G (GPRS and EDGE) and 3.5G (HSPA). Cellular systems commonly operate in 850, 900, 1800, and 1900 MHz frequency bands. [9]

Satellite Communication

This is a wireless communication technology for transferring signals between two nodes using a satellite. This is also known as space communication. In satellite communication, the modulated signal is sent towards the satellite. Then the satellite amplifies the signal and sent it back to the receiver on the earth's surface. It provides a global coverage between any pair of communication nodes, and provides date rate of up to 1 Mbps.

2.2. Simulation Tools

Network simulation is the process by which a computer network is modelled by identifying, analyzing and quantifying the interaction between various network devices and software.

There exist several simulation tools that could be used for throughput analysis of smart homes.

- Riverbed – Previously known as OPNET. It can simulate network behavior form single PTP (Point to point) topology to a global network as ISP (Internet service provider). It provides GUI (Graphical user interface) to construct the topology. Important feature of this modeler is splitting the simulation proces into three parts, (I) design of the network (II) simulation (III) analysis.
- OMNET++ – Also one of the simulators that features a GUI. This simulator provides extensions for many networking scenarios .

- NS-2 (Network simulator 2) – Text interface simulator created in 1995. It uses C++ and OTcl programming languages to create scenarios.
- NS-3 (Network simulator 3) – Very similar simulator to NS-2, it also uses the text interface for creating topologies, but for writing the code it is used C++ in combination with python scripts. [10]

2.3. Paper review

1. Research paper on "Impact of Smart Grid on the Connected Home"

In this paper, they focused on gathering data from technology providers, utilities, and other organizations, as well as stakeholders. Then the balanced their data about current status of smart home solutions and technologies. Then they researched a landscape for smart home depending on the current impacts of smart grid over smart homes. [1]

2. Research paper on "Performance analysis of wireless network based on IEEE 802.15.4 in smart home environment"

In this paper, they proposed an architecture of smart home with CSMA/CA network based on the analyses of IEEE802.15.4 standards. They used OPNET simulator for the module. They discussed some superframe order value on the network throughput and end-to-end delay. [11]

3. Research paper on "Research and Implementation of Wireless Home Network in Smart Home"

In this paper, they proposed a software and hardware model using Zigbee network for smart homes. They designed scheme including acquisition data accuracy, stability and performance of power consumption. [10]

4. Research paper on "A power line communication network infrastructure for the smart home"

In this paper, they proposed a PLC network in the existing home infrastructure. They showed that the PLC network can provide 3 low bit rate or 3 medium bit rate multimedia streams concurrently with no packet drops and jitters. For private home networks they connected PLC within homes to the internet. [13]

5. Research paper on “Simulations of Network Bottlenecks in Smart Grids”

In this paper, they simulated a PLC network using ns-3 simulator. They calculated the throughput value for the network bottleneck for different scenarios to be exact 10 scenarios. They considered low quality of the cables which causes multiple repeaters on communication line and impact of transmission protocols. [12]

6. Research paper on “Device-to-Device Communication for Smart Grid Devices”

In this paper, they propoesd a network using Peer-to-Peer (P2P) and decentralized Virtual Private Network (VPN) as the gateway of the homes. Here they considered the home as an agent. They provided a comparison of different approaches of P2P networks and mesh-VPNs. [14]

7. Research paper on “Assessment of communication technologies and network requirements for different smart grid applications”

In this paper, they did a comparative research on different communication technologies. The comparison was based on their data rates and coverage ranges. The also considered different smart grid applications requirements. Which helped a lot for selecting the network for smart homes. [9]

8. Research paper on “The Strategy of Smart Home Control System Design based on Wireless Network”

In this paper, firstly they proposed a Zigbee network for smart home based on star network as its network topology. Which manages the initial setting of terminal node and prevents illegal access. Secondly, they proposed a software and a hardware for the network which controllable by the owner of the smart home. [15]

9. Research paper on “Study on Architecture of Smart Home Management System and Key Devices”

In this paper they proposed an individual network for gateway, socket and terminal of a smart home. But different network protocol can work together. Basically, they proposed a whole smart home management system. They also designed a software run on Android PAD. [16]

10. Research paper on “On the Design of Gateway Node for Smart Gird Home Network”

In this paper, researcher proposed a hardware analysis and design using wireless network which can deliver billing information to the user through gateway. Which will help the user to maintain the electricity according to his budget. They also tested it in an 8 storied building. [17]

2.4. Research objectives

This research program addressed the following objectives:

- To study the design and modeling of a smart home network infrastructure.
- To study the various communication system.
- To study about the design, optimization and simulation in NS-3 software.
- To study about the design, optimize and simulate a CSMA network infrastructure in Bangladesh prospective.
- To study about the design, optimize and simulate a Wifi network infrastructure in Bangladesh prospective.

Chapter 3

Methodology

3.1. Methodology

methodology includes some topics like: How go about your research? What overall strategy adopted and why? What design and techniques used? [17] So, we already told that smart home is not for luxury anymore. It became necessity. To upgrade to a smart home, it's a must to select a efficient network for the infrastructure. Among all the network, we decided to go with CSMA and WiFi. Firstly, because it's never simulated before. Secondly these are the cheapest and available networks in Bangladesh. During this research we adopted some techniques. Those are discussed in this chapter.

3.2 Research Design

This research has been designed in provisional way. Those are:

- Study on smart home. It's most important to know about Smart homes first. We learned some basic which makes a normal home smart. There could be only one feature or multiple to make a smart home. It's totally depending on the user and his capacity. So bearing that in mind we can convert a home into a smart home.
- Study on communication networking. We studied types of communication systems. A bunch of them can be used in homes infrastructure. Then we selected network depending on Bangladesh's economical situation.
- Study on simulation softwires. There are a lots of simulation softwire. Among them we chased a latest softwire which is famous now a days in the research community.
- Study the procedure of the design a model in NS-3. It's a long process. Firsty we had to learn how it works. Then how to design on NS-3. We designed a five storied building which is very common in Bangladesh.
- Study on necessary parameters. To work with our selected network, we had to learn the effective permeates for the simulation.
- Implement the design.

3.3 Simulation Procedure

Simulation procedure is discussed step by step.

Firstly, A design of a building is created in prospective of Bangladesh. Which is a 5 storied building. Which have 2 units in each floor. In our design we considered a P2P connection for connecting each flat.

Secondly, 2 individual code is created for CSMA and Wifi. This softwire can use C++ and Python. We used C++ for our simulation. We used some build in header files from NS-3.20.

Thirdly, Simulate the code and optimized it.

Fourthly, Created PCAP and TR files has been saved.

Fifthly, Created PCAP files opened in Wireshark. From there, we took the I/O graphs and flowcharts.

Sixthly, Created TR files are opened in Trace Metrics. There we analyses the throughput and goodputs.

Seventhly, We came to an conclusion.

Chapter 4

Design, Simulation and Analysis

4.1. Module

4.1.1 Module design

In this paper a module is considered as a five storied building with two units. This is almost common in Bangladesh.

CSMA Module:

Here node 0 to 9 is the Gateway of the units. They are connected with P2P network. Inside of every units 5 more nodes considered which will be the smart devices.

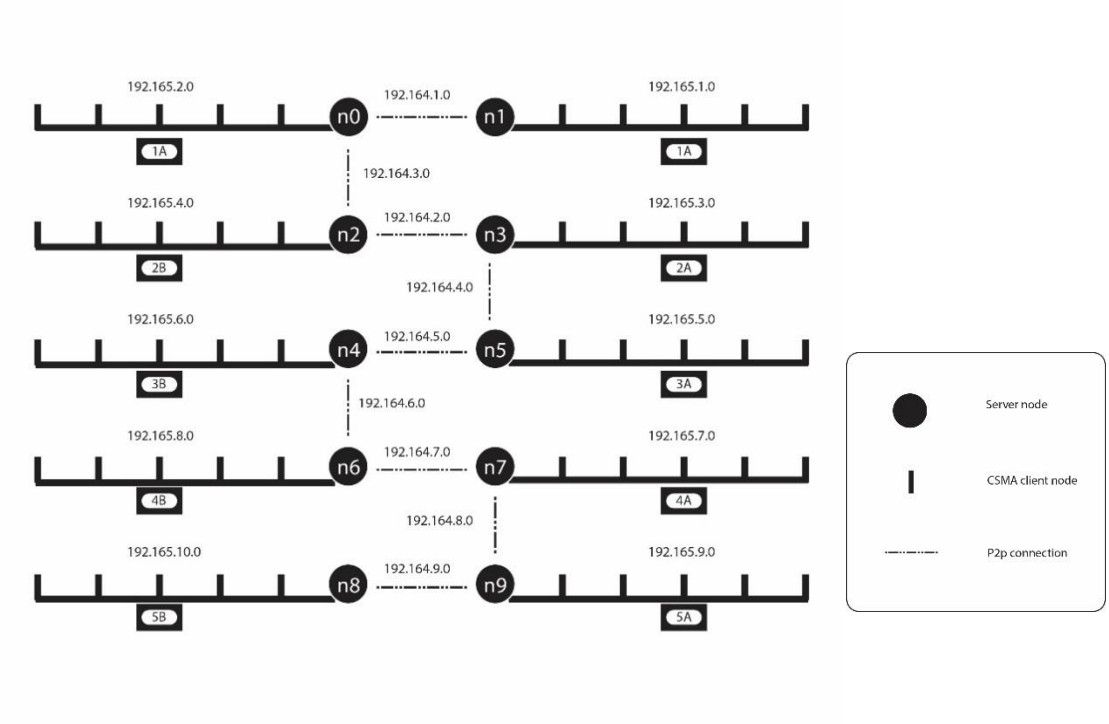

Figure 4.1 1 CSMA module

IPv4 address helper is used in this case. IP 192.165.1.0 to 192.165.10.0 is used inside of the units. In the gateway IP 192.164.1.0 to 192.164.9.0 is used. For connecting the units, we used one p2p connection per floor.

WIFI Module:

In this module node 0 to 9 is the gateway of the units or flats. They are connected with P2P connection. Inside of every home 5 more nodes considered which will be connected to the smart devices. IPv4 address helper is used in this case. IP 192.165.1.0 to192.165.10.0 is used inside of the units. In the gateway IP 192.164.1.0 to 192.164.9.0 is used. For connecting the units, we used one p2p connection per floor.

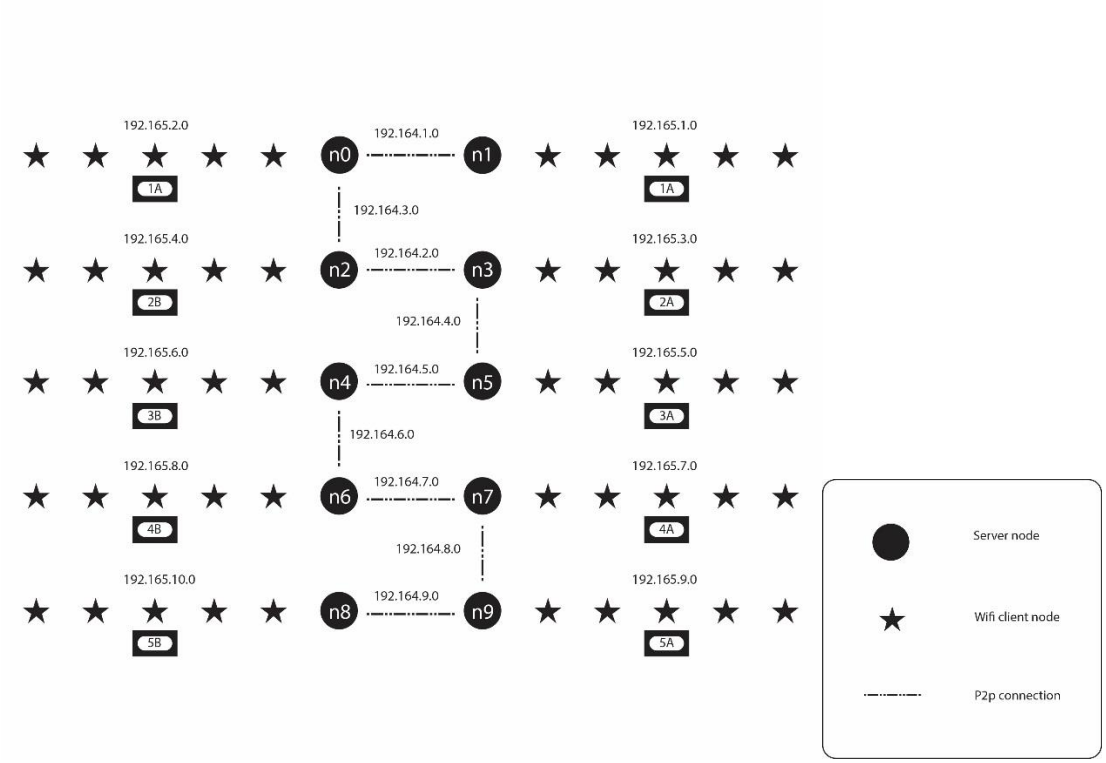

Figure 4.1 2. WiFi module

4.1.2 Simulation Parameters

Table 4.1.2. 1 Simulation Parameter

Simulation Parameter	
Packet Type	CBR (Constant Bit Rate)
Packet size	1024B
Packet Interval	1s
Data rate	100Mbps
Delay	6560ns
Transmission protocol	UDP (User Datagram Protocol)
Routing protocol	RIP (Routing Information Protocols)
Address helper	IPv4

4.2. Simulation Results

From the simulation we get an XML file, a PCAP file and a TR file. XML file is to show the visual animation of the data transfer of the module. It needs NetAnim software to run the file. To run the PCAP file Wireshark software is mostly used. Tr file refers as Asciitrace file. It can be opened with normal editor. Trace Metrice is popular to run this file.

