DEVELOPMENT OF PROGRAMMABLE FIBER-WIRELESS EDUCATIONAL TESTBED

MUHAMMAD HAQEEM BIN MOHD NASIR

COLLEGE OF GRADUATE STUDIES
UNIVERSITI TENAGA NASIONAL

2020

DEVELOPMENT OF PROGRAMMABLE FIBER-WIRELESS EDUCATIONAL TESTBED

MUHAMMAD HAQEEM BIN MOHD NASIR

A Thesis Submitted to the College of Graduate Studies, Universiti Tenaga Nasional in Fulfilment of the Requirement for Degree of Master of Electrical Engineering

FEBRUARY 2020

DECLARATION

I hereby declare that the thesis is my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously, and is not concurrently submitted for any other degree at Universiti Tenaga Nasional or at any other institutions. This thesis may be made available within the university library and may be photocopied and loaned to other libraries for the purpose of consultation.

_____

(Signature)

MUHAMMAD HAQEEM BIN MOHD NASIR

Date: 24 February 2020

ABSTRACT

Fiber-Wireless (FiWi) network is an integration of fiber optic and wireless connections in the same network. FiWi is needed due to rapid increment of Internet users and bandwidth-hungry services. Therefore, a lot of solutions have been proposed and created by researchers using embedded system-based hardware and off-the-shelves routers in order to study FiWi technology. However, off-the-shelves routers have a limitation on its ability to be reconfigurable and scalable to a certain extent. Hence, this thesis proposes the development and performance evaluation of a reprogrammable, fast configurable, scalable and flexible FiWi router testbed. The testbed is using embedded system-based hardware that can be used for lab-scale experiments for research and educational purposes. Raspberry Pi is used as the embedded system hardware to develop the router since it is reconfigurable, space friendly, cost-efficient and user friendly. Each router comprises of four Raspberry Pis; one Header Pi and three Forwarding Pis, which are connected via two Ethernet switches. For wireless router, an additional access point is used as the antennae of the router. The performance of the testbed in terms of throughput, end-to-end delay, and jitter for upstream and downstream are done in wireless network, fiber network and FiWi network. The performance of the proposed testbed is scaled up with off-the-shelf router and industrial grade routers in terms of throughput for each network where the throughput shows similar increasing trend proving that the testbed is working correctly. The end-to-end delay of the testbed behaves expectedly as the data size increases and comply with IEEE 802.15.4 routing scheme trend. Whereas the jitter complies the Cisco's standard which is under 30 *ms*. The maximum jitter in FiWi is 8.25 *ms*. A stress test on the testbed is conducted by sending two traffics of data simultaneously. The result shows that the end-to-end delay for two traffics is twice as much as single traffic as expected since router needs to process the data twice the amount of data. The maximum jitter of the proposed router for two traffics is 11.23 *ms* which is still under 30 *ms*. The scalability test is done for Wireless-Fiber-Wireless (Wi-FiWi) network and Fiber-Wireless-Fiber (Fi-WiFi) network. The results prove that the proposed testbed is suitable to be a reprogrammable, fast reconfigurable, scalable and flexible FiWi router testbed for research and educational purposes.

ACKNOWLEDGEMENT

I am humbly grateful for the love and support from my supervisor and co-supervisor, Ir. Dr. Nurul Asyikin binti Mohamed Radzi and Dr. Fairuz Abdullah for their guidance and teachings as well as their tireless dedication, motivations and enthusiasm on showing me the right path throughout my studies. Also, special thank you to Prof. Ir. Dr. Md Zaini bin Jamaludin, Dr. Wan Siti Halimatul Munirah binti Wan Ahmad, Mr. Aiman bin Ismail, Mr. Mohammad Azmi Ridwan, Nurshazlina Suhaimy, Faris Syahmi bin Samidi, and Nayli Adriana for their advices and knowledge that have been shared with me. Secondly, I would like to express my gratitude to Dr. Normy Norfiza, Mrs. Roslina Abdul Ghapar and Mrs. Husni for their constant morale support. Lastly, I want to express my eternal love and gratitude to my parents; Normah binti Hj Hambali and Mohd Nasir bin A. Wahab for their prayers and wisdom. Without them, I will never achieve what I have accomplished today.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$D_{i,j}$	Time of transmission
D_{ave}	Average transmission time
N_p	Total number of receiving packets

LIST OF ABBREVIATIONS

3GPP	Third Generation Partnership Project
4G	Fourth Generation
AID	Action Identifier
AP	Access Point
AWG	Arbitrary Waveform Generator
BATMAN	Better Approach To Mobile Ad-hoc Networking
BBU	Baseband Unit
CCN	Content Centric Network
CDMA	Code Division Multiple Access
CID	Connection Identifier
CO	Central Office
CSA	Client System Agent
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
D2D	Device-to-Device
DCF	Distributed Coordination Function
DFB	Distributed Feedback
DPDK	Data Plane Development Kit
DRAGON	Dynamic Resource Allocation via GMPLS Optical Network
DS	Distribution System
E/O	Electrical-to-Optical
EN	Emulation Network
FBMC	Filter-Bank Multicarrier
FDD	Frequency Division Duplex
FiWi	Fiber-Wireless
Fi-WiFi	Fiber-Wireless-Fiber
FM	Frequency Modulation
FMC	Fiber Media Converter
FPGA	Field-Programmable Gate Array
FSAN	Full Service Access Network

FSK	Frequency Shift Keying
GE-PON	Gigabit Ethernet PON
GMPLS	Generalized Multiprotocol Label Switching
GOOSE	Generic Object-Oriented Substation Event
G-PON	Gigabit-capable PON
GR	Guarantee Rate
GSM	Global System Mobile communication
HQoS	Hierarchical QoS
HWMN	Hybrid Wireless Mesh Network
IEEE	Institute of Electrical and Electronic Engineers
IoT	Internet of Things
IPv6	IP version 6
ITU	International Telecommunication Union
ITU-T	ITU – Telecommunication Standardization Sector
L2	Layer 2
L3	Layer 3
LCD	Liquid Crystal Display
LF-DLSTM	Local Feature-based Deep Long Short Term
LOS	Line of Sight
LSP	Label Switched Path
LTE	Long Term Evolution
M2M	Machine-to-Machine
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network
MFH	Mobile Frouhaul
MIMO	Multiple Input Multiple Output
MN	Management Network
MNO	Mobile Network Operator
MOFI	Mobile-Oriented Future Internet
MOR	Multichannel Opportunistic Routing
NDN	Named Data Network
NFD	NDN Forwarding Daemon
NG-PON	Next Generation PON

NS2	Network Simulator 2
NS3	Network Simulator 3
O/E	Optical-to-Electrical
OFDM PON	Orthogonal Frequency Division Multiplexing PON
OLSR	Optimized Link State Routing
OLT	Optical Line Terminal
ONU	Optical Network Unit
OVS	OpenvSwitch
P2MP	Point-to-Multipoint
ParaDiMe	Parallel Router Distributed Memory
PCE	Path Computation Element
PDV	Packet Delay Variation
PHY	Physical
PMT	Photomultiplier Tube
P-OLT	Programmable OLT
PON	Passive Optical Network
P-ONU	Programmable ONU
QoS	Quality of Service
R&F	Radio-and-Fiber
RAN	Radio Access Network
RAN	Remote Antenna Unit
RAT	Regulated Action Table
RF	Radio Frequency
ROADM	Reconfigurable Optical Add-Drop Multiplexer
RoF	Radio-over-Fiber
RTDS	Real Time Digital Simulator
SC/APC	Single Core/Angled Physical Contact
SDM	Spatial Division Multiplexing
SDN	Software Defined Network
SDOAN	Software Defined Optical Access Network
SMF	Single-Mode Fiber
SR	Segment Routing
TDD	Time Division Duplex

TDM-PON	Time Division Multiplexing PON
UE	User Equipment
UHD	Ultra High Definition
UMTS	Universal Mobile Telecommunications System
URLLC	Ultra-Reliable Low Latency Communications
USRP	Universal Software Radio Peripheral
VDS	Video Distribution System
VLSR	Virtual Label Switched Router
VoIP	Voice over IP
VR	Virtual Reality
VTR	Verilog-to-Routing
WASN	Wireless Ad-hoc Sensor Network
WCDMA/HSPA	Wide-band Code Division Multiple Access/High-Speed Packet Access
WDM	Wavelength Division Multiplexing
WDM-PON	Wavelength Division Multiplexed PON
Wi-FiWi	Wireless-Fiber-Wireless
WiMAX	World Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WMAN-SC	Wireless Metropolitan Area Network – Single Carrier
WMAN-OFDM	WMAN-Orthogonal Frequency Division Multiplexing
WMAN-OFDMA	WMAN-Orthogonal Frequency Division Multiplexing Access
WMAN-SCA	WMAN - Single Carrier Access
WMN	Wireless Mesh Network
WRON	Wavelength-Routed Optical Network

LIST OF PUBLICATIONS

The following list shows the publication done by author up to date of thesis.

1. M. H. M. Nasir, N. A. M. Radzi, W. S. H. M. W. Ahmad, F. Abdullah, M. Z. Jamaludin (2019, Sept.). Comparison of Router Testbeds: Embedded system-based, Software-based, and Multiprotocol Label Switching (MPLS). Indonesian Journal of Electrical Engineering and Computer Science (IJEECS). *15(3),* pp. 1250-1256.
2. F. S. Samidi, M. H. M. Nasir, N. A. M. Radzi, F. Abdullah, A. Ismail, M. Z. Jamaludin, Y. E. Erana, W. S. H. M. W. Ahmad. Performance Evaluation of Wireless Router Testbed using Raspberry Pi. International Journal of Recent Technology and Engineering (IJRTE). *8(4),* pp. 6415-6421.
3. M. H. M. Nasir, W. S. H. M. W. Ahmad, N. A. M. Radzi. Raspberry Pi as Reconfigurable Fiber-based Router and Its Performance Analysis. Accepted for presentation in International Conference on Photonics 2020 (ICP2020).

CHAPTER 1

INTRODUCTION

1.1 Background

Over the years, Internet has becoming more advanced each day. Most people uses Internet to communicate with friends and family, for online banking, video streaming and online gaming. Some uses an advanced Internet applications such as Virtual Reality (VR) for online gaming. For better or worse, the Internet has completely changed the way of life for humankind.

Having such advanced technologies cause high demands of more reliable and faster Internet because the services in these technologies are classified as bandwidth-intensive. According to Yu et al. [1], the exponential growth of mobile devices and bandwidth-intensive services with demanding Quality of Service (QoS) have increased the interest of fiber-wireless (FiWi) network deployment to provide high capacity, high flexibility and low cost broadband access. Hence, the needs of FiWi network to provide greater bandwidth allocation and mobility to the end users are necessary. According to Edholm's Law of Bandwidth [2], FiWi network is an integration of optical fiber and wireless to provide fixed and mobile services. One of the most basic architecture of FiWi is using Passive Optical Network (PON). It uses passive components that do not require any power which reduces operation and maintenance cost.

This thesis develops a scalable and fast integration testbed for FiWi that can be used in lab-scale experiments for research purposes and as an educational module. The testbed uses Raspberry Pi because it can be programmed by using Python to create a router. The key feature of Python for this project is its socket module which enables the Raspberry Pi to reliably communicate with one another via Transmission Control Protocol/Internet Protocol (TCP/IP). Furthermore, compared to other languages, the simplicity of Python makes the reconfiguration of the proposed router fast and flexible. Hence, making the proposed router scalable and fast integration. Each router in the

testbed consists of four Raspberry Pi 3B+ and two Ethernet switches via CAT5e Ethernet cable. Detail explanations about the testbed will be done in Chapter 3.

1.1.1 Fiber Wireless Network

FiWi is an integration of fiber optic and wireless in a network [2-6]. According to Yu et al. [1], a typical FiWi has a high-capacity PON that comprises of an Optical Line Terminal (OLT) and multiple Optical Network Units (ONU) connected to a cluster of wireless routers. In return, ONU is integrated with a wireless gateway that provides wide-area connectivity to users.

The typical architecture of FiWi is shown in Figure 1. FiWi consists of two major parts in the network; optical backhaul and wireless front end. At the optical backhaul, the OLT at the central office (CO) forms a root connected to the optical backbone via a fiber link. This is to provide cloud computing services. ONU is connected to the OLT via 1 : N (1 : 32 or 1 : 64) splitter to form a leaf-shaped nodes. The CO is responsible to manage the information transmission between mobile devices with ONUs and acting as a gateway to other networks [7]. In the wireless end of FiWi network, mobile devices such as smart phones [8, 9], VR glasses [10], smart watches [11] and other Internet of Things (IoT) devices [10-12] have access to the Internet either via ONU or multi-hop wireless mesh network.

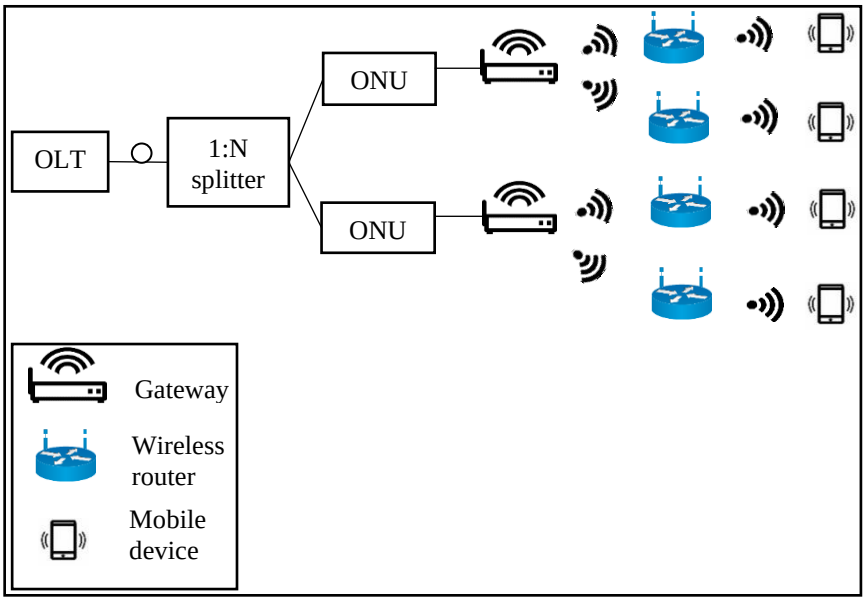

Figure 1 FiWi typical architecture

Based on a study by Chein and Reisslein [13], one of the advantages of FiWi is that it provides high speed optical backhaul for a wireless mesh network. Rimal et al. [14] added that the network has a broadband access where it uses wide range of frequencies enabling a large number of data to communicate simultaneously. Whereas, mobile backhaul combines reliability and capacity of Ethernet-based optical backhaul with the wide range of coverage and flexibility of Ethernet-based wireless devices. By utilizing the efficacy of both optical fiber and wireless, a fast speed and low cost of service areas can be achieved [15].

Other advantage of having FiWi technology is the combination of fiber optic and radio access technologies in multi-tier Radio Access Network (RAN) [16]. Radio access technologies will be used to deliver wireless services with high capacity, high link speed, and low latency [17]. The multi-tier RAN will improve the cell edge performance for mobile fronthaul and backhaul, resource sharing, and centralization of multiple standards with different frequency bands and modulation formats.

It can be concluded that FiWi is a promising technology to support high demand of bandwidth and large number of users in different types of topologies and geographical environments.

1.1.2 FiWi Testbed

Over the years, research communities have been working hard to improve current technologies. Hence, in order to study and perform extensive performance evaluations, testbeds are needed. Testbed is known as a prototype for proof-of-concept of a technology features where further experiments can be applied [18]. There are various types of testbed in various platforms such as lab-scale testbed and field testbed.

Numerous benefits of having a testbed are explained by Hurst et al. [19] Built in lab-scale environment is portable. Which means, it can be easily packed away, securely stored and safely transported. The testbed can be also reused and assembled by other researchers in the future.

Furthermore, according to a study by Gong et al. [20], a testbed can have flexible architecture which can be connected to any devices such as multiple sensors. Despite of its compactness, it has powerful local computation unit. A testbed can also be reconfigured to different topologies.

It can be deduced that, developing a lab-scale testbed is a promising and practical way of studying and experimenting a particular technology especially in FiWi for research and educational purposes before an actual implementation due to its simplicity, portability and flexibility.

1.2 Problem Statements

According to [16], the demand for Internet and leased line bandwidth are growing continuously at more than 20% per year due to more video streaming, cloud computing, social media and mobile data delivery. By 2020, the bandwidth demand will continue to grow due to an enhancement of video quality such as 8K Ultra High Definition (UHD) and increasing number of user subscriptions. Because of that, the estimated traffic in terms of mobile data and fixed systems will be 2600 times more than the traffic in 2010 [16]. Furthermore, this growth is accelerated by new type of communication services such as device-to-device (D2D) and machine-to-machine (M2M). Therefore, in order to ensure the users to experience the same QoS at anytime and anywhere while the demand is increasing, FiWi deployment has become a necessity because it can cover a large area and support large number of users.

According to Ridwan et al. [21], FiWi is still an ongoing study and there are plenty of rooms for improvement. Therefore, a development of fast integration and scalable testbed is crucial to provide a platform to investigate further in order to understand fundamental knowledge especially for undergraduate university students, graduate engineers and researchers. Some of the existing testbeds consist of complicated setup and require a big area to install. For example, Abraha et al. [22] conducted an experiment consist of 64 antennas for FiWi testbed setup. This makes the testbed impractical to be an educational module because the installation of the antennas involves a whole building. Hence, fast integration is an important criterion for a testbed

to be an educational module so that the installation of the testbed is easy and the lecturers can have more time on explaining rather than focusing on testbed installation. Therefore, the proposed testbed is able to help students on understanding the hands-on knowledge of FiWi, reprogrammability of the router and medium conversions in FiWi in a lab-scale area.

1.3 Research Objectives

The aim of this thesis is to design and evaluate the performance of a fast integration scalable FiWi testbed. The simplicity of the testbed in terms of development and setup makes it practical to be an educational module in FiWi network by using Raspberry Pi 3B+.

The specific objectives of this thesis are as follows:

1. Development of a reprogrammable and fast reconfigurable lab-scale FiWi testbed that supports the integration of fiber optic and wireless for research and educational purposes.
2. Evaluate performance of proposed FiWi testbed in terms of scalability of more than one traffics and flexibility to a different topology; such as Fiber-Wireless-Fiber (Fi-WiFi) and Wireless-Fiber-Wireless (Wi-FiWi).

1.4 Scope of Work

The scope of work for this thesis is shown in Figure 1.1. The system is focused on FiWi network because it is being used in the current global networking technology. Embedded system-based is chosen over software-based and industrial-based is due to its fast integration, scalability and suitability as an educational module especially for beginners. The chosen embedded system is Raspberry Pi 3B+ over Arduino and Banana Pi R1 because it is user friendly with low power consumption and fast implementation. Arduino is not chosen because it is not suitable to be used as a testbed for data communication field and it does not have enough processing power to handle the routing processes. Meanwhile, Banana Pi R1 is not chosen because the way it works is just by using “OpenWrt” command to start the routing process making the

users unable to create their own routing mechanism for research and educational purposes. Raspberry Pi 3B+ however, uses Python language making it more feasible because it has socket module, therefore, the routing mechanism can be built from scratch.

In order to achieve the objectives of this thesis, a FiWi testbed is designed and developed by using Raspberry Pi 3B+ instead of depending on theories, analytical calculations or a simulation. This is because the results obtained by the testbed are more accurate as it includes non-linear factors such as noise and heat loss in copper. However, the testbed is used in the lab-scale environment with short distance transmissions. Hence, the noise and heat loss are negligible. The fiber optics used in this project are mainly for data transmissions. While the results obtained theoretically, analytically and simulations come with many assumptions.

The performance of the testbed is evaluated for upstream and downstream communication in terms of throughput, delay and jitter. These performance parameters are affected by different data size which is the design parameter. The throughput of the testbed is then scaled and validated with industrial grade routers. The performance parameters are then re-evaluated for stress test and scalability test. During stress test, two traffics of data are sent simultaneously. Whereas, the scalability test is to test the performance of the testbed in other topologies which are fiber-wireless-fiber (Fi-WiFi) network and wireless-fiber-wireless (Wi-FiWi) network.

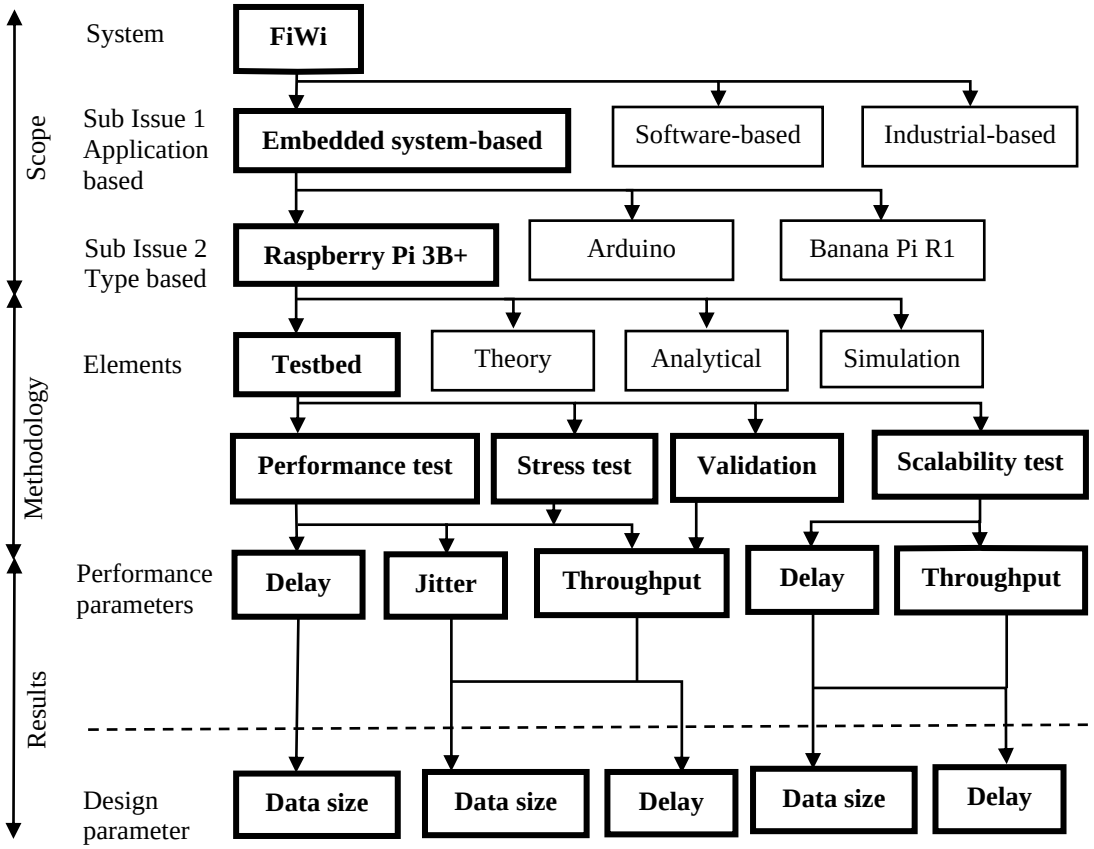

Figure 1.1 Scope of study

1.5 Thesis Outline

This thesis comprises of five chapters:

Chapter 1 begins by stating the current problem faced by the Internet. Then, the basic informations and advantages of FiWi network and testbed are presented in this chapter. The problem statements are discussed, and the objectives are outlined. Finally, Chapter 1 ends by explaining the scope of work to achieve project objectives.

Chapter 2 will begin by explaining in detail about the existing testbed architecture done by other researchers. Then, the routing media in terms of fiber, wireless and fiber-wireless that have been used in real life will be reviewed. The core subtopic in this chapter is FiWi. Hence, in-depth study on the theory of FiWi will be discussed in this chapter. Next, a review on the advantages and disadvantages of the existing router

testbeds which are software-based router, commercial-based router and embedded system-based router will be discussed in this chapter. This chapter ends by summarizing the literature review.

Chapter 3 will explain about the methodology of this project. Firstly, this chapter will explain about the proposed testbed architecture. The explanation includes router architecture, network environment which is the topology and the default system parameter. Then, the hardware setup and each of the components will be explained in details. The explanation about programming environment for the testbed will include the algorithm of the testbed, testbed scalability and how the testbed can be an educational module. This chapter also explains on the testbed parameters. Finally, will be ended by summarizing the whole chapter.

Chapter 4 will explain about the results and performance evaluation of the testbed for wireless transmission, fiber transmission and FiWi transmission. Firstly, the results in terms of throughput will be validated for each type of the transmission. Then, the explanation continues with the presentation of delay, throughput and jitter for each type of transmission. The performance for the stress test and scalability test will also be included. This chapter will be ended with the summary on the performance of the testbed.

Finally, Chapter 5 will present the conclusion and future work for this thesis. This chapter will be concluding the whole project and will describe on advantages and disadvantages of the testbed. The challenges and limitations while doing this project will be addressed in this chapter as well as the recommendations for future work to improvise the testbed.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A general information of FiWi has been briefly discussed in Chapter 1. According to Tornatore et al. [16], FiWi network architecture changes the future of Internet networking structure due to its capability to feed the bandwidth-hungry demand as well as the mobility and flexibility offered by wireless. This chapter explains in detail about FiWi network. However, FiWi still has a lot of rooms for improvements for future communication technologies. Therefore, numerous studies in various testbed architectures such as fiber, wireless, and FiWi have been studied thoroughly by many researchers. This chapter focuses on literature review of previous router testbeds such as software-based router, commercial-based router and embedded system-based router.

This chapter begins with a detail explanation of FiWi network in Section 2.2. Section 2.3 discusses on testbed architectures, such as fiber, wireless and FiWi. Then, Section 2.4 discuses on types of router testbeds, such as software-based router, commercial-based router and embedded system-based router. Finally, Section 2.5 summarizes this chapter.

2.2 FiWi Network

According to Martin Maier and Navid Ghazisaidi, FiWi network is a combination of fiber and wireless in the same network [23, 24]. FiWi provides cost effectiveness, robustness, flexibility, high capacity, reliability and self-organization [25-27]. A typical FiWi network consists of PON which comprises of OLT and ONU that are connected to wireless routers as shown in Figure 2.1.

In order to provide new high-speed services, PON has become one of the solutions to overcome bandwidth limitation of the last mile bottleneck access technologies as well

as increasing demand of high-spectrum bandwidth [28]. Currently, Time Division Multiplexing-based PON (TDM-PON) such as Gigabit Ethernet PON (GE-PON) and Gigabit-capable PON (G-PON) are widely used in different countries [28, 29]. Until today, the demand for high bandwidth is increasing rapidly due to advanced multimedia applications such as online gaming and video-on-demand. Hence, telecommunication groups such as Full Service Access Network (FSAN), Institute of Electrical and Electronic Engineers (IEEE) and International Telecommunication Union (ITU) have proposed Next Generation PON (NG-PON). NG-PON has two stages; NG-PON1 and NG-PON2. NG-PON1 is known as the midterm next generation which provides 10 *Gbps* for upstream and downstream. Whereas, NG-PON2 is a long-term next generation which provides not less than 40 *Gbps* [28, 30-32]. Moreover, according to [28], recent studies have been carried out on NG-PON2 enabling technologies such as 40 G TDM-PON and Wavelength Division Multiplexed PON (WDM-PON), Time and Wavelength Division Multiplexed PON (TWDM-PON) and Orthogonal Frequency Division Multiplexing PON (OFDM-PON). Amongst all these technologies, TWDM-PON has been selected as the best choice for NG-PON2 due to its capability to support backward compatibility, flexibility and static sharing [28, 33]. By 2020, NG-PON2 is expected to provide 128 *Gbps* to 500 *Gbps* of bandwidth, supports from 256 to 1024 ONUs, 20 *km* to 40 *km* extended passive reach option for the working path, low energy consumption, low capital and operational expenditures and coexistence with G-PON.

PON is a Point-to-Multipoint (P2MP) optical fiber network with no active elements in the signal's path which connects OLT with ONUs via 1: N optical splitters. PON typically uses tree topology to maximize the coverage, allow flexibility, and minimize the number of network splitting. Hence, it reduces the optical power loss and increases optical reach [28]. Moreover, the deployment of PON technology in access network provides end-to-end transparency, less processing and is not affected by electrical noises [28].

At the wireless side of FiWi network, it provides broadband services for not only mobile users but also fixed subscribers. Wireless technology in FiWi is necessary due to various limitations such as geographical constraints, economical balance, provider's

strategy, and damage situation in case of disasters that cause optical fiber to not be able to be deployed [34]. According to Van et al. [35], many studies on FiWi architectural have been done and the most widely considered FiWi network architecture is based on PON and wireless networking technologies such as Wireless Local Area Network (WLAN), World Interoperability for Microwave Access (WiMAX), Long Term Evolution (LTE), and Wireless Mesh Network (WMN).

According to Maier and Ghazisaidi [23], WLANs based on IEEE 802.11 have become very popular in providing different data services. In general WLAN architecture, an Access Point (AP) is connected to the internet or other WLANs through a distribution system (DS). In this architecture, wireless clients communicate with their associated AP using the medium access control (MAC) protocols defined in the IEEE 802.11 specifications [36-38]. The IEEE 802.11 MAC layer deploys the distributed coordination function (DCF) as a default access technique. In this contention-based scheme, wireless clients associated with the AP use their air interfaces for sensing the channel availability. If the channel is idle, the source sends its data to destination through the associated AP. If more than one wireless client tries to access the channel simultaneously, a collision occurs. Hence, carrier sense multiple access/collision avoidance (CSMA/CA) access protocol is introduced in this technology to avoid collisions.

The initial IEEE 802.16 WiMAX standard was established in the frequency band from 10 *GHz* to 66 *GHz* providing up to 75 *Mbps* line-of-sight (LOS) connections for both point-to-multipoint and mesh nodes [23]. The WiMAX physical (PHY) layer supports four different modulation schemes; wireless metropolitan area network-single carrier (WMAN-SC), WMAN-single carrier access (WMAN-SCA), WMAN-orthogonal frequency division multiplexing (WMAN-OFDM) and WMAN-orthogonal frequency division multiple access (WMAN-OFDMA). WMAN-SC is designed for the frequency band from 10 *GHz* to 66 *GHz*, whereas other modulation schemes are used for the frequency band from 2 *GHz* to 11 *GHz*. Additionally, WiMAX PHY layer can transfer data bidirectionally by using time division duplex (TDD) or frequency division duplex (FDD). Meanwhile, WiMAX MAC layer is responsible for assigning

connection identifiers (CIDs) as well as allocating bandwidth to subscriber stations [23].

LTE has been defined by the third generation partnership project (3GPP) as fourth generation (4G) cellular network technology for high-speed wireless end users [38-40]. The first amendment of LTE provides a transmission rate of 300 *Mbps* and operates in both TDD and FDD modes [38]. LTE operators are given the flexibility to define the size of bandwidth, ranging from below 5 *MHz* to 20 *MHz* [23]. LTE aims at providing a smooth evolution from 3GPP to 3GPP2 cellular networks such as wide-band code division multiple access/high-speed packet access (WCDMA/HSPA) and code division multiple access (CDMA). Typically, OFDM is used in the downlink radio transmission of LTE networks. Moreover, LTE supports advanced multi-antenna schemes such as multiple input multiple output (MIMO) antennas, transmit diversity, spatial multiplexing and beamforming.

WMN has been envisioned to enhance flexibility, increases reliability, and improve the performance of wireless networks [41, 42]. There are two main approaches in the design of wireless networks; infrastructure networks and infrastructure-less networks [23]. In infrastructure networks, the wireless clients rely on an underlying infrastructure for communication via a central control point such as AP. Global system for mobile communications (GSM) and universal mobile telecommunications system (UMTS) are the typical examples for infrastructure wireless networks. On the other hand, in infrastructure-less networks, wireless clients communicate with one another directly. It is also known as mobile ad-hoc networks (MANETs) which enable wireless clients to act as routers [23]. The convergence of these two approaches leads to WMN. WMNs employ multihop communications to forward traffic to and from wired DS. WMNs are expected to be widely deployed due to their ability to provide ubiquity, convenience, cost-efficiency, and simplicity [23, 41, 42].

These wireless networking technologies can be implemented by using Radio-over-Fiber (RoF) [43]. According to Rajpal and Goyal [34], RoF is potential to be the backbone of the wireless access network and it has gained significant momentum in the last decade as the last-mile access scheme. RoF is an analog optical link to transport

data over optical fiber by transmitting modulated radio frequency (RF) signals to and from CO to base station or Remote Antenna Unit (RAU) [44-46]. This modulation can be done directly with the radio signal or at an intermediate frequency. In other words, RoF transport information over optical fiber by modulating light pulses with the radio signals [34]. RoF serves as a high-speed WLAN. The frequencies of RoF systems span a wide range which is in GHz region. Most of signal processing including encoding, multiplexing, RF generation, and modulation are carried out by CO and shared with several base stations, hence making it easy to install and maintain [34, 43]. According to Singh and Singh [43], RoF has several applications other than WLAN, such as Video Distribution Systems (VDS), satellite control, cellular networks, vehicle communication, and mobile broadband services.

Another approach of wireless technology is by using Radio-and-Fiber (R&F) technology. R&F is an improvement of RoF which uses two MAC protocols that make simpler transportation of data from wireless to optical and vice versa. Besides, R&F has also distributed processing with storage capabilities and can perform additional functions. It reduces the functionality at CO and the task can be managed at ONU. Compared to RoF, R&F has better QoS, less propagation and larger network coverage [24, 47]. In this thesis, R&F is used in the proposed testbed because the wireless and fiber links are using different MAC protocols [21, 48, 49]. R&F provides extended coverage of the network without having to install optical backhaul which has limited length and size of fibers, making the testbed to be cost-efficient.

