Quality Of Service Design Of Openwifi At Telecom Telkom University Infra Project Lab - Provisioning Infrastructure

1st Gangan Hamamminata Jamalullail
Faculty of Electrical Engineering
Telkom University
Bandung, Indonesia
gangan@student.telkomuniversity.ac.id

2nd Dhoni Putra Setiawan
Faculty of Electrical Engineering
Telkom University
Bandung, Indonesia
setiawandhoni@telkomuniversity.ac.id

3rd Prananto Bayu Herlambang
Coordinator Telecom Infra Project Community Lab
Telkom University
Bandung, Indonesia
prananto.bayu@gcidesign.com

Abstract - This paper presents the results of a research study that investigated the feasibility of setting up and configuring OpenWiFi on a server. The research study was conducted in a controlled environment, and the results showed that OpenWiFi can be successfully set up and configured on a server. The physical layer configuration steps were straightforward and could be completed by following the instructions in the OpenWiFi documentation. The application layer configuration steps were more complex, but they were also well-documented and easy to follow. The future of OpenWiFi is promising. As an open-source project, OpenWiFi is constantly being developed and improved. The community of OpenWiFi users and developers is also growing, which will help to ensure that OpenWiFi remains a viable option for Wi-Fi QoS management. The limitations of this research are that it was conducted in a controlled environment. It is possible that the results would be different in a real-world setting. However, the methodology used in this research can be easily adapted to other environments. Overall, this research provides a valuable contribution to the field of Wi-Fi QoS management. The results of this research demonstrate that OpenWiFi is a viable option for Wi-Fi QoS management. The methodology used in this research can be easily adapted to other environments, and the future of OpenWiFi is promising.

Keywords—Wi-Fi, OpenWiFi, Docker

I. INTRODUCTION

OpenWiFi, an open-source and community-driven Wi-Fi software system, is revolutionizing the wireless technology landscape with its disruptive approach. This paper delves into the transformative impact of OpenWiFi on Quality of Service (QoS) management, emphasizing its innovative provisioning infrastructure. Before exploring the provisioning aspects, we must address the prerequisites: installation of Docker and OpenWiFi, along with Access Point configuration. By understanding these foundational steps, we can unveil how OpenWiFi empowers optimized network performance, enriched user experiences, and a more inclusive Wi-Fi development ecosystem. To validate its capabilities, testing with iPerf3 is employed, showcasing the potential of OpenWiFi in reshaping the future of wireless connectivity.

II. LITERATURE REVIEW

A. WiFi

Wi-Fi, short for Wireless Fidelity, is a wireless communication technology that allows devices to connect to the internet and local area networks without the need for physical cables. It utilizes radio waves to transmit data between devices, enabling seamless connectivity and mobility. The most used Wi-Fi standard for indoor environments is IEEE 802.11n, also known as Wi-Fi 4, which provides faster data transfer rates and increased range compared to its predecessors [1]. In recent years, the focus has shifted towards enhancing the Quality of Service (QoS) management in Wi-Fi networks. QoS management plays a crucial role in optimizing network performance by prioritizing certain types of data traffic, ensuring smoother video streaming, VoIP calls, and online gaming experiences [2]. Advanced QoS techniques like traffic shaping, packet prioritization, and band steering have been implemented to enhance the overall user experience in densely populated environments, where multiple devices compete for bandwidth. These efforts aim to deliver a stable, reliable, and efficient Wi-Fi connection, satisfying the demands of modern connected lifestyles.

B. OpenWiFi

OpenWiFi is an open-source, community-developed Wi-Fi software system that aims to break vendor lock-in and accelerate the innovation of Wi-Fi services. It is a disaggregated system, which means that the software and hardware components are decoupled, allowing for greater flexibility and choice [3]. OpenWiFi includes a cloud controller SDK and an enterprise-grade Access Point (AP) firmware, which are designed and validated to work seamlessly together [3].

C. Docker

Docker is a powerful containerization platform that allows developers to package applications and their dependencies into isolated containers. These containers operate independently and consistently across various environments, making it easier to deploy, manage, and scale applications. Docker works by utilizing OS-level virtualization to create lightweight containers that share the host system's kernel, enabling efficient resource utilization and swift deployment[4]. This approach eliminates compatibility issues and ensures applications run seamlessly, regardless of the underlying infrastructure, making Docker a game-changer in modern software development and deployment practices.

D. Access Point

An Access Point (AP) is a networking device that acts as a central hub for wireless communication within a local area network (LAN)[5]. It enables wireless devices such as laptops, smartphones, and IoT devices to connect to the network and gain internet access. The Access Point acts as a bridge between the wireless devices and the wired network, facilitating data transmission between them[6]. It uses radio frequency signals to communicate with wireless devices and converts data packets from wireless to wired format, allowing seamless connectivity. With its vital role in providing wireless connectivity, the Access Point is a fundamental component in modern networking, ensuring efficient and reliable wireless communication for various applications.

E. iPerf3

iPerf3 is a widely used open-source tool for testing network bandwidth and performance. It allows users to measure the data transfer rate between two endpoints in a network, providing valuable insights into network capacity and efficiency. iPerf3 works by establishing a client-server connection, where the client sends data packets to the server, and the server measures the throughput and reports the results[7]. It supports various parameters and options, making it highly versatile for conducting network performance tests under different scenarios. With its ability to gauge network performance accurately, iPerf3 serves as an essential tool for network administrators, researchers, and developers to optimize and troubleshoot networks for optimal efficiency and reliability.