4.2.1 CSMA Simulation

There were a lot of PCAP file for every transmission of every nodes. From those PCAP file we collected the I/O (Input/output) graph and the flowcharts.

Some I/O graphs of different nodes are shown in figure bellow. Here X-axis shows the time and Y-axis shows the packet rate. From the graph in most of the cases the first

packet rate is higher then the rest of the packet rate. Although most the packet rate is stable.

In Figure 4.2.1.1 , some packes are sent to node 0. At first the packet rate decreases .

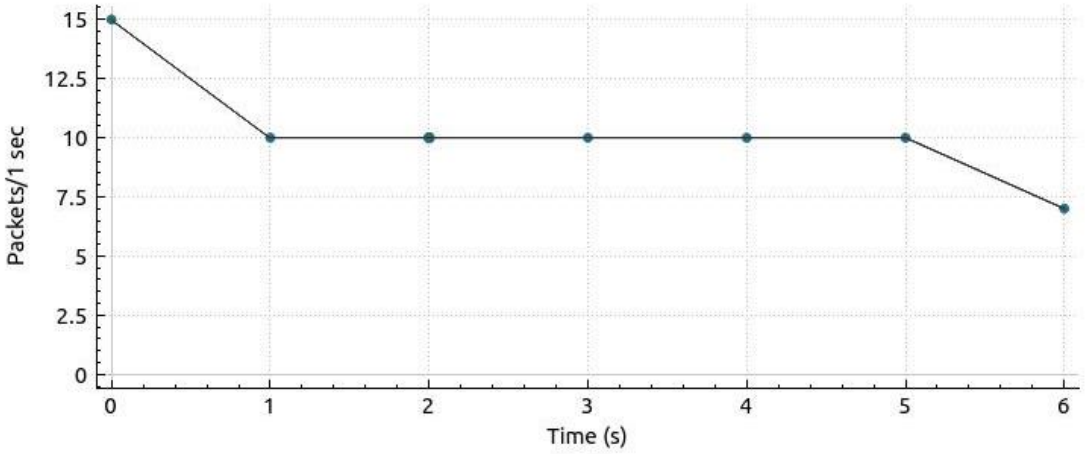

Figure 4.2.1. 1 I/O graph of node 0.

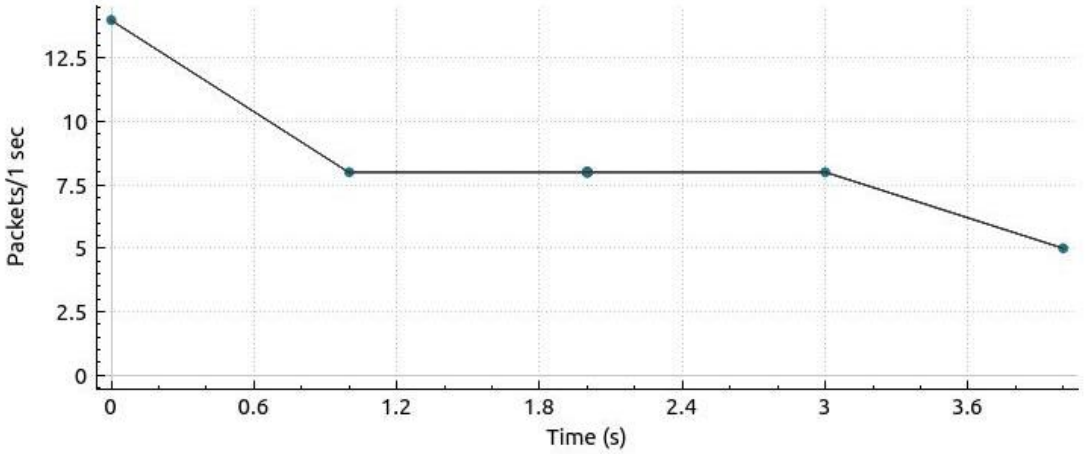

Figure 4.2.1. 2 I/O graph of node 1.

Then its becomes constant. Which is mostly the same in all graphs. In Figure 2.4.1.4 data rate increased for a small period. Its probably due to P2P connection.

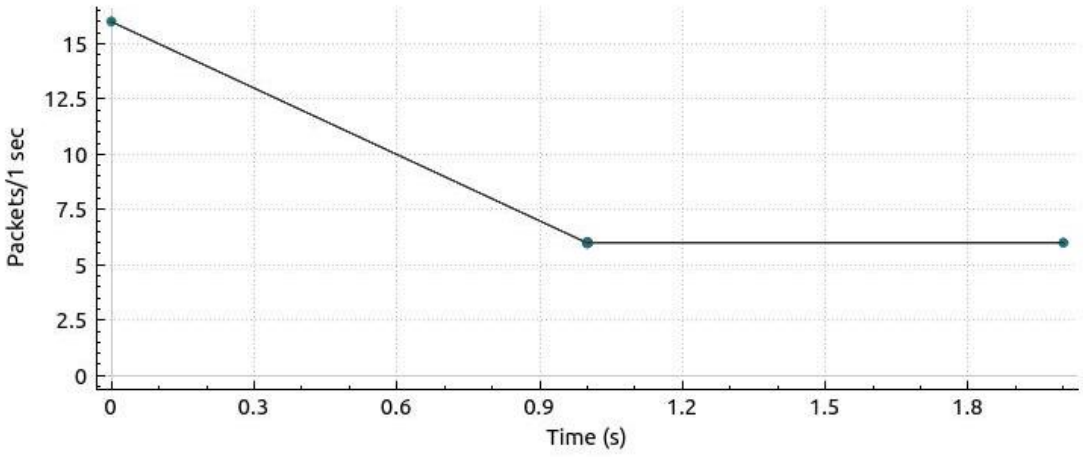

Figure 4.2.1. 3 I/O graph of node 2.

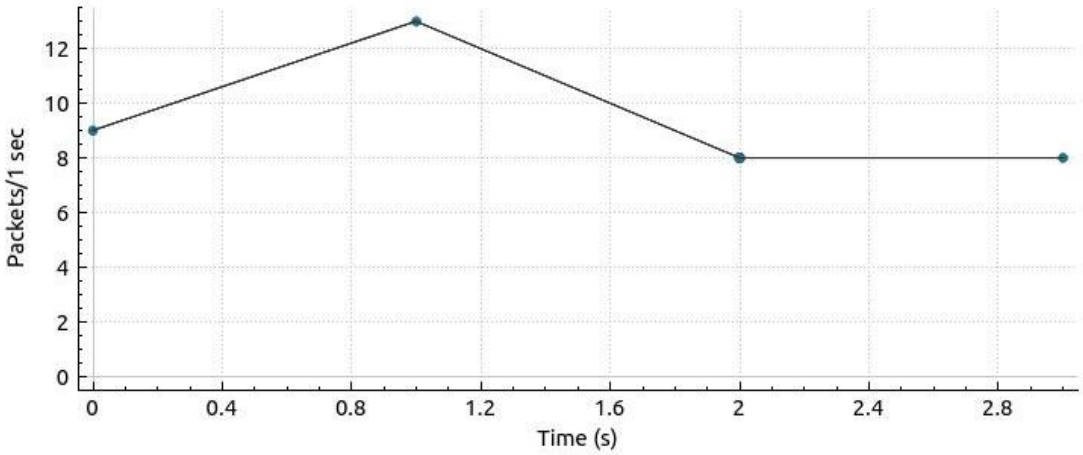

Figure 4.2.1. 4 I/O graph of node 3.

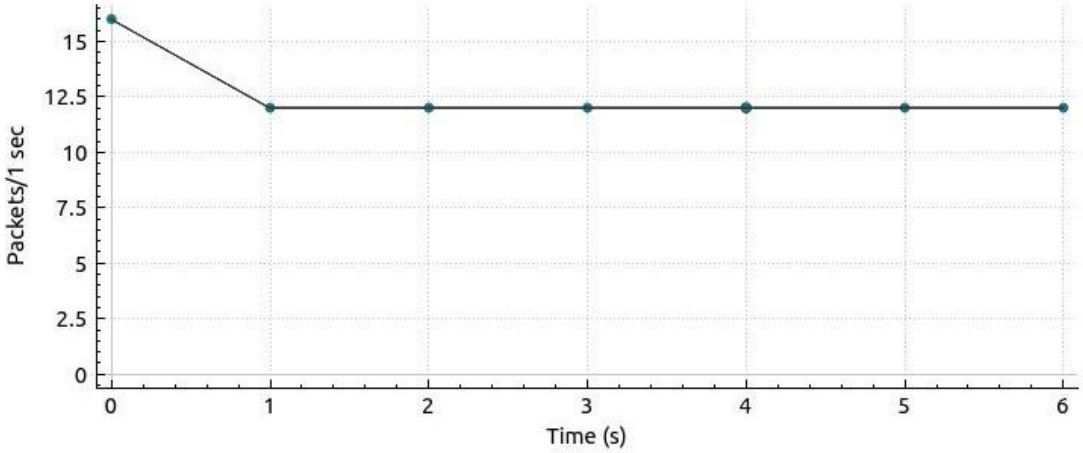

Figure 4.2.1. 5 I/O graph of node 4.

Some flowcharts of different nodes are shown in figuP re bellow. Is shows the transmission process. In which time, which node communicated with which IP is shown in this chart.