2.3 Testbed Architecture

Testbed is an equipment or setup which is used to test a new type of technology in order to prove a new concept or to enhance the current technology. There are two types of testbed; field testbed and lab-scale testbed. Field testbed is used in a wide scale area such as in an industrial area that requires heavy duty equipment and setup. Whereas, lab-scale testbed is used in a lab which uses small and simple equipment such as microcontroller and microprocessors. Testbed can be implemented for various architectures and setups in order to operate. Hence, this section explains testbed architectures in three different media; fiber, wireless and FiWi. Fiber architectures

such as PON is used in a testbed in order to provide very low latency, reliability, and robustness for the data transmission in a large amount of bandwidth to achieve a great performance. Whereas, wireless architectures such as WiFi is used in order to provide mobility and scalability on a testbed so that it can be implemented in wide area. The summary of these architectures are tabulated at the end of this section.

2.3.1 Fiber

Szymanski and Rezaee [50] proposed a testbed as a proof-of-concept of Guarantee-Rate (GR) by using Field-Programmable Gate Array (FPGA) and PON. The testbed comprises of two planes; control plane and forwarding plane. At the control plane, SDN is implemented to control 128 traffic flows through the packet switches. Whereas at the forwarding plane, it consists of eight controllers and Altera Cyclone IV FPGA as the packet switches. This testbed architecture reduces the end-to-end delay between clients. The control plane has the ability to bypass Layer 3 (L3) IP routers using a Layer 2 (L2) underlay which improves the energy efficiency significantly. The FPGA controller can handle routers and switches with aggregate data rates in hundreds of *Terabits per second (Tbps)*.

Yang et al. [51] developed a testbed in order to analyse the performance of software defined optical access network (SDOAN) architecture for remote unified control based on OpenFlow-enabled PON. The testbed comprises of two planes; control plane and data plane. The control plane is where OpenFlow is installed on a server computer to remotely control the traffic flows on the data plane. The data plane is where PON is deployed to transport all the data from a source to a destination. This architecture is designed to allocate the network bandwidth resources and detect the status of the network flows in real time flexibly and efficiently. The control plane and data plane communicate with one another via Reconfigurable Optical Add-Drop Multiplexers (ROADMs). SDOAN enhances the resource utilization and QoS guarantee of each user effectively through unified control plane and reduce operating expenses by remote interaction and operation. Besides, the separation of control and data plane has a positive impact on the network because if there is any broken links on the data plane, it will not affect the control plane. Also, if the control plane is broken, it will not affect

the data plane. Hence, this type of architecture saves the cost of maintaining the network. Another work by Yang et al. [52] has similar architecture with [51]. Instead of using server computers to enable OpenFlow, [52] used netFPGA at the control plane. However, netFPGA shows lower performance than server computers as netFPGA has lower specifications. Hence, the control and management on the network is less effective.

Okamoto et al. [53] proposed a programmable OLT (P-OLT) and programmable ONU (P-ONU) testbed. The aim of this work is to analyse the performance of the proposed P-OLT and P-ONU in terms of throughput and packet loss rate. The architecture of this testbed comprises of a proxy server, P-OLT, P-ONU, a server and a client. The proxy server is placed at the control plane of the testbed to be used as the buffer for the data, control the transmission timing periodically and provides traffic shaping of burst traffic. P-OLT and P-ONU are programmed by using FPGA. Whereas the server and client act as the end-to-end data transmission nodes. The advantage of this architecture is that it can be implemented for a large-scale testbed. The results of this testbed show that the packet loss rate is consistently at 0% even when the number of ONU increases. However, the throughput decreases as the number of ONU increases. Despite of having such advantages, FPGA is complex because it has a lot of configurable logic blocks where each of them comprises of many logic gates and look up tables. Due to this complexity, FPGA requires a lot of power consumptions compared to other embedded system hardware like Raspberry Pi. Furthermore, FPGA is also complex in terms of programming. For example, if the user wants to create a delay, the user needs to use arithmetic functions first to define the prescaler in order to slow down the default clock. Moreover, FPGA does not store the program when it turns off. This means that, once FPGA is switched off, all the programs and functions defined by the user will be erased. This makes FPGA takes longer boot time and takes longer time to do the experiments. Furthermore, FPGA needs an external flash memory in order to overcome this problem which makes it further cost-inefficient. It is also not durable since it is sensitive to electrostatic on human body.

Luis et al. [54] proposed a Spatial Division Multiplexing (SDM) network testbed that consists of three nodes that are connected together via 19-core multicore fibers. The

nodes communicate with each other via ROADM. This work is to provide performance evaluations for the proposed testbed. The results of the testbed show that SDM network can provide ultra-high capacity links and high-quality connections even when packet and circuit switching are in the same network.

In 2018, Azofra et al. [55] created a testbed with fiber medium to evaluate the performance of Software Defined Network (SDN)-based GPON. The testbed provides fast, efficient and accurate QoS management by using Raspberry Pi. It comprises of an OLT, two fiber splitters and 5 *km* standard Single Mode Fiber (SMF). The OLT has two main components, which are one L3 model Optical Network Terminal (ONT) which includes router functionalities and comply with International Telegraph Union – Telecommunication Standardization Sector (ITU-T) G.9894.x and G.988 standards and one Raspberry Pi to enable OpenFlow. Hence, the performance of the network can be improved by using this testbed because the user can freely manage and control the traffic flows.

Based on the literature review in this section, it can be found that most authors use FPGA and SDN-based control plane as the proposed testbed to enhance the performance of PON. Despite of having improvements such as programmability and flexibility, as well as fast, and efficient, the performance of FPGA and SDN-based control plane are limited compared to industrial grade hardware or server computers due to their limited specifications. Compared to FPGA, SDN-based control plane such as Raspberry Pi is simpler, fast reconfigurable and implementation, cost efficient, lower power consumption and space-friendly. This is because Python is used as the main the programming language in Raspberry Pi because the user does not require to do any complex mathematical or arithmetic expressions just to execute a simple function such as delay. It is fast reconfigurable and fast implementations because the user does not require to reprogram it when it is turned off. All the programs have been stored and can be automatically executed when it is turned on. It is more cost-efficient than FPGA because it can execute the same functions and program as FPGA at a lower price. Unlike FPGA, Raspberry Pi requires low power consumption to boot because it can also be powered by using power bank. Lastly, Raspberry Pi has more durability since it is not as sensitive to electrostatic as FPGA. Hence, an extensive experiment

can be done in various environments either it is indoor or outdoor. With that, it can be concluded that Raspberry Pi is more practical and simpler to be used as a testbed than FPGA.

2.3.2 Wireless

In 2015, Singh and Talasila [56] proposed a wireless testbed to provide comparative analysis of different routing protocols which are Better Approach to Mobile Ad hoc Networking (BATMAN)-advance and Hybrid Wireless Mesh Network (HWMN) in WSN. There are five TP-LINK WR1043ND routers that used in this testbed. TP-LINK WR1043ND routers are high speed gigabit wireless routers that capable of achieving up to 450 *Mbps* when operated at 2.4 *GHz*. These routers also have three antennas to provide the users with larger coverage and stronger wireless signals. Hence, more accurate results can be achieved.

In 2016, Prusty et al. [57] proposed a testbed to analyse the link quality in packet routing for Wireless Ad-hoc Sensor Network (WASN). The testbed comprises three types of hardware which are NI WSN-9791 Ethernet gateway, NI WSN-3202 programmable analogue input node, and NI WSN-3212 programmable thermocouple input node. NI WSN-9791 Ethernet gateway is to coordinate the communication between distributed WSN nodes and host controller or base station. NI WSN-3202 programmable analogue input node acts as the wireless route to transmit packets from other nodes to the gateway. NI WSN-3212 programmable thermocouple is a temperature sensor where it sends the temperature data to the wireless router. The testbed is also useful in understanding of the characteristics and behaviour of low-power links in WASN to help designing a suitable protocol.

Barolli et al. [58] proposed a wireless testbed to analyse the performance of OpenWRT-based testbed for Content Centric Network (CCN) by using Optimised Link State Routing (OLSR). In this testbed, Raspberry Pis are used as the wireless nodes. OpenWRT command is used on the Raspberry Pis for them to communicate with one another. The advantage of using Raspberry as the testbed is that since it operates on an open source kernel, it is reconfigurable because it can be embedded

with various algorithms. However, Raspberry Pi only suitable to be implemented in a small-scale experiment due to its small bandwidth and limited specifications.

Zhang et al. [59] proposed a wireless testbed by implementing Multichannel Opportunistic Routing (MOR) in order to improve the robustness of WS network to interference. In this testbed, 16 Zigbees are used as the wireless nodes to implement MOR. Zigbee is a low-cost and low-power wireless hardware. However, Zigbee has low-rate data transmission which only suitable for a small scale testbed such as in a lab.

Jecan et al. [60] proposed a testbed to evaluate the performance for industrial WSN dual-standards which are ISA100.1a and WirelessHART in star and mesh topologies. There are two types of hardware for the testbed; VR950 gateway and VS210 wireless sensor nodes. VR950 gateway is to enable the ISA100.1a and WirelessHART standards. VR950 is widely used in industrial field such as oil and gas, mining, and manufacturing where safety, security and reliability are the priorities. Whereas the VS210 wireless sensor nodes are to receive from other temperature, humidity and pressure sensors wirelessly. Since the experiment is for industrial WSN, VS210 has a short circuit protection for all input and output ports. In mesh topology, there are 100 wireless sensor nodes, whereas, in star topology, there are 50 wireless sensor nodes. By implementing dual-standards in one testbed, a better overall performance of industrial WSN can be achieved.

In 2019, Pakzad et al. [61] proposed a wireless testbed to provide performance comparison between two widely used protocols in WMN which are OLSR and BATMAN. The testbed comprises of 37 wireless nodes which are equipped with Intel core i7-2600 processor, 4 *GB* RAM, 240 *GB* SSD and two wireless interfaces with three antennas. The nodes were arranged in a 1 *m* grid covering 90 m^2 area. USRP is placed between the nodes to generate interference. The advantage of this architecture is that it is built in a secluded area where no other interference other than from USRP. Hence, the results obtained are accurate.

Based on the literature review in this section, we can find that most authors used high specifications for wireless nodes and routers in order to achieve robust and efficient results. There are only two authors who proposed a small scale testbed which are Jecan et al. [60] and Barolli et al. [58].

2.3.3 Fiber-Wireless

In 2016, Nguyen and Cheriet [62] proposed a work in order to provide a solution for rearchitecting a telecommunication company's CO. This solution offers services in a smart community by using SDN. The testbed has three main components. The first one is a core switching platform that provides optical multiservice to link the Smart Residence to Internet providers, as well as to international partners. This switching platform is integrated into the smart edge that makes it programmable by implementing SDN functions such as dynamic routing and traffic filtering. The second component is an optical access platform that consists of virtual home gateways and optical aggregation switches that link smart home and WiFi APs to the core switch. The third component is a set of telco-grade blade servers that provide various telecommunication services as well as monitoring, power management, and emergency alerting. Based on the monitoring services, a database containing user, power, and resource data, analytical services are developed to extract the information, then, optimizes the resource and service provisioning.

Xu et al. [63] proposed a work to compare the performances of filter-bank multicarrier (FBMC) and OFDM with and without centralized pre-equalization in a FiWi integrated mobile fronthaul (MFH) network. The testbed comprises of one baseband unit (BBU) pool, one RAU, two user equipment (UE) terminals, distributed feedback (DFB) laser that is used as downlink light source, and Tektronik 7122C arbitrary waveform generator (AWG).

In 2017, Rimal et al. [64] provide an enhancement for capacity-centric FiWi broadband access networks by implementing cloudlet-aware resource management scheme. The testbed comprises of Sun Telecom GE8100 as the OLT, four Sun Telecom GE8200 as the ONUs, WLAN access point, Dell Optiplex 9020 desktop as the cloud server and

Dell Inspiron 3521 laptop as the wireless edge device. However, in 2018, Rimal et al. [2] proposed an improvement by designing a capacity-centric FiWi broadband access networks enhanced with edge computing to guarantee low end-to-end latency. The testbed hardware are the same as before, but instead of using cloudlet server and WLAN access point, the author used Ubuntu 14.04 Desktop as the Edge Cloud and ZyXEL NWA570N wireless access point. Ubuntu 14.04 is a Linux-based Operating System (OS) that enables the user to freely reconfigure the network because it is an open source OS. ZyXEL NWA570N has 300 *Mbps* of throughput and can cover wide range of wireless communication.

In 2018, Ridwan et al. [21] conducted an experiment in order to achieve two objectives which are to study the performance evaluation of the upstream FiWi testbed transmission in terms of throughput, transmission time, and jitter and also its reconfigurability. The testbed comprises of four Universal Software Radio Peripheral (USRP) 2922, 1:4 splitter, optical to electrical (E/O) converters, electrical to optical (O/E) converters, and fiber optics. USRP are used as OLT and ONU as well as due to their reconfigurability. USRP wireless signals can reach up to 100 *m* when operated at 30 *dB* output power.

Liu et al. [65] proposed a FiWi testbed to provide performance evaluation of integrated heterogenous networking scheme for multi-access networks that uses network virtualization to achieve the dynamic orchestration of the network, storage and computing resource. This testbed has two planes; control plane and physical infrastructure plane or data plane. At the control plane, MEC server is used to install OpenFlow in order to control and manage the data flow in the data plane. SDN switches are used to enable control plane and data plane communicate intercommunication. At the data plane, the SDN switches acting as the routers are connected with one another via 20 *km* and 40 *km* fiber optics. One of the routers is connected with AP to enable the wireless access to the mobile devices, in this case, laptops.

Turk and Zeydan [66] conducted an experiment to introduce a guidance framework to Mobile Network Operators (MNOs) so that they can enable converged fixed and

mobile services with customized QoS support. The main hardware in this testbed are Nokia 7360 ISAM FX as the OLT, three Nokia 7750 SR-7 as the routers and security gateway, radio access network (RAN) to provide cellular services and Nokia 7705 SAR-A cell router. The routers for this testbed support from 10 *Gbps* to 2 *Tbps* of bandwidth as well as hierarchical QoS (HQoS).

Lastly, work by Alfadhli et al. [67] proposed a FiWi testbed that provides an experimental quantitative latency analysis of different low function split options at the fronthaul for ultra-reliable low latency communications (URLLC). The testbed comprises of two computers that act as a server and a client, E/O and O/E converters, USRP-b210 to generate interference, AWG to produce 1 *GHz* bandwidth OFDM signal at a carrier frequency of 1 *GHz*, photodiode and laser diode to produce light sources. USRP-b210 is an open and reprogrammable software defined radio that allow the user to immediately begin developing with GNU radio and develop a prototype with high performance.

Based on the literature review in this section, we can find that there are various field of studies involving FiWi from small scale testbed to an industrial scale testbed, proving that FiWi has a lot of rooms for improvements. The advantage of small scale testbed is that it is cost efficient, space friendly and able to produce results close to industrial scale testbed.

To conclude this section, Table 2.1 summarizes all of the testbed architectures including the objective of each work, the testbed details and the media used in the testbed.

Table 2.1 Testbed architecture summary

Title	**Objective**	**Testbed details**	**Medium**
An FPGA Controller for Deterministic Guaranteed-Rate Optical Packet Switching, 2015 [50]	- To provide a proof-of-concept GR testbed by using FPGA and PON.	- The forwarding plane of the testbed consists of eight controllers and Altera Cyclone IV FPGA as the packet switches. - SDN control plane of the testbed routes 128 traffic flows through the packet switches.	Fiber
Experimental Demonstration of Remote Unified Control for OpenFlow-Based Software-Defined Optical Access Networks, 2016 [51]	-To analyse the performance of SDOAN) architecture for remote unified control based on OpenFlow-enabled PON	- The testbed comprises of two planes; control plane and data plane - In the control plane, OpenFlow is installed in the servers - In the data plane, PON is deployed. - Control plane and data plane communicate with each other by using four (ROADMs)	Fiber
SUDOI: Software Defined Networking for Ubiquitous Data Center Optical Interconnection, 2016 [52]	-To study the feasibility and efficiency of the proposed architecture by using the testbed with OpenFlow-enabled optical nodes.	- The testbed comprises of two layers; control layer and data layer. - In control plane, OpenFlow is installed in netFPGA - In data plane, PON is deployed and can communicate with control plane via ROADM.	Fiber

Title	Objective	Testbed details	Medium
Logical Optical Line Terminal Technologies Towards Flexible And Highly Reliable Metro And Access-Integrated Networks, 2017 [53]	-To analyse the performance of P-OLT and P-ONU.	- Hardware used in this testbed: 1) proxy server – used as the buffer for the data, controls the transmission timing periodically, and provides traffic shaping of burst traffic 2) P-OLT and P-ONU 3) Server and client	Fiber
Demonstration of an SDM Network Testbed for Joint Spatial Circuit and Packet Switching, 2018 [54]	- To provide performance evaluations for the proposed testbed.	- SDM network testbed consists of three nodes that are connected together via 19-core multicore fibers. - The nodes communicate with each other via ROADM.	Fiber
Implementation of a Testbed to Analysis a SDN Based GPON, 2018 [55]	-To evaluate the performance of SDN-based GPON to provide fast, efficient and accurate QoS management.	- The testbed comprises of: 1) OLT – consists of one L3 model ONT and one Raspberry Pi to enable OpenFlow 2) two splitters 3) 5 *km* standard SMF	Fiber

Title	Objective	Testbed details	Medium
A Practical Evaluation for Routing Performance of BATMAN-ADV And HWMN in A Wireless Mesh Network Testbed, 2015 [56]	- To provide a comparative analysis of different routing protocols which are BATMAN-advance and HWSN in WMN.	- Five TP LINK WR1043ND routers are used for BATMAN-advance and HWSN experimental setup.	Wireless
Testbed for Link Quality Assessment in Wireless Ad-hoc Sensor Network, 2016 [57]	- To analyse the link quality in packet routing for WASN by conducting an experiment on a testbed.	- The testbed consists of three types of hardware: 1) NI WSN-9791 Ethernet Gateway – to coordinate the communication between distribute WSN nodes and host controller or base station. 2) NI WSN-3202 ±10 *V* Programmable Analog input node – act as the wireless router to transmit packets from other nodes to the gateway 3) NI WSN-3212 Programmable Thermocouple input node – sense the temperature, then send the data to the wireless router.	Wireless

Title	Objective	Testbed details	Medium
Experimental Results of A Raspberry Pi and OLSR-Based Wireless Content Centric Network Testbed Considering OpenWRT OS, 2016 [58]	- To analyse the performance of OpenWRT-based testbed for CCN by using OSLR protocol.	- Raspberry Pis are used as the wireless nodes. OpenWRT is used to communicate with one another.	Wireless
MOR: Multichannel Opportunistic Routing for Wireless Sensor Networks, 2017 [59]	- Implementing MOR to improve the robustness of WSN network to interference.	- 16 Zigbees are used as the wireless nodes to implement MOR.	Wireless
A Dual-Standard Solution for Industrial Wireless Sensor Network Deployment: Experimental Testbed and Performance Evaluation, 2018 [60]	- To evaluate the performance for a dual standard Industrial WSN by employing star and mesh topology.	- The testbed consists of two types of hardware; 1) VR950 Gateway – to enables the ISA100.1a and WirelessHART standards. 2) VS210 wireless sensor nodes – to receive data from temperature, humidity and pressure sensors wirelessly. - In mesh network, there are 100 wireless sensor nodes, whereas, star network has 50 wireless sensor nodes.	Wireless

Title	Objective	Testbed details	Medium
R2Lab Testbed Evaluation for Wireless Mesh Network Experiments, 2019 [61]	- To provide a comparison between two widely used protocols in WMN; OLSR and BATMAN	- Using 37 wireless nodes which are equipped with: 1) Intel Core i7-2600 processors, 4 *GB* RAM, 240 *GB* SSD. 2) Two wireless interfaces with 3 antennas each; Atheros 802.11 93xx a/b/g/n and Intel 5300 chips. - The distance for the wireless nodes are 1 *m* from each other in a 90 m^2 area. - Universal Software Radio Peripheral (USRP) is used to generate interference.	Wireless
Virtual Edge-Based Smart Community Network Management. IEEE Internet Computing, 2016 [62]	- To provide a solution for rearchitecting a telecommunication company's CO to offer services in a smart community by using virtual network function elements which is SDN.	- The testbed has three components: 1) a core switching platform is implemented with SDN functions 2) An optical access platform consists of virtual home gateways and optical aggregation switches linking smart home and WiFi APs to the core switch. 3) A set of telco-grade blade servers to provide various telecommunication services	FiWi

Title	Objective	Testbed details	Medium
FBMC in Next-Generation Mobile Fronthaul Networks With Centralized Pre-Equalization. IEEE Photonics Technology Letters, 2016 [63]	-To compare the performances of FBMC as well as OFDM with and without centralized pre-equalization in a fiber-wireless integrated MFH network.	- The testbed comprises of: 1) one BBU pool 2) one RAU 3) two UE terminals 4) DFB laser is used as DL light source 5) Tektronik 7122C AWG	FiWi
Cloudlet Enhanced Fiber-Wireless Access Networks for Mobile-Edge Computing, 2017 [64]	-To provide an enhancement for capacity-centric FiWi broadband access networks by implementing cloudlet-aware resource management scheme.	- The testbed comprises of: 1) Sun Telecom GE8100 as the OLT 2) four Sun Telecom GE8200 as the ONUs 3) WLAN access point 4) Dell Optiplex 9020 Desktop as the cloudlet server 5) Dell Inspiron 3521 laptop as the wireless edge device	FiWi

Title	Objective	Testbed details	Medium
Experimental Testbed for Edge Computing in Fiber-Wireless Broadband Access Networks, 2018 [2]	-To design capacity-centric FiWi broadband access networks enhanced with edge computing to guarantee low end-to-end latency.	- The testbed comprises of: 1) Sun Telecom GE8100 as the OLT 2) four Sun Telecom GE8200 as the ONUs 3) ZyXEL NWA570N wireless access point 4) Ubuntu 14.04 Desktop as the Edge Cloud 5) Dell Inspiron 3521 laptop as the wireless edge device	FiWi
Feasibility Study of a Reconfigurable Fiber-Wireless Testbed Using Universal Software Radio Peripheral, 2018 [21]	-To study the performance evaluation of the upstream FiWi testbed transmission in terms of throughput, transmission time, and jitter. -To test the testbed reconfigurability	- The testbed comprises of: 1) four USRP 2922 as the OLT and ONU 2) 1:4 splitter 3) O/E and O/E converters 4) fiber optic	FiWi
Performance Evaluation of Integrated Multi-Access Edge Computing And Fiber-Wireless Access Networks, 2018 [65]	-To provide performance evaluation of integrated heterogenous networking scheme for multi-access edge computing and FiWi access networks that uses network virtualization to achieve the dynamic orchestration of the network, storage and computing resource.	- The testbed comprises of: 1) 20 *km* and 40 *km* fiber optics 2) core switch 3) three SDN switches with optical and Ethernet forwarding capability 4) WLAN Access Points 5) MEC servers	FiWi

Title	Objective	Testbed details	Medium
An Experimental Measurement Analysis of Congestion Over Converged Fixed and Mobile Networks, 2018 [66]	-To introduce a guidance framework to MNOs so that they can enable converged fixed and mobile services with customized QoS support.	- The testbed comprises of: 1) Nokia 7360 ISAM FX as the OLT 2) three Nokia 7750 SR-7 routers and security gateway 3) radio access network to provide cellular services 4) Nokia 7705 SAR-A cell router	FiWi
Latency Performance Analysis of Low Layers Function Split for URLLC Applications in 5G Networks, 2019 [67]	-To provide an experimental quantitative latency analysis of different low function split options at the fronthaul for ultra-reliable low latency communications URLLC	- The testbed comprises of: 1) two computers to serve as client and server 2) E/O and O/E converters 3) USRP-b210 to generate RF interference 4) AWG is used to produce 1 *GHz* bandwidth OFDM signal at a carrier frequency of 1 *GHz*. 5) photodiode and laser diode	FiWi

2.4 Router Testbed

This section explains router testbed in three different approaches; software-based, commercial-based, and embedded system-based. Software-based router is a router that developed by using softwares and the produced results are purely from simulations. Commercial-based routers are routers that are available in the market either off-the-shelf or industrial grade routers because they are widely used in various industrial fields. Whereas, embedded system-based routers are routers that are built by using embedded system hardware such as FPGA and Raspberry Pi, which are mostly used in lab-scale experiments. The summary of these approaches are tabulated at the end of this section.

2.4.1 Software-based Router

In 2015, Bahnasy et al. [68] proposed a simulation evaluations by using OpenFlow to control both packet which is called as OpenFlow Messages Mapping and optical network called as OpenFlow Extension. The simulation results are then compared with Generalized Multiprotocol Label Switching (GMPLS) approach by using Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON). DRAGON comprises of two major components that are used as the control plane for GMPLS which are, client system agent (CSA) and virtual label switch router (VLSR). The comparison results show that the OpenFlow Extension solution outperforms OpenFlow Messages Mapping and GMPLS solutions because it has lower end-to-end light path setup time and lower blocking ratio and control traffic compared to GMPLS. One of the advantages of this work is VLSR and OpenFlow uses shortest path routing causing the delay for the packet is low. Secondly, VLSR is flexible because it can control different types of switches such as Ethernet, TDM or optical. Furthermore, VLSR is reliable because it uses TCP/IP protocol to transmit the packets. On the other hand, OpenFlow controller can recalculate alternative light path in case of light path failures. OpenFlow also has maximum flexibility and manageability because all the functionalities are controlled by a single OpenFlow controller. However, these approaches have disadvantages, such as, VLSR can cause high delay for the packet because the algorithm is complicated. Its flexibility and manageability are also low

because the signalization and reservation messages must be updated and exchanged among all intermediate VLSRs.

Runge et al. [69] proposed a QoS aware software router model by using Network Simulator 3 (NS3) and evaluate the performance optimizations. The results show that the different scheduling strategies of a software router have significant influence on the performance of handling real time traffic. The advantage of this QoS aware software router is it has dedicated receiver ring for prioritized packet processing which will be served according to a specific scheduling strategy. Hence, the packet will be process faster for low latency constraints real-time traffics such as Voice over IP (VoIP), video conferencing and online gaming. However, the disadvantages are explained in [70]. The first disadvantage is that the user does not build up all the functionalities from scratch, instead, they are utilizing the existing models. The users must always have to consider which models that are suitable to be used in a certain context because if it is used in a different context, it malfunctions completely. The second disadvantage is that it has scalability limits. Which means, there are limited memory and it requires certain amount of computation time. Every nodes, channels and other components require memory space, therefore, their numbers are limited by available memory.

Addie and Natarajan [71] proposed an analysis for Netml systems by using the simulations of NS3 and Click routers. The Netml system enables an XML description of a network to be converted into an NS3 program, then, run the simulation to collect and plot the results on Netml public server. A basic implementation of IPtables is also implemented. Users are able to specify any number of filtering rules within the forwarding chain of the filter table. The advantage of using Click router in this work is that it can build its own firewall. It is completely flexible, configurable and customizable according to user's needs. According to Suresh and Merz [72], the main disadvantage of using NS3 and Click is that some of Click's MAC elements are not supported by NS3 yet. Therefore, it will be a tedious work to find a compatible MAC elements making the work more complicated.

The software router by Goswami et al. [73] studied on how to achieve the estimation of optical-layer power consumption and cost for a long distance optical networks using Wavelength Division Multiplexing (WDM). The simulation was conducted by using C++ programming language. The test is implemented on two different networks which are IP-over-WDM and IP-over-MPLS-over-WDM networks. In the former network test, which is IP-over-WDM network, IP routers communicate directly with Wavelength-Routed Optical Network (WRON) and bandwidth requests are equal to light-path capacity. On the other hand, in IP-over-MPLS-over-WDM network, the IP routers communicate with WRON through MPLS routers, hence the bandwidth requests arrived as Label Switch Path (LSP). Therefore, it is lower than light-path capacity. The advantage of this work is that it is cost efficient and reconfigurable. However, the disadvantage of C++ in this work is that it is not a specific network simulation tool. Hence, a lot of assumptions must be made in order to simulate as close to real network as possible. This will make the produced results to be inaccurate and cannot be implemented in real network.

In 2016, Ohsugi et al. [74] developed a power consumption model of multicore software Named Data Network (NDN) router. By applying this model, it is reported that caching can reduce power consumption when the computation of caching is as low as during data forwarding. The advantage of using NDN software router is that it reduces the amount of traffic forwarded towards upstream routers and thus, reduces the power consumed by their forwarding devices. However, the disadvantage of NDN is that it has limited scalability, which means, it cannot manage when the network becomes too complex.

In 2017, In a different work by Xu et al. [75], a framework called Minos has been proposed to regulate router actions on data planes. Action Identifier (AID) is the input for Minos to perform lookups in Regulated Action Table (RAT). Minos is expected to get a couple of distinct factors that will affect the security of a programmable router such as cost and effectiveness. This is because the reprogrammability of a router makes it vulnerable and exposed to the risk of being hacked. Minos is implemented and evaluated separately on Click and Data Plane Development Kit (DPDK). DPDK comprises of a set of libraries that support efficient implementations of network

functions [76]. According to Rajesh et al. [77], Intel DPDK is able to boost packet processing performance and throughput, hence, allowing more time for data plane applications. However, DPDK does not have cache coherence and locality making its access time to be slow.

In 2018, the work by Tajiki et al. [78], focuses on a new traffic engineering architecture for SDN-MPLS network, where they proposed improving flow-level management flexibility. It is done by applying OpenFlow-enabled switches at the edge of the network while the MPLS routers are the core router of the network. The simulation was done by using MATLAB2016b. The proposed scheme also re-assigns flows in the LSP to highly utilize the network resource. MATLAB2016b has improved functionalities in the toolbox that enable users to produce better equations and algorithms. Hence, more accurate results can be produced. The main disadvantage of using MATLAB in this work is that it can be slow to process and compute for a such complex hybrid network like in proposed work.

Lastly, in 2019, Kim et al. [79] proposed a new Internet architecture for the future mobile network called Mobile-Oriented Future Internet (MOFI). The architecture of MOFI comprises of two main components; the separation of control and data plane for getting an optimal data path and distributed identifier as the locator mapping control for alleviating traffic overhead at the central agent. The architecture of MOFI is built by using OpenFlow and Click modular router on a Linux platform. The results show that proposed MOFI able to provide mobility management efficiently and support the backward compatibility for the current IP network and IP version 6 (IPv6) applications. The advantage of Click modular router in this work is it is a flexible software router that enables the user to configure and customize freely according to the user's needs. According to Kohler et al. [80], Click can achieve a maximum loss-free forwarding rate of 333,000 of 64 *bytes* packets per second when run on Linux computers, proving that Click's modular and flexibility architecture is compatible with good performance. On the other hand, OpenFlow supports IPv6, hence, it can support more users compared to IPv4. However, user usually uses small elements to create configurations to develop Click routers. These elements cannot solve problems such as when control flow and data flow do not match the flow of packets. Therefore, large

elements are required to overcome these problems. Hence, this makes the development of a router complicated since the big elements run a complex process for a protocol like 802.1d [81].

As conclusion, various of software are used for the software router simulation such as VLSR, Click, NS3, DPDK, MATLAB2016b, C++ and NDN. Most of the authors were using Click modular router due to its flexibility, configurability and customizability. However, the result produced by the software routers do not consider the non-linear effects because there are no hardware involved.

2.4.2 Commercial-based Router

In 2014, Blair et al. [82] used four MPLS routers in their experiment, with multiple protocols such as IEEE C37.94, IEC 61850-9-2 Sampled Values, and IEC 61850-8-1 Generic Object-Oriented Substation Event (GOOSE). The study is to demonstrate and analyse the use of commercial IP/MPLS protocol to carry protection relay hardware to support power system protection functions. They also used Real-time Digital Simulator (RTDS) to simulate the power system and interfaced with the hardware protection relays. RTDS allows power system faults and other events to be simulated in real-time. The authors run two different experiments. For the first experiment, the author runs the experiment for nine times. The first five tests are to get the propagation delay results for a chain of IP/MPLS routers. According to the results, the jitter buffer size increases as the payload size increases. Thus, the propagation delay increases. For sixth test onwards are to get the results of propagation delay for ring topology of IP/MPLS routers. The advantage for the ring setup in sixth test compared to fifth test is that it has rerouting capability because the setup has an alternate path for the packets in case of broken link. There is small difference in terms of propagation delay for the sixth and seventh test due to the variation of delay in the hardware. At the eighth and ninth test, the propagation delay increases when number of C37.94 slots decreases even when the payload size is very small. This is because the main purpose of the slots is to define the end-to-end usable bandwidth for the packets. Hence, if the number of slots is small, the usable bandwidth will also be small causing the delay to be high. For the second experiment, the results show that IEC 61850 SV and GOOSE are better

than IEEE C37.94 in terms of circuit breaker tripped time, and backup intertrip time. Even the bandwidth used by these two protocols is greater, they are able to transfer much more information for three-phase voltage and current waveforms. The advantage of IP/MPLS routers in this work is that the routers has the capability to support ring topology. Hence, the data has better protection because it can be rerouted to another path in case of broken links. This feature is crucial especially in power system since the data must flow continuously all the time to detect faults in the system. However, as mentioned in earlier, the traffic flows in this router are still prone to congestions because the data for all the services are transmitted in one LSP.

In 2015, Feng et al. [83] proposed a testbed using an OpenFlow module that is embedded in the control plane of a commercial router. It is used to set up a datapath with an external OpenFlow controller using OpenFlow protocol. The commercial router used is DCRM 5980. The testbed proves that it could not only implement the software-defined networking functionalities for network control flexibility, but also it is easy for rapid deployment with updating the software image instead of adding or changing any hardware. The advantage of using DCRM 5980 router in this testbed is that it can be implemented with other firmware such as OpenFlow. The router also supports IPv6, hence it can support more users connected to it. It is also flexible and scalable. Furthermore, it supports VLAN and MPLS. However, after installing OpenFlow, the routing is based on the shortest path. Hence, the data will use the shortest path even the path is congested which can cause high delay in data transmissions and increase the number of packet loss.

In 2016, Sgambelluri et al. [84], implemented Segment Routing (SR) in two different networks, which are SDN-based and Path Computation Element (PCE)-based. SR technology is proposed to provide minimum depth segment list encoding for the data path, hence, less computation delay for both networks. In SDN-based network, SR controller is implemented in an OpenFlow controller. While in PCE-based network, SR is implemented where the nodes consist of commercial IP/MPLS routers. The results for both implementations are similar because the same path and segment list computation were implemented in both controllers. The routing mechanism of a commercial IP/MPLS routers is based on the short label header on the data rather than

long network addresses. Hence, this router avoids complex lookups in a routing table to speed up the traffic flows. However, the traffic flows in this router are still prone to congestions because the data for all the services are transmitted in one LSP.

In 2017, Tantayakul et al. [85] used two commercial routers to evaluate the performance of open source OpenFlow switches, which are OpenvSwitch (OVS) and ofsoftswitch13 (CPqD) in terms of UDP throughput, TCP throughput and percentage of packet loss. The two commercial routers are TP-LINK WR1043ND routers. One of the routers sets as an OpenFlow switch. Ryu controller is used to manage the routing path of OpenFlow switches while iperf tool is used to generate UDP and TCP traffics in order to measure and evaluate the performance. The results show that the maximum bandwidth of OVS is higher than CPqD for both UDP and TCP traffics. However, although CPqD has limited bandwidth of 50 *Mbps*, it provides faster OpenFlow handshake and uses less memory space compared to OVS. In this work, the advantage of using TP-LINK WR1043ND routers is that the firmware of the routers can be reinstall or upgraded to another firmware such as OpenFlow by using OpenWRT. These routers also provide accurate results in real-time because it has gigabit bandwidth by default, hence, more data can be transmitted at once. However, these routers have high packet loss which is around 57% to 58%. This is because the router was running on CPqD which only has 50 *Mbps* of bandwidth but the data rate generated by Iperf is 100 *Mbps*.

Lastly, in 2019, Chen et al. [86] proposed a local feature-based deep long short-term memory (LF-DLSTM) approach for WiFi fingerprinting indoor localization by using TP-LINK WDR4300 router. The experiments were done in two different environments, which are in a research lab and in an office. The proposed approach is compared with state-of-the-art methods for indoor localization. The results show that the proposed approach achieved the best localization performance in these two environments. When compared with state-of-the-art methods, it achieved improvements from 18.9% to 53.46% indicating the effectiveness of LF-DLSTM. The advantage of using TP-LINK WDR4300 router is that it has big coverage because it has three dual band external antennas. It also has high bandwidth that supports

simultaneous data transmission. Hence, more data can be transmitted at once to obtain accurate results. However, this router is not configurable.

As conclusion, commercial routers are able to produce accurate and high performance results. However, most of the commercial routers are not reconfigurable in terms of tweaking its existing program. Even though there are routers in the literature that embedded with OpenFlow by the authors, its current program is still cannot be reconfigured. From the literature, we can identify that commercial routers are prone to frequent congestions, high packet loss and not reconfigurable.

2.4.3 Embedded system-based Router

In 2016, Sivarman et al. [87] used netFPGA as the router to study the role of packet buffer memory on the power consumption of backbone routers. They have developed an algorithm for the memory components to sleep and awake when needed, while being able to control the resulting traffic performance degradation in the form of packet loss during congestion. They conducted a comprehensive evaluation of their algorithm by using the simulation of offline traffic traces taken from carrier/enterprise networks as well as online TCP flow in Network Simulator 2 (NS2). The evaluation also being implemented on a programmable router testbed which is the netFPGA. NetFPGA is connected to traffic generators and delay emulators to demonstrate the feasibility of implementing the algorithm in the hardware. After implemented on netFPGA, the algorithm has saved 40% of the energy when it is under very heavy load. The advantage of using netFPGA is it can be reprogrammable by using VHDL programming language.

Hoo and Kumar [88] proposed a distributed memory parallel FPGA router called Parallel Router Distributed Memory (ParaDiMe) to speculatively routes in parallel and dynamically detects the need of reducing the number of active processes in order to speed up the routing process in the network. The results were compared with other type of FPGA routing which is called Verilog-to-routing (VTR). Compared to VTR, ParaDiMe achieved an average speed up of 19.8 times with 32 processes while producing similar quality of results. The advantages of using FPGA is that it is

reconfigurable and reprogrammable. It also has high performance to produce accurate results.

In 2018, Concatto et al. [89] proposed a custom-made, FPGA-based router with a simple arithmetic routing engine which is expected to be much more efficient in terms of power and area utilization. The results of the experiment show that the power consumption when using arithmetic routing is less than 5 *W* which is only 12.5% of the power delivered by FPGA. The throughput and latency of the proposed work show a promising figure of 8 *Gbps* and 500 *ns* per hop respectively. The advantage of using FPGA in this testbed is that it is reconfigurable and reprogrammable. It also provides high bandwidth up to 40 *Gbps*. It has high performance of computation to obtain accurate results. However, the disadvantages of [87-89] are that they are FPGA-based. FPGA contains a lot of configurable logic blocks and complicated programming, making the computation time contributes to overall network latency. This complexity can degrade the network performance.

Posch et al. [90] proposed a testbed using Banana Pi R1 as a router. This work demonstrates NDN-based multimedia delivery using adaptive bit-rate streaming. Additionally, a graphical user interface is provided to create their own network topology, configure a streaming scenario and observe in near-real time. Two dedicated networks which are Management Network (MN) and Emulation Network (EN) are connected with Banana Pi in a star topology. The role of Management Network (MN) is for configuration and monitoring. While EN is a virtual network overlay that is created using networking tools such as *iptables* and traffic control. The purpose of having the separation between MN and EN is to prevent management and control interference with running network emulation. Their results presented in [91] show that the delay is big despite the packets are sent from one node to its adjacent node via CAT6 Ethernet cable. This is because the Banana Pi is connected with two networks, hence the processor cannot keep up with the networks, operating systems, and data transmission at the same time. Supported by the results obtained in [92], it shows that Banana Pi gives low throughput and high delay for point-to-multipoint connection in LAN and WAN network despite it has Gigabit Ethernet ports. The advantage of using Banana Pi is that it has more than one Ethernet port making it space friendly to be a

testbed. Furthermore, it already has built-in routing algorithm. However, according to Lech and Włodarski [92], Banana Pi has its own weakness. It produces a huge delay which is up to 43 *ms* for such a simple point-to-multipoint LAN and Wide Area Network (WAN) topology. Furthermore, according to the author, Banana Pi can cause a serious problem to the network when it receives a large amount of traffic. These problems can be overcome by using Raspberry Pi because the user can set an individual route for each traffic by using socket module in Python. Furthermore, built-in routing algorithm of Banana Pi is not reconfigurable.

In 2016, Jang et al. [93] proposed a testbed architecture using Raspberry Pi which allows dynamic configuration of mesh networks and coordination of each flow of traffic to support application-aware QoS. The router testbed is compatible with legacy network architecture in IEEE 802.11 ad-hoc network. The proposed testbed prioritizes the video streaming application instead of file transfer. This limits the file transfer traffic so that it does not harm the video streaming throughput. Hence, the video stream has less delay because intermediate router prioritizes the traffic in order to comply with QoS requirement predefined by the router testbed algorithm. The advantage of using Raspberry Pi as router testbed is that it is reconfigurable and scalable. It is also cost-efficient to be the testbed. However, Raspberry Pi has low performance due to its hardware limitations that might cause high delay during data transmission. In this work, the Raspberry Pi router testbed has no rerouting mechanism in case of congestion.

Piao et al. [94] proposed a wireless communication prototype by using Raspberry Pi to communicate with Android smartphones. The Raspberry Pi is installed with NDN routing mechanism called NDN Forwarding Daemon (NFD). The routing mechanism is based on shortest path. The results show the relationship between communication delay and number of pings. Based on the results, the communication delay reached its peak when the number of ping is 45. The advantage of using Raspberry Pi in this work is that it is reconfigurable and scalable. The kernel of Raspberry Pi is an open source kernel which can be used to write and execute custom algorithm. However, the routing mechanism in this work is based on shortest path. Hence, it does not reroute in case of congested path. As mentioned earlier, Raspberry Pi has low performance due to its

hardware limitations. But it is acceptable because the results produced are very close to the results produced by commercial routers and they are still within an acceptable range of communication standards.

Lastly, Gupta et al. [95] has proposed a small size, low cost and portable SDN switch testbed using Raspberry Pi. The Raspberry Pi only has 1 Ethernet port, but the authors extended it by using three low-cost USB-based LAN cards. In order to make the Raspberry Pi as SDN switch, it is installed with Ubuntu MATE 15.04 instead of Raspbian to support the latest version of OpenFlow switch. Four laptops are connected to the SDN switch as client, POX/RYU controller, and the other two laptops act as the servers. The testbed has no numerical results yet but it is claimed to support OpenFlow Specification 1.0 to 1.4. The main advantage of using Raspberry Pi in this work is that it is space-friendly for a lab-scale testbed.

As conclusion, embedded system-based routers are reconfigurable, reprogrammable, scalable, space friendly and cost-efficient. These advantages of embedded system-based routers make them an ideal choice as the lab-scale router testbed for research and academic purposes. In this section, the main embedded system hardware used in the literature are FPGA, Banana Pi and Raspberry Pi. However, embedded system like FPGA is complex to be reconfigured. It is also sensitive to electrostatic charges on human body and not cost efficient. Whereas, built-in routing of Banana Pi is not reconfigurable. These problems are solved by using Raspberry Pi since both the kernel and algorithms are completely configurable.

In [93], the proposed wireless Raspberry Pi routers are connected in mesh setup. Therefore, each router needs extra computational time to decide which route to use just for point-to-point communications. Hence, the redundant computational time contributes to overall degradation of network performance. Furthermore, proposed FiWi router testbed has better scalability since the testbed covers both fiber and wireless transmissions. In [94], the proposed router uses Raspberry Pi 2 which has lower hardware specifications than proposed FiWi router in this thesis. Furthermore, built-in routing mechanism in NDN that is implemented in Raspberry Pi 2 is not reconfigurable. Compared to proposed FiWi router in this thesis, the Raspberry Pi 3B+

used has better hardware specifications, hence, it is able to produce better performance. Furthermore, its routing mechanism can be reconfigured freely according to the user's needs. Lastly in [95], Raspberry Pi is used to run OpenFlow to become a router. However, the built-in routing program cannot be reconfigured freely by the user unlike proposed FiWi router in this thesis

Table 2.2 Router testbed summary

Title	Description	Advantages	Disadvantages
Software-based			
OpenFlow and GMPLS Unified Control Planes: Testbed Implementation And Comparative Study, 2015 [68]	-This paper proposed and experimentally evaluates two solutions using Open Flow (OF) and OpenFlow Extension. The results are evaluated and compared with Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) -DRAGON comprises of two major components; client system agent (CSA) and virtual label switch router (VLSR). -VLSR and CSA are used as the control plane for GMPLS.	- VLSR and OF uses shortest path routing causing the low delay for the packets. - VLSR can control different types of switches. - VLSR is reliable - OF controller can recalculate alternative light path in case of light path failures. - OF has maximum flexibility and manageability	- VLSR causes high delay for the packet because the algorithm is complicated. The flexibility and manageability are low because the signalization and reservation messages must be updated and exchanged among all intermediate VLSRs. - OF and VLSR are not intelligent since it will choose the shortest path even though the path is congested. - No hardware involved.

Title	Description	Advantages	Disadvantages
Software-based			
Towards Low Latency Software Routers, 2015 [69]	-Proposed a QoS aware software router model by using NS3 and evaluate the performance optimizations.	- The QoS aware software router model has dedicated receiver ring for prioritized packet processing which will be served according to a specific scheduling strategy. Hence, the packet will be process faster for low latency constraints real-time traffics such as VoIP, video conferencing, and online gaming.	- There is no hardware involved, hence, no non-linear effects are considered in the simulations. - Disadvantages: 1) user does not build up all the functionalities from scratch, instead, they are utilizing the existing models. 2) scalability limits
Netml-ns3-click: modeling of routers in Netml/ns 3 by means of the click modular router, 2015 [71]	-Proposed an analysis for Netml systems by using the simulations of NS3 and Click routers. The Netml system enables an XML description of a network to be converted into an NS3 program, then, run the simulation to collect and plot the results on Netml public server. A basic implementation of IPtables is also implemented.	- The advantage of using Click router in this work is that it can build its own firewall. - Click router is completely flexible, configurable and customizable according to user's needs.	- No hardware involved - Some of Click's MAC elements is not supported by NS3 yet.

Title	Description	Advantages	Disadvantages
Software-based			
On methodologies to estimate optical-layer power consumption and cost for long-haul WDM networks with optical reach constrain, 2015 [73]	-Studied how to achieve the estimation of optical-layer power consumption and cost for a long distance optical networks using Wavelength Division Multiplexing (WDM). The test is implemented on two different networks which are IP-over-WDM and IP-over-MPLS-over-WDM networks. -The simulation was conducted by using C++ programming language.	- Cost efficient - reconfigurable	- No hardware involved - A lot of assumptions must be made in order to simulate as close to real network as possible. This will make the produced results to be inaccurate and cannot be implemented in real network.

Title	Description	Advantages	Disadvantages
Software-based			
Power consumption model of NDN-based multicore software router based on detailed protocol analysis, 2016 [74]	-Developed a power consumption model of multicore software Named Data Network (NDN) router.	- It reduces the amount of traffic forwarded towards upstream routers and thus, reduces the power consumed by their forwarding devices.	- Has limited scalability - No hardware involved
MINOS: regulating router dataplane actions in dynamic runtime environments, 2017 [75]	- A framework called Minos has been proposed to regulate router actions on data planes. - Minos is implemented and evaluated separately on Click and Data Plane Development Kit (DPDK).	- DPDK comprises of a set of libraries that support efficient implementations of network functions [76]. - DPDK able to boost packet processing performance and throughput, hence, allowing more time for data plane applications [77].	- DPDK does not have cache coherence and locality making its access time to be slow. - No hardware involved in this work

Title	Description	Advantages	Disadvantages
Software-based			
SDN-Based Resource Allocation in MPLS Networks: A Hybrid Approach, 2018 [78]	- Proposed on a new traffic engineering architecture for SDN-MPLS network, where they improve flow-level management flexibility. It is done by applying OpenFlow-enabled switches at the edge of the network while the MPLS routers are the core router of the network. - The simulation was done by using MATLAB2016b.	- MATLAB2016b has improved functionalities in the toolbox that enable users to produce better equations and algorithms. Hence, more accurate results can be produced.	- No hardware involved - It can be slow to process and compute for a such complex hybrid network like in proposed work.

Title	Description	Advantages	Disadvantages
Software-based			
Mobile-oriented Future Internet: Implementation And Experimentations Over EU–Korea Testbed, 2019 [79]	-A new Internet architecture for the future mobile network was proposed, named Mobile-Oriented Future Internet (MOFI). The MOFI architecture comprises of two main components: (1) separation of data and control planes (2) distributed identifier–locator mapping control for alleviating traffic overhead at a central agent. MOFI architecture is implemented using OpenFlow and Click Modular Router over a Linux platform, operated.	- Click modular router is a flexible software router that enables the user to configure and customize freely according to the user's needs. - Click can achieve a maximum loss-free forwarding rate of 333,000 of 64 *bytes* packets per second when run on Linux computers, proving that Click's modular and flexible architecture is compatible with good performance [80]. - OF supports IPv6, hence, it supports more users than IPv4.	- Click modular router is not considering non-linear effects such as noises and fluctuations because there is no hardware involved. - Small elements cannot solve problems like when control and data flows do not match with the flow of packets

Title	Description	Advantages	Disadvantages
Commercial-based			
Demonstration and analysis of IP/MPLS communications for delivering power system protection solutions using IEEE C37. 94, IEC 61850 Sampled Values, and IEC 61850 GOOSE protocols, 2014 [82]	-Used four MPLS routers in their experiment. The study is to demonstrate and analyse the use of commercial IP/MPLS protocol to carry protection relay hardware to support power system protection functions.	- The advantage of IP/MPLS routers in this work is that the routers has the capability to support ring topology. Hence, the data has better protection because it can be rerouted to another path in case of broken links.	- The traffic flows in IP/MPLS router are still prone to congestions because the data for all the services are transmitted in one LSP.

Title	Description	Advantages	Disadvantages
Commercial-based			
Hybrid SDN Architecture To Integrate With Legacy Control And Management Plane: An Experiences-based Study, 2015 [83]	-Proposed a testbed that an OpenFlow module is embedded in the control plane of a commercial router -The commercial router used is DCRM 5980.	- The router can be implemented with other firmware such as OpenFlow - The router supports IPv6, hence supports more users connected to it. - Flexible and scalable - supports VLAN and MPLS	- After installing OpenFlow, the routing is based on shortest path. Which means, the data will use the shortest path even the path is congested.
Experimental demonstration of segment routing, 2016 [84]	-Implemented Segment Routing (SR) in two different networks, which are SDN-based and Path Computation Element (PCE)-based. In PCE-based network, SR is implemented where the nodes consist of commercial IP/MPLS routers.	- The routing mechanism of a commercial IP/MPLS routers is based on the short label header on the data rather than long network addresses. Hence, this router avoids complex lookups in a routing table to speed up the traffic flows.	- The traffic flows in IP/MPLS router are still prone to congestions because the data for all the services are transmitted in one LSP.

Title	Description	Advantages	Disadvantages
Commercial-based			
Experimental Analysis in SDN Open Source Environment, 2017 [85]	-Used two commercial routers to evaluate the performance of open source OpenFlow switches, which are OpenvSwitch (OVS) and ofsoftswitch13 (CPqD -The two commercial routers are TP-LINK WR1043ND routers. One of the routers sets as an OpenFlow switch. -Ryu controller is used to manage the routing path of OpenFlow switches while iperf tool is used to generate UDP and TCP traffics.	- The firmware of the router can be reinstall or upgraded to another firmware such as OpenFlow. - Provides accurate results in real-time. - This router has gigabit bandwidth; hence, more data can be transmitted at once. Thus, more accurate results can be obtained.	- Has high packet loss which about 57% to 58% - After reinstall to another firmware, the bandwidth of the router reduced significantly.

Title	Description	Advantages	Disadvantages
Commercial-based			
WiFi Fingerprinting Indoor Localization Using Local Feature-Based Deep LSTM, 2019 [86]	-Proposed a local feature-based deep long short-term memory (LF-DLSTM) approach for WiFi fingerprinting indoor localization by using TP-LINK WDR4300 router. The experiments were done in two different environments, which are in a research lab and in an office. The proposed approach is compared with state-of-the-art methods for indoor localization.	- The router has big coverage because it has three dual band external antennas. - Has high bandwidth which supports simultaneous data transmission. Hence, more data can be transmitted at once to obtain accurate results.	- The router is not reconfigurable

Title	Description	Advantages	Disadvantages
Embedded system-based			
Greening Router Line-Cards via Dynamic Management of Packet Memory, 2016 [87]	-Used netFPGA as the router to study the role of packet buffer memory on the power consumption of backbone routers. -The evaluation also being implemented on a programmable router testbed which is the netFPGA. NetFPGA is connected to traffic generators and delay emulators.	- The advantage of using netFPGA is it can be reprogrammable by using VHDL programming language.	- The netFPGA is not cost-efficient and sensitive to electrostatic on human body. - netFPGA is complex as FPGA
ParaDiMe: A Distributed Memory FPGA Router Based On Speculative Parallelism and Path Encoding, 2017 [88]	- Proposed a distributed memory parallel FPGA router called Parallel Router Distributed Memory (ParaDiMe) to reduce the number of active processes.	- FPGA is reconfigurable and reprogrammable - High performance to provide accurate results	- Sensitive to electrostatic charges on human body - FPGA is very complex - Not cost-efficient

Title	Description	Advantages	Disadvantages
Embedded system-based			
A CAM-free Exascalable HPC Router For Low-Energy Communications, 2018 [89]	- Proposed a custom-made, FPGA-based router with a simple arithmetic routing engine which is expected to be much more efficient in terms of power and area utilization.	- FPGA is reconfigurable and reprogrammable - Provide high bandwidth up to 40 *Gbps* - High performance of computation to obtain accurate results	- Sensitive to electrostatic charges on human body. - FPGA is very complex - Not cost-efficient
Emulating NDN-based multimedia delivery, 2016 [90]	-Proposed a testbed using Banana Pi R1 as a router. This work demonstrates NDN-based multimedia delivery using adaptive bit-rate streaming. Two dedicated networks which are Management Network (MN) and Emulation Network (EN) are connected with Banana Pi R1 in a star topology.	- It has more than one ethernet port making it space friendly to be a testbed - Built-in routing algorithm	- It produces a huge delay which is up to 43 *ms* for such a simple point-to-multipoint LAN and Wide Area Network (WAN) topology [92] - Banana Pi can cause a serious problem to the network when it receives a large amount of traffic [92] - Built-in algorithm not reconfigurable

Title	Description	Advantages	Disadvantages	Differences
Embedded system-based				
Implementing a Dynamically Reconfigurable Wireless Mesh Network Testbed for Multi-Faceted QoS Support, 2016 [93]	-Proposed a testbed architecture by using Raspberry Pi which allow dynamic configuration of mesh networks and coordination of each flow of traffic to support application-aware QoS.	- Raspberry Pi is reconfigurable and scalable - Cost-efficient	- Low performance due to hardware limitations causing high delay when data is transmitted. - Has no rerouting mechanism in case of congestion	- Proposed wireless router has extra computational time to decide which route to use in mesh network. - Only use wireless transmission. - Covers fiber and wireless transmissions.
The Real Implementation of NDN Forwarding Strategy On Android Smartphone, 2016 [94]	-Proposed a wireless communication prototype by using Raspberry Pi to communicate with Android smartphones. The Raspberry Pi is installed with NDN routing mechanism called NFD.	- Raspberry Pi is reconfigurable and scalable - The kernel is open source which can be used to write and execute custom algorithms.	- Raspberry Pi has low performance due to its hardware limitations - The routing algorithm is based on shortest path, hence, it does not reroute in case of congested path.	- Raspberry Pi 2 has lower specifications. - NDN is not reconfigurable. - Proposed FiWi router has higher specifications. - Proposed FiWi router is freely reconfigurable.

Title	Description	Advantages	Disadvantages	Differences
Embedded system-based				
Developing Small Size Low-Cost Software-Defined Networking Switch Using Raspberry Pi, 2018 [95]	-Has proposed a small size, low cost and portable SDN switch testbed using Raspberry Pi. Then, four laptops are connected to the SDN switch as client, POX/RYU controller, and the other two laptops act as the servers.	- Space-friendly - Scalable	- Limited hardware specifications	- OpenFlow-based Raspberry Pi router is not freely reconfigurable. - Proposed FiWi router is freely reconfigurable.

2.5 Summary

In conclusion, FiWi is a technology that combines optical and wireless communication. It is widely used in current communication system due to its robustness and mobility. Despite it still has plenty rooms for improvements. Hence, many testbeds have been proposed by researchers to study numerous issues. Testbed is an ideal approach to test a setup of equipment in order to test or enhance current technology or as a proof-of-concept. There are also various architecture of testbeds which are fiber, wireless and FiWi. From the literature review, it can be concluded that FiWi testbeds can be implemented from a lab-scale testbed to an industrial-scale testbed which is summarized in Table 2.3.

To the best of our knowledge, routers are installed at the backhaul of FiWi network. Hence, we further scope down our literature review on router testbeds which include software-based routers, commercial routers and embedded system-based routers which have been summarized in Figure 2.4. From the literature review, it can be concluded that most authors were using software-based routers as their testbeds because these routers are reconfigurable, customizable, flexible, cost-efficient and able to produce high performance results. However, there are no hardware involved in their work, hence, the results produced were not including non-linear affects such as noises and fluctuations. Meanwhile, commercial routers are able to produce accurate and high-performance results. However, most of these routers are not reconfigurable, not cost-efficient and not space friendly. Therefore, embedded-system-based routers were used as the testbed due to their open source kernel, reconfigurability, scalability, space-friendly and cost-efficient. Due to their limited hardware specifications, embedded system-based routers are not able to produce as high performance as software-based routers and commercial routers. To the best of our knowledge, embedded system-based routers are the best choice in order to build a testbed as the proof-of-concept of a fundamental of a technology or to enhance the current technology. Compared to other embedded system hardware, Raspberry Pi 3B+ used in this project has socket module that enables data communication between hardwares. It is simpler, more cost-efficient, space friendly and durable compared to others such as FPGA. Thus, development of reprogrammable and fast reconfigurable lab-scale FiWi testbed that supports the integration of fiber optic and wireless for research and

educational purposes is feasible. Furthermore, due to its simplicity, Raspberry Pi used in this project able to make the testbed scalable to more than one traffics and flexible to different topologies such as Fi-WiFi and Wi-FiWi. Therefore, we have come to our conclusion that Raspberry Pi as the main embedded system hardware is the best choice to be our testbed in FiWi network.

Table 2.3 Overall summary of testbed architecture in FiWi

Testbed Architecture in FiWi		
Hardware used	**Advantages**	**Disadvantages**
Industrial grade hardware [2, 62-66]	- High performance and accurate results - Support long distance transmission	- Not reconfigurable and not fully reprogrammable - Not cost-efficient
USRP [21, 67]	- Reconfigurable and reprogrammable - Flexible and scalable - Space friendly	- Not cost-efficient

Table 2.4 Overall summary of router testbed

Router type	Advantages	Disadvantages
Software-based [68, 69, 71, 73-75, 78, 79]	- Freely reprogrammable - Flexible - Customizable - High performance - Cost-efficient	- The parameters are based on assumptions - Not involved non-linear effects
Commercial-based [82-86]	- High performance and accurate results - Support many setups and topologies - Have better securities	- Not reprogrammable and reconfigurable - Not cost-efficient
Embedded system-based	1) FPGA/netFPGA [87-89]: - Reprogrammable and reconfigurable - High performance 2) Banana Pi [90, 91]: - Has more than 1 Ethernet port - Has built-in algorithm 3) Raspberry Pi [93-95]: - Reprogrammable and reconfigurable - Simple, scalable and flexible - Space-friendly and portable - Cost-efficient	1) FPGA/netFPGA: - Complex and complicated in terms of programming and setup - Not fast integration - Not cost-efficient 2) Banana Pi: - Can cause serious problem to the network [92] - Built-in algorithm not reconfigurable 3) Raspberry Pi: - Limited performance (RAM, Ethernet performance)

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter proposes an embedded system-based FiWi testbed using Raspberry Pi 3B+ that is able to perform various experiments and testings in point-to-point communication. The proposed FiWi testbed consists of four routers that are connected in tree topology. This topology is chosen because it is a typical topology for FiWi network [1, 7]. However, the topology of the testbed can be varied as desired. Each router consists integration of four Raspberry Pi 3B+ and switches. Raspberry Pi 3B+ is chosen because it is user friendly especially for beginners. It is also cost efficient making it more affordable. This makes it an ideal choice as an educational module with low power consumption and fast implementation.

Overview of the FiWi testbed such as testbed topology, router architecture, network environment and default system parameters is explained in Section 3.2. The hardware used for this testbed are listed and discussed in Section 3.3. Section 3.4 explains on the testbed programming environment in terms of flowchart of the system, testbed scalability and fast integration. The design parameters and performance parameters are discussed in Section 3.5. Finally, Section 3.6 summarizes this chapter.

In order to get better understanding on the flow of this chapter, the research flow in Figure 3.1 is referred. This chapter begins with a description of testbed topology and router architecture. Then, it continues with the description on the hardware used for the testbed such as Raspberry Pi 3B+, Ethernet switches, Fiber Media Converter (FMC) and optical fiber. Next, this chapter discusses about the network environment of the testbed, ie., the traffic of data in terms of data size. Finally, the parameters for the input and output of the testbed is deliberated.

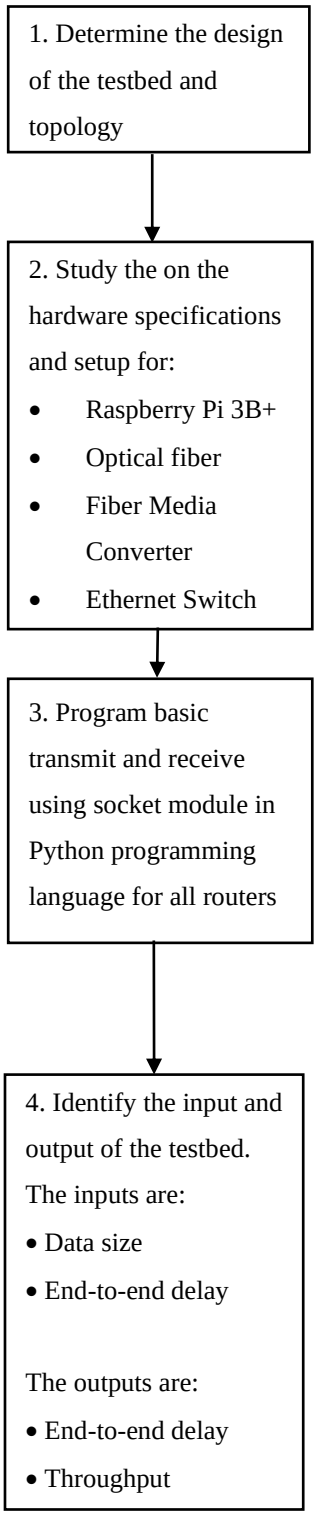

Figure 3.1 Research flow

Figure 3.2 shows a flowchart that reflects the overview of technical development of the proposed testbed. Firstly, after the proposed routers are connected with one another in FiWi architecture, a packet is sent from one client to another client to observe its reliability. If the client does not receive the packet, then, an additional processing delay is increased in the proposed router. This process is repeated until the clients are able to receive the packet without fail. The value of additional processing delay is further discussed in Section 3.2.3. Afterwards, the performance of the proposed router testbed is tested in point-to-point wireless network, fiber network and FiWi network in terms of throughput, end-to-end delay and jitter in order to validate its correctness and comply with the current communication standards. After it is validated, a stress test is done by sending two traffics at a time to test the scalability of the proposed router testbed. Then, the topology is change to Fi-WiFi and Wi-FiWi in order to test its flexibility and stability.

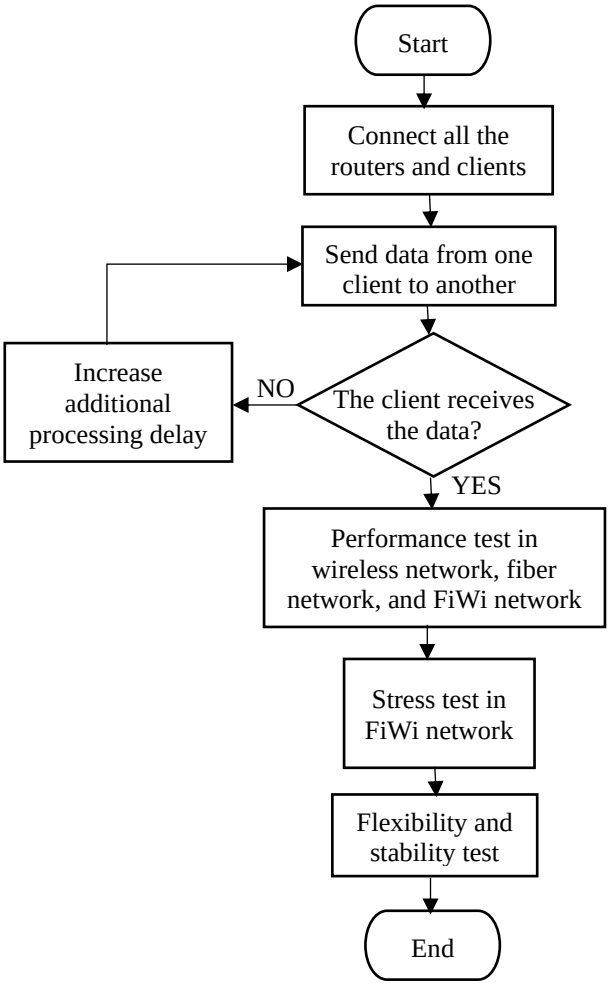

Figure 3.2 Research flowchart

3.2 Proposed Fiber-Wireless Testbed

3.2.1 Router Architecture

This section explains on the proposed router architecture of the testbed. The proposed router architecture consists of four Raspberry Pi 3B+; one Header Pi and three Forwarding Pi which are then connected with two Ethernet switches as shown in Figure 3.3. The Header Pi is to identify the final destination of the data desired by the user. Whereas, the Forwarding Pis are to forward the data to desired destination. Ethernet Switch 1 is to represent the internal circuit of router, while Ethernet Switch 2 is to represent the external connection between router and other routers. Each of the Raspberry Pi in the router has portable Liquid Crystal Display (LCD) screen to monitor the destination of data and to make sure that the routing mechanism is correct. These Raspberry Pis are connected to each other via CAT5e Ethernet cable through Ethernet switches. CAT5e Ethernet cable is used because it supports Gigabit Ethernet, therefore, it is compatible with Raspberry Pi 3B+ Ethernet port which also has Gigabit bandwidth.