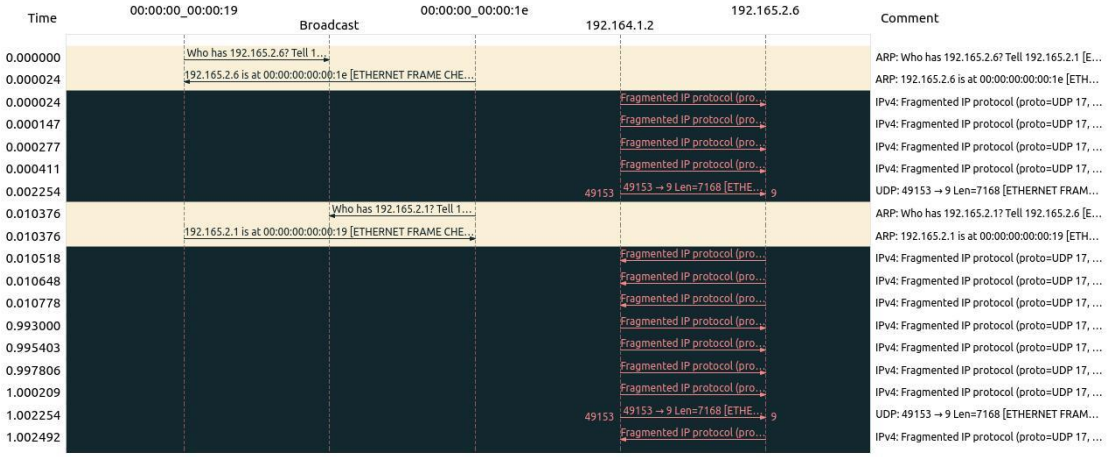

Figure 4.2.1. 6 Flowchat of node 0

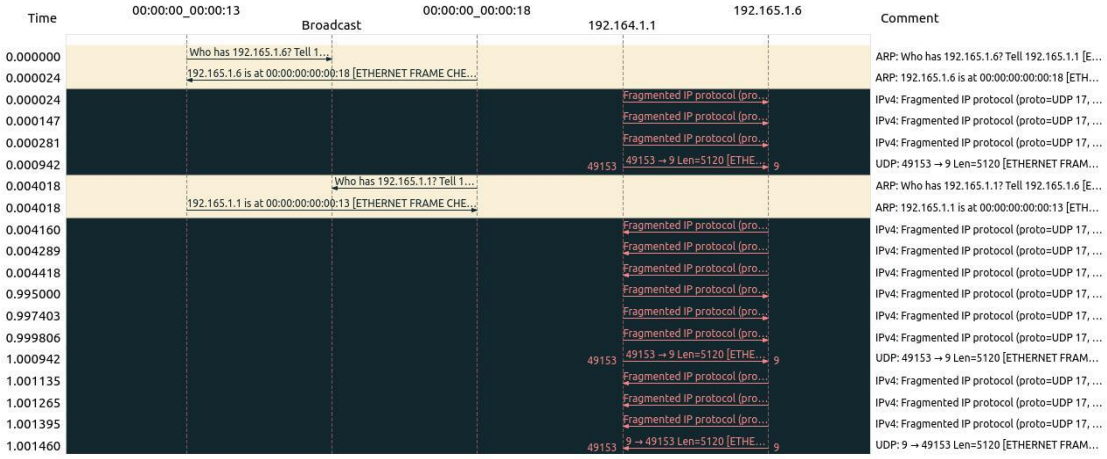

Figure 4.2.1. 7 Flowchat of node 1

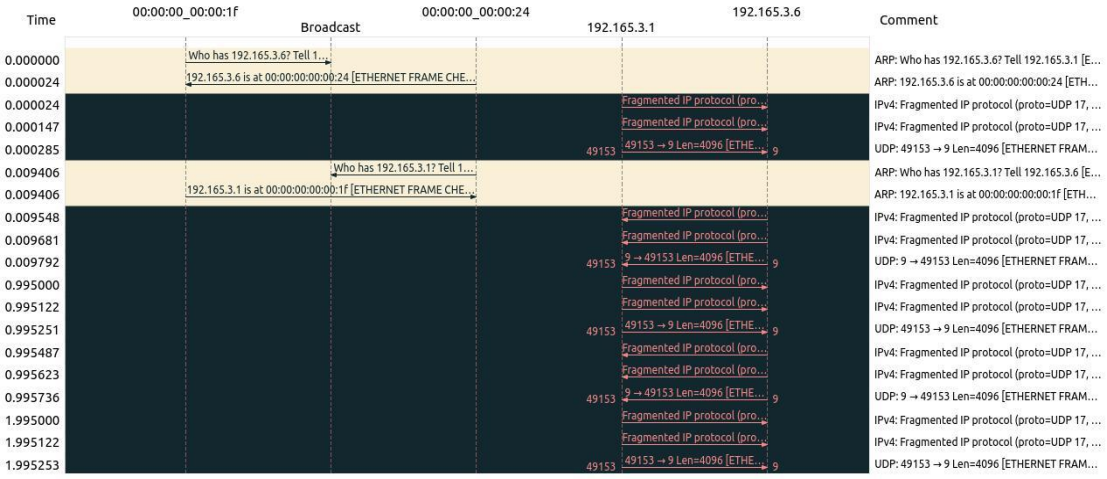

Figure 4.2.1. 8 Flowchat of node 2

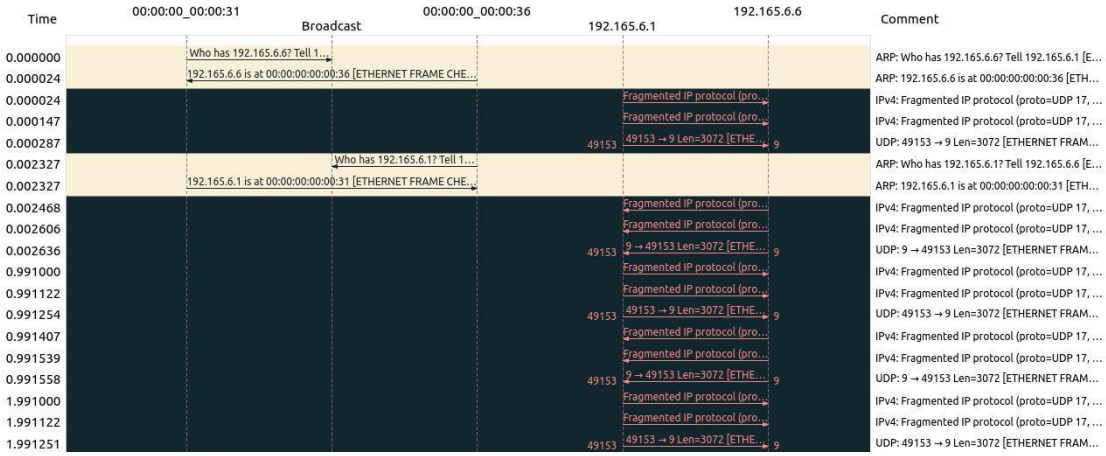

Figure 4.2.1. 9 Flowchat of node 3

Figure 4.2.1. 10 Flowchat of node 4

Figure 4.2.1. 11 Flowchat of node 5

In Trace Metrics software TR file was opened. We collected the throughput and goodput value. From equation (1) , we can see that lower throughput is better.

$$\text{Throughput} = \frac{Transmitted\ Data}{Trnsmitted\ time} \qquad \text{---------- (1)}$$

Some throughput and goodput value of some nodes are shown below.

Table 4.2.1. 1. Goodput and throughput value of some nodes.

Goodput	Throughput
6275.354747758534	6369.8375046554
3203.9152854604035	3258.905249791913
132.4758231622729	135.9751845288235
2052.1254870986045	2086.1192832308107
1156.2889772616497	1181.7843243608042

4.2.2. Wifi Simulation

From the WiFI PCAP file we collected the I/O (Input/output) graph and the flowcharts.

Some I/O graphs of different nodes are shown in figure bellow. Here X-axis shows the time and Y-axis shows the packet rate. From the graph transmission starts with a high packet rate and it pullback. The packet rate is good but its not stable. Its fluctuates more overtime.

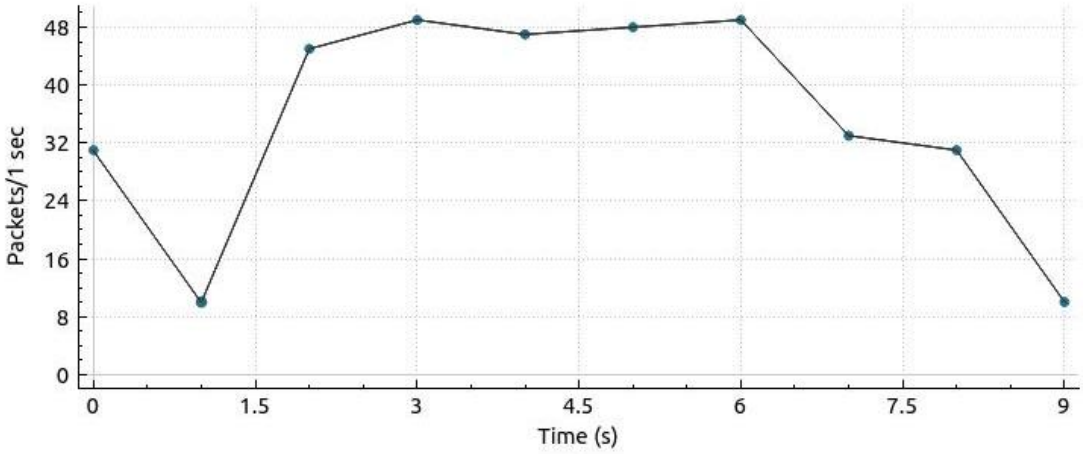

Figure 4.2.2. 1 I/O graph of node 0.

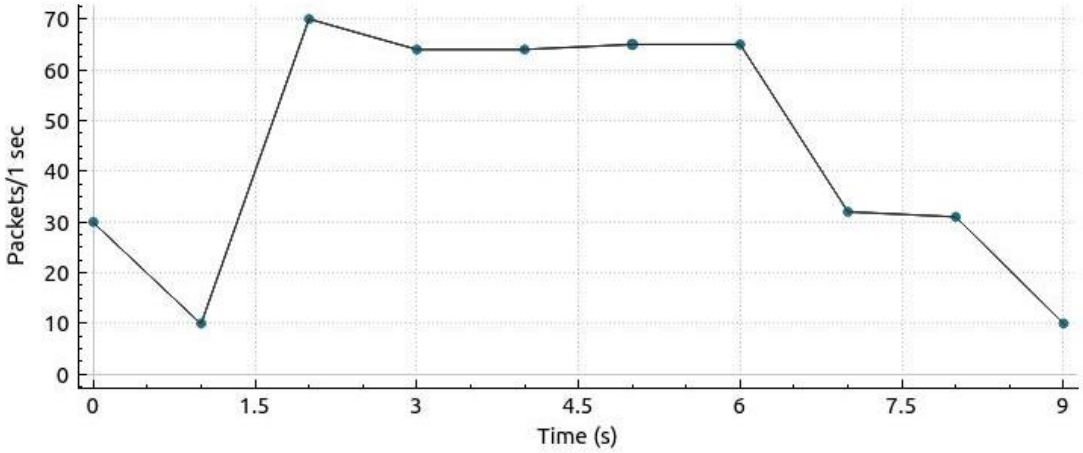

Figure 4.2.2. 2 I/O graph of node 1.

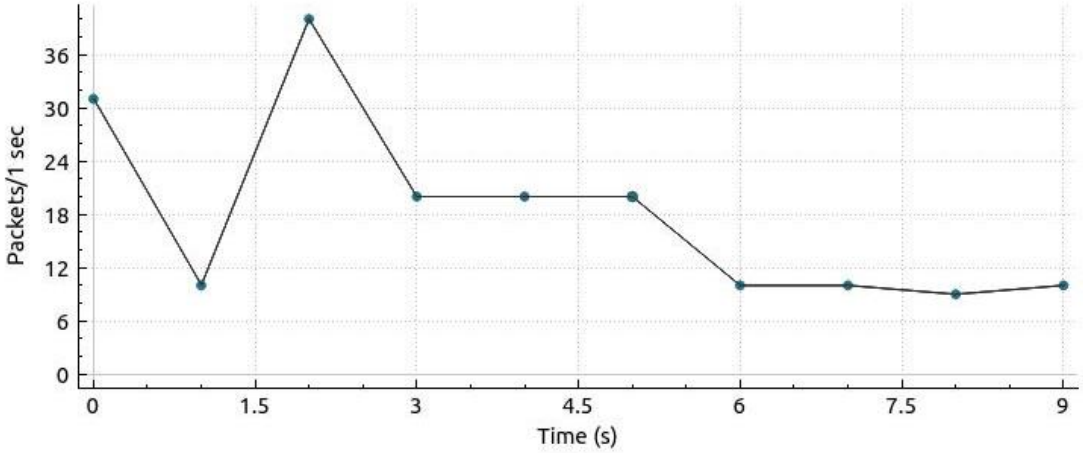

Figure 4.2.2. 3 I/O graph of node 2.

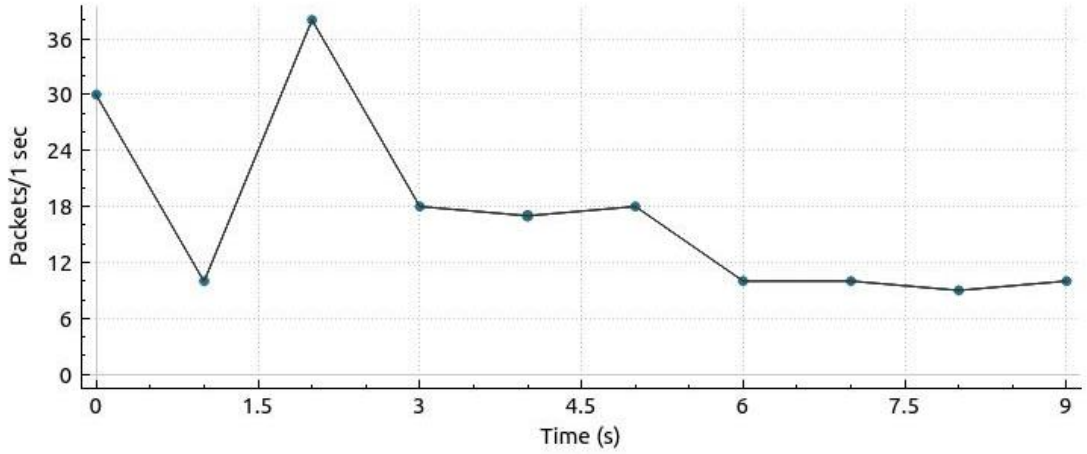

Figure 4.2.2. 4 I/O graph of node 3.

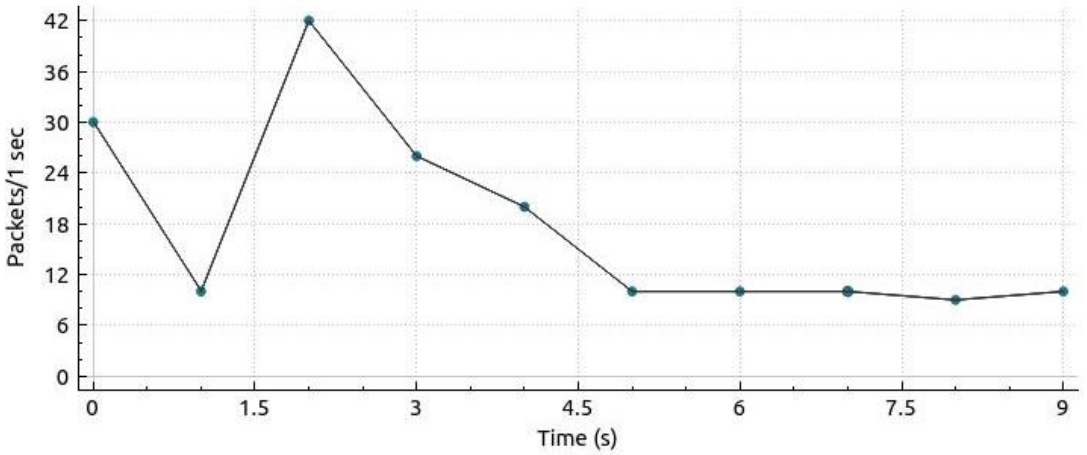

Figure 4.2.2. 5 I/O graph of node 4.

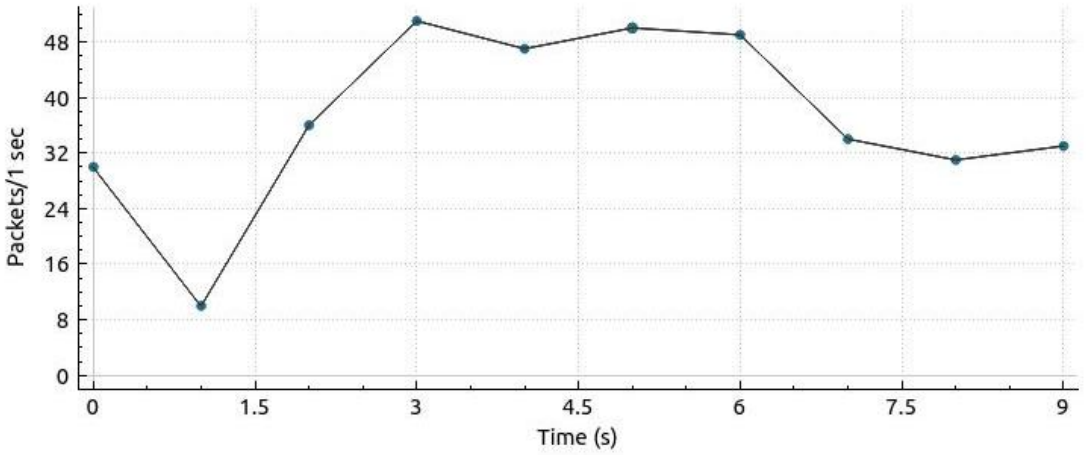

Figure 4.2.2. 6 I/O graph of node 5.