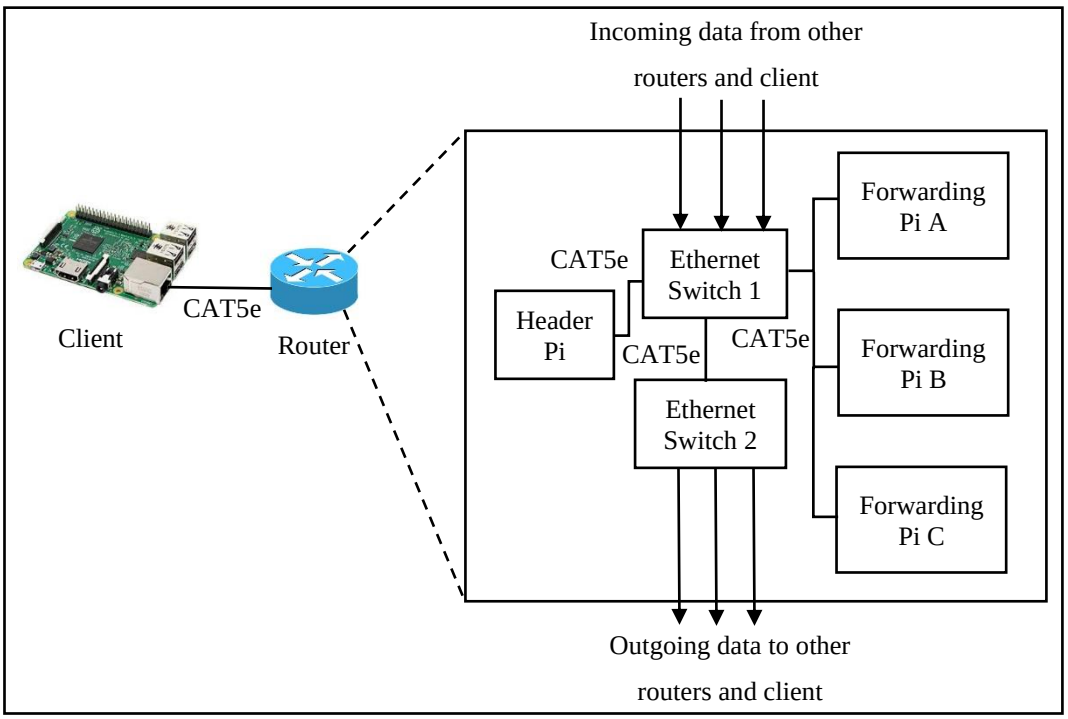

Figure 3.3 Basic router architecture

In order to make the router supports wireless transmission, an extra component is added to the architecture which is the AP as shown in Figure 3.4. The AP acts as the wireless transmitter component and antennae for the router.

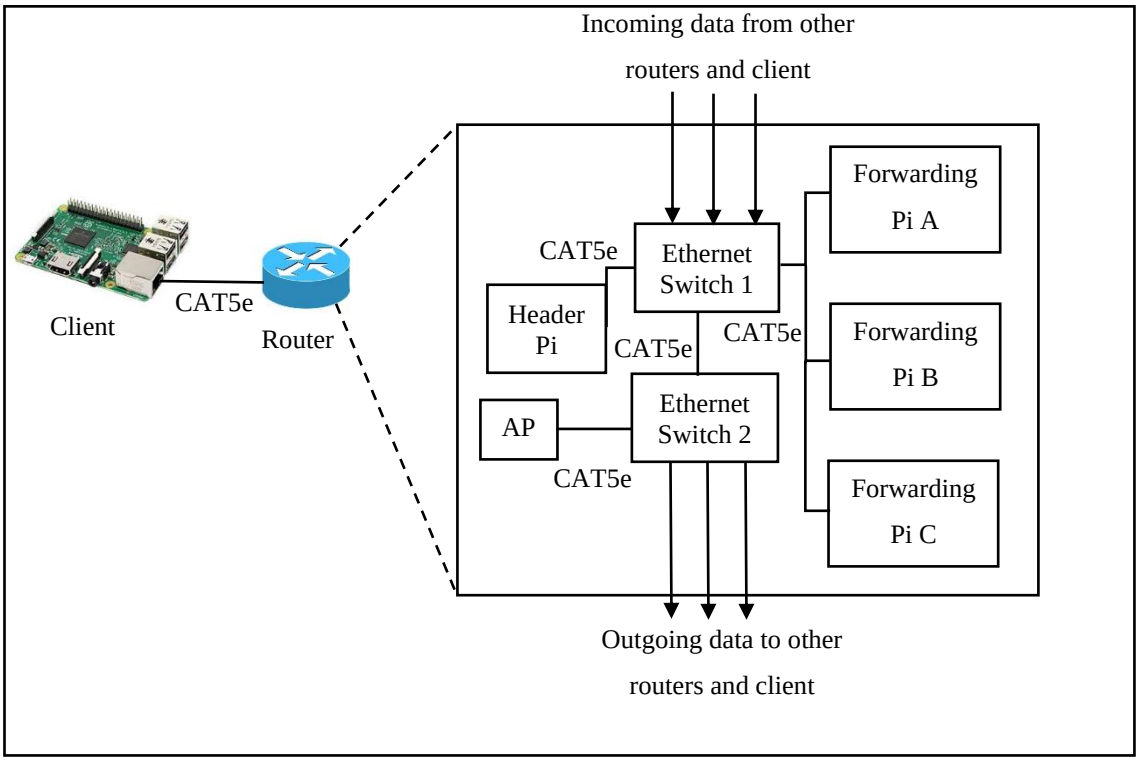

Figure 3.4 Basic fiber-wireless router architecture

3.2.2 Traffic Modelling

This section discusses on the internal process of a router of this testbed. The flow of data represented by the dotted arrow in Figure 3.5. When a client sends a packet of data to the destination, a pre-label is added to the payload of the packet to indicate where the data should end. The client sends the data to Header Pi. Then, the Header Pi checks the label to know the beginning and end of data. After the Header Pi has identified the destination of the data, it replaces the old label with a new label onto the payload and then broadcast the data to each Forwarding Pis. Each of the Forwarding Pi has its own unique label because each identity label corresponds to one destination only. When the Forwarding Pi receives data from Header Pi, it will check the label on the payload of the packet with its own identity label. If both labels are the same, the Forwarding Pi will continue to forward the data to desired destination set by the client.

Otherwise, if both labels are not the same, the Forwarding Pi turns the data to zero and drops the packet. Figure 3.5 illustrates how the data is processed in the router.

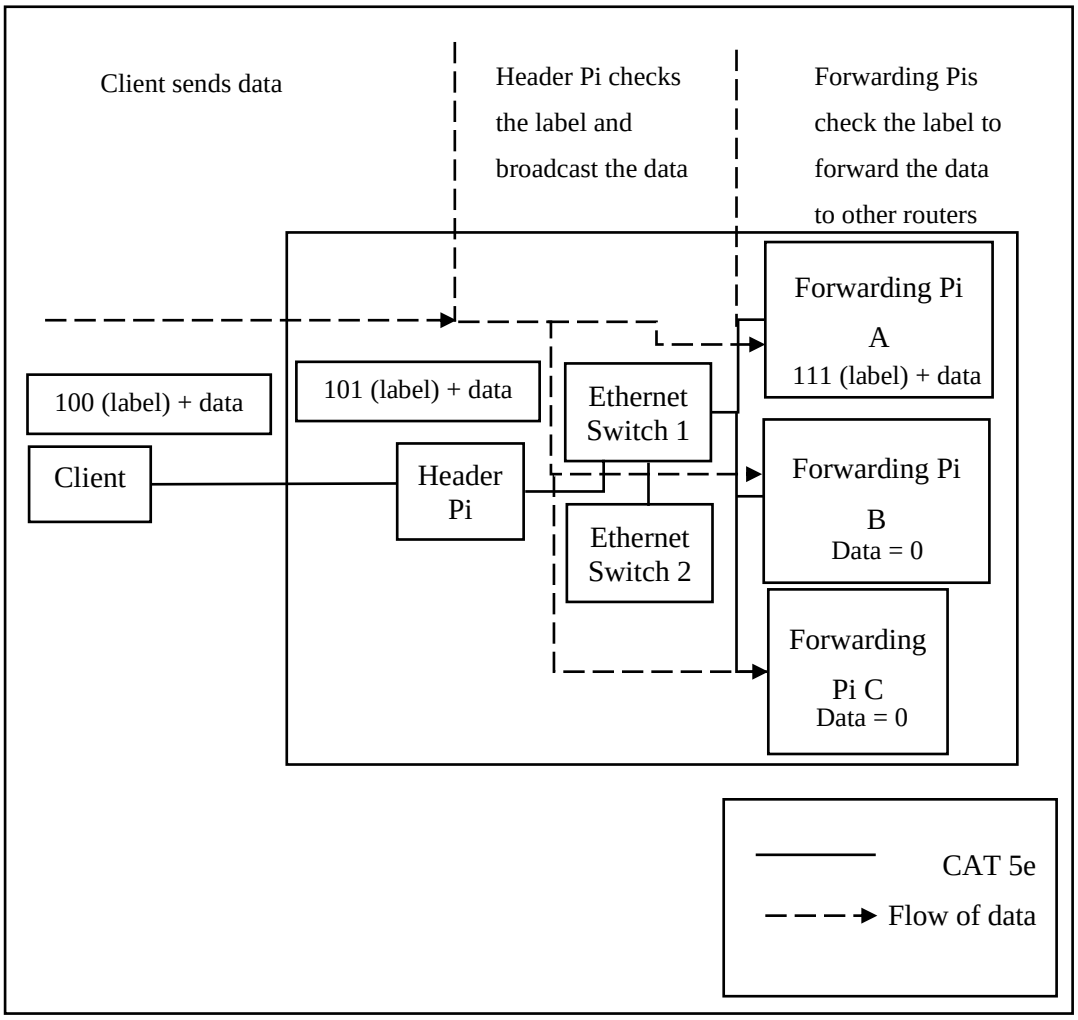

Figure 3.5 Data flow in a router

For analytics purposes, the maximum data size generated is 1448 *bytes* because it is the default maximum size of a packet set by Python in Raspberry Pi. The 1448 *bytes* of data includes 3 *bytes* reserved for the labels. The minimum data size transmitted is 100 *bytes* because according to Brown [96], it is the most reliable data size to achieve 0% packet loss in a transmission. Then, the data transmitted will be increased every 100 *bytes* each time until it reaches 1445 *bytes* which is the maximum data size for a packet in socket module. Furthermore, the increment of 100 *bytes* will make the trend of the graph clearer.

3.2.3 Default System Parameter

In order to make sure that the data transmission is reliable with no packet loss, each of the Raspberry Pis in the router needs an extra processing time. This extra processing time can be added by importing another Python module called "*time*". This module is already inside the Python programming folder. Therefore, function call is used to call for this module. By introducing extra processing time, the router has just enough time to process the packets including label injecting, label checking and packet forwarding. In this case, 80 *ms* is added to each Raspberry Pis of the router. This value is obtained by several iterations in an experiment. 1445 *bytes* of data is sent from one client to another client through their respective routers. An observation is done to check whether the client receives the data or not. Initially, there is no additional processing delay in the router. If the client does not receive the data, the delay of each Raspberry Pi in the router is increased by 10 *ms* until the client receives the data. 10 *ms* is chosen because through the experiment, it is the minimum amount of processing delay for 100 *bytes* of data to arrive at the client. Lesser than 10 *ms*, none of the data arrived at the desired client. The default system parameter is summarized in Table 3.1.

Table 3.1 Summary of default system parameter

Default system parameter	Value
Extra processing delay	80 *ms*

3.3 Raspberry Pi-based Fiber-Wireless Testbed

3.3.1 Hardware Setup

The testbed is setup in tree topology which consists of four routers as shown in Figure 3.6 because other than limited amount of Raspberry Pis, it is the minimum number of routers to test the performance for all type of connections; fiber, wireless and Fi-Wi. The connection between routers is by using Single Core/Angled Physical Contact (SC/APC) fiber optic patch cord. Raspberry Pi only has one Ethernet port. Hence, a FMC is needed as an adapter between Raspberry Pi and fiber optic. Figure 3.7 shows the hardware setup for FiWi network environment that reflects block diagram in Figure 3.6.

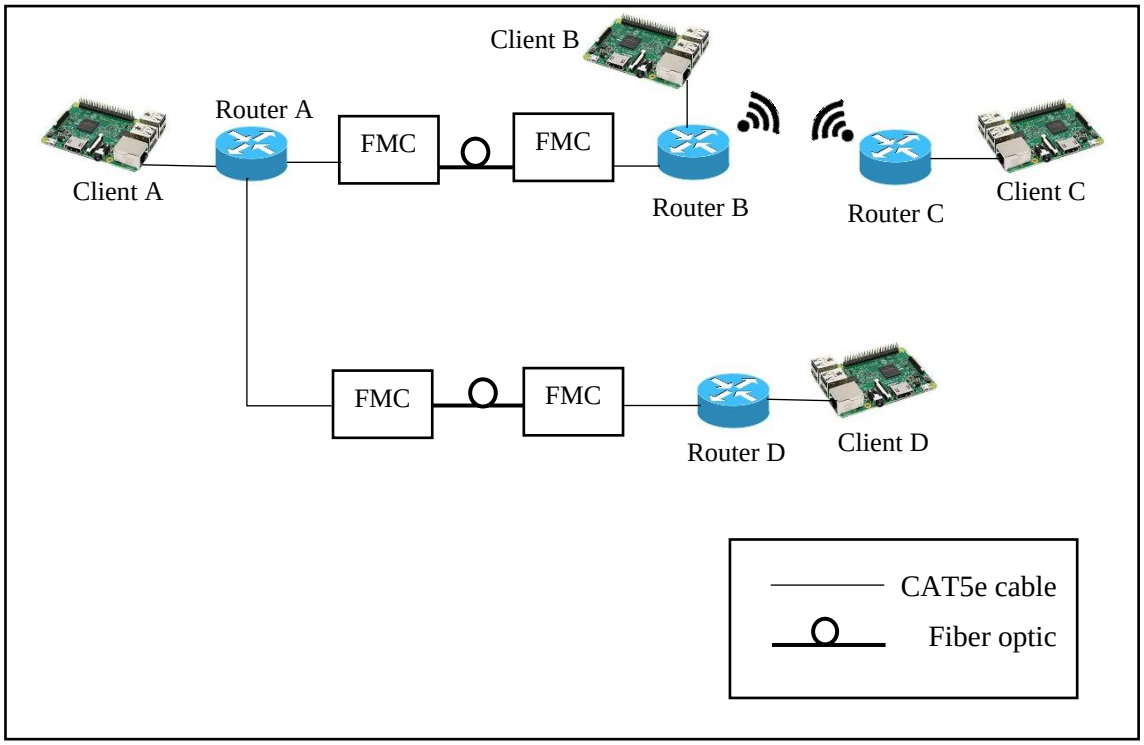

Figure 3.6 FiWi testbed block diagram

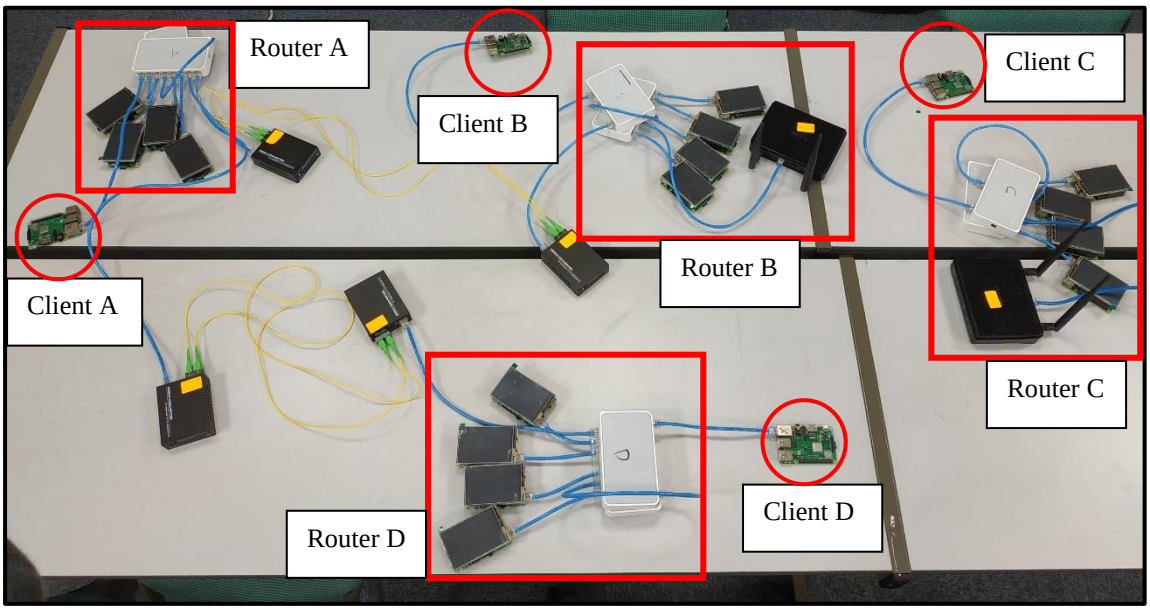

Figure 3.7 FiWi testbed architecture

Then, the topology is expanded to fiber-wireless-fiber (Fi-WiFi) and wireless-fiber-wireless (Wi-FiWi) as shown in Figure 3.8 and Figure 3.9 respectively because compared to other topologies such as ring and mesh, we can test the performance and the reliability of the proposed router after gone through a number of medium changes from wireless to fiber and vice versa. Moreover, the purpose of these setups is to test to scalability of the testbed.

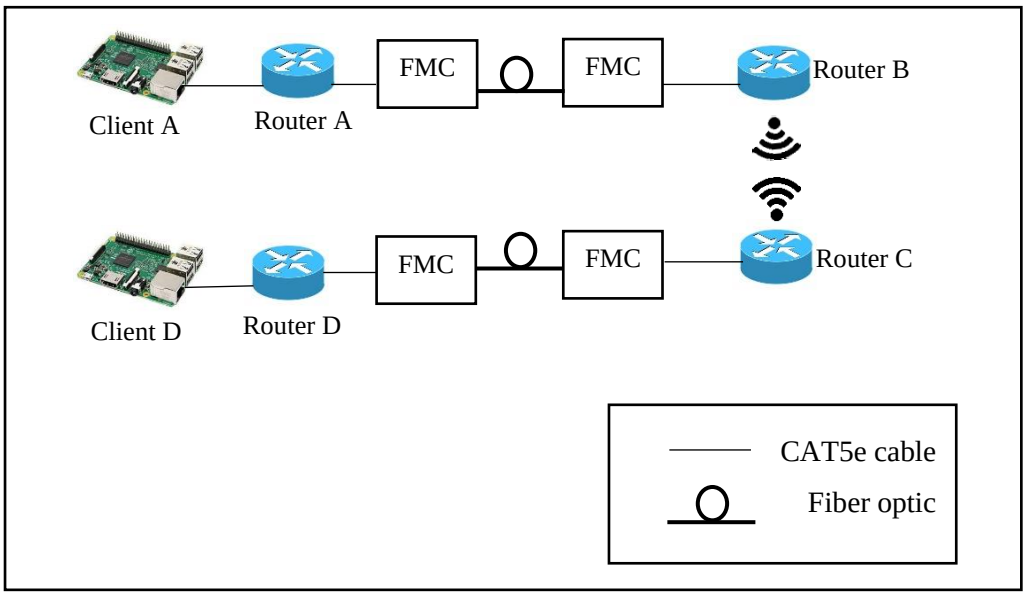

Figure 3.8 Fi-WiFi setup

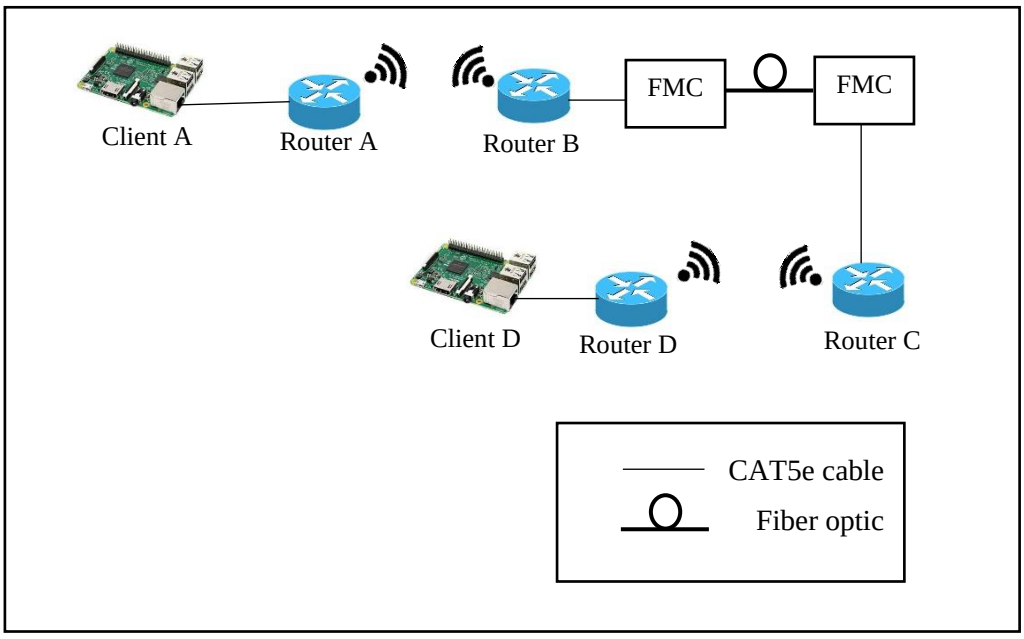

Figure 3.9 Wi-FiWi setup

3.3.2 Raspberry Pi Router Connections

Figure 3.10 shows the connection between four Raspberry Pi 3B+ and two switches to form a router. As mentioned in Section 3.2.1, Header Pi is to check and identify the final destination based on the pre-label injected by the client. There are three Forwarding Pis in this router because, in a typical router, each of the port will have a unique IP address. Therefore, to emulate the real router, three Forwarding Pis are needed because each of them have a unique static IP address. Ethernet Switch 1 is to represent the internal connection of the routers, which means, the connection between Header Pi and Forwarding Pis. While Ethernet Switch 2 is to represent the external connection between router and other routers. Figure 3.11 shows the connection for fiber-wireless router.

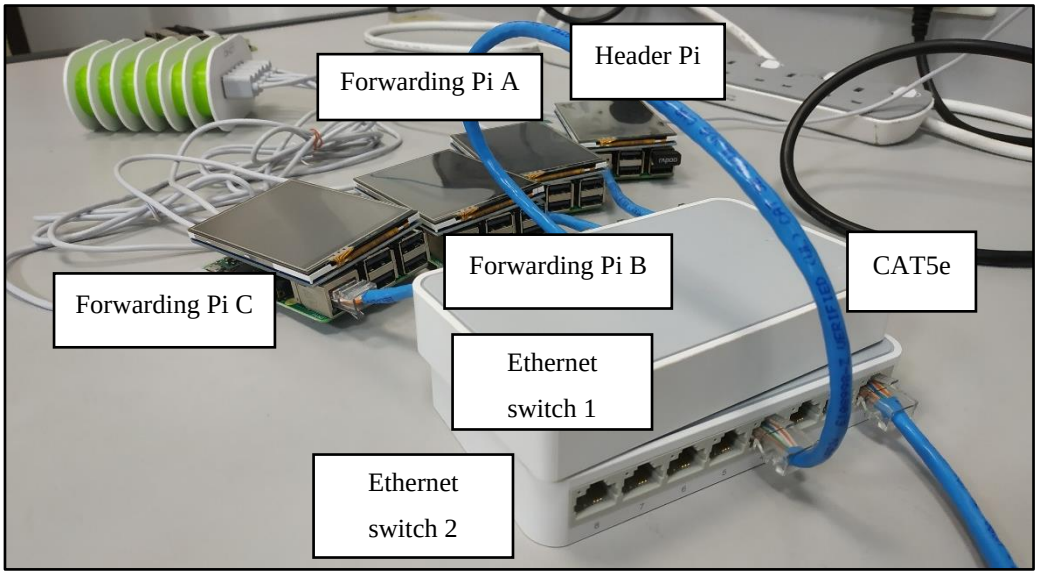

Figure 3.10 Raspberry Pi router connection

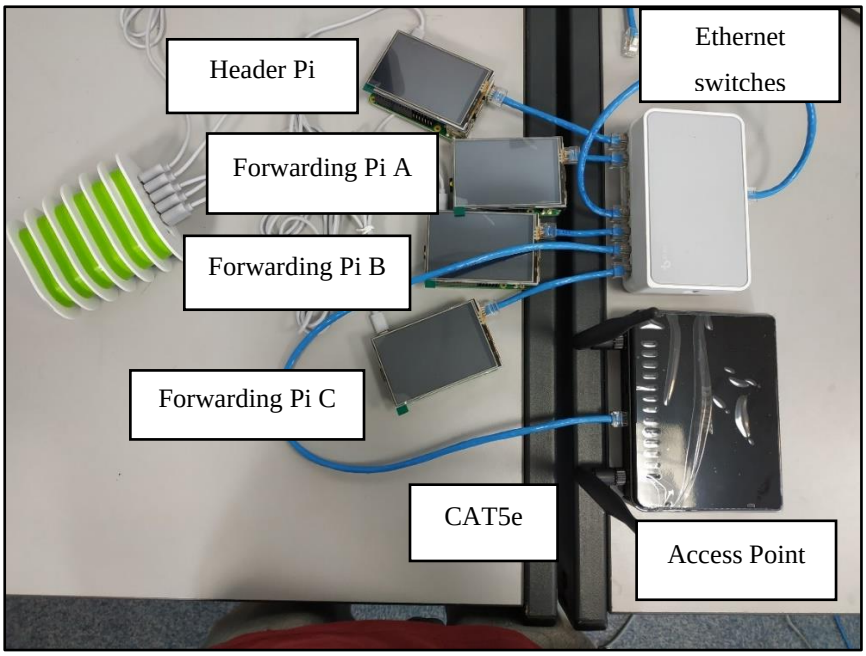

Figure 3.11 Raspberry Pi fiber-wireless router connection

3.3.3 Optical Fiber

Optical fiber used in this project is a single-mode fiber patch cord. The patch cord length is fixed at 1 m. The insertion loss for this fiber optic ranges from 0.11 dB to 0.18 dB. The return loss ranges from 61.4 dB to 62.1 dB.

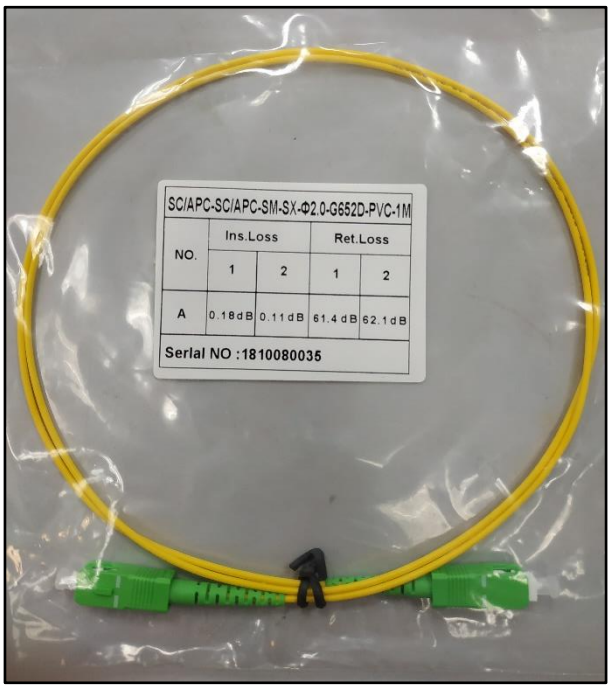

Figure 3.12 SC/APC to SC/APC fiber optic

3.3.4 Fiber Media Converter (FMC)

FMC is used as an adapter for Raspberry Pi to connect to the fiber optic. The supported data rate is up to 100 *Mbps*. The FMC also supports 20 *km* transmission while operating at 1310 *nm* wavelength for transmitting and receiving because it has individual port for transmit and receive as shown in Figure 3.14.

Figure 3.13 Fiber Media Converter

3.3.5 Ethernet Switch

Ethernet switch is used to provide more ports because Raspberry Pi has only one ethernet port. The ethernet switch has eight ports and it supports up to 100 *Mbps* of bandwidth.

Figure 3.14 Ethernet switch

3.4 Raspberry Pi Fiber-Wireless Testbed Programming Environment

3.4.1 Transmit and Receive Flowchart

This section shows the flowchart of routing mechanism of this testbed for the client, Header Pi and Forwarding Pi that have been explained in Section 3.2.2. Figure 3.15, Figure 3.16 and Figure 3.17 show the flowchart for client, Header Pi and Forwarding Pi respectively. The red boxes in Figure 3.15 and Figure 3.16 are the examples of processes that take place in Figure 3.4. The flowchart shown in Figure 3.17 is only for Forwarding Pi A because all Forwarding Pis has the exact same process but with different labels.

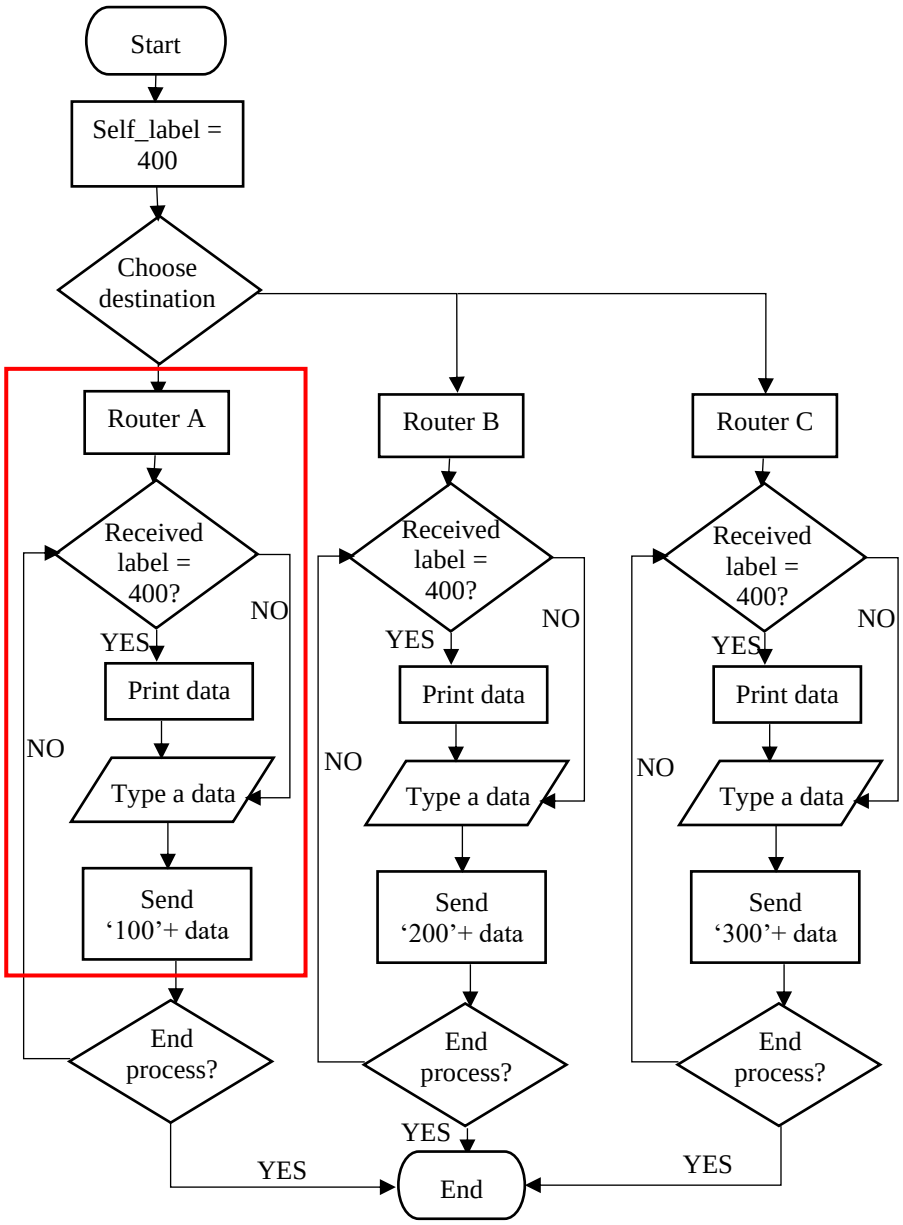

Figure 3.15 Client flowchart

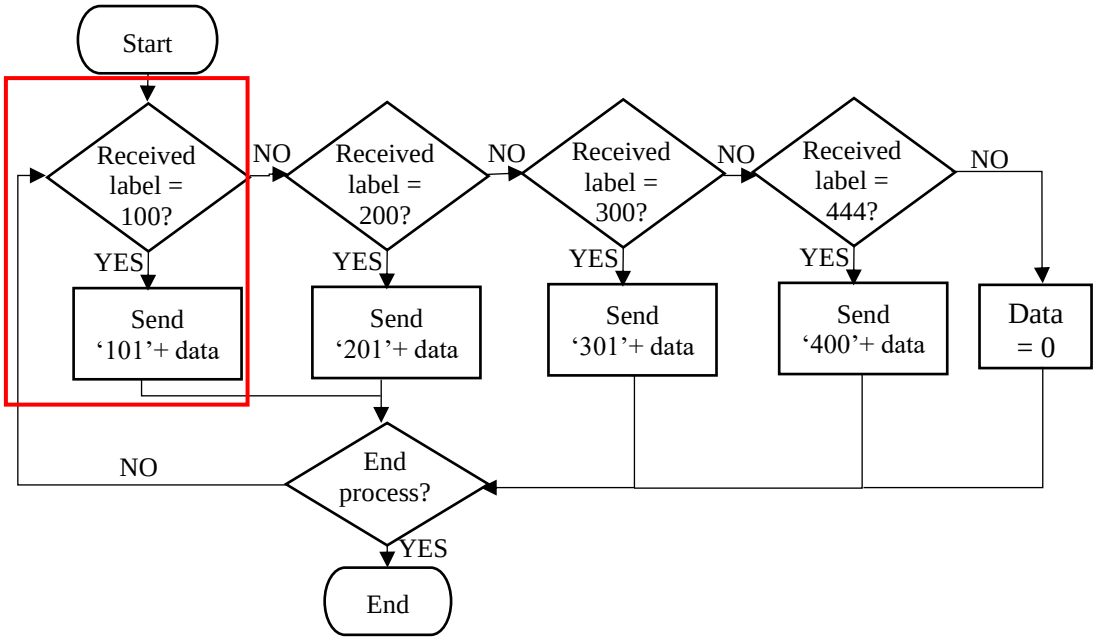

Figure 3.16 Header Pi flowchart

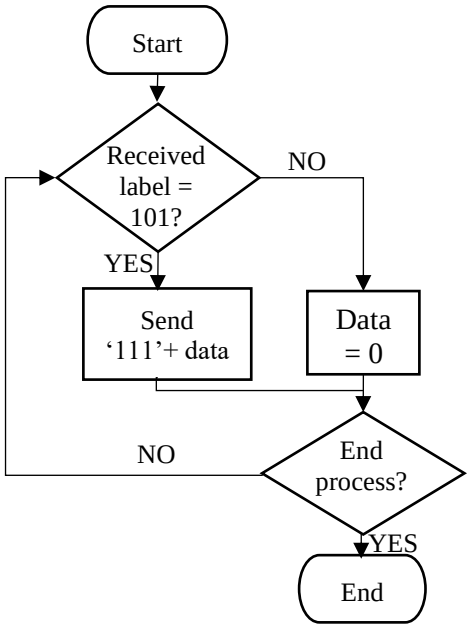

Figure 3.17 Forwarding Pi A flowchart

This testbed is connection-oriented communication which has three-way handshake by using socket module in Python. In order to declare the communication is connection-oriented, the program must include “s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)”. The variable “s” is to simplify the whole line and will be easier to use later on in the program. Then, the first “socket” in “socket.socket” is the function call from socket module in Python module, while the second one is the function name. Next, “socket.AF_INET” means the communication is based on Internet Protocol version 4 (IPv4). Finally, “socket.SOCK_STREAM” means the protocol used is Transmission Control Protocol (TCP), which means it is connection-oriented communication.