Some flowcharts of different nodes are shown in figure bellow. Is shows the transmission process. In which time, which node communicated with which IP is shown in this chart.

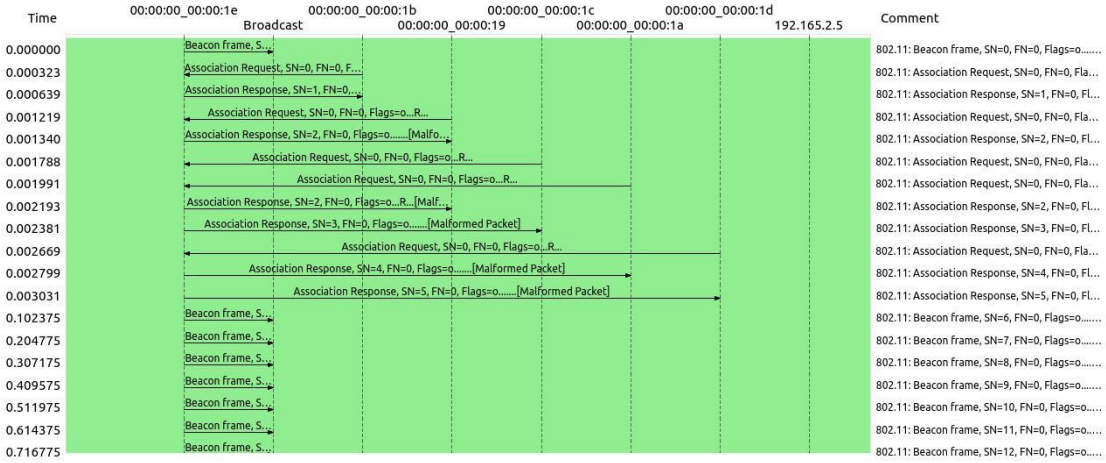

Figure 4.2.2. 7 Flowchat of node 0

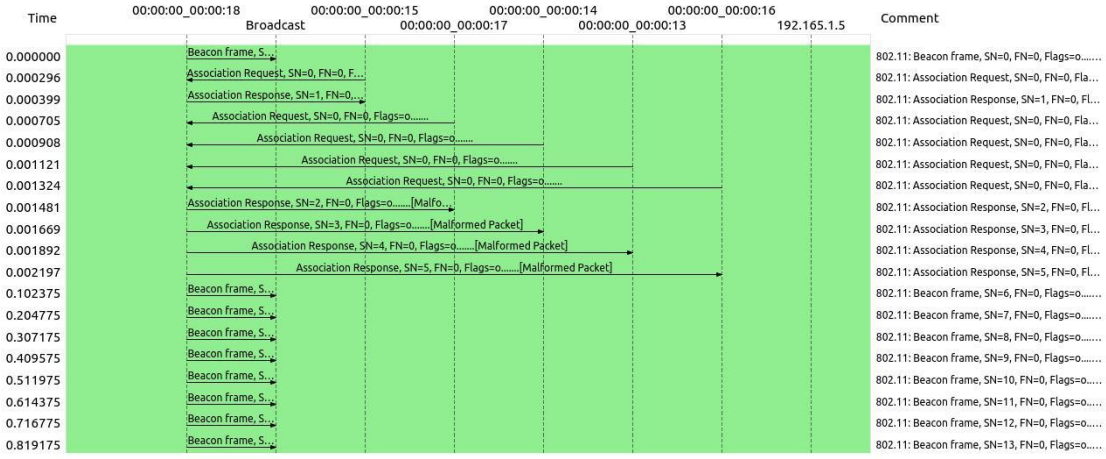

Figure 4.2.2. 8 Flowchat of node 1

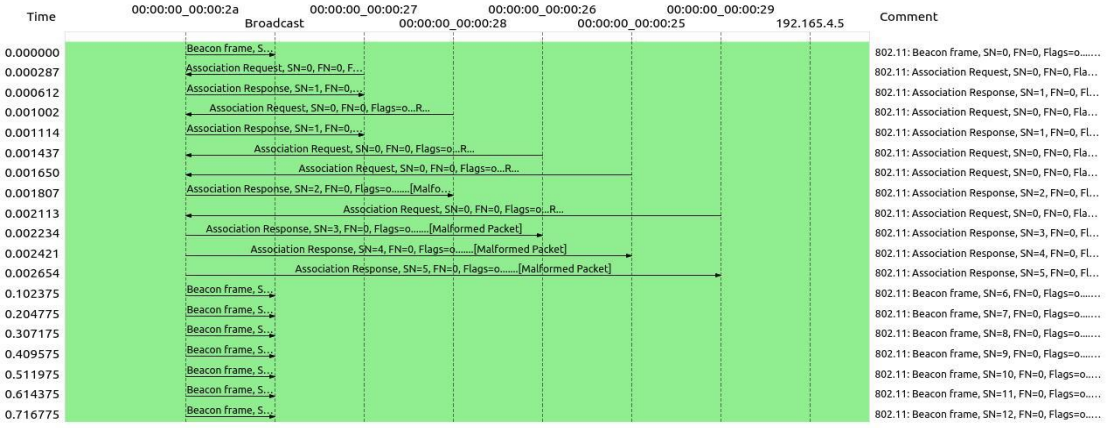

Figure 4.2.2. 9 Flowchat of node 2

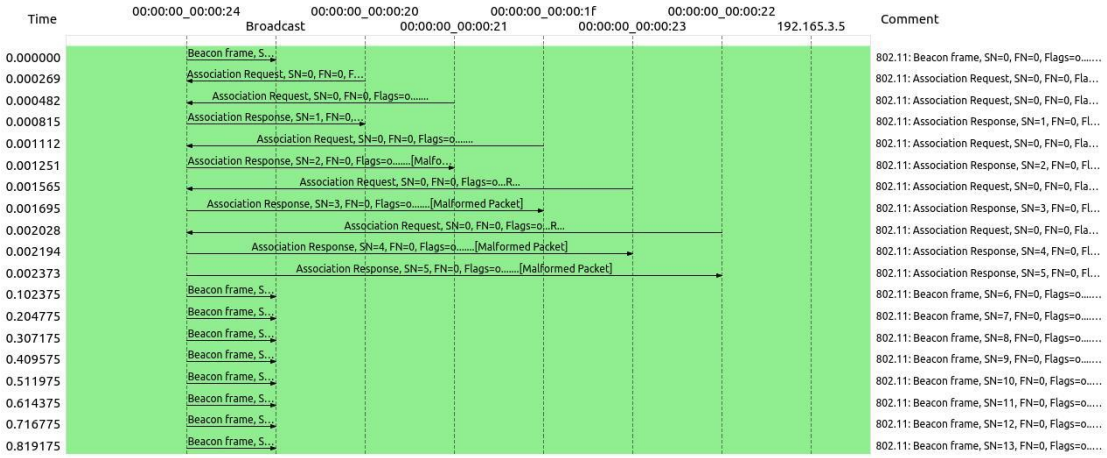

Figure 4.2.2. 10 Flowchat of node 3

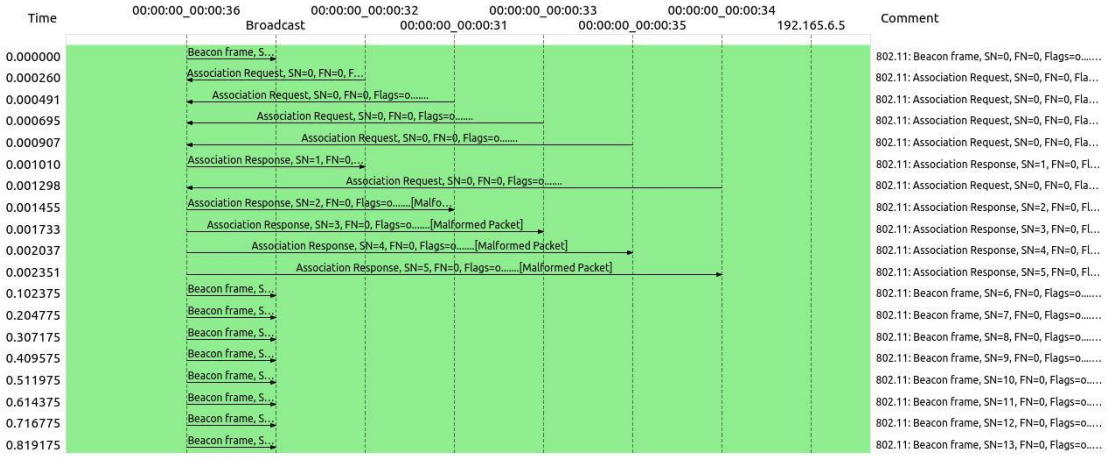

Figure 4.2.2. 11 Flowchat of node 4

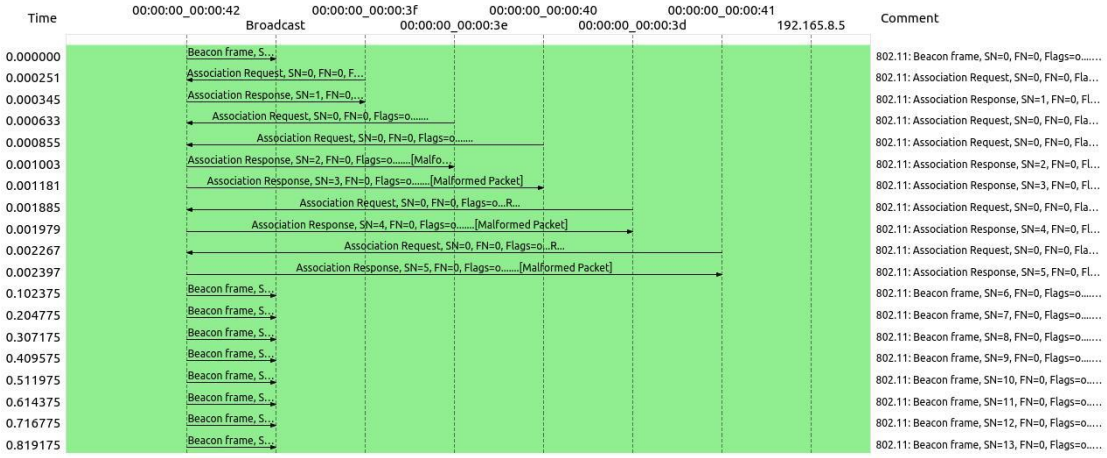

Figure 4.2.2. 12 Flowchat of node 5

In Trace Metrics software TR file was opened. We collected the throughput and goodput value. From equation (1), we can see that lower throughput is better.

$$\text{Throughput} = \frac{Transmitted\ Data}{Trnsmitted\ time} \qquad \text{---------- (1)}$$

Some throughput and goodput value of some nodes are shown below.

Table 4.2.2. 1 Goodput and throughput value of some nodes.

Goodput	Throughput
6986.874943999292	7132.652968767461
10547.563604220717	10764.619834469457
1900.7521461485105	1941.8277221881604
1014.8083492148827	1036.9569441382234
1421.5370923525936	1453.7532304229073

4.2.3 P2P Simulation

In both Modules we used P2P network as Gateway. For Education Purposes we also took the PCAP file and TR file of p2p network.

Some of the I/O graphs of p2p network for CSMA Module shown below.

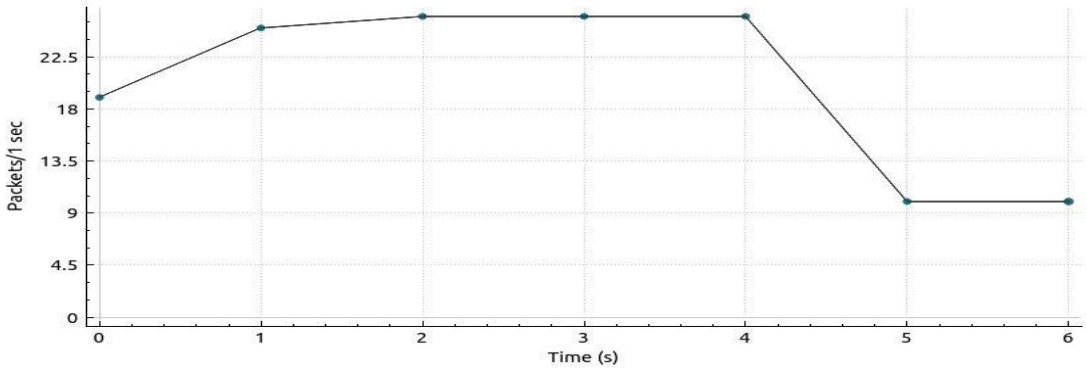

Figure 4.2.3. 1 I/O graph of node 0 of p2p for csma.

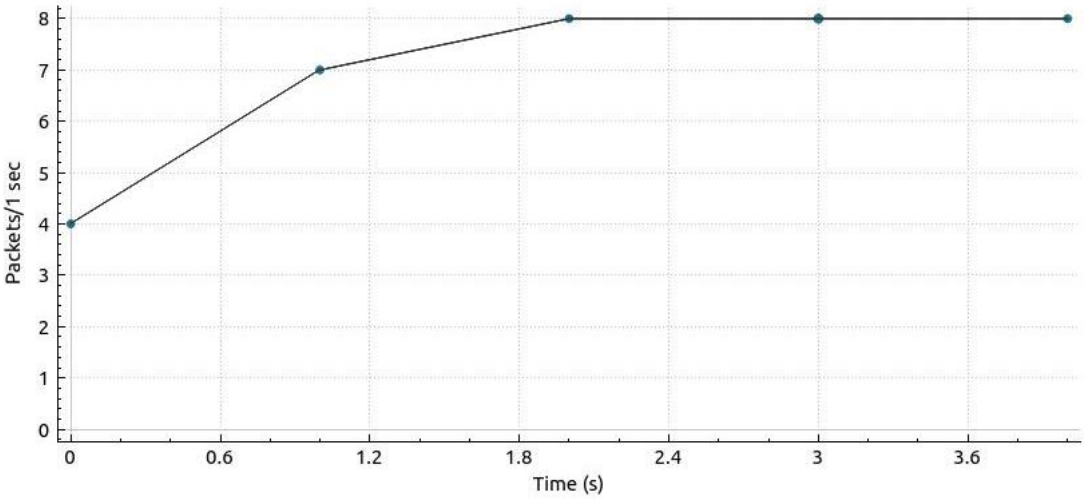

Figure 4.2.3. 2 I/O graph of node 1 of p2p for csma.