After this declaration, the program checks whether or not the router receives the data or not by using “data = socket.recv(4096)”. In the program, “data” means the variable where the received data is stored. “socket.recv” is module used to receive the data. “4096” refers to 4096 *bytes* which is the buffer size for the incoming data. Once the data is received, the program checks the first 3 *bytes* of the data to check the label in “if-else” statements. In the “if-else” statement, if the label is the same with the label of Header Pi or Forwarding Pi, the data is forwarded by using “socket.send(data)”. Otherwise, the data turns to 0 by using “data = ‘0’”. The rest of the program is a repetitive of “if-else” statement but with different label values.

3.4.2 Testbed scalability

One of the objectives of this project is to develop a scalable FiWi testbed. Scalability means the ability for the testbed to adapt to a new architecture or arrangement easily and its ability to expand the number of new components. In order to add or remove a component in the FiWi testbed, minor changes need to be done to the testbed arrangement. In this case, the testbed setup is changed to Fi-WiFi and Wi-FiWi to test testbed’s scalability performance and to prove the router works in various setup.

Fi-WiFi is an architecture where the data travels from fiber medium to another fiber medium via wireless medium as shown in Figure 3.10. When Client A sends a data to Client D, the data is processed by Router A. Then, it is sent to Router B via fiber and

Router C wirelessly. Router C processes the data again and sends it to router D via fiber optic. Finally, Router D processes the data and sends it to Client D. The process for each router is the same as in Figure 3.15, Figure 3.16 and Figure 3.17. However, simple changes to the program is needed in terms of label values so that the router can perform routing mechanism for Fi-WiFi architecture in a correct manner. Each label for Fi-WiFi must be unique from FiWi architecture and Wi-FiWi architecture to prevent the router from misinterpret the destination of the data. For example, if the Client A, Router A, Client B, Router B, Client C and Router C in FiWi setup as shown in Figure 3.7 have self-labels of 100, 111, 200, 222, 300, and 333 respectively, then, the self-label values for Client A, Router A, Client B, Router B, Client C, Router C, Client D, and Router D in Fi-WiFi setup in Figure 3.8 must be different, such as 500, 501, 600, 601, 700, 701, 800, and 801 respectively. This is because the Fi-WiFi setup is different from FiWi setup since Client D and Router D in FiWi setup has fiber transmission only but not FiWi transmission. On the other hand, in Fi-WiFi includes all the clients and routers to transmit from one client to other client. This method applies to Wi-FiWi as well because it is also a different setup from FiWi and Fi-WiFi setups. The summary of the process is summarized in Table 3.2.

Table 3.2 Summary of labels for FiWi, Fi-WiFi, Wi-FiWi

Topology	Labels							
	Client A	Router A	Client B	Router B	Client C	Router C	Client D	Router D
FiWi	100	111	200	222	300	333	400	444
Fi-WiFi	500	501	600	601	700	701	800	801
Wi-FiWi	502	504	602	604	702	704	802	804

3.4.3 Raspberry Pi Fiber-Wireless Testbed as an Educational Module

Proposed Raspberry Pi router specifications are presented in Appendix D. However, this testbed is not only limited to research, but it is also a good platform for academicians to teach engineering students about how routing in data communication works. The teaching of communication often taught to the students using conventional methods such as notes, textbooks and slides which require more effort and time for the students to understand compared to hands-on experience on the testbed.

Using this testbed, students can learn data transmission in a typical FiWi architecture from a client to another client. When the user sends a data through Ethernet cable, the data needs to undergo medium conversions; electrical signals to optical signals and electrical signals to wireless signals. At the fiber side of FiWi architecture, the data undergo a conversion from electrical signals to optical signals because the travels from Ethernet cable to fiber optic via FMC. The FMC has a circuit board to process and translates the electrical signals to equivalent coded optical signals. A Light Emitting Diode (LED) or laser can be used to generate the optical pulses. These pulses are reflected to fiber optic medium by using lenses, hence, the data is travels through the fiber optic. Once the data arrived at the other FMC, the data is converted back from light signals to electrical signals by using photodiode.

At the wireless side, when the router forwards the data wirelessly, the AP converts the data from electrical signals to electromagnetic signals. These data undergo a modulation called Frequency Modulation (FM) or Frequency Shift Keying (FSK) modulation. The digital data is translated based on the frequency of the waves. Then, at the other end, the AP receives the data wirelessly and demodulates it back to electrical signal. Thus, the client receives the data via Ethernet cable.

3.5 Testbed Parameters

The testbed's design parameters and performance parameters are discussed in this section.

3.5.1 Design Parameters

Design parameters are defined as the input for the FiWi testbed. There are two types of design parameters that have been identified; data size and the end-to-end delay.

The data size is the length of data in terms of *bytes* (*B*). The data size can be varied easily by the user at the client side. In this experiment, the increment of the transmitted data size is 100 *bytes* for each transmission. The data size affects the end-to-end delay

where greater data size may result in greater end-to-end delay. The end-to-end delay is used to study the throughput and jitter of the testbed.

Table 3.3 Design parameters

Description	**Units**
Data size	*B*
End-to-end delay	*s*

3.5.2 Performance Parameters

Performance parameters are the output of the FiWi testbed. These outputs indicate the FiWi performance. There are three performance parameters for this testbed; end-to-end delay, throughput and jitter.

End-to-end delay is the time taken when the user sends the data from a client to a client. Throughput is the overall performance of the testbed in *bit per second* (*bps*) when the data size in *bit* (*b*) is divided by end-to-end delay in *second* (*s*). Finally, jitter is defined as the variation of end-to-end delay in *second* (*s*). The lower the jitter, the better the performance of the testbed.

Table 3.4 Performance parameters

Description	**Units**
Delay	*s*
Throughput	*bps*
Jitter	*s*

3.6 Summary

This chapter describes the proposed FiWi testbed by using Raspberry Pi 3B+ that supports data transmission for research and education purposes. The architecture of this testbed can be scaled easily to Fi-WiFi and Wi-FiWi without any tedious hardware rearrangement. The routing mechanism for FiWi, Fi-WiFi and Wi-FiWi can be achieved by changing the label for each client, Header Pi and Forwarding Pi in Python. The testbed is not only easy to input the parameters but also fast to obtain the result. This chapter also explains on the design parameters such as data size and end-to-end delay. The performance parameters such as delay, throughput and jitter are also discussed in this chapter. The next chapter will elaborate on the results and discussion for FiWi, Fi-WiFi, Wi-FiWi.

CHAPTER 4

RESULTS AND PERFORMANCE EVALUATION

4.1 Introduction

This chapter discusses on the performance evaluation of proposed FiWi testbed. In Section 4.2, the testbed's performance such as validation of the proposed testbed with off-the-shelf router is reviewed. The validation is done in terms of end-to-end delay of the data transmission of the testbed, as well as the throughput and the jitter. The performance of the testbed covers for wireless transmission, fiber transmission and FiWi transmission. This section also presents the outcome of performance evaluation for FiWi stress test in terms of end-to-end delay, throughput and jitter. Whereas for scalability performance test, this section presents the performance of Fi-WiFi and Wi-FiWi. Finally, Section 4.7 presents the summary of this chapter

4.2 Wireless Transmission Performance Test

For this test, Raspberry Pi client is used to send a packet to another client via a pair of wireless routers. The initial data size transmitted is at 100 *bytes*. Then, the data size is increased by 100 *bytes* each time up to 1445 *bytes*. As mentioned in Chapter 3, 100 *bytes* is chosen to achieve maximum reliability [96]. The point-to-point transmitting and receiving experiments are done for upstream and downstream transmissions. The performance of the experiment in terms of end-to-end delay, throughput and jitter are plotted on the graph.

The proposed router is compared with off-the-shelf router to check its functionality and to observe its correctness. A pair of off-the-shelf routers used in this experiment are D-LINK DAP1360 because they can intercommunicate directly via MAC address by using Bridge configuration [97]. The setup for the off-the-shelves router is the same as the proposed testbed, which is point-to-point. The upstream and downstream transmissions are done by using two computers. One computer sends the data in the form of a file from one computer to another computer. The initial data size for the file is 10 *kB*. The data size is increased by 10 *kB* each time until 100 *kB*. It is increased

until 100 *kB* because the maximum bandwidth of the off-the-shelf router is not more than 1 *Mbps* [97]. The maximum bandwidth of the off-the-shelf router obtained through numerous experiments are consistent to 700 *kbps*. This is deemed acceptable as typical router has a max of 60% to 70% of its throughput in datasheet due to its header and congestion algorithm [98]. Then, the throughput of the experiment for upstream and downstream are captured using Wireshark.

4.2.1 Throughput

Figure 4.1 and Figure 4.2 show the downstream and upstream graphs of throughput for the proposed router and off-the-shelf router respectively. Based on Figure 4.1 the trend of the off-the-shelf router for upstream is increasing as the offered load increases from 0 *bps* at 0% offered load to 700 *kbps* at 100% offered load. Meanwhile, the throughput of the proposed router is also increasing as the offered load increases from 0 *bps* at 0% offered load to 677 *kbps* offered load. Figure 4.2 shows that the throughput of the off-the shelf-router for downstream is increasing as the offered load increases from 0 *bps* at 0% offered load to 774 *kbps* at 100% offered load. Meanwhile, the throughput for the proposed router also increases from 0 *bps* at 0% offered load to 752 k*bps* at 100% offered load. The throughput of the proposed router is scaled up to the throughput of the off-the-shelf router to observe the trend of the graph. The results show that the throughput of the proposed router has similar increasing trend with the throughput of off-the-shelf router, hence proving that proposed router performance is correct. For both Figure 4.1 and Figure 4.2, the off-the-shelf router has higher capability than the proposed router. This is expected because the proposed router has lower processing power than off-the-shelf router. Hence, it takes longer time to forward the data. The throughput for the proposed router is calculated by using Equation 4.1.

$$Throughput\ (bps) = \frac{Data\ transmitted\ (bit) - Data\ loss\ (bit)}{End-to-end\ delay\ (s)} \tag{4.1}$$

For both downstream and upstream transmissions, off-the-shelf router is able to transmit data up to 700 *kbps* and 774 *kbps* respectively at 100% offered with 1445 *bytes* of in each packets. But for the proposed router, one packet of 1445 *bytes* is able

to achieve up to 52.097 *kbps* for downstream transmission and 54.335 *kbps* for upstream transmission. Hence, if the proposed router sends as many packets as the off-the-shelf router, the data that can be transmitted in the proposed router is vertically scaled up by factor of 13.436 ($\frac{700\ kbps}{52.097\ kbps} = 13.436$) for downstream and 12.883 for upstream ($\frac{774\ kbps}{54.335\ kbps} = 12.883$). Vertical scaling is a method to translate a graph without losing the original properties where all y-values in the graph are multiplied by a specific factor. Therefore, for the proposed router, the throughput for each offered load is multiplied with 13.436 and 12.883 for downstream and upstream respectively.

It is expected that the throughput for the proposed router is lower than the off-the-shelf router because off-the-shelf router has higher specifications, processing power and price. The differences in bandwidth values for both off-the-shelf router and proposed router are not of our concern because the one of the objectives of this project is to create a simple and fast reconfigurable router testbed that supports FiWi transmission and able to serve as FiWi educational module as stated in Appendix D.

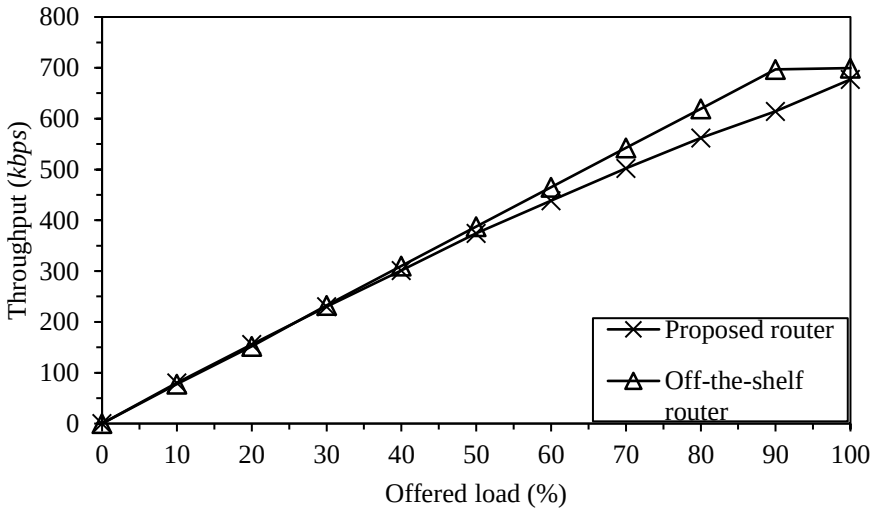

Figure 4.1 Downstream throughput

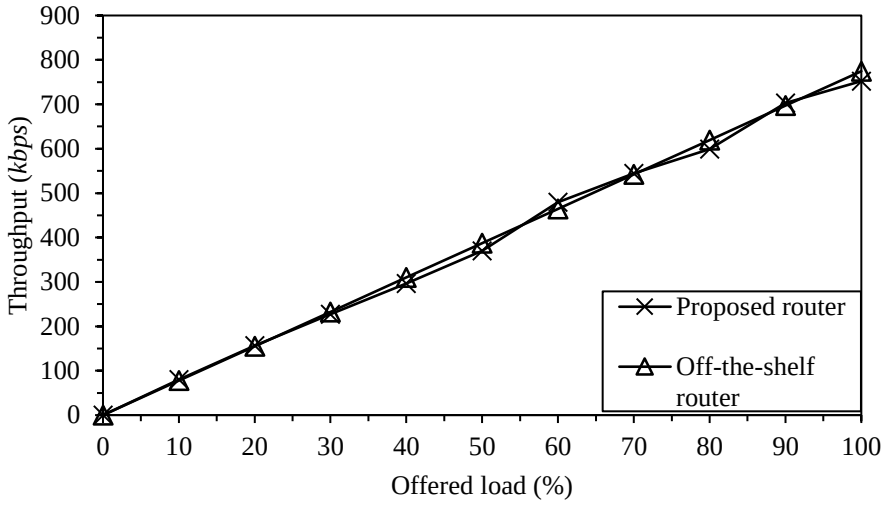

Figure 4.2 Upstream throughput

4.2.2 End-to-end delay

This section explains the end-to-end delay of the wireless router. End-to-end delay is the delay between a client to another client via a pair of proposed routers. From the graph in Figure 4.3, the end-to-end delay increases as the data size increases for both upstream and downstream transmissions. This is as expected because larger data size requires more processing time. Based on Figure 4.3, the downstream end-to-end delay starts from 0.13 *s* at 100 *bytes* to 0.19 *s* at 1445 *bytes*. Whereas upstream end-to-end delay starts from 0.14 *s* at 100 *bytes* to 0.19 *s* at 1445 *bytes*. For both graphs, the end-to-end delay increases because the proposed router needs more time to process as the data size getting bigger. In Figure 4.3, the end-to-end delay at 1445 *bytes* has sudden increase from 1400 *bytes*. This is due to the proposed router that has reached its processing limit due to hardware limitations. Upstream transmission has higher end-to-end delay compared to downstream transmission. However, there are not much difference between downstream and upstream delay which is only about 8 *ms*. Despite this difference, the end-to-end delay is acceptable due to the trend of both graphs satisfying the behaviour of the trend of end-to-end delay in IEEE 802.15.4 routing scheme [99] where it increases as the data size increases. Hence, it can be concluded that, the proposed router is suitable to be a wireless router for research and educational purposes because the behaviour of the graph is satisfying the nature of typical router.

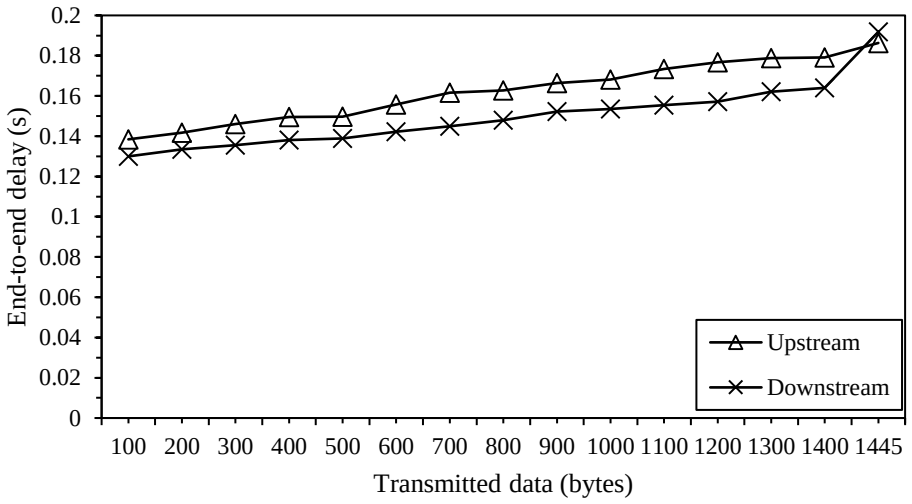

Figure 4.3 End-to-end delay for wireless transmission

4.2.3 Jitter

The downstream and upstream jitter graphs for the wireless router are shown in Figure 4.4 and Figure 4.5 respectively. The purpose of analysing the jitter of the proposed router is to check whether the jitter values are in the acceptable range. Jitter is defined as the variation in time between packets arriving, caused by network congestion, timing drift, or route changes. The lower the jitter, the better the performance of the system can be. The jitter, *J* is calculated in Equation 4.2. In which, Packet Delay Variation (PDV) is achieved by using Equation 4.3. According to [100], $D_{i,j}$ is the time of a transmission *i*, in transmitter *j* and *Np* is the total number of receiving packets. D_{ave} is the average transmission time of *i* in transmitter *j* that is achieved by using Equation 4.4.

$$Jitter, J\ (s) = \sqrt{PDV} \tag{4.2}$$

$$PDV = \frac{\sum_{c=1}^{Np}(D_{i,j} - D_{ave})^2}{Np} \tag{4.3}$$

$$D_{ave} = \frac{\sum_{c=1}^{Np} D_{i,j}}{Np} \tag{4.4}$$

In Figure 4.4 and Figure 4.5, there is no observable trend of graph can be monitored unlike end-to-end delay and throughput. This is because the jitter relates to the variation in the end-to-end delay. Therefore, the jitter varies throughout the transmitted data. In Figure 4.5, the jitter varies from 0.929 *ms* to 9.80 *ms*. The high jitter at 1445 *bytes* is caused by the high end-to-end delay as shown in Figure 4.3. Meanwhile the jitter graph in Figure 4.6, the values vary from 1.57 *ms* to 6.73 *ms*. Based on Figure 4.6, the highest jitter is shown at 1445 *bytes*. This is due to the end-to-end delay graph in Figure 4.4, 1445 *bytes* gives the highest end-to-end delay. In this wireless transmission, it can be concluded that the jitter relates closely with the delay in which when the delay is high. Moreover, despite there is no observable trend of jitter graphs, the jitter values are still within the acceptable range because according to Cisco in [101], the acceptable jitter values must be below 30 *ms*. Hence, proving that the proposed testbed is suitable to be used as a wireless router.

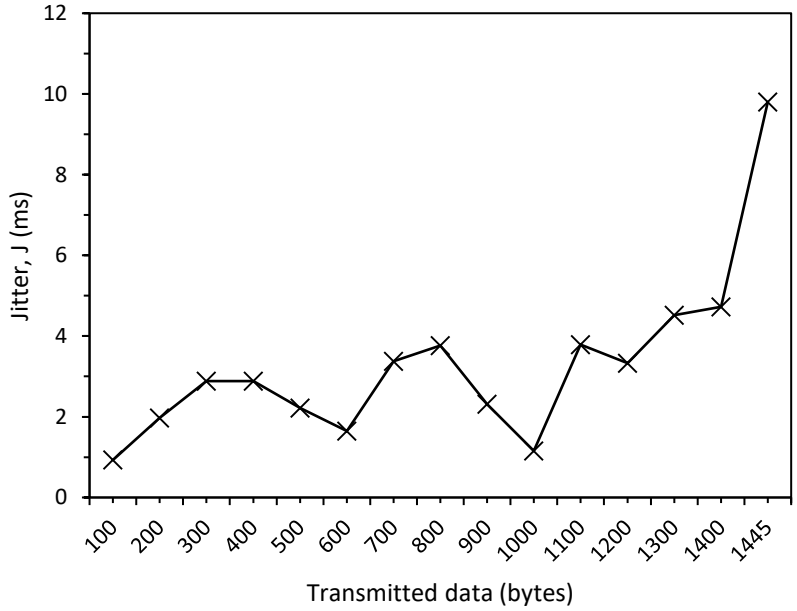

Figure 4.4 Proposed router downstream jitter

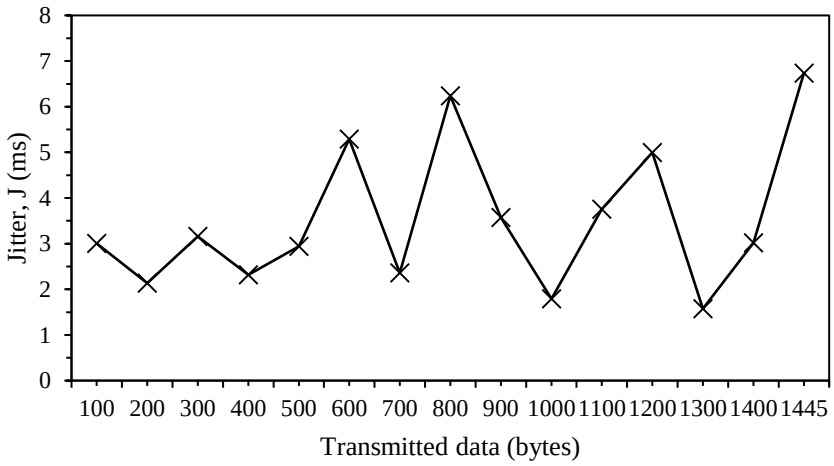

Figure 4.5 Proposed router upstream jitter

4.3 Fiber Transmission Performance Test

The setup for this test is similar to the wireless performance test but instead of using wireless transmission, the data are transmitted by using fiber. The throughput of the testbed is validated with industrial grade router. The data size for the industrial grade router is initially set at 10 *MB* and it is increased by 10 MB for each transmission until it reaches 100 *MB*. The limit is set at 100 *MB* because 100 *MB* is a reasonable maximum data size for 1 *Gb* data size considering the total bandwidth needs to include the packet headers as well. The throughput of the transmission is recorded by using Wireshark.

4.3.1 Throughput

Figure 4.6 and Figure 4.7 show throughput graphs for downstream and upstream transmissions respectively between industrial grade router and the proposed router. For a packet of 1445 bytes, the throughput of proposed router is 92 *kbps* and 91 *kbps* for downstream and upstream respectively. Then, the scaling method used same as in wireless transmission. The graphs show that the throughput of proposed router have similar increasing trends with industrial grade routers as the offered load increases. Hence, the performance of the proposed router is validated. In Figure 4.6, the trends of the throughput for proposed router and industrial grade router for downstream are increasing as the offered load increases. For proposed router, the throughput increases from 0 *bps* at 0% offered load to 767 *Mbps* at 100% offered load. Meanwhile, the

throughput for industrial grade router increases from 0 *bps* at 0% offered load to 540 *Mbps* at 80% offered load. However, at 50% of offered load, the throughput of the industrial grade router is increasing at slower rate because it approaches the limit of the bandwidth. Then, at 80% of the offered load, the throughput is stagnant until 100% of offered load. It is due to the bandwidth has been fully utilized.

Meanwhile in Figure 4.7, the trends of the throughput for both proposed router and industrial grade router for upstream are also increasing as the offered load increases. In Figure 4.7, the throughput of proposed router increases from 0 *bps* at 0% offered load to 785 *Mbps* at 100% offered load. Meanwhile, the throughput for industrial grade router increases from 0 *bps* at 0% offered load to 480 *Mbps* at 60% because the throughput of the industrial grade router reaches its limit at 60% of the offered load. The reason of why the proposed router's throughput keeps increasing for both graphs is because there is no QoS in the proposed router. On the other hand, the industrial grade router has QoS to prevent congestions. Hence, it limits the maximum throughput to reserve the bandwidth in case of flooding. From the validation, we can conclude that the trends of the proposed router are correct. It is expected that the industrial grade router has higher throughput before 60% of offered load because it has greater specifications and data processing power compared to proposed router.

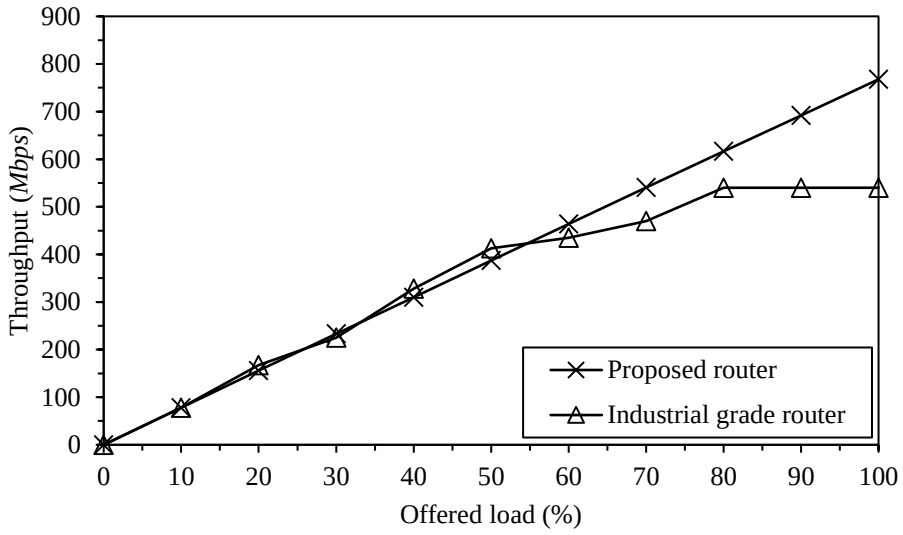

Figure 4.6 Downstream throughput

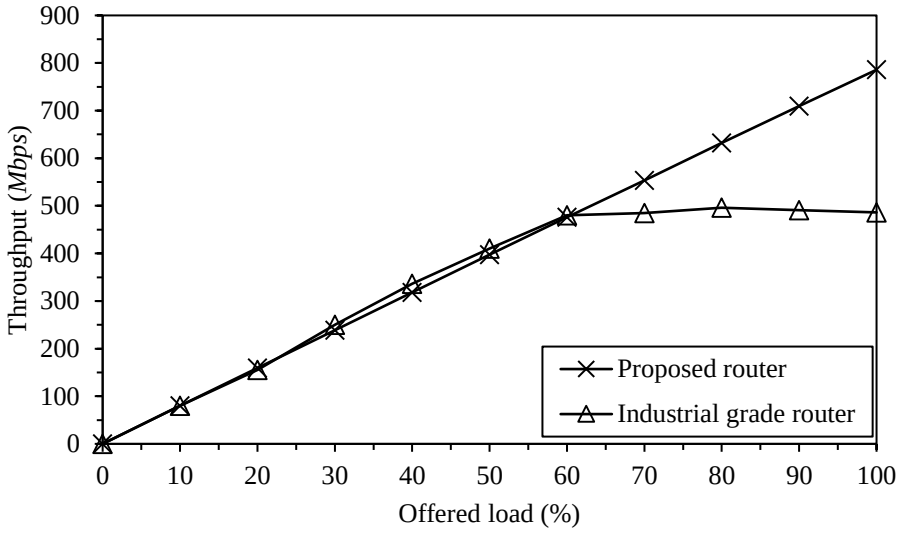

Figure 4.7 Upstream throughput

4.3.2 End-to-end delay

This section explains the end-to-end delay of the proposed router in fiber transmission. Based on the graph in Figure 4.8, the end-to-end delays are increasing as the data size increases for downstream and upstream transmissions. This is due to more time taken to process the data when the data are transmitted. There is not much change in delay for both downstream and upstream. This is due to the bandwidth in fiber is large, but the transmitted data size is small. In Figure 4.8, the minimum downstream end-to-end delay is 0.122 *s* at 100 *bytes*, while the maximum end-to-end delay is 0.126 *s* at 1445 *bytes*. As for upstream transmission, the minimum end-to-end delay is 0.123 *s* at 100 *bytes*, while the maximum end-to-end delay is 0.126 *s* at 1445 *bytes*. The end-to-end delay is expected to be higher than a typical fiber-supported router like industrial grade router because of hardware limitation. Furthermore, one of the objectives of this thesis is to create a reconfigurable router testbed that supports fiber transmission for educational module.

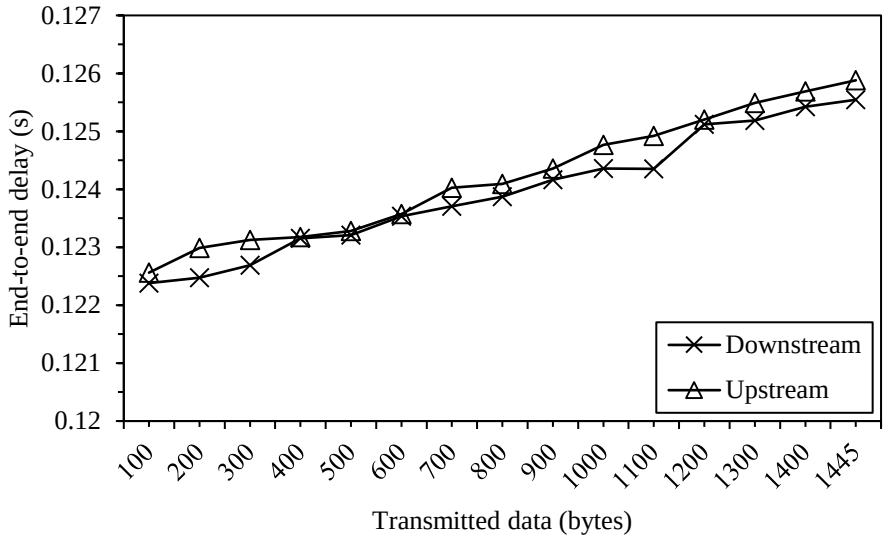

Figure 4.8 End-to-end delay for fiber transmissions

4.3.3 Jitter

The jitter for downstream and upstream transmissions are calculated using Equation 4.2, Equation 4.3 and Equation 4.4. The jitter graphs for downstream and upstream are shown in Figure 4.9 and Figure 4.10 respectively. In Figure 4.9 and Figure 4.10, there is no observable trend like end-to-end delay and throughput. This is because the jitter relates to the variation of end-to-end delay for each transmitted data. If the variation is high, then the jitter is high and otherwise. For downstream jitter ranges from 0.04 *ms* to 0.38 *ms*. The jitter reaches its peak at 0.38 *ms* which is when 1200 *bytes* is transmitted. Whereas, the jitter for upstream transmission ranges from 0.12 *ms* to 0.49 *ms*. The jitter reaches its peak for upstream transmission is when 1445 *bytes* is transmitted which is at 0.49 *ms*. Based on these values, it can be concluded that the transmission for fiber is stable because the jitter values are very small due to large bandwidth and minimal noise in fiber. The jitter values are below 30 *ms* which is within acceptable range [101]. This proves that the testbed is suitable to be used as fiber-based router.