Some of the I/O graphs of p2p network for CSMA Module shown below

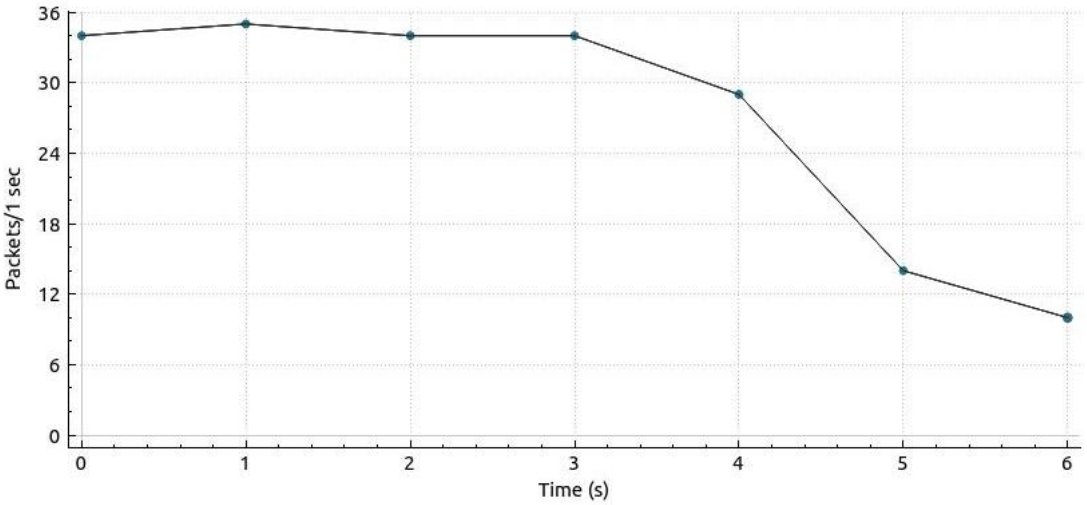

Figure 4.2.3. 3 I/O graph of node 0 of p2p for wifi

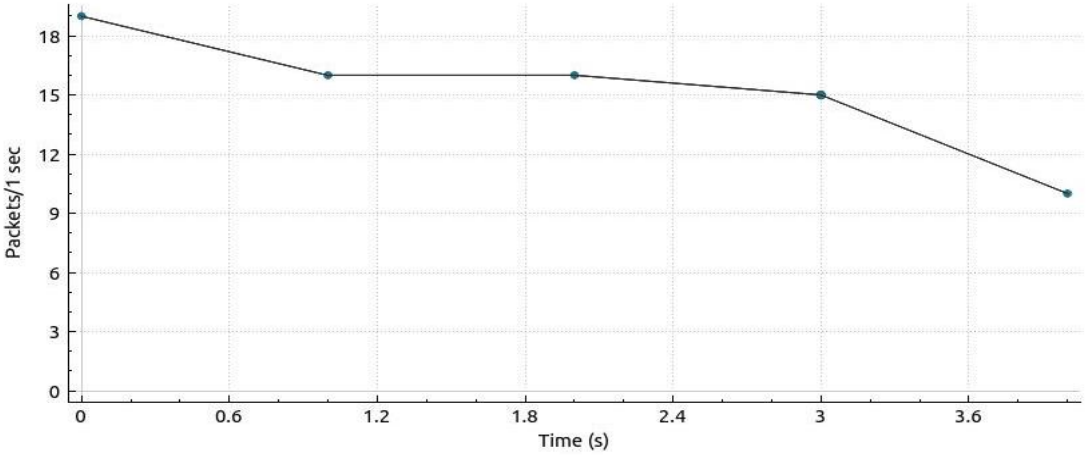

Figure 4.2.3. 4 I/O graph of node 1 of p2p for wifi.

Some of the nodes Throughput and Goodputs of p2p network for CSMA Module shown below.

Table 4.2.3. 1 Goodput and throughput value of some nodes of p2p for csma.

CSMA	
Goodput	Throughput
16852.53039821844	17151.9235010779
14327.610618209761	14578.622706348586
6459.15005940904	6578.9642193936725
6459.15005940904	6578.9642193936725
6459.15005940904	6578.9642193936725

Table 4.2.3. 2 Goodput and throughput value of some nodes of p2p for wifi.

CSMA	
Goodput	Throughput
13999.040313468116	14317.682777164768
10415.915759866086	10653.847259475799
6775.298464722675	6932.076748387439
7585.503868598128	7761.74123401536
7132.6379659952545	7299.145790536252

Chapter 5

Conclusion and Future works

5.1. Achievements

In this paper we successfully simulated two individual communication system for home infrastructure. One is CSMA network, another is Wifi network based on IEEE 802.11ac standard. We also managed to decreased the throughputs.

5.2. Limitations

This thesis has some limitation. Firstly, in CSMA we avoided collision. To implement it in real life, it has to upgrade in CSMA/CD. Secondly, the throughput is still quite high.

5.3. Future Work Field

In future will help to learn about CSMA simulation to upgrade it in CSMA/CD. It will help the researchers to learn about the IEEE 802.11ac standard simulation, and further more to upgrade it into IEEE 802.11ax standard. Depending on the economic situation of Bangladesh, this paper future work can be implemented in the future smart homes of Bangladesh.

References

[1] Continental Automated Buildings Association (CABA), "Impact of Smart Grid on the Connected Home Landmark Research Study," *IEEE,* pp. 6-8, 2013.

[2] J. Callaham, "Android authority," 15 april 2019. [Online]. Available: https://www.androidauthority.com/what-is-a-smart-home-806483/.

[3] techopedia, "Techopedia," [Online]. Available: https://www.techopedia.com/definition/5649/carrier-sense-multiple-access-csma.

[4] ns-3, "Network Simulator," 2011. [Online]. Available: https://www.nsnam.org/docs/release/3.12/models/html/csma.html. [Accessed 2011].

[5] Actiontech, "Actiontech," Actiontech, 2019. [Online]. Available: https://www.actiontec.com/wifihelp/complete-guide-to-wifi-networking/.

[6] J. KIM, "The university of Arizona," 8 April 2011. [Online]. Available: http://www2.engr.arizona.edu/~junseok/ns3_wifi.htm. [Accessed 8 April 2011].

[7] creation123, "GeeksforGeeks," [Online]. Available: https://www.geeksforgeeks.org/network-simulator-3/.

[8] ns-3, "ns-3," ns-3, 2011. [Online]. Available: https://www.nsnam.org/about/. [Accessed 2011].

[9] M. P. Murat Kuzlu, "Assessment of Communication Technologies and Network Requirements for Different Smart Grid Applications," *IEEE,* pp. 1-4, 2013.

[10] J. Pokorny, P. Masek, J. Hosek, P. Mlynek and P. Seda, "Simulations of Network Bottlenecks in Smart Grids," *IEEE,* 2018.

[11] Y. Ma, N. Wei and M. Lv, "Performance Analysis of Wireless Network Based on IEEE 802.15.4 in Smart Home Environment," *IEEE,* 2012.

[12] L.-x. Wang, "Research and Implementation of Wireless Home Network in Smart Home," *IEEE,* 2014.

[13] Y.-J. Lin, H. Latchman, M. Lee and S. Katar, "A Power Line Communication Network Infrastructure for The Smart Home," *IEEE,* vol. 9, no. 6, pp. 104 - 111, 2002.

[14] S. Fey, P. Benoit, G. Rohbogner, A. H. Christ and C. Wittwer, "Device-to-device communication for Smart Grid devices," *IEEE,* 2012.

[15] X. Zhao, "The Strategy of Smart Home Control System Design based on Wireless Network," *IEEE,* 2010.

[16] C. Zhou, "Study on architecture of smart home management system and key devices," *IEEE,* 2013.

[17] M.-T. Nguyen, L.-L. Nguyen and T.-D. Nguyen, "On the Design of Gateway Node for Smart Gird home network," *IEEE,* 2015.

[18] T. Lynch, Writing up Qualitative Research, English Language Teaching Centre.

Appendices

Codes of CSMA Module

```
#include "ns3/core-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/network-module.h"

#include "ns3/applications-module.h"

#include "ns3/wifi-module.h"

#include "ns3/mobility-module.h"

#include "ns3/csma-module.h"

#include "ns3/internet-module.h"

#include "ns3/netanim-module.h"


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

int

main (int argc, char *argv[])

{

  bool verbose = true;

  uint32_t csma1a = 5;

  uint32_t csma1b = 5;

  uint32_t csma2a = 5;

  uint32_t csma2b = 5;

  uint32_t csma3a = 5;

  uint32_t csma3b = 5;
```

```
  uint32_t csma4a = 5;
  uint32_t csma4b = 5;
  uint32_t csma5a = 5;
  uint32_t csma5b = 5;
  bool tracing = true;

  CommandLine cmd;
  cmd.AddValue ("csma1a", "Number of \"extra\" CSMA nodes/devices", csma1a);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.Parse (argc,argv);
  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }
////////////////p2p nodes/////////////////
  NodeContainer p2pNodes;
  p2pNodes.Create (10);
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

///1s,2nd
  NetDeviceContainer p2pDevices1;
```

```
  p2pDevices1 = pointToPoint.Install (p2pNodes.Get(0),p2pNodes.Get(1));
  NetDeviceContainer p2pDevices2;
  p2pDevices2 = pointToPoint.Install (p2pNodes.Get(2),p2pNodes.Get(3));
  //for 1st Left
  NetDeviceContainer p2pDevicesl;
  p2pDevicesl = pointToPoint.Install (p2pNodes.Get(0),p2pNodes.Get(2));
///3rd
  NetDeviceContainer p2pDevices3;
  p2pDevices3 = pointToPoint.Install (p2pNodes.Get(4),p2pNodes.Get(5));
  //for 1st Right
  NetDeviceContainer p2pDevicesr;
  p2pDevicesr = pointToPoint.Install (p2pNodes.Get(3),p2pNodes.Get(5));
///4rd
  NetDeviceContainer p2pDevices4;
  p2pDevices4 = pointToPoint.Install (p2pNodes.Get(6),p2pNodes.Get(7));
  //for 2nd Left
  NetDeviceContainer p2pDevicesL;
  p2pDevicesL = pointToPoint.Install (p2pNodes.Get(4),p2pNodes.Get(6));
 ///5rd
  NetDeviceContainer p2pDevices5;
  p2pDevices5 = pointToPoint.Install (p2pNodes.Get(8),p2pNodes.Get(9));
  //for 2nd Right
  NetDeviceContainer p2pDevicesR;
  p2pDevicesR = pointToPoint.Install (p2pNodes.Get(7),p2pNodes.Get(9));
```

```
//////////////////////csma/////////////////////
//1A csma nodes
  NodeContainer csma1aNodes;
  csma1aNodes.Add (p2pNodes.Get (1));
  csma1aNodes.Create (csma1a);
  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
  csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
  NetDeviceContainer csma1aDevices;
  csma1aDevices = csma.Install (csma1aNodes);
//1B Csma nodes
  NodeContainer csma1bNodes;
  csma1bNodes.Add (p2pNodes.Get (0));
  csma1bNodes.Create (csma1b);
  NetDeviceContainer csma1bDevices;
  csma1bDevices = csma.Install (csma1bNodes);

//2A Csma nodes
  NodeContainer csma2aNodes;
  csma2aNodes.Add (p2pNodes.Get (3));
  csma2aNodes.Create (csma2a);
  NetDeviceContainer csma2aDevices;
  csma2aDevices = csma.Install (csma2aNodes);
//2B Csma nodes
```

```
  NodeContainer csma2bNodes;
  csma2bNodes.Add (p2pNodes.Get (2));
  csma2bNodes.Create (csma2b);
  NetDeviceContainer csma2bDevices;
  csma2bDevices = csma.Install (csma2bNodes);
//3A Csma nodes
  NodeContainer csma3aNodes;
  csma3aNodes.Add (p2pNodes.Get (5));
  csma3aNodes.Create (csma3a);
  NetDeviceContainer csma3aDevices;
  csma3aDevices = csma.Install (csma3aNodes);
//3B Csma nodes
  NodeContainer csma3bNodes;
  csma3bNodes.Add (p2pNodes.Get (4));
  csma3bNodes.Create (csma3b);
  NetDeviceContainer csma3bDevices;
  csma3bDevices = csma.Install (csma3bNodes);
//4A Csma nodes
  NodeContainer csma4aNodes;
  csma4aNodes.Add (p2pNodes.Get (7));
  csma4aNodes.Create (csma4a);
  NetDeviceContainer csma4aDevices;
  csma4aDevices = csma.Install (csma4aNodes);
//4B Csma nodes
```

```
  NodeContainer csma4bNodes;
  csma4bNodes.Add (p2pNodes.Get (6));
  csma4bNodes.Create (csma3b);
  NetDeviceContainer csma4bDevices;
  csma4bDevices = csma.Install (csma4bNodes);
//5A Csma nodes
  NodeContainer csma5aNodes;
  csma5aNodes.Add (p2pNodes.Get (9));
  csma5aNodes.Create (csma5a);
  NetDeviceContainer csma5aDevices;
  csma5aDevices = csma.Install (csma5aNodes);
//5B Csma nodes
  NodeContainer csma5bNodes;
  csma5bNodes.Add (p2pNodes.Get (8));
  csma5bNodes.Create (csma5b);
  NetDeviceContainer csma5bDevices;
  csma5bDevices = csma.Install (csma5bNodes);

  InternetStackHelper stack;
  stack.Install (csma1aNodes);
  stack.Install (csma1bNodes);
  stack.Install (csma2aNodes);
  stack.Install (csma2bNodes);
  stack.Install (csma3aNodes);
```

```
stack.Install (csma3bNodes);
stack.Install (csma4aNodes);
stack.Install (csma4bNodes);
stack.Install (csma5aNodes);
stack.Install (csma5bNodes);


Ipv4AddressHelper address;

address.SetBase ("192.164.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces1;
p2pInterfaces1 = address.Assign (p2pDevices1);
address.SetBase ("192.164.2.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces2;
p2pInterfaces2 = address.Assign (p2pDevices2);
address.SetBase ("192.164.3.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfacesl;
p2pInterfacesl = address.Assign (p2pDevicesl);
address.SetBase ("192.164.4.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfacesr;
p2pInterfacesr = address.Assign (p2pDevicesr);
address.SetBase ("192.164.5.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces3;
p2pInterfaces3 = address.Assign (p2pDevices3);
```

```
address.SetBase ("192.164.6.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfacesL;
p2pInterfacesL = address.Assign (p2pDevicesL);
address.SetBase ("192.164.7.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces4;
p2pInterfaces4 = address.Assign (p2pDevices4);
address.SetBase ("192.164.8.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfacesR;
p2pInterfacesR = address.Assign (p2pDevicesR);
address.SetBase ("192.164.9.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces5;
p2pInterfaces5 = address.Assign (p2pDevices5);
////////////////CSMA
address.SetBase ("192.165.1.0", "255.255.255.0");
Ipv4InterfaceContainer csma1aInterfaces;
csma1aInterfaces = address.Assign (csma1aDevices);
address.SetBase ("192.165.2.0", "255.255.255.0");
Ipv4InterfaceContainer csma1bInterfaces;
csma1bInterfaces = address.Assign (csma1bDevices);
address.SetBase ("192.165.3.0", "255.255.255.0");
Ipv4InterfaceContainer csma2aInterfaces;
csma2aInterfaces = address.Assign (csma2aDevices);
address.SetBase ("192.165.4.0", "255.255.255.0");
Ipv4InterfaceContainer csma2bInterfaces;
```

```
csma2bInterfaces = address.Assign (csma2bDevices);
address.SetBase ("192.165.5.0", "255.255.255.0");
Ipv4InterfaceContainer csma3aInterfaces;
csma3aInterfaces = address.Assign (csma3aDevices);
address.SetBase ("192.165.6.0", "255.255.255.0");
Ipv4InterfaceContainer csma3bInterfaces;
csma3bInterfaces = address.Assign (csma3bDevices);
address.SetBase ("192.165.7.0", "255.255.255.0");
Ipv4InterfaceContainer csma4aInterfaces;
csma4aInterfaces = address.Assign (csma4aDevices);
address.SetBase ("192.165.8.0", "255.255.255.0");
Ipv4InterfaceContainer csma4bInterfaces;
csma4bInterfaces = address.Assign (csma4bDevices);
address.SetBase ("192.165.9.0", "255.255.255.0");
Ipv4InterfaceContainer csma5aInterfaces;
csma5aInterfaces = address.Assign (csma5aDevices);
address.SetBase ("192.165.10.0", "255.255.255.0");
Ipv4InterfaceContainer csma5bInterfaces;
csma5bInterfaces = address.Assign (csma5bDevices);

UdpEchoServerHelper echoServer (9);
//1A
ApplicationContainer serverApps1a = echoServer.Install (csma1aNodes.Get
(csma1a));
serverApps1a.Start (Seconds (1.0));
```

```
  serverApps1a.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient1a (csma1aInterfaces.GetAddress (csma1a), 9);
  echoClient1a.SetAttribute ("MaxPackets", UintegerValue (5));
  echoClient1a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient1a.SetAttribute ("PacketSize", UintegerValue (5120));
  ApplicationContainer clientApps1a = echoClient1a.Install(p2pNodes.Get(0));
  clientApps1a.Start (Seconds (1.0));
  clientApps1a.Stop (Seconds (10.0));
  //1B
  ApplicationContainer serverApps1b = echoServer.Install (csma1bNodes.Get
(csma1b));
  serverApps1b.Start (Seconds (1.0));
  serverApps1b.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient1b (csma1bInterfaces.GetAddress (csma1b), 9);
  echoClient1b.SetAttribute ("MaxPackets", UintegerValue (7));
  echoClient1b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient1b.SetAttribute ("PacketSize", UintegerValue (7168));
  ApplicationContainer clientApps1b = echoClient1b.Install(p2pNodes.Get(1));
  clientApps1b.Start (Seconds (1.0));
  clientApps1b.Stop (Seconds (10.0));
  //2A
  ApplicationContainer serverApps2a = echoServer.Install (csma2aNodes.Get
(csma2a));
  serverApps2a.Start (Seconds (1.0));
  serverApps2a.Stop (Seconds (10.0));
```

```
  UdpEchoClientHelper echoClient2a (csma2aInterfaces.GetAddress (csma2a), 9);
  echoClient2a.SetAttribute ("MaxPackets", UintegerValue (4));
  echoClient2a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient2a.SetAttribute ("PacketSize", UintegerValue (4096));
  ApplicationContainer clientApps2a = echoClient2a.Install(p2pNodes.Get(3));
  clientApps2a.Start (Seconds (1.0));
  clientApps2a.Stop (Seconds (10.0));
  //2B
  ApplicationContainer serverApps2b = echoServer.Install (csma2bNodes.Get
(csma2b));
  serverApps2b.Start (Seconds (1.0));
  serverApps2b.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient2b (csma2bInterfaces.GetAddress (csma2b), 9);
  echoClient2b.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient2b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient2b.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer clientApps2b = echoClient2b.Install(p2pNodes.Get(2));
  clientApps2b.Start (Seconds (1.0));
  clientApps2b.Stop (Seconds (10.0));
  //3A
  ApplicationContainer serverApps3a = echoServer.Install (csma3aNodes.Get
(csma3a));
  serverApps3a.Start (Seconds (1.0));
  serverApps3a.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient3a (csma3aInterfaces.GetAddress (csma3a), 9);
```

```
  echoClient3a.SetAttribute ("MaxPackets", UintegerValue (2));
  echoClient3a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient3a.SetAttribute ("PacketSize", UintegerValue (2048));
  ApplicationContainer clientApps3a = echoClient3a.Install(p2pNodes.Get(5));
  clientApps3a.Start (Seconds (1.0));
  clientApps3a.Stop (Seconds (10.0));
  //3B
  ApplicationContainer serverApps3b = echoServer.Install (csma3bNodes.Get
(csma3b));
  serverApps3b.Start (Seconds (1.0));
  serverApps3b.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient3b (csma3bInterfaces.GetAddress (csma3b), 9);
  echoClient3b.SetAttribute ("MaxPackets", UintegerValue (3));
  echoClient3b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient3b.SetAttribute ("PacketSize", UintegerValue (3072));
  ApplicationContainer clientApps3b = echoClient3b.Install(p2pNodes.Get(4));
  clientApps3b.Start (Seconds (1.0));
  clientApps3b.Stop (Seconds (10.0));
  //4A
  ApplicationContainer serverApps4a = echoServer.Install (csma4aNodes.Get
(csma4a));
  serverApps4a.Start (Seconds (1.0));
  serverApps4a.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient4a (csma4aInterfaces.GetAddress (csma4a), 9);
  echoClient4a.SetAttribute ("MaxPackets", UintegerValue (8));
```

```
  echoClient4a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient4a.SetAttribute ("PacketSize", UintegerValue (8192));
  ApplicationContainer clientApps4a = echoClient4a.Install(p2pNodes.Get(7));
  clientApps4a.Start (Seconds (1.0));
  clientApps4a.Stop (Seconds (10.0));
  //4B
  ApplicationContainer serverApps4b = echoServer.Install (csma4bNodes.Get
(csma4b));
  serverApps4b.Start (Seconds (1.0));
  serverApps4b.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient4b (csma4bInterfaces.GetAddress (csma4b), 9);
  echoClient4b.SetAttribute ("MaxPackets", UintegerValue (5));
  echoClient4b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient4b.SetAttribute ("PacketSize", UintegerValue (5120));
  ApplicationContainer clientApps4b = echoClient4b.Install(p2pNodes.Get(6));
  clientApps4b.Start (Seconds (1.0));
  clientApps4b.Stop (Seconds (10.0));
  //5A
  ApplicationContainer serverApps5a = echoServer.Install (csma5aNodes.Get
(csma5a));
  serverApps5a.Start (Seconds (1.0));
  serverApps5a.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient5a (csma5aInterfaces.GetAddress (csma5a), 9);
  echoClient5a.SetAttribute ("MaxPackets", UintegerValue (3));
```

```
  echoClient5a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient5a.SetAttribute ("PacketSize", UintegerValue (3072));
  ApplicationContainer clientApps5a = echoClient5a.Install(p2pNodes.Get(9));
  clientApps5a.Start (Seconds (1.0));
  clientApps5a.Stop (Seconds (10.0));
  //5b
  ApplicationContainer serverApps5b = echoServer.Install (csma5bNodes.Get
(csma5b));
  serverApps5b.Start (Seconds (1.0));
  serverApps5b.Stop (Seconds (10.0));
  UdpEchoClientHelper echoClient5b (csma5bInterfaces.GetAddress (csma5b), 9);
  echoClient5b.SetAttribute ("MaxPackets", UintegerValue (5));
  echoClient5b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient5b.SetAttribute ("PacketSize", UintegerValue (5120));
  ApplicationContainer clientApps5b = echoClient5b.Install(p2pNodes.Get(1));
  clientApps5b.Start (Seconds (1.0));
  clientApps5b.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  Simulator::Stop (Seconds (10.0));
  if(tracing == true){
    pointToPoint.EnablePcapAll("csmap2p");
    csma.EnablePcapAll("csma", true);
  }
////////Ascii trace
```

```
  AsciiTraceHelper ascii;
  csma.EnableAsciiAll(ascii.CreateFileStream("csma.tr"));
  pointToPoint.EnableAsciiAll(ascii.CreateFileStream("csmap2p.tr"));

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

Codes of WiFi Module

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"



using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
```

```
int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t wifi1a = 5;
  uint32_t wifi1b = 5;
  uint32_t wifi2a = 5;
  uint32_t wifi2b = 5;
  uint32_t wifi3a = 5;
  uint32_t wifi3b = 5;