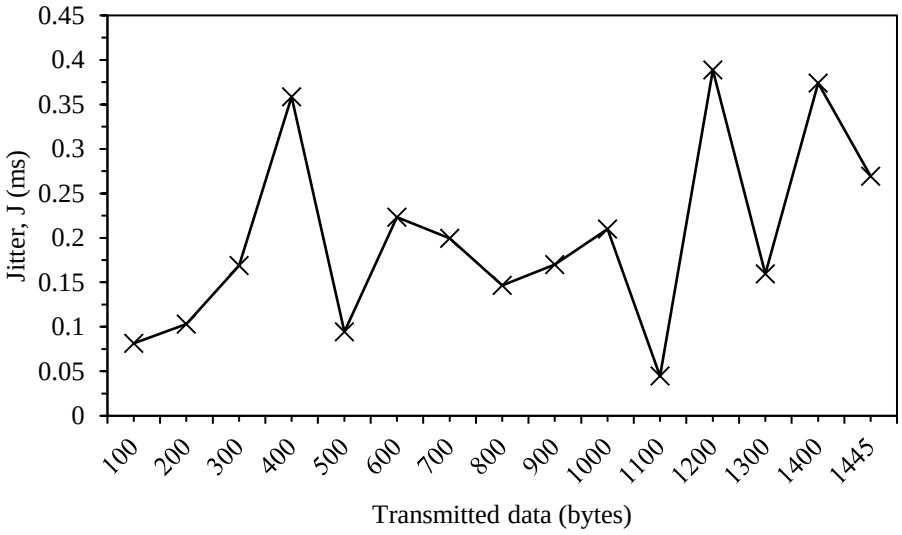

Figure 4.9 Downstream jitter

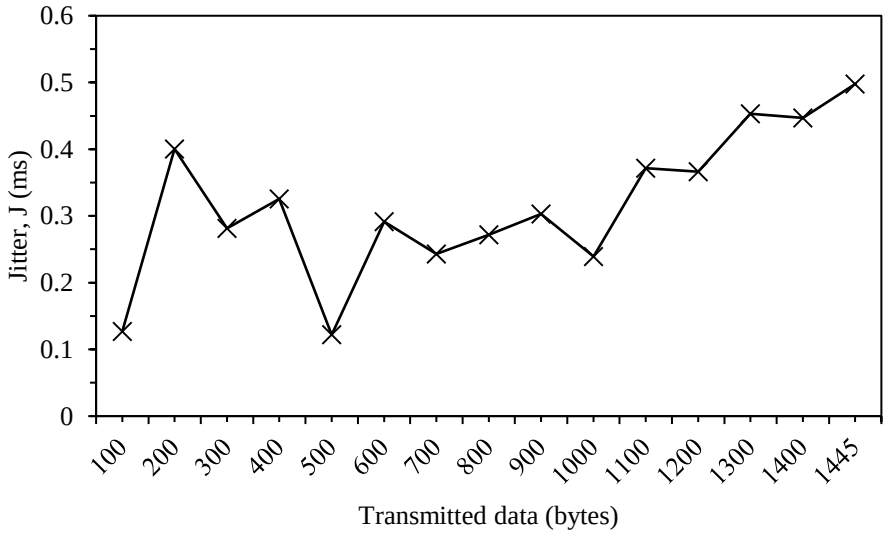

Figure 4.10 Upstream jitter

4.4 Fiber-Wireless Transmission Performance Test

The setup for this test is the combination of fiber and wireless transmission as shown in Figure 3.7 where the data is sent from Client A to Client C. The data size for this test is the same as the wireless performance test and fiber performance test. The throughput of the FiWi proposed router is validated with industrial grade router. The bandwidth of the industrial grade router is set to 1 *Gbps*. However, the initial data size is set to 10 *kB* and increased by 10 *kB* for each transmission until it reaches 100 *kB*.

This is because the industrial grade router is connected to an access point to match its setup with the proposed router setup. Therefore, the actual bandwidth is 1 *Mbps*.

4.4.1 Throughput

Figure 4.11 and Figure 4.12 show the throughput graphs for downstream and upstream transmission respectively. The purpose of this validation is to observe the correctness of the proposed router. For a packet of 1445 bytes, the throughput of proposed router is 79.8 *kbps* and 93 *kbps* for downstream and upstream respectively. Then, the scaling method used same as in wireless transmission. Based on the graphs, the throughput of the proposed router has similar increasing trend as the industrial grade router. This shows that the transmission of the proposed router is correct. Based on Figure 4.11, the throughput for proposed router increases from 0 *bps* at 0% offered load to 719 *kbps* at 100% offered load. Meanwhile, the throughput of industrial grade router increases from 0 *bps* at 0% offered load to 697 *kbps* at 90%. However, the throughput of the industrial grade router decreases at 100% offered load. Based on Figure 4.12, the throughput of the proposed router increases as the offered load increases from 0 *bps* at 0% offered load to 770.351 *kbps* at 100% offered load. Meanwhile, the throughput for industrial grade router is also increasing from 0 *bps* at 0% to 774 *kbps* at 100% offered load.

In Figure 4.11, the throughput of the industrial grade router decreases at 100% offered load because the router has reached its bandwidth limit. Meanwhile, the throughput of the proposed router is keep increasing. As mention earlier, this is due to the proposed router does not have QoS, whereas, the industrial grade router has QoS algorithm to limit the maximum throughput. Overall, the throughput of the proposed router is lesser than industrial grade router due to its limited specs, bandwidth and processing power. This is expected because the proposed router is built for lab-scale experiment, but, the industrial grade router is meant for industrial purposes. From the graph, we can conclude that the trend of the proposed router is correct. Unlike individual throughput of fiber and wireless, the throughput of FiWi has a bottleneck at the wireless side of the setup. Hence, the throughput is limited at around 700 *kbps*.

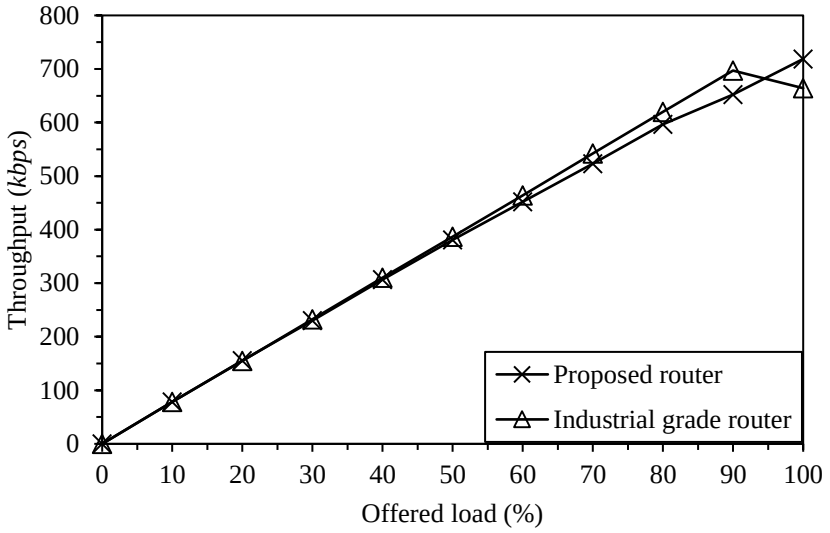

Figure 4.11 Downstream throughput

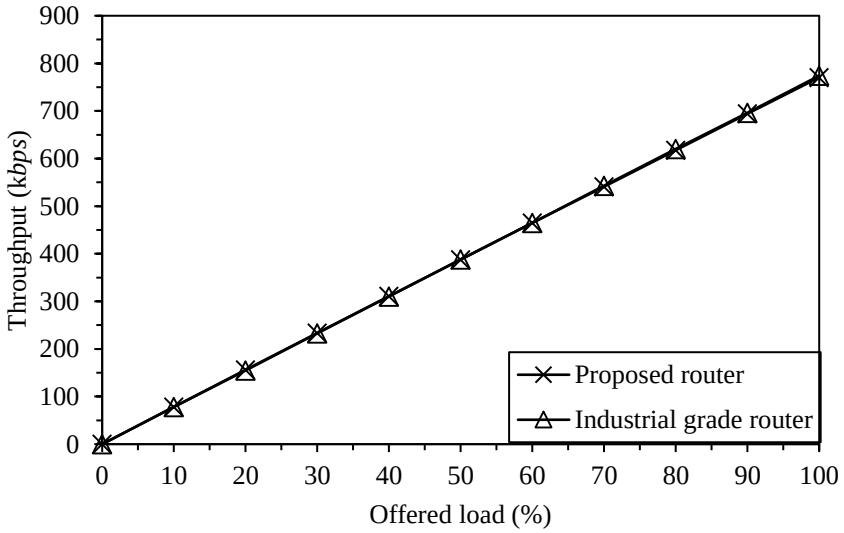

Figure 4.12 Upstream throughput

4.4.2 End-to-end delay

Figure 4.13 show the end-to-end delay for downstream and upstream transmissions in FiWi. The graphs show increasing trend of end-to-end delay for both downstream and upstream transmissions due to increase in data size. Based on Figure 4.13, the minimum downstream end-to-end delay is 0.125 *s* at 100 *bytes*, while its maximum is 0.145 *s* at 1445 *bytes*. Whereas the minimum upstream end-to-end delay is 0.122 *s* at 100 *bytes*, while its maximum is 0.124 *s* at 1445 *bytes*.

The end-to-end delay for both downstream and upstream increases because as the data size increases, the router takes longer time to process the data. The reason why there are not many differences in the graphs is because the data size is very small for a large bandwidth of fiber which is 1 *Gbps*. Furthermore, the data is transmitted by using light pulses in fiber. Hence, it transmits faster compared to electrical pulses in copper. For example, during downstream transmission, the end-to-end delay at 100 *bytes* is 0.1245 *s*, whereas at 200 *bytes* the end-to-end delay is 0.1249 *s*. The overall end-to-end delay for downstream is higher than upstream as expected in [102]. However, even though the end-to-end delay of the proposed testbed is higher, but, its increasing trend complies with the trend in IEEE 802.15.4 routing scheme [99]. It is expected to be higher than the standard because of hardware limitation. Furthermore, one of the objectives of this thesis is to create a reconfigurable router testbed that supports fiber-wireless transmission for educational module.

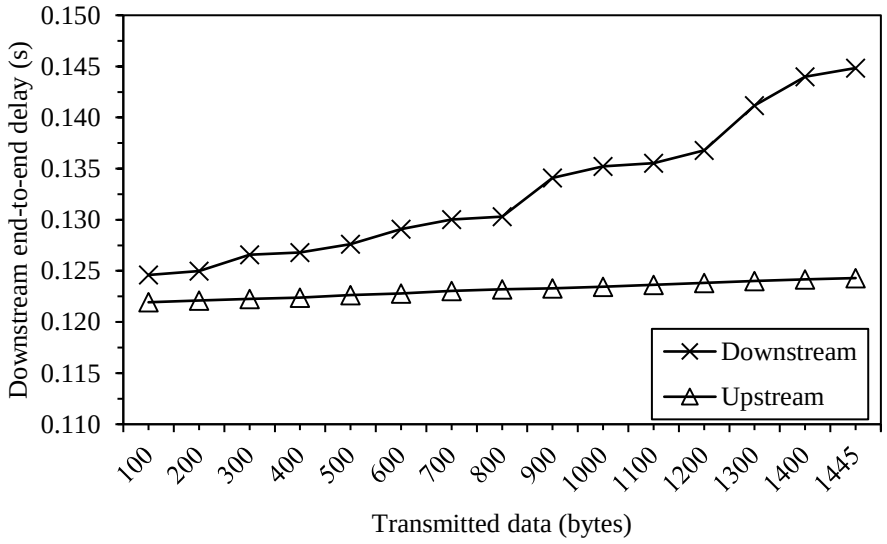

Figure 4.13 End-to-end delay for FiWi transmission

4.4.3 Jitter

Figure 4.14 and Figure 4.15 show the graphs for downstream and upstream jitter respectively. In Figure 4.14, the trend of the jitter is increasing as the data size increases from 0 *s* at 0 *bytes* to 3.96 *ms* at 900 *bytes* because as the data size increases, the delay variation between transmitted data increases, hence the jitter increases. However, the trend of the graph is not consistent from 1000 *bytes* to 1445 *bytes*

because the delay variations between transmitted data are not consistent. For example, the jitter from 900 *bytes* to 1000 *bytes* decreases because the delay variation from 1000 *bytes* is smaller than 900 *bytes*. Whereas, the delay variation from 1000 *bytes* to 1100 *bytes* increases, hence the jitter increases. Based on the graph, the jitter values are having a huge gap from 1300 *bytes* to 1445 *bytes* ranging from 2.49 *ms* to 8.25 *ms* because the proposed router is reaching its limit, hence making the data transmission unstable which causes the delay variation to be high. Based on Figure 4.15, the jitter graph for upstream transmission varies from 0 *s* at 0 *bytes* to 0.108 *ms* at 800 *bytes*. This is due to the upstream transmission is done right after the downstream transmission. Therefore, the hardware became heated causing the transmission not stable. Hence, there is the inconsistency in the jitter graph for upstream transmission. Despite there not observable trend in both graphs, the jitter values are acceptable because according to Cisco, the jitter values must be below 30 *ms*.

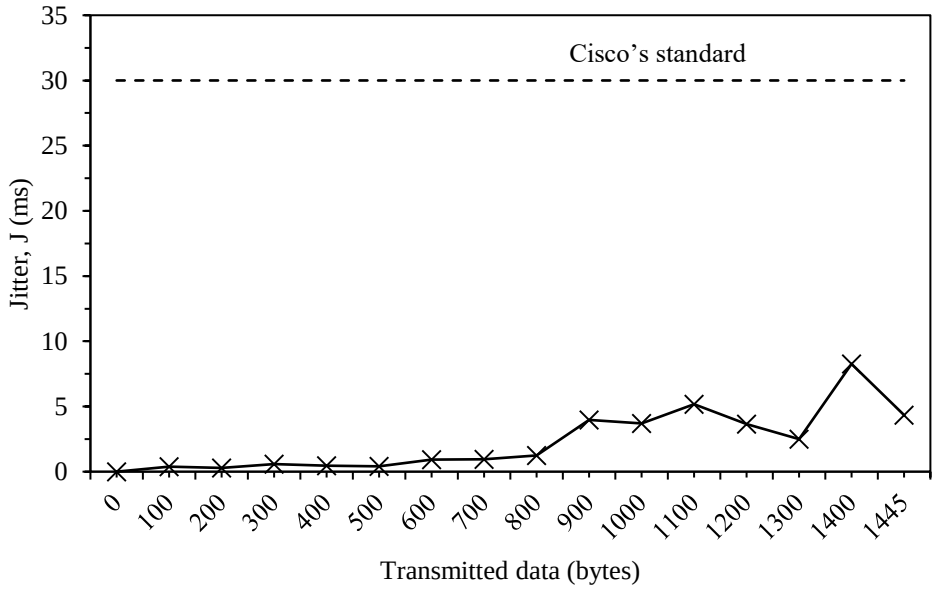

Figure 4.14 Downstream jitter

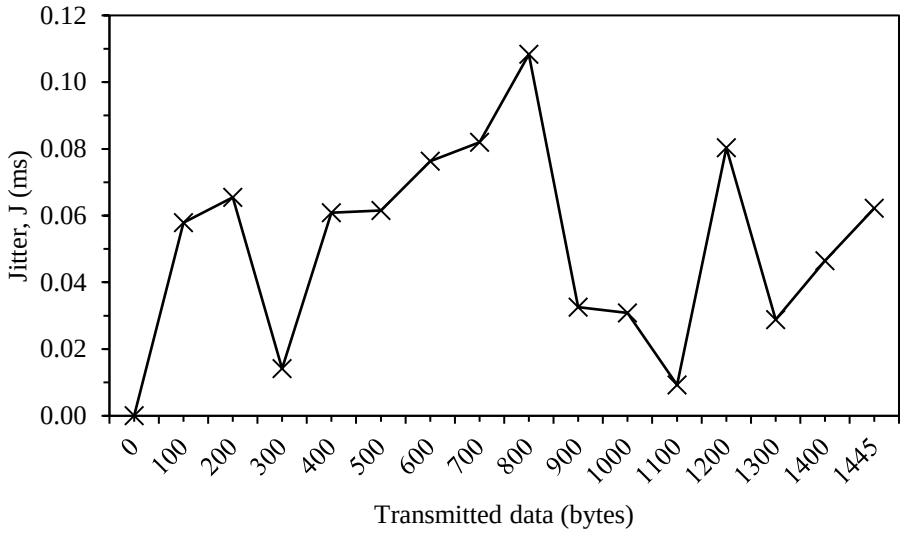

Figure 4.15 Upstream jitter

4.5 Fiber-Wireless Stress Test

Unlike FiWi performance test, there is no need to scale up the throughput of this test because it is already proven that the proposed router is suitable to be a FiWi router. Hence, in this section, the limit of the FiWi testbed is tested by transmitting two traffics simultaneously from one router to another. This experiment is done by connecting two clients for each proposed router. This section presents the throughput, end-to-end delay and jitter for downstream and upstream transmission. The purpose of this section is to analyse the performance of the proposed router when there are two traffics transmitted simultaneously.

4.5.1 Throughput

Figure 4.16 and Figure 4.17 show the throughput graphs for downstream and upstream transmissions respectively. Based on Figure 4.16, the trend of the throughput graph is increasing as the data size increases. The minimum throughput is 3.210 *kbps* at 100 *bytes*, while the maximum throughput is 40.047 *kbps* at 1400 *bytes*. The throughput decreases at 1445 *bytes* from 40 *kbps* to 39.5 *kbps* because it is the limit of the router. Compared with the throughput of a single traffic, the throughput of two traffics is half of the throughput of single traffics. The reason of this occurrence is two traffics causing the proposed router to process the data twice. For example, at 100 *bytes* in

downstream transmission, the throughput for a single traffic is 6.42 *kbps* whereas the throughput for two traffics is 3.21 *kbps*. This statement is supported by Xu et al. [103] where more traffics contribute to less throughput.

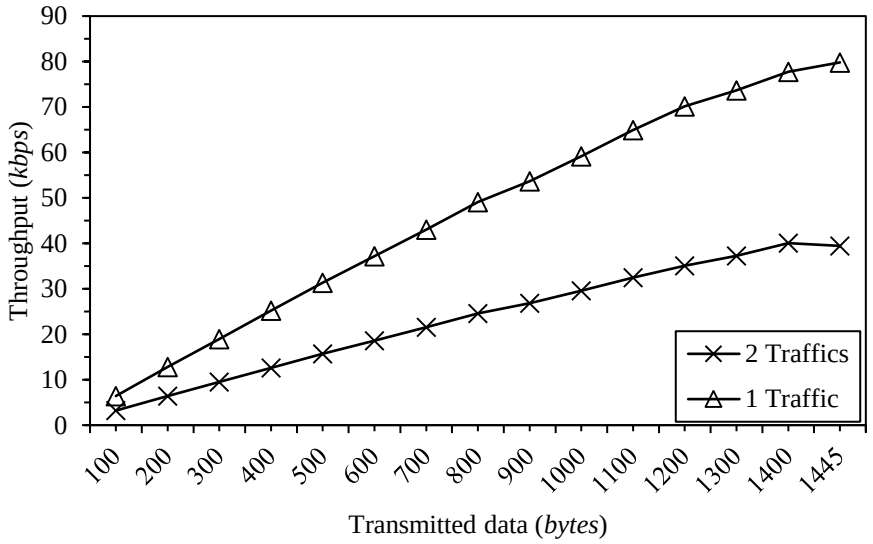

Figure 4.16 Downstream throughput

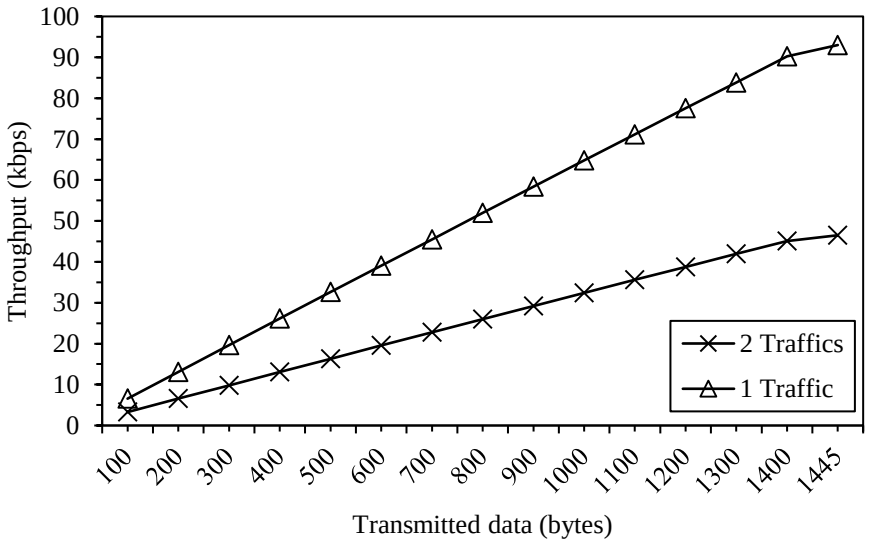

Figure 4.17 Upstream throughput

4.5.2 End-to-end delay

Figure 4.18 and Figure 4.19 show the downstream and upstream end-to-end delay graphs respectively. Based on Figure 4.18, the trend of the graph is increasing from

0.249 *s* at 100 *bytes* to 0.293 *s* at 1445 *bytes*. The increasing trend of the graph is due to increasing values of data size. At 1445 *bytes*, there is a sudden increase of end-to-end delay. This is due to the proposed router is at its limit of processing two traffics simultaneously. Theoretically, when number of traffics increases, the end-to-end delay also increases [104]. Therefore, comparing the graph in Figure 4.19, the end-to-end delay of two traffics is twice as big as single traffic. This is due to the proposed router needs to process the data twice compared to a single traffic which the proposed router processes the data only once. For example, the end-to-end delay at 1445 *bytes* for single traffic and two traffics are 144.838 *ms* and 293.01 *ms* respectively.

Based on Figure 4.19, the upstream end-to-end delay graph increases from 243.863 *ms* at 100 *bytes* to 248.591 *ms* at 1445 *bytes*. Like downstream transmission, the end-to-end delay for upstream transmission is twice as high as single traffic. This is due to the proposed router needs to process the data twice compared to a single traffic. For example, at 1445 *bytes*, the end-to-end delay for two traffics and single traffic are 248.591 *ms* and 124.296 *ms* respectively. Hence, we can conclude that even though the trends of the graphs are looking constant, but, actually it is increasing as the data increases. The constant trends of the graphs are due to the presence of fiber which make the difference in end-to-end delay between two data such as 100 *bytes* and 200 *bytes* are not significant.

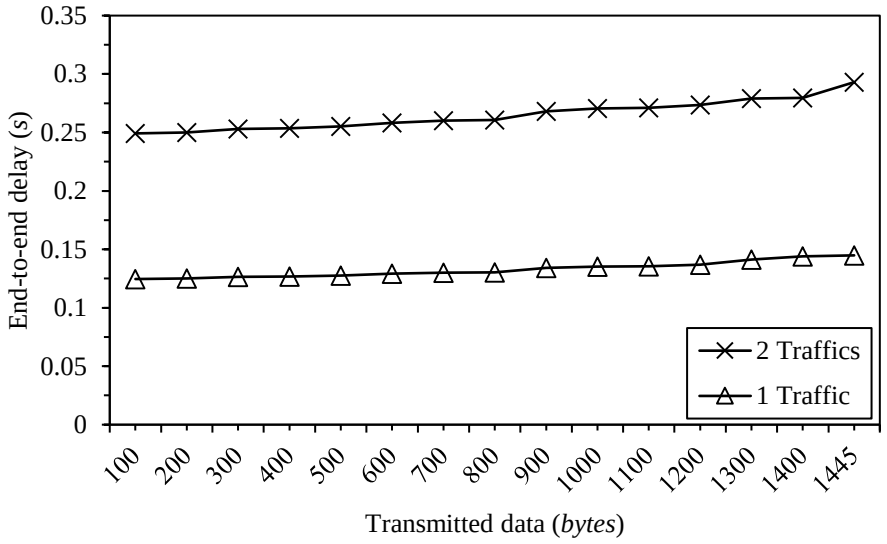

Figure 4.18 Downstream end-to-end delay

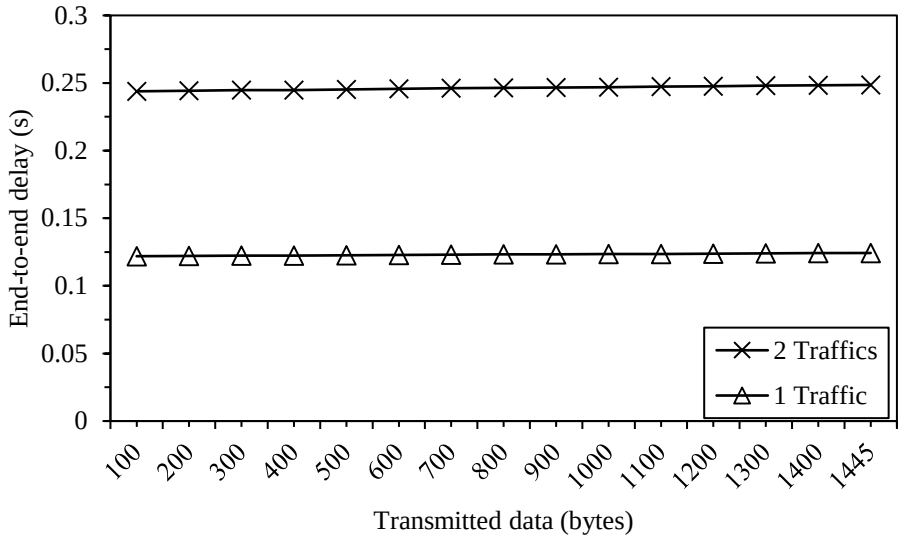

Figure 4.19 Upstream end-to-end delay

4.5.3 Jitter

Figure 4.20 and Figure 4.21 show the jitter for upstream and downstream transmissions for two traffics. The purpose of this section is to check whether the jitter values for two traffics are within acceptable range or not. Based on the graph in Figure 4.20, the jitter increases from 0.757 *ms* at 100 *bytes* to 7.91 *ms* at 900 *bytes*. Then, the jitter starts to vary from 7.393 *ms* at 1000 *bytes* to 3.454 *ms* at 1300 *bytes*. The jitter continues to increase from 3.454 *ms* at 1300 *bytes* to 11.22 *ms* at 1445 *bytes*. Based on Figure 4.24, the jitter for upstream transmission varies from 18.49 *μs* at 1100 *bytes* to 0.217 *ms* at 800 *bytes*.

The jitter reaches its peak at 1445 *bytes* in Figure 4.20 because it is the limit of the proposed router. Therefore, the delay variation between transmissions at this point is very high. Whereas in Figure 4.21, the jitter varies and inconsistent due to the proposed router becomes heated when running the routing for too long. Compared to single traffic, two traffics have greater jitter because the proposed router needs to process the data twice. Therefore, it takes longer time to process for each data causing the delay variation for each data transmission to be higher. For example, at 1445 *bytes*, the downstream jitter for single traffic is 4.35 *ms*, while the downstream jitter for two traffics is 11.22 *ms*. As for upstream transmission, the jitter for single traffic is 0.06 *ms* at 1445 *bytes*, whereas the jitter for two traffics is 0.12 *ms* at 1445 bytes. Despite

having higher jitter compared to a single traffic, the jitter values for two traffics are still within acceptable range. Hence, we can conclude that the testbed is scalable.

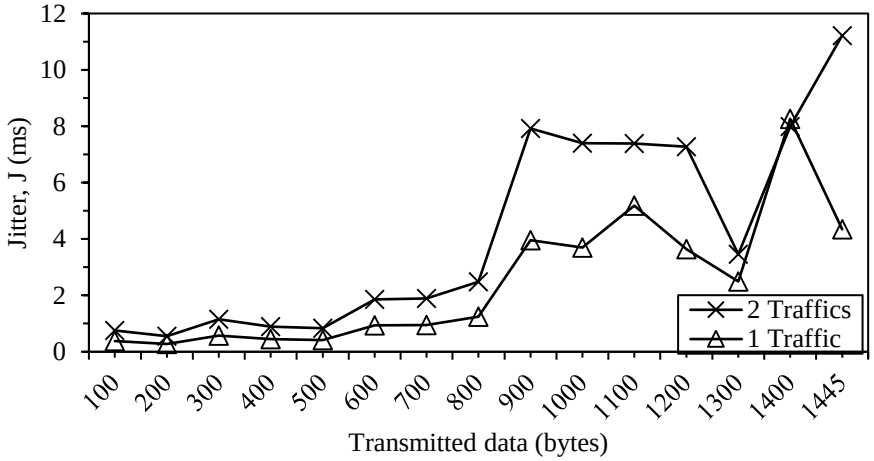

Figure 4.20 Downstream jitter

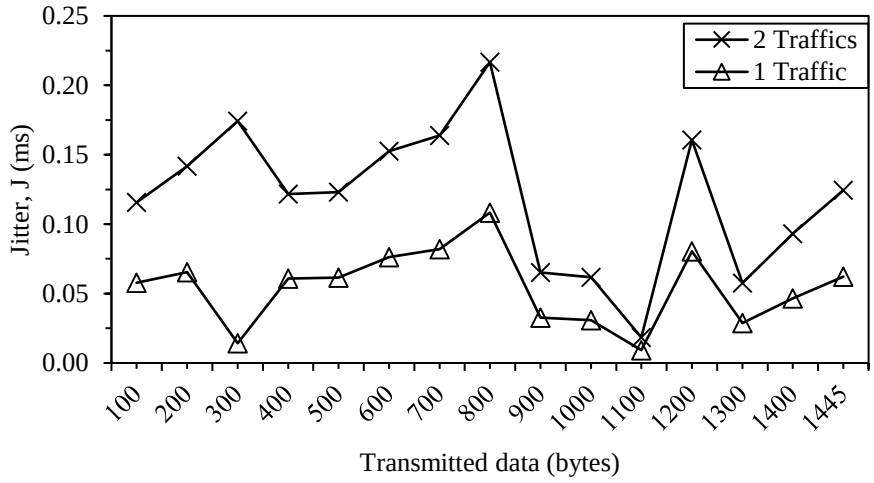

Figure 4.21 Upstream jitter

4.6 Scalability Performance Test

This section presents the performance for Fi-WiFi and Wi-FiWi in terms of end-to-end delay and throughput. Unlike FiWi performance test, the throughput of both Fi-WiFi and Wi-FiWi are not scaled up because it is proven the proposed router is suitable to be FiWi router testbed. Hence, the purpose of this section is to analyse the scalability performance and to compare which architecture makes the router works more stable.

4.6.1 Fiber-Wireless-Fiber Performance Test

Figure 4.22 shows the downstream and upstream throughput for Fi-WiFi network. Based on Figure 4.22, the downstream throughput increases as the data size increases from 2.626 *kbps* at 100 *bytes* to 22.403 *kbps* at 1445 *bytes*. Meanwhile, upstream throughput graph is increasing up until 1100 *bytes* only. Then, the graph becomes unstable from 1200 *bytes* to 1300 *bytes*. Then, the throughput continues to decrease from 1400 *bytes* to 1455 *bytes*.

Even though the downstream throughput is increasing, but the gradient of the graph is gradually decreasing starting at 700 *bytes*. Whereas the upstream throughput becomes unstable at 1200 *bytes* to 1300 *bytes* because the proposed router is almost at its limit. Furthermore, the instability of throughput at these points is because the gradient of the graph of the upstream end-to-end delay suddenly gets bigger at 1200 *bytes*. Then, the gradient of the graph of upstream end-to-end delay becomes smaller at 1300 *bytes* causing the upstream throughput at 1300 *bytes* to increase. Then, the throughput decreases at 1400 *bytes* and 1445 *bytes* due to the sudden increase of upstream end-to-end delay at 1400 *bytes* in and it continues to increase until 1445 *bytes*.