  uint32_t wifi4a = 5;
  uint32_t wifi4b = 5;
  uint32_t wifi5a = 5;
  uint32_t wifi5b = 5;
  bool tracing = true;
  CommandLine cmd;
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.Parse (argc,argv);
  if (wifi1a > 18) {
     std::cout << "Number of wifi nodes " << wifi1a <<
                  " specified exceeds the mobility bounding box" << std::endl;
     exit (1);
    }
  if (wifi1b > 18) {
```

```
      std::cout << "Number of wifi nodes " << wifi1b <<
                " specified exceeds the mobility bounding box" << std::endl;
      exit (1);
    }
  if (verbose) {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }
//////////////////p2p/////////////////
  NodeContainer p2pNodes;
  p2pNodes.Create (10);
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
///1s,2nd
  NetDeviceContainer p2pDevices1;
  p2pDevices1 = pointToPoint.Install (p2pNodes.Get(0),p2pNodes.Get(1));
  NetDeviceContainer p2pDevices2;
  p2pDevices2 = pointToPoint.Install (p2pNodes.Get(2),p2pNodes.Get(3));
  NetDeviceContainer p2pDevicesl;
  p2pDevicesl = pointToPoint.Install (p2pNodes.Get(0),p2pNodes.Get(2));
///3rd
  NetDeviceContainer p2pDevices3;
  p2pDevices3 = pointToPoint.Install (p2pNodes.Get(4),p2pNodes.Get(5));
```

```
  //for 1st Right
  NetDeviceContainer p2pDevicesr;
  p2pDevicesr = pointToPoint.Install (p2pNodes.Get(3),p2pNodes.Get(5));
///4rd
  NetDeviceContainer p2pDevices4;
  p2pDevices4 = pointToPoint.Install (p2pNodes.Get(6),p2pNodes.Get(7));
  //for 2nd Left
  NetDeviceContainer p2pDevicesL;
  p2pDevicesL = pointToPoint.Install (p2pNodes.Get(4),p2pNodes.Get(6));
 ///5rd
  NetDeviceContainer p2pDevices5;
  p2pDevices5 = pointToPoint.Install (p2pNodes.Get(8),p2pNodes.Get(9));
  //for 2nd Right
  NetDeviceContainer p2pDevicesR;
  p2pDevicesR = pointToPoint.Install (p2pNodes.Get(7),p2pNodes.Get(9));
//////////////////wifi1a/////////////////
  NodeContainer wifiStaNodes1a;
  wifiStaNodes1a.Create (wifi1a);
  NodeContainer wifiApNode1a = p2pNodes.Get (1);
  YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
  phy.SetChannel (channel.Create ());

  WifiHelper wifi = WifiHelper::Default ();
```

```
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
NqosWifiMacHelper mac1a = NqosWifiMacHelper::Default ();

Ssid ssid1a = Ssid ("ns-3-ssid1a");
mac1a.SetType ("ns3::StaWifiMac",
        "Ssid", SsidValue (ssid1a),
        "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevices1a;
staDevices1a = wifi.Install (phy, mac1a, wifiStaNodes1a);
mac1a.SetType ("ns3::ApWifiMac",
        "Ssid", SsidValue (ssid1a));
NetDeviceContainer apDevices1a;
apDevices1a = wifi.Install (phy, mac1a, wifiApNode1a);
///////mobility
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                "MinX", DoubleValue (0.0),
                "MinY", DoubleValue (0.0),
                "DeltaX", DoubleValue (5.0),
                "DeltaY", DoubleValue (10.0),
                "GridWidth", UintegerValue (3),
                "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
```

```
                    "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
  mobility.Install (wifiStaNodes1a);
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (wifiApNode1a);
//////////////////wifi1b//////
  NodeContainer wifiStaNodes1b;
  wifiStaNodes1b.Create (wifi1b);
  NodeContainer wifiApNode1b = p2pNodes.Get (0);
  YansWifiChannelHelper channel1b = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy1b = YansWifiPhyHelper::Default ();
  phy1b.SetChannel (channel1b.Create ());
  // phy1b.Set("ChannelNumber", UintegerValue(6));
  WifiHelper wifib = WifiHelper::Default ();
  wifib.SetRemoteStationManager ("ns3::AarfWifiManager");
  NqosWifiMacHelper mac1b = NqosWifiMacHelper::Default ();
  Ssid ssid1b = Ssid ("ns-3-ssid1b");
  mac1b.SetType ("ns3::StaWifiMac",
          "Ssid", SsidValue (ssid1b),
          "ActiveProbing", BooleanValue (false));
  NetDeviceContainer staDevices1b;
  staDevices1b = wifib.Install (phy1b, mac1b, wifiStaNodes1b);
  mac1b.SetType ("ns3::ApWifiMac",
          "Ssid", SsidValue (ssid1b));
  NetDeviceContainer apDevices1b;
```

```
apDevices1b = wifib.Install (phy1b, mac1b, wifiApNode1b);
MobilityHelper mobility1b;
mobility1b.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (0.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (5.0),
                                 "DeltaY", DoubleValue (10.0),
                                 "GridWidth", UintegerValue (3),
                                 "LayoutType", StringValue ("RowFirst"));
mobility1b.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                             "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility1b.Install (wifiStaNodes1b);
mobility1b.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility1b.Install (wifiApNode1b);
//////////////////wifi2a/////////////////
NodeContainer wifiStaNodes2a;
wifiStaNodes2a.Create (wifi2a);
NodeContainer wifiApNode2a = p2pNodes.Get (3);
YansWifiChannelHelper channel2a = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy2a = YansWifiPhyHelper::Default ();
phy2a.SetChannel (channel2a.Create ());
// phy2a.Set("ChannelNumber", UintegerValue(6));
WifiHelper wific = WifiHelper::Default ();
wific.SetRemoteStationManager ("ns3::AarfWifiManager");
```

```
NqosWifiMacHelper mac2a = NqosWifiMacHelper::Default ();


Ssid ssid2a = Ssid ("ns-3-ssid2a");
mac2a.SetType ("ns3::StaWifiMac",
            "Ssid", SsidValue (ssid2a),
            "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevices2a;
staDevices2a = wific.Install (phy2a, mac2a, wifiStaNodes2a);
mac2a.SetType ("ns3::ApWifiMac",
            "Ssid", SsidValue (ssid2a));
NetDeviceContainer apDevices2a;
apDevices2a = wific.Install (phy2a, mac2a, wifiApNode2a);
///////mobility
MobilityHelper mobility2a;
mobility2a.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (0.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (5.0),
                                 "DeltaY", DoubleValue (10.0),
                                 "GridWidth", UintegerValue (3),
                                 "LayoutType", StringValue ("RowFirst"));
mobility2a.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                             "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility2a.Install (wifiStaNodes2a);
```

```
  mobility2a.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility2a.Install (wifiApNode2a);
//////////////////wifi2b//////
  NodeContainer wifiStaNodes2b;
  wifiStaNodes2b.Create (wifi2b);
  NodeContainer wifiApNode2b = p2pNodes.Get (2);
  YansWifiChannelHelper channel2b = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy2b = YansWifiPhyHelper::Default ();
  phy2b.SetChannel (channel2b.Create ());
  // phy2b.Set("ChannelNumber", UintegerValue(6));
  WifiHelper wifid = WifiHelper::Default ();
  wifid.SetRemoteStationManager ("ns3::AarfWifiManager");
  NqosWifiMacHelper mac2b = NqosWifiMacHelper::Default ();
  Ssid ssid2b = Ssid ("ns-3-ssid2b");
  mac2b.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid2b),
                 "ActiveProbing", BooleanValue (false));
  NetDeviceContainer staDevices2b;
  staDevices2b = wifid.Install (phy2b, mac2b, wifiStaNodes2b);
  mac2b.SetType ("ns3::ApWifiMac",
                 "Ssid", SsidValue (ssid2b));
  NetDeviceContainer apDevices2b;
  apDevices2b = wifid.Install (phy2b, mac2b, wifiApNode2b);
  MobilityHelper mobility2b;
```

```
mobility2b.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (0.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (5.0),
                                 "DeltaY", DoubleValue (10.0),
                                 "GridWidth", UintegerValue (3),
                                 "LayoutType", StringValue ("RowFirst"));
mobility2b.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                             "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility2b.Install (wifiStaNodes2b);
mobility2b.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility2b.Install (wifiApNode2b);

//////////////////wifi3a/////////////////
NodeContainer wifiStaNodes3a;
wifiStaNodes3a.Create (wifi3a);
NodeContainer wifiApNode3a = p2pNodes.Get (5);
YansWifiChannelHelper channel3a = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy3a = YansWifiPhyHelper::Default ();
phy3a.SetChannel (channel3a.Create ());
WifiHelper wifie = WifiHelper::Default ();
wifie.SetRemoteStationManager ("ns3::AarfWifiManager");
NqosWifiMacHelper mac3a = NqosWifiMacHelper::Default ();
Ssid ssid3a = Ssid ("ns-3-ssid3a");
```

```
mac3a.SetType ("ns3::StaWifiMac",
            "Ssid", SsidValue (ssid3a),
            "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevices3a;
staDevices3a = wifie.Install (phy3a, mac3a, wifiStaNodes3a);
mac3a.SetType ("ns3::ApWifiMac",
            "Ssid", SsidValue (ssid3a));
NetDeviceContainer apDevices3a;
apDevices3a = wifie.Install (phy3a, mac3a, wifiApNode3a);
///////mobility
MobilityHelper mobility3a;
mobility3a.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (0.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (5.0),
                                 "DeltaY", DoubleValue (10.0),
                                 "GridWidth", UintegerValue (3),
                                 "LayoutType", StringValue ("RowFirst"));

mobility3a.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                             "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility3a.Install (wifiStaNodes3a);
mobility3a.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility3a.Install (wifiApNode3a);
```

```
//////////////////wifi3b//////
  NodeContainer wifiStaNodes3b;
  wifiStaNodes3b.Create (wifi3b);
  NodeContainer wifiApNode3b = p2pNodes.Get (4);
  YansWifiChannelHelper channel3b = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy3b = YansWifiPhyHelper::Default ();
  phy3b.SetChannel (channel3b.Create ());
  WifiHelper wifif = WifiHelper::Default ();
  wifif.SetRemoteStationManager ("ns3::AarfWifiManager");
  NqosWifiMacHelper mac3b = NqosWifiMacHelper::Default ();

  Ssid ssid3b = Ssid ("ns-3-ssid3b");
  mac3b.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid3b),
                 "ActiveProbing", BooleanValue (false));
  NetDeviceContainer staDevices3b;
  staDevices3b = wifif.Install (phy3b, mac3b, wifiStaNodes3b);
  mac3b.SetType ("ns3::ApWifiMac",
                 "Ssid", SsidValue (ssid3b));
  NetDeviceContainer apDevices3b;
  apDevices3b = wifif.Install (phy3b, mac3b, wifiApNode3b);
  MobilityHelper mobility3b;
  mobility3b.SetPositionAllocator ("ns3::GridPositionAllocator",
                                   "MinX", DoubleValue (0.0),
```

```
                                "MinY", DoubleValue (0.0),
                                "DeltaX", DoubleValue (5.0),
                                "DeltaY", DoubleValue (10.0),
                                "GridWidth", UintegerValue (3),
                                "LayoutType", StringValue ("RowFirst"));
  mobility3b.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                               "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
  mobility3b.Install (wifiStaNodes3b);
  mobility3b.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility3b.Install (wifiApNode3b);
//////////////////wifi4a/////////////////
  NodeContainer wifiStaNodes4a;
  wifiStaNodes4a.Create (wifi4a);
  NodeContainer wifiApNode4a = p2pNodes.Get (7);
  YansWifiChannelHelper channel4a = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy4a = YansWifiPhyHelper::Default ();
  phy4a.SetChannel (channel4a.Create ());
  // phy4a.Set("ChannelNumber", UintegerValue(6));
  WifiHelper wifig = WifiHelper::Default ();
  wifig.SetRemoteStationManager ("ns3::AarfWifiManager");
  NqosWifiMacHelper mac4a = NqosWifiMacHelper::Default ();
  Ssid ssid4a = Ssid ("ns-3-ssid4a");
  mac4a.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid4a),
```

```
            "ActiveProbing", BooleanValue (false));
  NetDeviceContainer staDevices4a;
  staDevices4a = wifig.Install (phy4a, mac4a, wifiStaNodes4a);
  mac4a.SetType ("ns3::ApWifiMac",
            "Ssid", SsidValue (ssid4a));
  NetDeviceContainer apDevices4a;
  apDevices4a = wifig.Install (phy4a, mac4a, wifiApNode4a);
///////mobility
  MobilityHelper mobility4a;
  mobility4a.SetPositionAllocator ("ns3::GridPositionAllocator",
                        "MinX", DoubleValue (0.0),
                        "MinY", DoubleValue (0.0),
                        "DeltaX", DoubleValue (5.0),
                        "DeltaY", DoubleValue (10.0),
                        "GridWidth", UintegerValue (3),
                        "LayoutType", StringValue ("RowFirst"));
  mobility4a.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                       "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
  mobility4a.Install (wifiStaNodes4a);
  mobility4a.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility4a.Install (wifiApNode4a);
//////////////////wifi4b//////
  NodeContainer wifiStaNodes4b;
  wifiStaNodes4b.Create (wifi4b);
```

```
NodeContainer wifiApNode4b = p2pNodes.Get (6);
YansWifiChannelHelper channel4b = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy4b = YansWifiPhyHelper::Default ();
phy4b.SetChannel (channel4b.Create ());
// phy4b.Set("ChannelNumber", UintegerValue(6));
WifiHelper wifih = WifiHelper::Default ();
wifih.SetRemoteStationManager ("ns3::AarfWifiManager");
NqosWifiMacHelper mac4b = NqosWifiMacHelper::Default ();
Ssid ssid4b = Ssid ("ns-3-ssid4b");
mac4b.SetType ("ns3::StaWifiMac",
        "Ssid", SsidValue (ssid4b),
        "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevices4b;
staDevices4b = wifih.Install (phy4b, mac4b, wifiStaNodes4b);
mac4b.SetType ("ns3::ApWifiMac",
        "Ssid", SsidValue (ssid4b));
NetDeviceContainer apDevices4b;
apDevices4b = wifih.Install (phy4b, mac4b, wifiApNode4b);
MobilityHelper mobility4b;
mobility4b.SetPositionAllocator ("ns3::GridPositionAllocator",
                "MinX", DoubleValue (0.0),
                "MinY", DoubleValue (0.0),