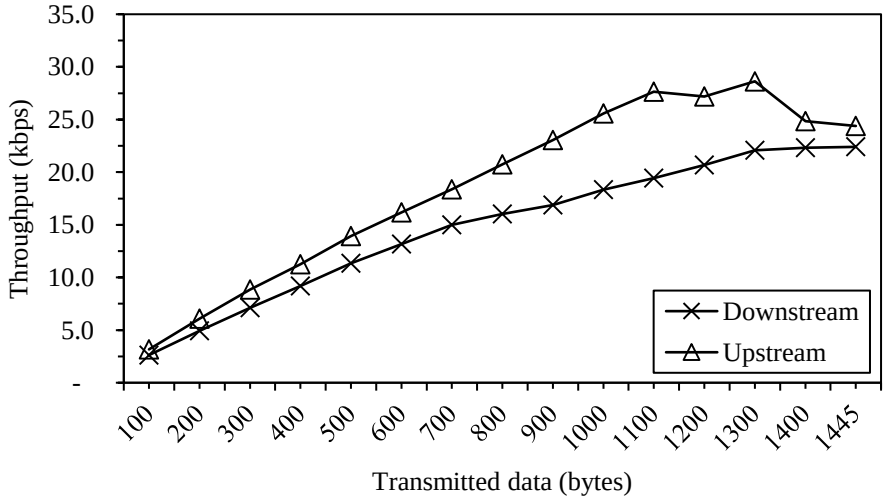

Figure 4.22 Fi-WiFi downstream throughput

Figure 4.23 shows the downstream and upstream end-to-end delay for Fi-WiFi network. Based on Figure 4.23, the end-to-end delay increases as the data size increases from 0.609 *s* at 100 *bytes* to 1.032 *s* at 1445 *bytes*. Whereas upstream the

end-to-end delay increases as the data size increases from 0.505 *s* at 100 *bytes* to 0.948 *s* at 1445 *bytes*. The end-to-end delay for this setup is higher than the other setup, such as FiWi and wireless. This is because the data has to undergo multiple routers and medium conversions from fiber to wireless and then wireless to fiber. For downstream transmission, there is a sudden increase in end-to-end delay at 1400 *bytes*. This is due to the proposed router is already approaching its limit. Hence, the data processing time takes longer at this point.

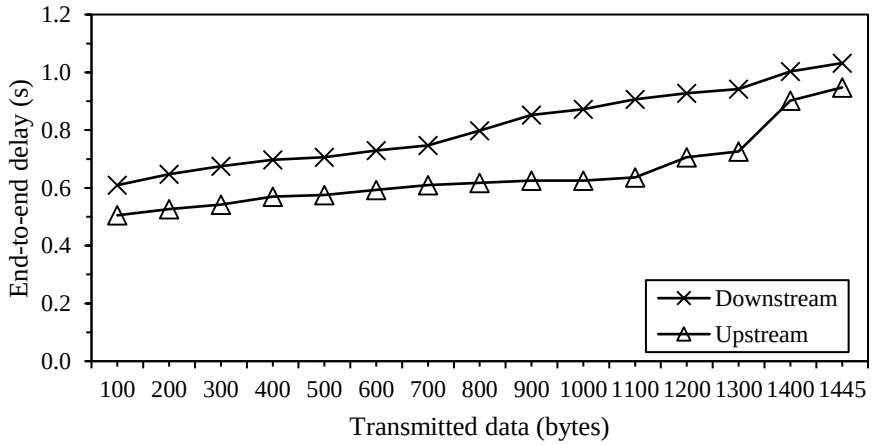

Figure 4.23 Fi-WiFi end-to-end delay

4.6.2 Wireless-Fiber-Wireless Performance Test

Figure 4.24 shows the downstream and upstream throughput respectively in Wi-FiWi network. The trend of the downstream throughput increases as the transmitted data size increases from 2.833 *kbps* at 100 *bytes* to 38.09 *kbps* at 1445 *bytes*. While, the upstream throughput increases as the transmitted data increases from 2.81 *kbps* at 100 *bytes* to 38.348 *kbps* at 1445 *bytes*.

At 100 *bytes*, the throughput for upstream transmission is lower than downstream transmission. However, as the transmitted data increases, the throughput for upstream transmission is higher than downstream transmission. For example, at 1100 *bytes*, the throughput for upstream transmission is 29.754 *kbps*, while the throughput for downstream transmission is 29.941 *kbps*. However, starting 1200 *bytes*, the throughput for upstream transmission is 32.276 *kbps*, while the throughput for downstream

transmission is 32.087 *kbps*. At 1445 *bytes*, the performance for both downstream and upstream transmissions are slightly decreased based on the slope of the graph at this point. This is due to the proposed router has reached its limit. Despite some differences in throughput values, the throughput for downstream and upstream do not have much difference proving that the proposed router in Wi-FiWi is stable for both transmissions. Therefore, it can be concluded that this proposed router is suitable to be a scalable testbed.

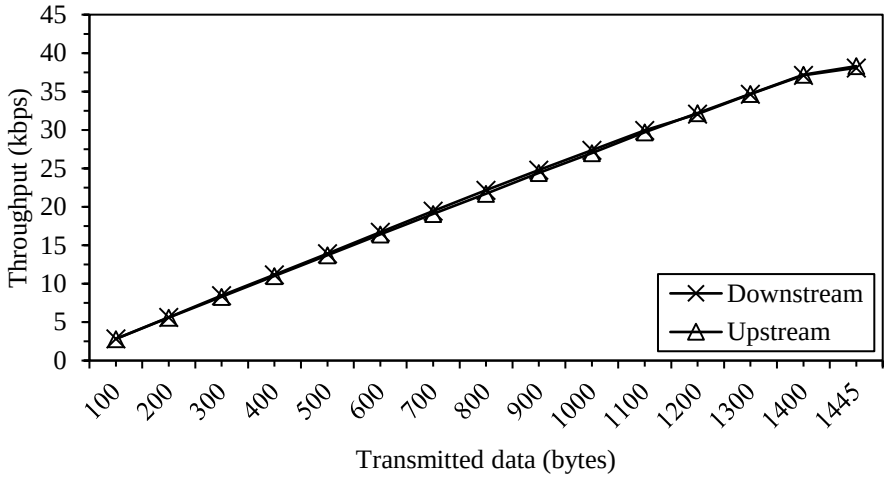

Figure 4.24 Wi-FiWi throughput

Figure 4.25 shows the end-to-end delay of downstream and upstream transmissions respectively for Wi-FiWi network. The trend of the downstream end-to-end delay increases as the transmitted data size increases from 0.565 *s* at 100 *bytes* to 0.607 *s* at 1445 *bytes*. In Figure 4.25, there is a sudden increase to the downstream end-to-end delay at 1200 *bytes*. This is due to the proposed router is already approaching its limit. Hence, the data processing time takes longer at this point. Meanwhile, the trend of the uupstream end-to-end delay increases as the transmitted data increases from 0.569 *s* at 100 *bytes* to 0.603 *s* at 1445 *bytes*. The downstream and upstream end-to-end delay for Wi-FiWi network is also higher than wireless, fiber and FiWi network because there are more medium changes in between clients. Therefore, the process of medium conversion contributes to higher end-to-end delay.

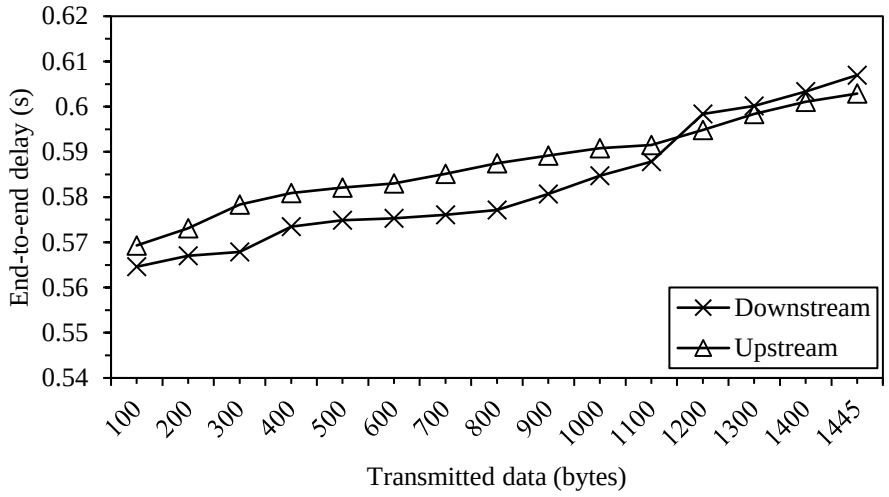

Figure 4.25 Wi-FiWi end-to-end delay

4.6.3 Fi-WiFi and Wi-FiWi performance comparison

Figure 4.26 and Figure 4.27 show the comparisons in terms of throughput between Fi-WiFi and Wi-FiWi for downstream and upstream transmissions from 100 *bytes* to 1445 *bytes*. Both graphs show that the performance for Wi-FiWi is more stable than Fi-WiFi based on the behaviour of the graphs. This is because Fi-WiFi has more conversion from electrical pulses to light pulses and vice versa compared to Wi-FiWi causing instability in Fi-WiFi. Therefore, we can conclude that, the overall performance in Wi-FiWi is more stable because it has better observable graphs compared to Fi-WiFi.

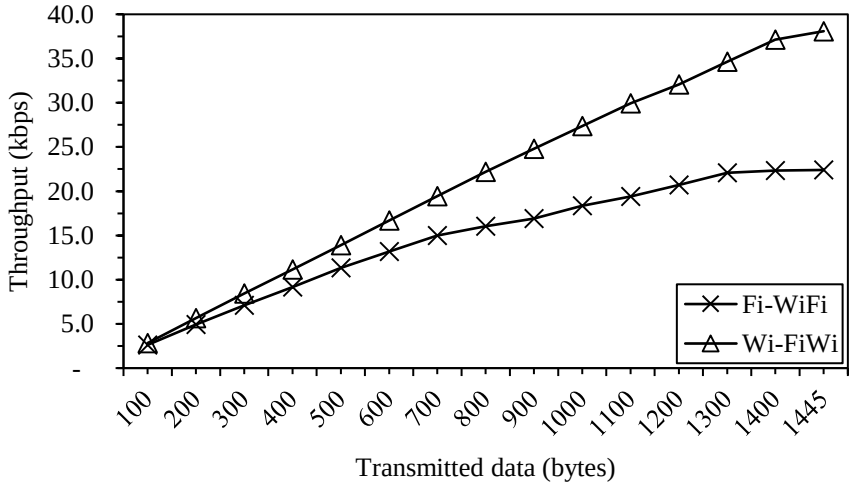

Figure 4.26 Fi-WiFi vs Wi-FiWi downstream throughput

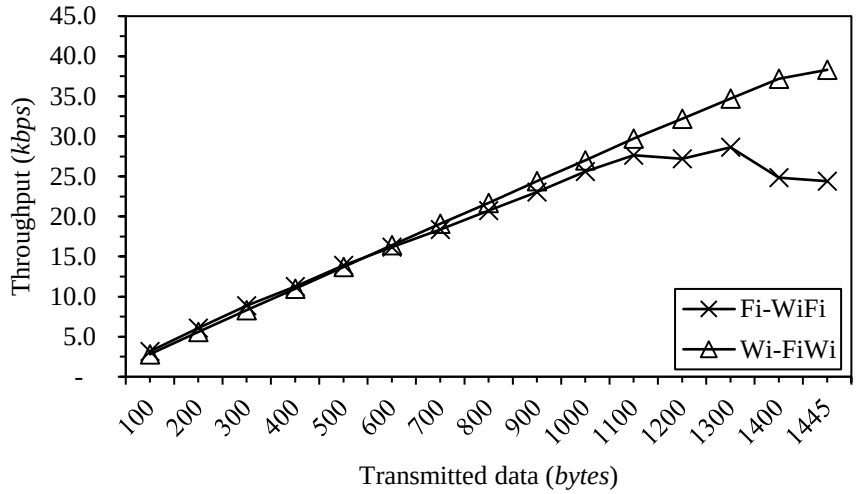

Figure 4.27 Fi-WiFi and Wi-FiWi upstream throughput

Figure 4.28 and Figure 4.29 show the end-to-end delay comparisons between Fi-WiFi and Wi-FiWi for downstream and upstream transmissions. Both of them have increasing end-to-end delay when the transmitted data size increased from 100 *bytes* to 1445 *bytes*. However, Fi-WiFi architecture has higher delay than Wi-FiWi. This is because in Fi-WiFi, there are more FMCs compared to Wi-FiWi. Hence, the medium conversions from light pulses to electrical pulses or vice versa in Fi-WiFi is more than Wi-FiWi. Thus, it contributes more delay. From the graphs, we can conclude that Wi-FiWi has more stable transmission because throughout the transmissions, the end-to-end delay do not change much.

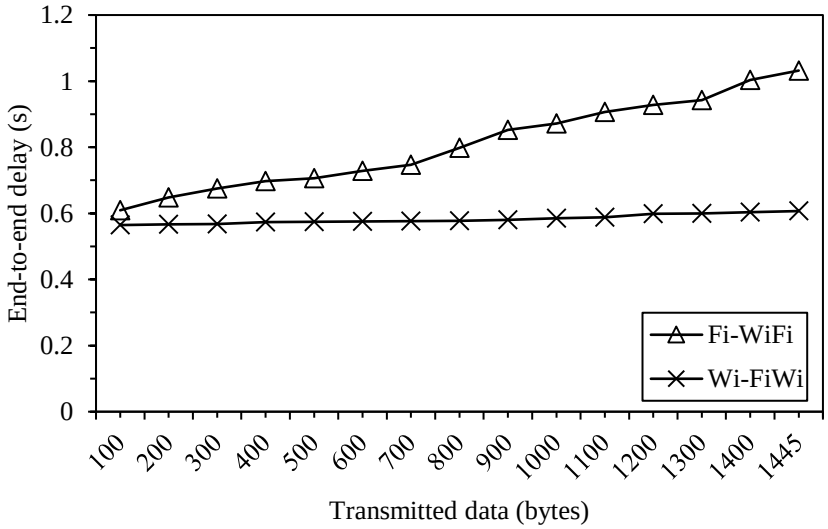

Figure 4.28 Fi-WiFi vs Wi-FiWi downstream end-to-end delay

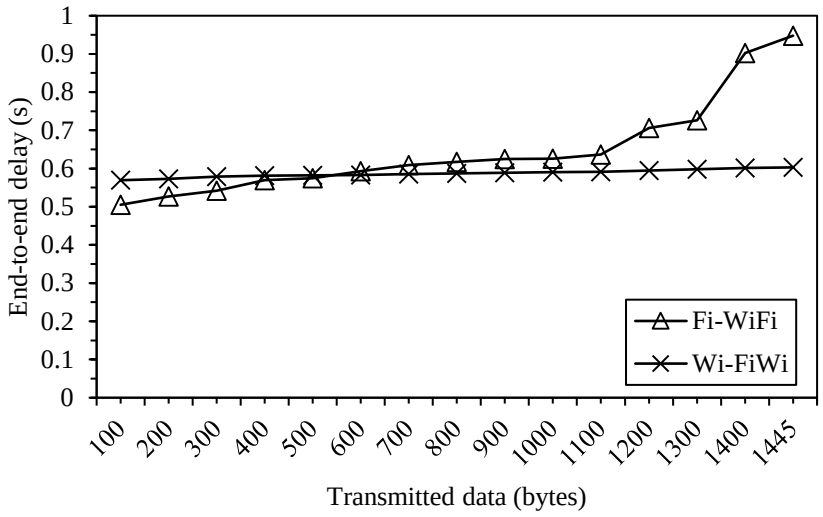

Figure 4.29 Fi-WiFi vs Wi-FiWi upstream end-to-end delay

4.7 Summary

In this chapter, the testbed's performance using wireless, fiber, and FiWi are presented. FiWi stress test and scalability test are also presented to prove that the proposed FiWi testbed is functioning as expected. The results in terms of throughput for wireless transmission, fiber transmission and FiWi transmission are validated with off-the-shelf router and industrial grade router to prove its the correctness. The end-to-end delay and throughput in wireless, fiber and FiWi transmission increase as the data size increases. The end-to-end delay of the proposed router is acceptable because its behaviour satisfying the behaviour of end-to-end delay of IEEE 802.15.4 routing scheme. However, as the proposed router approaches its limit, the throughput is maintained. The jitter varies within acceptable range which is 30 *ms* for downstream and upstream as the data is transmitted. Hence, the proposed router is suitable to be a simple, reconfigurable, low cost and fast implementation testbed. A stress test is conducted for FiWi transmission. The results show that the end-to-end delay for two traffics is twice as high as single traffic due to the proposed router needs to process the data twice. Hence, causing the throughput for two traffics is halved of the throughput of single traffics. The jitter of two traffics is also higher than a single traffic. However, the jitter is still within the acceptable range. Hence, making the proposed router suitable to be a scalable router. For scalability performance test, the results for Fi-WiFi and Wi-FiWi are presented in terms of end-to-end delay and throughput. The results show that the end-to-end delay in Fi-WiFi is higher than Wi-

FiWi due to more electrical pulses to light pulses conversion and vice versa from FMC in Fi-WiFi. Furthermore, Wi-FiWi is more stable compared to Fi-WiFi. With that, after the results are validated, it is proven that the proposed testbed is suitable to be a simple, reconfigurable, low cost, and fast implementation wireless, fiber and FiWi router. It is also suitable to be implemented in various setups proving that the testbed is scalable. The next chapter will discuss on future works and conclusion.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

FiWi is seen to be one of the best technologies for future global data communication network architecture due to the advantages of providing robustness and mobility to the consumers by deploying fiber and wireless in one network. Therefore, consumers can have better Internet connection and services. Hence, FiWi network is able to provide a promising solution for future networking technologies. From the literature, it can be concluded that there are still ongoing research in order to enhance FiWi current technology either only a part of FiWi such as fiber and wireless or the whole architecture. Due to that reasons, researchers are motivated to conduct intensive experiments by developing lab-scale testbed and industrial-scale testbed. The focus of this thesis is on lab-scale environment router testbed in FiWi architecture because to provide a proof-of-concept solution on reconfigurable router testbed in FiWi network. Three types of router testbeds exist; software-based router, commercial routers and embedded system-based routers. Embedded system-based router is the most appropriate choice for this thesis due to the open source kernel, reconfigurability, scalability, space-friendly and cost efficient.

Raspberry Pi is chosen to be the embedded system hardware in this project in order to develop a scalable and reconfigurable FiWi routers. It also has a module called socket which makes this project feasible. The module enables the user to program Raspberry Pi to communicate with each other by using IP addresses. The architecture of the testbed is in tree topology because it is the typical architecture in FiWi. The setup comprises of four Raspberry Pi routers; one acts as source router and the other three as destinations. Each router comprises of four Raspberry Pis; one Header Pi and three Forwarding Pi, which are connected via two Ethernet switches to represent the internal connection of the router and external connection to other routers. Whereas for wireless router, an additional AP is used as the antennae of the router. The role Header Pi in the router is to check the destination decided by the sender and insert a label onto the

data that represents the destination. Then, the labelled data is forwarded to Forwarding Pi so that the data can be forwarded to the destination. Header Pi also checks the incoming data whether the data belongs to the correct router or not by comparing the labels on the data with its self-label. Since there are no specific port for fiber on the Raspberry Pi, FMCs are used to interconnect one router with the other via fiber patch cord. After the FiWi setup is complete, the performance test of the testbed for wireless transmission, fiber transmission and FiWi transmission are tested in terms of throughput, end-to-end delay and jitter for upstream and downstream transmissions. By using the same design and performance parameters, the stress test is also conducted on the testbed by sending two traffics simultaneously. Then, the architecture of the testbed is reconfigured to Fi-WiFi and Wi-FiWi to test the router's scalability. The program of each router also needs to be reconfigured in terms of labels so that it works as intended. The performance of Fi-WiFi and Wi-FiWi are tested in terms of end-to-end delay and throughput.

In wireless transmission, the throughput of the proposed router is scaled up with the throughput of the off-the-shelf router. Whereas in fiber and FiWi transmissions, the throughput of the proposed router is scaled up with the throughput of industrial grade routers. This scaling method is done to check the proposed router functionality and to observe its correctness. The results show that, after the scaling method, the throughput of the proposed router has similar increasing trend with throughput of the off-the shelf router and industrial grade router, where, as the offered load increases, the throughput increases. This proves that the proposed router behaves correctly as intended. For end-to-end delay, the behaviour of the results for all transmissions are verified with IEEE 802.15.4 routing scheme, where, as the data size increases, the end-to-end delay increases. This behaviour is due to the proposed router requires more processing time as the data size increases. As for the jitter, the results for each transmission is verified with Cisco, where, the jitter values for proposed routers are below 30 *ms*, which is still within the acceptable range. In FiWi network, the proposed router is able to achieve maximum jitter of 8.25 *ms* for downstream and 0.11 *ms* for upstream. Hence, the proposed router is suitable to be wireless, fiber and FiWi router.

In stress test, the end-to-end delay of the proposed router is increasing as the data size increases. However, the proposed router shows that it requires the double amount of end-to-end delay compared to a single traffic. As for the throughput, it shows that the throughput is halved compared to a single traffic. This is due to the proposed router requires to process the data twice compared to a single traffic. Meanwhile, the jitter shows that two traffics have higher values compared to a single traffic. However, despite having higher jitter, it is still within the acceptable range which is below 30 *ms* which is 11.22 *ms* for downstream and 0.21 *ms* for upstream. Hence, this process that the proposed router is scalable.

In Fi-WiFi and Wi-FiWi architectures, the performance of the proposed router is tested in order to check its scalability and stability. The results show that the end-to-end delay for both architectures are increasing as the data size increases. However, at a certain point, the values become unstable. This is due to the proposed router is reaching its limit as the data size increases. The throughput of the proposed router in these architectures show an increasing trend as the data size increases. However, at a certain point, the throughput decreases because the proposed router has reached the limit. The end-to-end delay results for these two architectures are compared in order to check which architecture has more stability. The results show that Wi-FiWi has more stability compared to Fi-WiFi because it has lesser optical to electrical conversions and vice versa. This can be proven by observing the consistent trend of the end-to-end delay in Wi-FiWi. Hence, this proves that the proposed router is not only scalable, but also flexible and stable. The summary of performance for FiWi, Fi-WiFi and Wi-FiWi are tabulated in Table 5.1.

Table 5.1 Results summary

Topology	**Parameters**		
	Throughput	**End-to-end delay**	**Maximum jitter (< 30 *ms*)**
Wireless	- 677 *kbps* at 100% offered load (downstream) - 752 *kbps* at 100% offered load (upstream)	- 0.19 *s* at 1445 *bytes* (downstream) - 0.19 *s* at 1445 *bytes* (upstream)	- 9.8 *ms* (downstream) - 6.3 *ms* (upstream)
Fiber	- 767 *Mbps* at 100% offered load (downstream) - 785 *Mbps* at 100% offered load (upstream)	- 0.126 *s* at 1445 *bytes* (downstream) - 0.123 *s* at 1445 *bytes* (upstream)	- 0.38 *ms* (downstream) - 0.49 *ms* (upstream)
FiWi	- 719 *kbps* at 100% offered load (downstream) - 770 *kbps* (upstream) at 100% offered load	- 0.144 *s* at 1445 *bytes* (downstream) - 0.124 s at 1445 *bytes* (upstream)	- 8.25 *ms* (downstream) - 0.11 *ms* (upstream)
FiWi stress test	- 39.45 *kbps* at 1445 bytes (downstream) - 46.50 *kbps* at 1445 bytes (upstream)	- 0.29 *s* at 1445 *bytes* (downstream) - 0.25 *s* at 1445 *bytes* (upstream)	- 11.22 *ms* (downstream) - 0.21 *ms* (upstream)
Fi-WiFi	- 22.403 *kbps* at 1445 *bytes* (downstream) - 24.4 *kbps* at 1445 *bytes* (upstream)	- 1.03 *s* at 1445 *bytes* (downstream) - 0.948 *s* at 1445 *bytes* (upstream)	-
Wi-FiWi	- 38.09 *kbps* at 1445 bytes (downstream) - 38.35 *kbps* at 1445 *bytes* (upstream)	- 0.61 *s* at 1445 *bytes* (downstream) - 0.60 *s* at 1445 *bytes* (upstream)	-

Overall, a working reprogrammable, fast reconfigurable and scalable educational FiWi router testbed has been developed by using Raspberry Pi in a lab-scale environment. In wireless, fiber and FiWi architecture, the proposed router is proven to work correctly for downstream and upstream based on the scaling method. Its end-to-end delay is acceptable because the increasing trend is satisfying the IEEE 802.15.4 routing scheme. The jitter of the proposed router is also acceptable because the values are within an acceptable range produced by Cisco which is below 30 *ms*. In conclusion, it is proven that Raspberry Pi can be used to build a reconfigurable, flexible, and scalable educational FiWi router testbed. The proposed router also shows a promising stability in order to test various architectures.

5.2 Future Work and Recommendations

The development of the FiWi testbed using Raspberry Pi has been deliberated in this thesis. The future works are as follows:

- The overall throughput of the proposed testbed is small. There are better programmable embedded system hardware that can provide greater performance such as Raspberry Pi 4 which has just been released in June 2019. It is better than Raspberry Pi used in this project has Raspberry Pi 4 as it has better processor, RAM, and Ethernet capacity. In terms of processor, it uses Broadcom BCM2711 which is more powerful because it can process input and output data faster. Hence, more data can be processed at the same time which makes future proposed testbed to be more scalable than the current one. The processor also has metal cover which provides better heat dissipation. According to one of its website, Raspberry Pi 4 also has True Gigabit Ethernet, which means, the throughput produced is close to 1 *Gbps* [105]. In terms of RAM, Raspberry Pi 4 has maximum RAM capacity up to 4 *GB* which is four times bigger than current Raspberry Pi in proposed testbed. This makes the proposed testbed in the future to be able to handle more data at once.
- Current proposed router is using a low-cost AP to provide a reconfigurable FiWi testbed in a lab-scale environment. Hence, it is more than sufficient to use current AP as a proof-of-concept first. Then, as for future work, a better AP that supports dual-band with greater bandwidth and wider coverage such as TP-LINK WR1043ND can be used, so that more data can be transmitted simultaneously through wireless transmission and it is more stable than current AP in the proposed router. Hence, future proposed testbed can be implemented in wider scale area to get better results.
- Current proposed testbed has no protection for data transmission as it is only for proof-of-concept. Hence, for future work, physical link redundancy can be installed on the testbed to provide a better protection for data transmissions in case of broken link.

REFERENCES

[1] Y. Yu, C. Ranaweera, C. Lim, L. Guo, Y. Liu, A. Nirmalathas, *et al.* (2017, June). Hybrid Fiber-Wireless Network: An Optimization Framework For Survivable Deployment. Journal of Optical Communications and Networking. *9(6)*, pp. 466-478.

[2] B. P. Rimal, M. Maier, and M. Satyanarayanan. (2018, Aug.). Experimental Testbed for Edge Computing in Fiber-Wireless Broadband Access Networks. IEEE Communications Magazine. *56(8)*, pp. 160-167.

[3] V. Mishra, R. Upadhyay, and U. R. Bhatt, "A Review of Recent Energy-Efficient Mechanisms for Fiber-Wireless (FiWi) Access Network" in *Progress in Advanced Computing and Intelligent Engineering*, Springer, 2018, pp. 539-545.

[4] J. Liu, H. Guo, H. Nishiyama, H. Ujikawa, K. Suzuki, and N. Kato. (2015, Nov.). New Perspectives on Future Smart FiWi Networks: Scalability, Reliability, and Energy Efficiency. IEEE Communications Surveys & Tutorials. *18(2)*, pp. 1045-1072.

[5] Y. Liu, L. Guo, B. Gong, R. Ma, X. Gong, L. Zhang, *et al.* (2012, Mar.). Green Survivability in Fiber-Wireless (FiWi) Broadband Access Network. Optical Fiber Technology. *18(2)*, pp. 68-80.

[6] Q. Dai, G. Shou, Y. Hu, and Z. Guo, "A General Model for Hybrid Fiber-Wireless (FiWi) Access Network Virtualization," in 2013 IEEE International Conference on Communications Workshops (ICC), Hungary, 2013, pp. 858-862.

[7] H. Guo and J. Liu. (2018, Jan.). Collaborative Computation Offloading for Multi-Access Edge Computing Over Fiber–Wireless Networks. IEEE Transactions on Vehicular Technology. *67(5)*, pp. 4514-4526.

[8] Z. Zhang, J. Kong, C. Huang, Q. Wu, J. Wu, and J. Li, "Virtual Network Embedding Algorithm Considering Resource Fragmentation in Virtualized Industrial Fiber-Wireless (FiWi) Access Network," in Proceedings of the International Conference on Imaging, Signal Processing and Communication, Malaysia, 2017, pp. 148-152.

[9] H. Beyranvand, W. Lim, M. Maier, C. Verikoukis, and J. A. Salehi. (2015, Feb.). Backhaul-Aware User Association in FiWi Enhanced LTE-A Heterogeneous Networks. IEEE Transactions on Wireless Communications. *14(6)*, pp. 2992-3003.

[10] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb. (2018, June). Survey On Multi-Access Edge Computing For Internet of Things Realization. IEEE Communications Surveys & Tutorials. *20(4)*, pp. 2961-2991.

[11] W. Sun, J. Liu, and H. Zhang. (2017, June). When Smart Wearables Meet Intelligent Vehicles: Challenges and Future Directions. IEEE Wireless Communications. *24(3)*, pp. 58-65.

[12] B. P. Rimal, D. P. Van, and M. Maier. (2017, May). Mobile-Edge Computing Versus Centralized Cloud Computing Over A Converged FiWi Access Network. IEEE Transactions on Network and Service Management. *14(3)*, pp. 498-513.

[13] P.-Y. Chen and M. Reisslein. (2018, Apr.). FiWi Network Throughput-Delay Modeling with Traffic Intensity Control and Local Bandwidth Allocation. Optical Switching and Networking. *28(*pp. 8-22.

[14] B. P. Rimal, D. P. Van, and M. Maier. (2017, Feb.). Mobile Edge Computing Empowered Fiber-Wireless Access Networks in The 5G Era. IEEE Communications Magazine. *55(2)*, pp. 192-200.

[15] H.-H. Lu, C.-Y. Li, T.-C. Lu, C.-J. Wu, C.-A. Chu, A. Shiva, *et al.* (2016, Feb.). Bidirectional Fiber-Wireless and Fiber-VLLC Transmission System Based On An OEO-Based BLS and A RSOA. Optics Letters. *41(3)*, pp. 476-479.

[16] M. Tornatore, G.-K. Chang, and G. Ellinas, *Fiber-Wireless Convergence in Next-Generation Communication Networks.* Davis, USA, Springer, 2017, pp. 3-395.

[17] J. Liu, H. Guo, Z. M. Fadlullah, and N. Kato. (2016, Nov.). Energy Consumption Minimization for FiWi Enhanced LTE-A HetNets with UE Connection Constraint. IEEE Communications Magazine. *54(11)*, pp. 56-62.

[18] N. Choosri, Y. Park, S. Grudpan, P. Chuarjedton, and A. Ongvisesphaiboon. (2015, Mar.). IoT-RFID Testbed for Supporting Traffic Light Control.

International Journal of Information and Electronics Engineering. *5(2)*, pp. 102-106.

[19] W. Hurst, N. Shone, A. El Rhalibi, A. Happe, B. Kotze, and B. Duncan, "Advancing The Micro-CI Testbed for IoT Cyber-Security Research and Education," in The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, Greece, 2017, pp. 129-134.

[20] Z. Gong, W. Xue, Z. Liu, Y. Zhao, R. Miao, R. Ying, *et al.*, "Design of a Reconfigurable Multi-Sensor Testbed for Autonomous Vehicles and Ground Robots," in 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Japan, 2019, pp. 1-5.

[21] M. Ridwan, N. A. M. Radzi, F. Abdullah, N. M. Din, and C. Rashidi. (2018, Sep.). Feasibility Study of a Reconfigurable Fiber-Wireless Testbed Using Universal Software Radio Peripheral. International Journal of Engineering and Technology Innovation. *8(4)*, pp. 274-283.

[22] S. T. Abraha, D. F. Castellana, X. Liang, A. Ng'oma, and A. Kobyakov, "Experimental Study of Distributed Massive MIMO (DM-MIMO) in In-building Fiber-Wireless Networks," in 2018 Optical Fiber Communications Conference and Exposition (OFC), USA, 2018, pp. 1-3.

[23] M. Maier and N. Ghazisaidi, *FiWi Access Networks.* Cambridge university press, 2011, pp. 1-236.

[24] N. Ghazisaidi and M. Maier. (2011, Jan.). Fiber-wireless (FiWi) Access Networks: Challenges and Opportunities. IEEE network. *25(1)*, pp. 36-42.

[25] P. Singh and S. Prakash. (2017, Jul.). Optical Network Unit Placement In Fiber-Wireless (FiWi) Access Network by Moth-Flame Optimization Algorithm. Optical Fiber Technology. *36(*pp. 403-411.

[26] B. Kantarci, N. Naas, and H. T. Mouftah, "Energy-Efficient DBA and QoS in FiWi Networks Constrained To Metro-Access Convergence," in 2012 14th International Conference on Transparent Optical Networks (ICTON), UK, 2012, pp. 1-4.

[27] M. Lévesque, M. Maier, F. Aurzada, and M. Reisslein, "Analytical Framework for The Capacity and Delay Evaluation of Next-Generation FiWi Network Routing Algorithms," in 2013 IEEE Wireless Communications and Networking Conference (WCNC), China, 2013, pp. 1926-1931.

[28] S. Bindhaiq, A. S. M. Supa, N. Zulkifli, A. B. Mohammad, R. Q. Shaddad, M. A. Elmagzoub, *et al.* (2015, Aug.). Recent Development On Time And Wavelength-Division Multiplexed Passive Optical Network (TWDM-PON) For Next-Generation Passive Optical Network Stage 2 (NG-PON2). Optical Switching and Networking. *15(*pp. 53-66.

[29] S. S. Ahmed and M. M. Islam. (2018, Dec.). A Technical Review on Optical Access Networks. Nonlinear Dynamics. *6(2)*, pp. 79-95.

[30] D. Nesset. (2015, Dec.). NG-PON2 Technology and Standards. Journal of Lightwave Technology. *33(5)*, pp. 1136-1143.