                "DeltaX", DoubleValue (5.0),
                "DeltaY", DoubleValue (10.0),
```

```
                    "GridWidth", UintegerValue (3),
                    "LayoutType", StringValue ("RowFirst"));
  mobility4b.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                    "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
  mobility4b.Install (wifiStaNodes4b);
  mobility4b.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility4b.Install (wifiApNode4b);
//////////////////wifi5a/////////////////
  NodeContainer wifiStaNodes5a;
  wifiStaNodes5a.Create (wifi5a);
  NodeContainer wifiApNode5a = p2pNodes.Get (9);
  YansWifiChannelHelper channel5a = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy5a = YansWifiPhyHelper::Default ();
  phy5a.SetChannel (channel5a.Create ());
  WifiHelper wifii = WifiHelper::Default ();
  wifii.SetRemoteStationManager ("ns3::AarfWifiManager");
  NqosWifiMacHelper mac5a = NqosWifiMacHelper::Default ();
  Ssid ssid5a = Ssid ("ns-3-ssid5a");
  mac5a.SetType ("ns3::StaWifiMac",
          "Ssid", SsidValue (ssid5a),
          "ActiveProbing", BooleanValue (false));
  NetDeviceContainer staDevices5a;
  staDevices5a = wifii.Install (phy5a, mac5a, wifiStaNodes5a);
  mac5a.SetType ("ns3::ApWifiMac",
```

```
            "Ssid", SsidValue (ssid5a));
  NetDeviceContainer apDevices5a;
  apDevices5a = wifii.Install (phy5a, mac5a, wifiApNode5a);
///////mobility
  MobilityHelper mobility5a;
  mobility5a.SetPositionAllocator ("ns3::GridPositionAllocator",
                                   "MinX", DoubleValue (0.0),
                                   "MinY", DoubleValue (0.0),
                                   "DeltaX", DoubleValue (5.0),
                                   "DeltaY", DoubleValue (10.0),
                                   "GridWidth", UintegerValue (3),
                                   "LayoutType", StringValue ("RowFirst"));
  mobility5a.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                               "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
  mobility5a.Install (wifiStaNodes5a);
  mobility5a.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility5a.Install (wifiApNode5a);
////////////////////wifi5b//////
  NodeContainer wifiStaNodes5b;
  wifiStaNodes5b.Create (wifi5b);
  NodeContainer wifiApNode5b = p2pNodes.Get (8);
  YansWifiChannelHelper channel5b = YansWifiChannelHelper::Default ();
  YansWifiPhyHelper phy5b = YansWifiPhyHelper::Default ();
  phy5b.SetChannel (channel5b.Create ());
```

```
WifiHelper wifij = WifiHelper::Default ();
wifij.SetRemoteStationManager ("ns3::AarfWifiManager");
NqosWifiMacHelper mac5b = NqosWifiMacHelper::Default ();
Ssid ssid5b = Ssid ("ns-3-ssid5b");
mac5b.SetType ("ns3::StaWifiMac",
        "Ssid", SsidValue (ssid5b),
        "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevices5b;
staDevices5b = wifij.Install (phy5b, mac5b, wifiStaNodes5b);
mac5b.SetType ("ns3::ApWifiMac",
        "Ssid", SsidValue (ssid5b));
NetDeviceContainer apDevices5b;
apDevices5b = wifij.Install (phy5b, mac5b, wifiApNode5b);
MobilityHelper mobility5b;
mobility5b.SetPositionAllocator ("ns3::GridPositionAllocator",
                "MinX", DoubleValue (0.0),
                "MinY", DoubleValue (0.0),
                "DeltaX", DoubleValue (5.0),
                "DeltaY", DoubleValue (10.0),
                "GridWidth", UintegerValue (3),
                "LayoutType", StringValue ("RowFirst"));
mobility5b.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
```

```
mobility5b.Install (wifiStaNodes5b);
mobility5b.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility5b.Install (wifiApNode5b);
InternetStackHelper stack;
stack.Install (wifiApNode1a);
stack.Install (wifiApNode1b);
stack.Install (wifiStaNodes1b);
stack.Install (wifiStaNodes1a);
stack.Install (wifiApNode2a);
stack.Install (wifiApNode2b);
stack.Install (wifiStaNodes2b);
stack.Install (wifiStaNodes2a);
stack.Install (wifiApNode3a);
stack.Install (wifiApNode3b);
stack.Install (wifiStaNodes3b);
stack.Install (wifiStaNodes3a);
stack.Install (wifiApNode4a);
stack.Install (wifiApNode4b);
stack.Install (wifiStaNodes4b);
stack.Install (wifiStaNodes4a);
stack.Install (wifiApNode5a);
stack.Install (wifiApNode5b);
stack.Install (wifiStaNodes5b);
stack.Install (wifiStaNodes5a);
```

```
  Ipv4AddressHelper address;
//////p2p
  address.SetBase ("192.164.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces1;
  p2pInterfaces1 = address.Assign (p2pDevices1);
  address.SetBase ("192.164.2.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces2;
  p2pInterfaces2 = address.Assign (p2pDevices2);
  address.SetBase ("192.164.3.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfacesl;
  p2pInterfacesl = address.Assign (p2pDevicesl);
  address.SetBase ("192.164.4.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfacesr;
  p2pInterfacesr = address.Assign (p2pDevicesr);
  address.SetBase ("192.164.5.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces3;
  p2pInterfaces3 = address.Assign (p2pDevices3);
  address.SetBase ("192.164.6.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfacesL;
  p2pInterfacesL = address.Assign (p2pDevicesL);
  address.SetBase ("192.164.7.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces4;
  p2pInterfaces4 = address.Assign (p2pDevices4);
  address.SetBase ("192.164.8.0", "255.255.255.0");
```

```
  Ipv4InterfaceContainer p2pInterfacesR;
  p2pInterfacesR = address.Assign (p2pDevicesR);
  address.SetBase ("192.164.9.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces5;
  p2pInterfaces5 = address.Assign (p2pDevices5);
/////wifi
  address.SetBase ("192.165.1.0", "255.255.255.0");
  Ipv4InterfaceContainer wifiInterface1a;
  wifiInterface1a = address.Assign (staDevices1a);
  wifiInterface1a = address.Assign (apDevices1a);
  address.SetBase ("192.165.2.0", "255.255.255.0");
  Ipv4InterfaceContainer wifiInterface1b;
  wifiInterface1b = address.Assign (staDevices1b);
  wifiInterface1b = address.Assign (apDevices1b);
  address.SetBase ("192.165.3.0", "255.255.255.0");
  Ipv4InterfaceContainer wifiInterface2a;
  wifiInterface2a = address.Assign (staDevices2a);
  wifiInterface2a = address.Assign (apDevices2a);
  address.SetBase ("192.165.4.0", "255.255.255.0");
  Ipv4InterfaceContainer wifiInterface2b;
  wifiInterface2b = address.Assign (staDevices2b);
  wifiInterface2b = address.Assign (apDevices2b);
  address.SetBase ("192.165.5.0", "255.255.255.0");
  Ipv4InterfaceContainer wifiInterface3a;
```

```
wifiInterface3a = address.Assign (staDevices3a);
wifiInterface3a = address.Assign (apDevices3a);
address.SetBase ("192.165.6.0", "255.255.255.0");
Ipv4InterfaceContainer wifiInterface3b;
wifiInterface3b = address.Assign (staDevices3b);
wifiInterface3b = address.Assign (apDevices3b);
address.SetBase ("192.165.7.0", "255.255.255.0");
Ipv4InterfaceContainer wifiInterface4a;
wifiInterface4a = address.Assign (staDevices4a);
wifiInterface4a = address.Assign (apDevices4a);
address.SetBase ("192.165.8.0", "255.255.255.0");
Ipv4InterfaceContainer wifiInterface4b;
wifiInterface4b = address.Assign (staDevices4b);
wifiInterface4b = address.Assign (apDevices4b);
address.SetBase ("192.165.9.0", "255.255.255.0");
Ipv4InterfaceContainer wifiInterface5a;
wifiInterface5a = address.Assign (staDevices5a);
wifiInterface5a = address.Assign (apDevices5a);
address.SetBase ("192.165.10.0", "255.255.255.0");
Ipv4InterfaceContainer wifiInterface5b;
wifiInterface5b = address.Assign (staDevices5b);
wifiInterface5b = address.Assign (apDevices5b);
UdpEchoServerHelper echoServer (9);
```

```
///////1a
```

```
  ApplicationContainer serverApps1a = echoServer.Install (wifiStaNodes1b.Get(1));
  serverApps1a.Start(Seconds(1.0));
  serverApps1a.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient1a (wifiInterface1b.GetAddress(1), 9);
  echoClient1a.SetAttribute ("MaxPackets", UintegerValue (5));
  echoClient1a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient1a.SetAttribute ("PacketSize", UintegerValue (5120));
  ApplicationContainer clientApps1a = echoClient1a.Install (wifiStaNodes1a.Get
(wifi1a - 1));
  clientApps1a.Start (Seconds (2.0));
  clientApps1a.Stop (Seconds (10.0));
////////1b
  ApplicationContainer serverApps1b = echoServer.Install (wifiStaNodes1a.Get(1));
  serverApps1b.Start(Seconds(1.0));
  serverApps1b.Stop(Seconds(10.0));

  UdpEchoClientHelper echoClient1b (wifiInterface1a.GetAddress(1), 9);
  echoClient1b.SetAttribute ("MaxPackets", UintegerValue (7));
  echoClient1b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient1b.SetAttribute ("PacketSize", UintegerValue (7168));
  ApplicationContainer clientApps1b = echoClient1b.Install (wifiStaNodes1b.Get
(wifi1b - 1));
  clientApps1b.Start (Seconds (2.0));
  clientApps1b.Stop (Seconds (10.0));
///////2a
```

```
  ApplicationContainer serverApps2a = echoServer.Install (wifiStaNodes2b.Get(1));
  serverApps2a.Start(Seconds(1.0));
  serverApps2a.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient2a (wifiInterface2b.GetAddress(1), 9);
  echoClient2a.SetAttribute ("MaxPackets", UintegerValue (4));
  echoClient2a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient2a.SetAttribute ("PacketSize", UintegerValue (4096));
  ApplicationContainer clientApps2a = echoClient2a.Install (wifiStaNodes2a.Get
(wifi2a - 1));
  clientApps2a.Start (Seconds (2.0));
  clientApps2a.Stop (Seconds (10.0));
////////2b
  ApplicationContainer serverApps2b = echoServer.Install (wifiStaNodes2a.Get(1));
  serverApps2b.Start(Seconds(1.0));
  serverApps2b.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient2b (wifiInterface2a.GetAddress(1), 9);
  echoClient2b.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient2b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient2b.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer clientApps2b = echoClient2b.Install (wifiStaNodes2b.Get
(wifi2b - 1));
  clientApps2b.Start (Seconds (2.0));
  clientApps2b.Stop (Seconds (10.0));
///////3a
  ApplicationContainer serverApps3a = echoServer.Install (wifiStaNodes3b.Get(1));
```

```
  serverApps3a.Start(Seconds(1.0));
  serverApps3a.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient3a (wifiInterface3b.GetAddress(1), 9);
  echoClient3a.SetAttribute ("MaxPackets", UintegerValue (2));
  echoClient3a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient3a.SetAttribute ("PacketSize", UintegerValue (2048));
  ApplicationContainer clientApps3a = echoClient3a.Install (wifiStaNodes3a.Get
(wifi3a - 1));
  clientApps3a.Start (Seconds (2.0));
  clientApps3a.Stop (Seconds (10.0));
////////3b
  ApplicationContainer serverApps3b = echoServer.Install (wifiStaNodes3a.Get(1));
  serverApps3b.Start(Seconds(1.0));
  serverApps3b.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient3b (wifiInterface3a.GetAddress(1), 9);
  echoClient3b.SetAttribute ("MaxPackets", UintegerValue (3));
  echoClient3b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient3b.SetAttribute ("PacketSize", UintegerValue (3072));
  ApplicationContainer clientApps3b = echoClient3b.Install (wifiStaNodes3b.Get
(wifi3b - 1));
  clientApps3b.Start (Seconds (2.0));
  clientApps3b.Stop (Seconds (10.0));
///////4a
  ApplicationContainer serverApps4a = echoServer.Install (wifiStaNodes4b.Get(1));
  serverApps4a.Start(Seconds(1.0));
```

```
  serverApps4a.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient4a (wifiInterface4b.GetAddress(1), 9);
  echoClient4a.SetAttribute ("MaxPackets", UintegerValue (8));
  echoClient4a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient4a.SetAttribute ("PacketSize", UintegerValue (8192));
  ApplicationContainer clientApps4a = echoClient4a.Install (wifiStaNodes4a.Get
(wifi4a - 1));
  clientApps4a.Start (Seconds (2.0));
  clientApps4a.Stop (Seconds (10.0));
////////4b
  ApplicationContainer serverApps4b = echoServer.Install (wifiStaNodes4a.Get(1));
  serverApps4b.Start(Seconds(1.0));
  serverApps4b.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient4b (wifiInterface4a.GetAddress(1), 9);
  echoClient4b.SetAttribute ("MaxPackets", UintegerValue (5));
  echoClient4b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient4b.SetAttribute ("PacketSize", UintegerValue (5120));
  ApplicationContainer clientApps4b = echoClient4b.Install (wifiStaNodes4b.Get
(wifi4b - 1));
  clientApps4b.Start (Seconds (2.0));
  clientApps4b.Stop (Seconds (10.0));
///////5a
  ApplicationContainer serverApps5a = echoServer.Install (wifiStaNodes5b.Get(1));
  serverApps5a.Start(Seconds(1.0));
  serverApps5a.Stop(Seconds(10.0));
```

```
  UdpEchoClientHelper echoClient5a (wifiInterface5b.GetAddress(1), 9);
  echoClient5a.SetAttribute ("MaxPackets", UintegerValue (3));
  echoClient5a.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient5a.SetAttribute ("PacketSize", UintegerValue (3072));
  ApplicationContainer clientApps5a = echoClient5a.Install (wifiStaNodes5a.Get
(wifi5a - 1));
  clientApps5a.Start (Seconds (2.0));
  clientApps5a.Stop (Seconds (10.0));
////////5b
  ApplicationContainer serverApps5b = echoServer.Install (wifiStaNodes1a.Get(1));
  serverApps5b.Start(Seconds(1.0));
  serverApps5b.Stop(Seconds(10.0));
  UdpEchoClientHelper echoClient5b (wifiInterface1a.GetAddress(1), 9);
  echoClient5b.SetAttribute ("MaxPackets", UintegerValue (5));
  echoClient5b.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient5b.SetAttribute ("PacketSize", UintegerValue (5120));
  ApplicationContainer clientApps5b = echoClient5b.Install (wifiStaNodes5b.Get
(wifi5b - 1));
  clientApps5b.Start (Seconds (2.0));
  clientApps5b.Stop (Seconds (10.0));


  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  Simulator::Stop (Seconds (10.0));
  if(tracing == true){
    pointToPoint.EnablePcapAll("wifi2p");
```

```
    phy.EnablePcapAll("wifi");
  }
////////Ascii trace
  AsciiTraceHelper ascii;
  pointToPoint.EnableAsciiAll(ascii.CreateFileStream("wifip2p.tr"));
  phy.EnableAsciiAll(ascii.CreateFileStream("wifi.tr"));
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```