[31] Y. Luo, X. Zhou, F. Effenberger, X. Yan, G. Peng, Y. Qian, *et al.* (2012, Feb.). Time and Wavelength-Division Multiplexed Passive Optical Network (TWDM-PON) for Next-Generation PON Stage 2 (NG-PON2). Journal of Lightwave Technology. *31(4)*, pp. 587-593.

[32] J. S. Wey, D. Nesset, M. Valvo, K. Grobe, H. Roberts, Y. Luo, *et al.* (2016, Jan.). Physical Layer Aspects of NG-PON2 Standards—Part 1: Optical Link Design. IEEE/OSA Journal of Optical Communications and Networking. *8(1)*, pp. 33-42.

[33] D. Iida, S. Kuwano, J.-i. Kani, and J. Terada. (2013, Oct.). Dynamic TWDM-PON for Mobile Radio Access Networks. Optics Express. *21(22)*, pp. 1-10.

[34] S. Rajpal and R. Goyal. (2017, June). A Review On Radio-over-Fiber Technology-based Integrated (Optical/Wireless) Networks. Journal of Optical Communications. *38(1)*, pp. 19-25.

[35] D. P. Van, B. P. Rimal, M. Maier, and L. Valcarenghi. (2016, Feb.). ECO-FiWi: An Energy Conservation Scheme for Integrated Fiber-Wireless Access Networks. IEEE Transactions on Wireless Communications. *15(6)*, pp. 3979-3994.

[36] K. Ahmavaara, H. Haverinen, and R. Pichna. (2003, Nov.). Interworking Architecture Between 3GPP and WLAN Systems. IEEE Communications Magazine. *41(11)*, pp. 74-81.

[37] M. Youssef and A. Agrawala, "The Horus WLAN Location Determination System," in The 3rd International Conference on Mobile Systems, Applications, and Services, Washington, 2005, pp. 205-218.

[38] Y. Mekonnen, M. Haque, I. Parvez, A. Moghadasi, and A. Sarwat, "LTE and Wi-Fi Coexistence in Unlicensed Spectrum with Application to Smart Grid: A Review," in 2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D), USA, 2018, pp. 1-5.

[39] S. Radhakrishnan, S. Neduncheliyan, and K. K. Thyagharajan. (2016, Jan.). A Review of Downlink Packet Scheduling Algorithms for Real Time Traffic in LTE-Advanced Networks. Indian Journal of Science and technology. *9(4)*, pp. 1-5.

[40] M. A. Gadam, M. A. Ahmed, C. K. Ng, N. K. Nordin, A. Sali, and F. Hashim. (2016, Mar.). Review of Adaptive Cell Selection Techniques in LTE-Advanced Heterogeneous Networks. Journal of Computer Networks and Communications. *2016(3)*, pp. 1-12.

[41] S. Sakamoto, R. Obukata, T. Oda, L. Barolli, M. Ikeda, and A. Barolli. (2017, Nov.). Performance Analysis of Two Wireless Mesh Network Architectures by WMN-SA and WMN-TS Simulation Systems. Journal of High Speed Networks. *23(4)*, pp. 311-322.

[42] Y. Liu, K.-F. Tong, X. Qiu, Y. Liu, and X. Ding, "Wireless Mesh Networks in IoT Networks," in 2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition, UK, 2017, pp. 183-185.

[43] B. Singh and D. Singh. (2016, June). A Review on Advantages and Applications of Radio over Fiber System. International Journal of Current Engineering and Technology. *6(3)*, pp. 1042-1044.

[44] S. R. A. Sharma and S. Rana. (2017, Jul.). Comprehensive Study of Radio Over Fiber With Different Modulation Techniques–A Review. International Journal of Computer Applications. *170(4)*, pp. 22-25.

[45] D. Novak, R. B. Waterhouse, A. Nirmalathas, C. Lim, P. A. Gamage, T. R. Clark, *et al.* (2015, Nov.). Radio-over-Fiber Technologies for Emerging Wireless Systems. IEEE Journal of Quantum Electronics. *52(1)*, pp. 1-11.

[46] C. Lim, Y. Tian, C. Ranaweera, T. A. Nirmalathas, E. Wong, and K.-L. Lee. (2018, Oct.). Evolution of Radio-Over-Fiber Technology. Journal of Lightwave Technology. *37(6)*, pp. 1647-1656.

[47] U. R. Bhatt, A. Chhabra, and R. Upadhyay, "Fiber-Wireless (Fi-Wi) Architectural Technologies: A Survey," in 2016 International Conference on

Electrical, Electronics, and Optimization Techniques (ICEEOT), India, 2016, pp. 519-524.

[48] G. Kalfas, N. Pleros, L. Alonso, and C. Verikoukis. (2016, Apr.). Network Planning for 802.11ad and MT-MAC 60 GHz Fiber-Wireless Gigabit Wireless Local Area Networks Over Passive Optical Networks. Journal of Optical Communications and Networking. *8(4)*, pp. 206-220.

[49] D. P. Van, B. P. Rimal, J. Chen, P. Monti, L. Wosinska, and M. Maier. (2016, Nov.). Power-Saving Methods for Internet of Things over Converged Fiber-Wireless Access Networks. IEEE Communications Magazine. *54(11)*, pp. 166-175.

[50] T. H. Szymanski and M. Rezaee, "An FPGA Controller for Deterministic Guaranteed-Rate Optical Packet Switching," in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Canada, 2015, pp. 1177-1183.

[51] H. Yang, J. Zhang, Y. Zhao, J. Wu, Y. Ji, Y. Lin, *et al.* (2016, Aug.). Experimental Demonstration of Remote Unified Control for OpenFlow-Based Software-Defined Optical Access Networks. Photonic Network Communications. *31(3)*, pp. 568-577.

[52] H. Yang, J. Zhang, Y. Zhao, J. Han, Y. Lin, and Y. Lee. (2016, Feb.). SUDOI: Software Defined Networking for Ubiquitous Data Center Optical Interconnection. IEEE Communications Magazine. *54(2)*, pp. 86-95.

[53] S. Okamoto, T. Sato, and N. Yamanaka, "Logical Optical Line Terminal Technologies Towards Flexible And Highly Reliable Metro And Access-Integrated Networks," in Optical Metro Networks and Short-Haul Systems IX, United States, 2017, pp. 1-15.

[54] R. S Luis, H. Furukawa, G. Rademacher, B. J Puttnam, and N. Wada, "Demonstration of an SDM Network Testbed for Joint Spatial Circuit and Packet Switching," in 2017 European Conference on Optical Communication (ECOC), Sweden, 2018, pp. 1-3.

[55] J. Azofra, N. Merayo, J. C. Aguado, I. De Miguel, R. Durán, F. Ruíz, *et al.*, "Implementation of A Testbed to Analysis A SDN Based GPON," in 2018 European Conference on Optical Communication (ECOC), Italy, 2018, pp. 1-3.

[56] M. S. Singh and V. Talasila, "A Practical Evaluation for Routing Performance of BATMAN-ADV And HWMN in A Wireless Mesh Network Testbed," in 2015 International Conference on Smart Sensors and Systems (IC-SSS), India, 2015, pp. 1-6.

[57] A. R. Prusty, S. Sethi, and A. K. Nayak, "Testbed for Link Quality Assessment in Wireless Ad-hoc Sensor Network," in 2016 International Conference on Computing, Analytics and Security Trends (CAST), India, 2016, pp. 329-334.

[58] A. Barolli, T. Oda, L. Barolli, and M. Takizawa, "Experimental Results of A Raspberry Pi and OLSR-Based Wireless Content Centric Network Testbed Considering OpenWRT OS," in 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), Switzerland, 2016, pp. 95-100.

[59] P. Zhang, O. Landsiedel, and O. Theel, "MOR: Multichannel Opportunistic Routing for Wireless Sensor Networks," in Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks, Sweden, 2017, pp. 36-47.

[60] E. Jecan, C. Pop, Z. Padrah, O. Ratiu, and E. Puschita, "A Dual-Standard Solution for Industrial Wireless Sensor Network Deployment: Experimental Testbed and Performance Evaluation," in 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Italy, 2018, pp. 1-9.

[61] F. Pakzad, M. Portmann, T. Turletti, T. Parmentelat, M. N. Mahfoudi, and W. Dabbous, "R2Lab Testbed Evaluation for Wireless Mesh Network Experiments," in 2018 28th International Telecommunication Networks and Applications Conference, ITNAC 2018, Australia, 2019, pp. 1-6.

[62] K.-K. Nguyen and M. Cheriet. (2016, Dec.). Virtual Edge-Based Smart Community Network Management. IEEE Internet Computing. *20(6)*, pp. 32-41.

[63] M. Xu, J. Zhang, F. Lu, J. Wang, L. Cheng, H. J. Cho, *et al.* (2016, June). FBMC in Next-Generation Mobile Fronthaul Networks With Centralized Pre-Equalization. IEEE Photonics Technology Letters. *28(18)*, pp. 1912-1915.

[64] B. P. Rimal, D. P. Van, and M. Maier. (2017, Mar.). Cloudlet Enhanced Fiber-Wireless Access Networks for Mobile-Edge Computing. IEEE Transactions on Wireless Communications. *16(6)*, pp. 3601-3618.

[65] J. Liu, G. Shou, Y. Liu, Y. Hu, and Z. Guo. (2018, May). Performance Evaluation of Integrated Multi-Access Edge Computing And Fiber-Wireless Access Networks. IEEE Access. *6(*pp. 30269-30279.

[66] Y. Turk and E. Zeydan. (2018, Oct.). An Experimental Measurement Analysis of Congestion Over Converged Fixed and Mobile Networks. Wireless Networks. pp. 1-16.

[67] Y. Alfadhli, Y.-W. Chen, S. Liu, S. Shen, S. Yao, D. Guidotti, *et al.* (2019, Oct.). Latency Performance Analysis of Low Layers Function Split for URLLC Applications in 5G Networks. Computer Networks. *162(*pp. 1-7.

[68] M. Bahnasy, K. Idoudi, and H. Elbiaze. (2015, Apr.). OpenFlow and GMPLS Unified Control Planes: Testbed Implementation And Comparative Study. Journal of Optical Communications and Networking. *7(4)*, pp. 301-313.

[69] T. M. Runge, D. Raumer, F. Wohlfart, B. E. Wolfinger, and G. Carle. (2015, Apr.). Towards Low Latency Software Routers. JNW. *10(4)*, pp. 188-200.

[70] S. Rampfl, "Network Simulation and Its Limitations," in Proceeding zum Seminar Future Internet (FI), Innovative Internet Technologien und Mobilkommunikation (IITM) und Autonomous Communication Networks (ACN), 2013, pp.

[71] R. G. Addie and J. P. R. Natarajan, "Netml-NS3-Click: Modeling of Routers in Netml/NS3 By Means of The Click Modular Router," in SimuTools, Greece, 2015, pp. 293-295.

[72] P. L. Suresh and R. Merz, "NS-3-Click: Click Modular Router Integration For NS3," in Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques, Belgium, 2011, pp. 423-430.

[73] P. Goswami, S. K. Ghosh, and D. Datta. (2015, Feb.). On Methodologies to Estimate Optical-Layer Power Consumption and Cost For Long-Haul WDM Networks With Optical Reach Constraint. Photonic Network Communications. *29(1)*, pp. 12-31.

[74] K. Ohsugi, J. Takemasa, Y. Koizumi, T. Hasegawa, and I. Psaras. (2016, May). Power Consumption Model of NDN-based Multicore Software Router Based on Detailed Protocol Analysis. IEEE Journal on Selected Areas in Communications. *34(5)*, pp. 1631-1644.

[75] L. Xu, K. Xu, M. Shen, K. Ren, J. Fan, C. Guan, *et al.*, "MINOS: Regulating Router Dataplane Actions In Dynamic Runtime Environments," in Proceedings of the ACM Turing 50th Celebration Conference, China, 2017, pp. 1-10.

[76] M. A. Kourtis, G. Xilouris, V. Riccobene, M. J. McGrath, G. Petralia, H. Koumaras, *et al.*, "Enhancing VNF Performance By Exploiting SR-IOV And DPDK Packet Processing Acceleration," in 2015 IEEE Conference on Network Function Virtualization and Software Defined Network, NFV-SDN 2015, USA, 2016, pp. 74-78.

[77] R. Rajesh, K. B. Ramia, and M. Kulkarni, "Integration of LwIP Stack Over Intel (R) DPDK for High Throughput Packet Delivery to Applications," in 2014 Fifth International Symposium on Electronic System Design, 2014, pp. 130-134.

[78] M. M. Tajiki, B. Akbari, N. Mokari, and L. Chiaraviglio. (2018, Aug.). SDN-based Resource Allocation In MPLS Networks: A Hybrid Approach. Concurrency and Computation Practice and Experience. *31(8),* pp. 1-13.

[79] J. I. Kim, N. J. Choi, T. W. You, H. Jung, Y. W. Kwon, and S. J. Koh. (2019, Mar.). Mobile-oriented Future Internet: Implementation And Experimentations Over EU–Korea Testbed. Electronics (Switzerland). *8(3),* pp. 1-24.

[80] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. (2000, Aug.). The Click Modular Router. ACM Transactions on Computer Systems (TOCS). *18(3),* pp. 263-297.

[81] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. (1999, Dec.). The Click Modular Router. ACM SIGOPS Operating Systems Review. *33(5),* pp. 217-231.

[82] S. M. Blair, F. Coffele, C. Booth, B. De Valck, and D. Verhulst. (2014, Aug.). Demonstration And Analysis of IP/MPLS Communications for Delivering Power System Protection Solutions Ssing IEEE C37. 94, IEC 61850 Sampled Values, And IEC 61850 GOOSE Protocols. 2014 CIGRE Session. pp. 1-8.

[83] T. Feng, J. Bi, P. Xiao, and X. Zheng, "Hybrid SDN Architecture To Integrate With Legacy Control And Management Plane: An Experiences-based Study,"

in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Canada, 2015, pp. 754-757.

[84] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi. (2016, Jan.). Experimental Demonstration of Segment Routing. Journal of Lightwave Technology. *34(1)*, pp. 205-212.

[85] K. Tantayakul, R. Dhaou, B. Paillassa, and W. Panichpattanakul, "Experimental Analysis in SDN Open Source Environment," in 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Thailand, 2017, pp. 334-337.

[86] Z. Chen, H. Zou, J. Yang, H. Jiang, and L. Xie. (2019, June). WiFi Fingerprinting Indoor Localization Using Local Feature-Based Deep LSTM. IEEE Systems Journal. pp. 1-10.

[87] V. Sivaraman, A. Vishwanath, D. Ostry, and M. Thottan. (2016, Aug.). Greening Router Line-Cards via Dynamic Management of Packet Memory. IEEE Journal on Selected Areas in Communications. *34(12)*, pp. 3843-3853.

[88] C. H. Hoo and A. Kumar, "ParaDiMe: A Distributed Memory FPGA Router Based On Speculative Parallelism and Path Encoding," in 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), USA, 2017, pp. 172-179.

[89] C. Concatto, J. A. Pascual, J. Navaridas, J. Lant, A. Attwood, M. Lujan, *et al.*, "A CAM-free Exascalable HPC Router For Low-Energy Communications," in International Conference on Architecture of Computing Systems, Germany, 2018, pp. 99-111.

[90] D. Posch, B. Rainer, S. Theuermann, A. Leibetseder, and H. Hellwagner, "Emulating NDN-based Multimedia Delivery," in Proceedings of the 7th International Conference on Multimedia Systems, Austria, 2016, pp. 1-4.

[91] B. Rainer, D. Posch, A. Leibetseder, S. Theuermann, and H. Hellwagner. (2016, Sep.). A Low-Cost NDN Testbed On Banana Pi Routers. IEEE Communications Magazine. *54(9)*, pp. 105-111.

[92] P. Lech and P. Włodarski, "IoT WiFi Home Network Stress Test," in International Conference on Image Processing and Communications, Poland, 2017, pp. 247-254.

[93] S. Y. Jang, B. H. Shin, and D. Lee, "Implementing a Dynamically Reconfigurable Wireless Mesh Network Testbed for Multi-Faceted QoS Support," in Proceedings of the 11th International Conference on Future Internet Technologies, China, 2016, pp. 95-98.

[94] X. Piao, L. Huang, K. Yuan, J. Yuan, and K. Lei, "The Real Implementation of NDN Forwarding Strategy On Android Smartphone," in 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), USA, 2016, pp. 1-6.

[95] V. Gupta, K. Kaur, and S. Kaur, "Developing Small Size Low-Cost Software-Defined Networking Switch Using Raspberry Pi," in 50th Annual Convention of Computer Society of India : Next-Generation Networks, India, 2018, pp. 147-152.

[96] S. Brown, "An Analysis of Loss-free Data Aggregation for High Data Reliability in Wireless Sensor Networks," in 2017 28th Irish Signals and Systems Conference (ISSC), Ireland, 2017, pp. 1-6.

[97] D-Link, Wireless N300 Access Point & Router - DAP 1360 Datasheet at ftp://ftp.dlink.ru/pub/Wireless/DAP-1360/Data_sh/DAP-1360_A_E1_DS_v.2.5.4_09.09.14_EN.pdf

[98] A. Minakhmetov, C. Ware, and L. Iannone, "Optical Networks Throughput Enhancement via TCP Stop-and-Wait on Hybrid Switches," in 2018 Optical Fiber Communications Conference and Exposition (OFC), San Diego, USA, 2018, pp. 1-3.

[99] E. Leão, C. Montez, R. Moraes, P. Portugal, and F. Vasques. (2017, May). Alternative Path Communication in Wide-Scale Cluster-Tree Wireless Sensor Networks Using Inactive Periods. Sensors. *17(5)*, pp. 1049.

[100] M. Ridwan, "A Re-Programmable Testbed for Fiber Wireless Network Using NI-Sofware Defined Radio," Master in Electrical Engineering, Universiti Tenaga Nasional, Malaysia, 2017.

[101] C. Hattingh and T. Szigeti, *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs.* Cisco Press, 2004, pp. 1-768.

[102] L. Kumar, A. Singh, and V. Sharma, "Convergence of Bidirectional PON with Single-Sink Wireless Sensor Network Using Queue Theory" in *Ambient Communications and Computer Systems*, Springer, 2018, pp. 249-259.

[103] H. Xu, X.-Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang. (2017, Feb.). Incremental Deployment and Throughput Maximization Routing for A Hybrid SDN. IEEE/ACM Transactions on Networking. *25(3)*, pp. 1861-1875.

[104] Q. T. Minh, T. K. Dang, T. Nam, and T. Kitahara. (2019, May). Flow Aggregation for SDN-based Delay-insensitive Traffic Control in Mobile Core Networks. IET Communications. *13(8)*, pp. 1051-1060.

[105] Cytron. (2019). *Raspberry Pi 4 Model B.* Available: https://my.cytron.io/p-raspberry-pi-4-model-b-4gb?gclid=EAIaIQobChMI4smA_7Lo5AIVkyQrCh3Xvw0dEAQYASABEgIHI_D_BwE

APPENDIX A
PROGRAM FOR CLIENT

```
import sys,socket,select

def client_pi():
        host = '169.254.249.122'
        port = 9009
        s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

        try:
                s.connect((host,port))
        except:
                print 'Unable to connect'
                sys.exit()
        print 'Connected to the network\n'
        while 1:
                print 'Press: \n1) Router A\n2) Router B\n3) Router C\n'
                x = input('Choose one destination: ')
                if x == 1:
                        print 'Your destination is Router A.\n'
                        sys.stdout.write('[Me] ');sys.stdout.flush()
                        while 1:
                                socket_list = [sys.stdin,s]
                                read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                                for sock in read_sockets:
                                        if sock == s:
                                                data = sock.recv(4096)
                                                #print (data)
                                                if not data:
                                                        print 'Disconnected from
network.\n'
                                                elif data[1:4] == '400':
                                                        msg = data[4:]
```

```
                                                    sys.stdout.write('Received:
'+msg+'\n');sys.stdout.flush()
                                                    sys.stdout.write('[Me]
');sys.stdout.flush()
                                            elif data[1:4] == '400':
                                                    msg = data[4:]
                                    sys.stdout.write('Received: '+msg+'\n');sys.stdout.flush()
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                            elif data[1:4] == '400':
                                    msg = data[4:]
                                    sys.stdout.write('Received: '+msg+'\n');sys.stdout.flush()
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                                            else:
                                                    data = '0\n'
                                                    sys.stdout.flush()
                                    else:
                                                    sys.stdout.write('[Me] '); sys.stdout.flush()
                                                    msg = sys.stdin.readline()
                                                    s.send(str(100) + msg)
            if x == 2:
                    print 'Your destination is Router B.\n'
                            print 'Which client do you want to receive the message?\n1) Wired
client only\n2) Wireless client 1 only\n3) Wireless client 2 only\n4) Wireless client 3
only\n5) Broadcast the message\n'
                            y = input('Your choice: ')
                    if y == 1:
                            print 'You wish to send to wired client.\n'
                            sys.stdout.write('[Me] ');sys.stdout.flush()
                            while 1:
                                    socket_list = [sys.stdin,s]
                                    read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                                    for sock in read_sockets:
                                            if sock == s:
                                                    data = sock.recv(4096)
                                                    if not data:
```

```
                print 'Disconnected from network.\n'
            elif data[1:4] == '400':
                if data[4:6] == '10':
                                        msg = data[6:]
                 sys.stdout.write('Received message for service 1:
'+msg+'\n')
                 sys.stdout.write('[Me] ');sys.stdout.flush()
                elif data[4:6] == '20':
                                        msg = data[6:]
                 sys.stdout.write('Received message for service 2:
'+msg+'\n')
                 sys.stdout.write('[Me] ');sys.stdout.flush()
                elif data[4:6] == '30':
                                        msg = data[6:]
                 sys.stdout.write('Received message for service 3:
'+msg+'\n')
                 sys.stdout.write('[Me] ');sys.stdout.flush()
                                    else:
                                        data = '0\n'
                                        sys.stdout.flush()
            else:
                data = '0\n'
                sys.stdout.flush()

        else:
            sys.stdout.write('[Me] '); sys.stdout.flush()
            msg = sys.stdin.readline()
                                print 'Mark your message as:\n1)
Service 1\n2) Service 2\n3) Service 3\n'
                                z = input('Your choice: ')
                                if z == 1:
                 s.send(str(21110) + msg)
                                elif z == 2:
                                    s.send(str(21120) + msg)
                                elif z == 3:
                                    s.send(str(21130) + msg)
```

```
                elif y == 2:
                        print 'You wish to send to wireless client 1.\n'
                        sys.stdout.write('[Me] ');sys.stdout.flush()
                        while 1:
                                socket_list = [sys.stdin,s]
                                read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                                for sock in read_sockets:
                                        if sock == s:
                                                data = sock.recv(4096)
                                                if not data:
                                                        print 'Disconnected from network.\n'
                                                elif data[1:4] == '400':
                                                        if data[4:6] == '10':
                                                                                msg = data[6:]
                                                         sys.stdout.write('Received message from service 1:
'+msg+'\n')
                                                         sys.stdout.write('[Me] ');sys.stdout.flush()
                                                        elif data[4:6] == '20':
                                                                                msg = data[6:]
                                                         sys.stdout.write('Received message from service 2:
'+msg+'\n')
                                                         sys.stdout.write('[Me] ');sys.stdout.flush()
                                                        elif data[4:6] == '30':
                                                                                msg = data[6:]
                                                         sys.stdout.write('Received from service 3:
'+msg+'\n')
                                                         sys.stdout.write('[Me] ');sys.stdout.flush()
                                                        else:
                                                                data = '0\n'
                                                                sys.stdout.flush()
                                                else:
                                                        data = '0\n'
                                                        sys.stdout.flush()

                                        else:
```

```
                                    sys.stdout.write('[Me] '); sys.stdout.flush()
                                    msg = sys.stdin.readline()
                                    print 'Mark your message as:\n1) Service 1\n2) Service
2\n3) Service 3\n'
                                    z = input('Your choice: ')
                                    if z == 1:
                                        s.send(str(22110) + msg)
                                    elif z == 2:
                                        s.send(str(22120) + msg)
                                    elif z == 3:
                                        s.send(str(22130) + msg)

            elif y == 3:
                print 'You wish to send to wireless client 2.\n'
                sys.stdout.write('[Me] ');sys.stdout.flush()
                while 1:
                    socket_list = [sys.stdin,s]
                    read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                    for sock in read_sockets:
                        if sock == s:
                            data = sock.recv(4096)
                            if not data:
                                print 'Disconnected from network.\n'
                            elif data[1:4] == '400':
                                if data[4:6] == '10':
                                                                msg = data[6:]
                                    sys.stdout.write('Received message from service 1:
'+msg+'\n')
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                                elif data[4:6] == '20':
                                                                msg = data[6:]
                                    sys.stdout.write('Received message from service 2:
'+msg+'\n')
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                                elif data[4:6] == '30':
```

```
                                                    msg = data[6:]
                                    sys.stdout.write('Received message from service 3:
'+msg+'\n')
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                                else:
                                    data = '0\n'
                                    sys.stdout.flush()
                            else:
                                data = '0\n'
                                sys.stdout.flush()

                        else:
                            sys.stdout.write('[Me] '); sys.stdout.flush()
                            msg = sys.stdin.readline()
                            print 'Mark your message as:\n1) Service 1\n2) Service
2\n3) Service 3\n'
                            z = input('Your choice: ')
                            if z == 1:
                                s.send(str(23110) + msg)
                            elif z == 2:
                                s.send(str(23120) + msg)
                            elif z == 3:
                                s.send(str(23130) + msg)

            if y == 4:
                print 'You wish to send to wireless client 3.\n'
                sys.stdout.write('[Me] ');sys.stdout.flush()
                while 1:
                    socket_list = [sys.stdin,s]
                    read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                    for sock in read_sockets:
                        if sock == s:
                            data = sock.recv(4096)
                            if not data:
                                print 'Disconnected from network.\n'
```

```
                elif data[1:4] == '400':
                    if data[4:6] == '10':
                                                msg = data[6:]
                        sys.stdout.write('Received from service 1:
'+msg+'\n')
                        sys.stdout.write('[Me] ');sys.stdout.flush()
                    elif data[4:6] == '20':
                                                msg = data[6:]
                        sys.stdout.write('Received from service 2:
'+msg+'\n')
                        sys.stdout.write('[Me] ');sys.stdout.flush()
                    elif data[4:6] == '30':
                                                msg = data[6:]
                        sys.stdout.write('Received from service 3:
'+msg+'\n')
                        sys.stdout.write('[Me] ');sys.stdout.flush()
                    else:
                        data = '0\n'
                        sys.stdout.flush()
                else:
                    data = '0\n'
                    sys.stdout.flush()

            else:
                sys.stdout.write('[Me] '); sys.stdout.flush()
                msg = sys.stdin.readline()
                print 'Mark your message as:\n1) Service 1\n2) Service
2\n3) Service 3\n'
                z = input('Your choice: ')
                if z == 1:
                    s.send(str(24110) + msg)
                elif z == 2:
                    s.send(str(24120) + msg)
                elif z == 3:
                    s.send(str(24130) + msg)
```

```
                elif y == 5:
                        print 'You wish to broadcast the message.\n'
                        sys.stdout.write('[Me] ');sys.stdout.flush()
        while 1:
                socket_list = [sys.stdin,s]
                read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                for sock in read_sockets:
                        if sock == s:
                        data = sock.recv(4096)
                        if not data:
                                print 'Disconnected from network.\n'
                        elif data[1:4] == '400':
                                msg = data[4:]
                                sys.stdout.write('Received: '+msg+'\n')
                                sys.stdout.write('[Me] ');sys.stdout.flush()
                        elif data[0:3] == '400':
                                msg = data[3:]
                                sys.stdout.write('Received: '+msg+'\n')
                                sys.stdout.write('[Me] ');sys.stdout.flush()
                        elif data[2:5] == '400':
                                msg = data[5:]
                                sys.stdout.write('Received: '+msg+'\n')
                                sys.stdout.write('[Me] ');sys.stdout.flush()
                        else:
                                data = '0\n'
                                sys.stdout.flush()


                        else:
                        sys.stdout.write('[Me] '); sys.stdout.flush()
                        msg = sys.stdin.readline()
                        s.send(str(200) + msg)
    if x == 3:
        print 'Your destination is Router C.\n'
                print 'Choose mode:\n1) FiWi\n2) FiWi-Fi\n'
                a = input('Your mode: ')
```

```
                    if a == 1:
                            print 'FiWi mode\n'
                            sys.stdout.write('[Me] ');sys.stdout.flush()
                while 1:
                    socket_list = [sys.stdin,s]
                    read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                    for sock in read_sockets:
                            if sock == s:
                            data = sock.recv(4096)
                            if not data:
                                    print 'Disconnected from network.\n'
                            elif data[1:4] == '400':
                                    msg = data[4:]
                                    sys.stdout.write('Received: '+msg+'\n')
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                            elif data[1:4] == '400':
                                    msg = data[3:]
                                    sys.stdout.write('Received: '+msg+'\n')
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                            elif data[2:5] == '400':
                                    msg = data[5:]
                                    sys.stdout.write('Received: '+msg+'\n')
                                    sys.stdout.write('[Me] ');sys.stdout.flush()
                            else:
                                    data = '0\n'
                                    sys.stdout.flush()


                            else:
                            sys.stdout.write('[Me] '); sys.stdout.flush()
                            msg = sys.stdin.readline()
                            s.send(str(300) + msg)
                if a == 2:
                            print 'FiWi-Fi mode\n'
                    sys.stdout.write('[Me] ');sys.stdout.flush()
                    while 1:
```

```
                socket_list = [sys.stdin,s]
                read_sockets,write_sockets,error_sockets =
select.select(socket_list,[],[])
                for sock in read_sockets:
                    if sock == s:
                        data = sock.recv(4096)
                        if not data:
                            print 'Disconnected from network.\n'
                        elif data[1:4] == '400':
                            msg = data[4:]
                            sys.stdout.write('Received: '+msg+'\n')
                            sys.stdout.write('[Me] ');sys.stdout.flush()
                        elif data[2:4] == '400':
                            msg = data[3:]
                            sys.stdout.write('Received: '+msg+'\n')
                            sys.stdout.write('[Me] ');sys.stdout.flush()
                        elif data[2:5] == '400':
                            msg = data[5:]
                            sys.stdout.write('Received: '+msg+'\n')
                            sys.stdout.write('[Me] ');sys.stdout.flush()
                        else:
                            data = '0\n'
                            sys.stdout.flush()

                    else:
                        sys.stdout.write('[Me] '); sys.stdout.flush()
                        msg = sys.stdin.readline()
                        s.send(str(320) + msg)

if __name__ == "__main__":
        sys.exit(client_pi())
```

APPENDIX B
PROGRAM FOR HEADER PI

```
import sys,socket,select

def header_pi():
        host = '169.254.249.122'
        port = 9009
        s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

        try:
                s.connect((host,port))
        except:
                print 'Unable to connect'
                sys.exit()
        print 'Connected to the network\n'
        while 1:
                socket_list = [sys.stdin,s]
                read_sockets,write_sockets,error_sockets = select.select(socket_list,[],[])
                for sock in read_sockets:
                        if sock == s:
                                data = sock.recv(4096)
                                #print (data)
                                if not data:
                                        print 'Disconnected from network.\n'
                                elif data[1:4] == '444':
                                        msg = data[4:]
                                        sock.send(str(400)+msg)
                                elif data[1:4] == '100':
                                        msg = data[4:]
                                        sock.send(str(101)+msg)
                                elif data[1:4] == '200':
                                        msg = data[4:]
                                        sock.send(str(202)+msg)
                                elif data[1:4] == ‘300’:
                                        msg = data[4:]
```

```
                    sock.send(str(303)+msg)
                else:
                    data = '0\n'
                    sys.stdout.flush()
if __name__ == "__main__":
        sys.exit(header_pi())
```

APPENDIX C
PROGRAM FOR FORWARDING PI

```
import sys,socket,select

def forwarding_pi():
        host = '169.254.249.122'
        port = 9009
        s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

        try:
                s.connect((host,port))
        except:
                print 'Unable to connect'
                sys.exit()
        print 'Connected to the network\n'
        while 1:
                socket_list = [sys.stdin,s]
                read_sockets,write_sockets,error_sockets = select.select(socket_list,[],[])
                for sock in read_sockets:
                        if sock == s:
                                data = sock.recv(4096)
                                #print (data)
                                if not data:
                                        print 'Disconnected from network.\n'
                                elif data[1:4] == '101':
                                        msg = data[4:]
                                        sock.send(str(111)+msg)
                                else:
                                        data = ‘0’
                                        sys.stdout.flush()
if __name__ == "__main__":
        sys.exit(forwarding_pi())
```

APPENDIX D

PROPOSED ROUTER DATASHEET

Key Features

1. Reprogrammable and reconfigurable
2. Scalable to Fi-WiFi and Wi-FiWi
3. Low power consumption

Connectivity

1. Multiple Ethernet connections.
2. Able to communicate wirelessly with other routers and devices.

Performance

1. Provides up to 700 *kbps* of throughput for FiWi transmissions at 100% load.
2. Jitter for fiber, wireless and FiWi transmissions are less than 30 *ms*.

Hardware specifications

Total maximum power input	5 *V*/10 *A*
RAM	1 *GB* Low Power Double Data Rate (LPDDR2)
Router dimension (Height x width x depth)	11 *cm* x 42 *cm* x 28 *cm*
Operating system	Raspbian
Ethernet ports	6 ports
Programming	Python
Protocols	TCP/IP, IPv4
Wireless characteristic	802.11b/g/n