III. METHOD

To start the research there are several things that need to be properly built and configured. Below will be listed the next step divided into two layers, namely physical layer, and application layer.

A. Physical Layer

1. Server

The first thing is to check the physical connection and make sure that all the necessary cables are connected. The next thing is to check the network connection. There are two ways for checking this we can look on the light indicator or we can use ping test.

2. AP (Access Point)

To check the physical layer of AP (Access Point) there are several steps. The first step is to check the physical condition if there are no sign of defect or physical damage. After making sure there are no problems in the previous step check the indicator light and verify it with the instructional guidebook. Then we can test the connectivity by using ping to the server. If the ping fails, try to use another cable and if the problem consists of it there may be some problem with the port.

3. Ethernet Cable

o ensure the ethernet cable is working properly we can use ping test after making sure the port both on server and AP is working if the ping is unsuccessful there might be some problem with the cable.

B. Application Layer

1. Server

There are several things that need to be completed in order to have successful research:

a. Docker

The first step to do is to make sure that your ubuntu live server is up to date by using this command.

```
Sudo apt update
```

After making sure that your operating system is up to date you can start by adding dependencies so that docker can run smoothly by using this command.

```
sudo apt install -y apt-transport-
https ca-certificates curl
software-properties-common
```

After that you can add docker's GPG key by using this command

```
curl -fsSL
https://download.docker.com/linux/
ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-
archive-keyring.gpg
```

GPG or the GNU privacy guard is necessary because it will ensure that your docker installation is authenticated and came from the official docker site. After the GPG key is added you can add docker repositories by using this command.

```
echo "deb [arch=amd64 signed-
by=/usr/share/keyrings/docker-
archive-keyring.gpg]
https://download.docker.com/linux/
ubuntu $(lsb_release -cs) stable" |
sudo tee
/etc/apt/sources.list.d/docker.lis
t > /dev/null
```

That step is necessary because by doing so the system would be able to find docker packages when using simple packet management commands such as `apt-get` or `apt`. To test if the package is added to the APT source, we can do it by using this command.

```
sudo apt update
```

If done correctly you should be able to go to the next step, that is to install docker by using this command.

```
sudo apt install -y docker-ce
docker-ce-cli containerd.io
```

If there are some problems and the docker cannot be installed there could be several factors such as the operating system not being up to date or there may be some change to the address from the previous code.

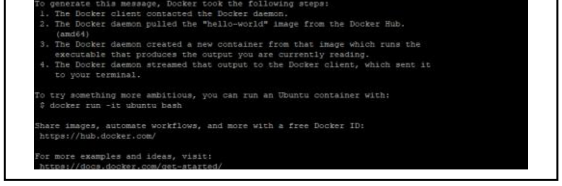
To verify the docker installation you could use this command.

```
sudo systemctl start docker
sudo systemctl enable docker
```

The start command is necessary to start the docker service if it isn't running and the enabled command is necessary so that docker would run every time the system boots up. To verify the installation, you could start a test container by using this command.

```
sudo docker run hello-world
```

When done correctly there should be an output like this image below.



```
Unable to find image 'hello-world:latest' locally
719385e32844: Pull complete
Digest: sha256:926fac19d22aa2d60f1a276b66a20eb765fbeea2db5dbdaafeb456ad8ce81598 2.457kB/2.457kB
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Fig. 1. Docker test run

If the image isn't available locally docker would pull an image from the official repository and load it like the image above. The next step is to install an add-on to the docker called docker-compose that is necessary for the research. Before installing docker-compose we need to make sure that we have all the dependencies in our system by trying to install them by using this code.

```
sudo apt install -y curl
```

After that we can use the dependencies, we get from before to download the docker compose by using this command.

```
sudo curl -L
"https://github.com/docker/compose
/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -
o /usr/local/bin/docker-compose
```

After successfully pulling it, we can change several of the properties to make sure that it able to run perfectly by using this code.

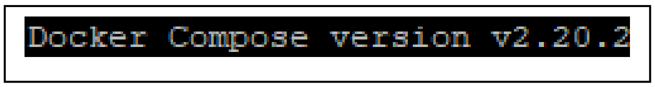
```
sudo chmod +x
/usr/local/bin/docker-compose

sudo ln -s /usr/local/bin/docker-
compose /usr/bin/docker-compose
```

The first code is necessary so that we would be able to execute the file by using `chmod` the file permissions and by using `+x` after that we specify that we want to be able to execute the file after that is just the path of the file I want to modify. The second step is added so docker compose is able to be used without typing the full path to binary every time it was used. To verify docker compose installation this code was used.

```
docker-compose –version
```

If the installation is done correctly the output should be similar to the image below. By the time of writing the latest version is 2.20.2.



```
Docker Compose version v2.20.2
```

Fig. 2. Example of lastest docker compose

b. OpenWiFi

The first thing to do is to clone the repositories into the local machine so it would be available locally. To achieve that we can use this line of code.

```
git clone
https://github.com/Telecominfrapro
ject/wlan-cloud-ucentral-deploy
```

There may be so difference to the address in the future so it would be strongly suggested to periodically check it. After it was done cloning move to the directory. The folder name is /wlan-cloud-ucentral-deploy/docker-compose after doing so we would be able to initialize the deployment by using this command.

```
docker-compose up -d
```

To check whether the deployment is successful or not we can use this line of code.

```
docker ps --format
"{{.Names}}\t{{.Command}}\t{{.Port
s}}"
```

The expected output would be similar to this:

TABLE I. DOCKER RUNNING CONTAINER

Name	Command	Ports
openwifi-owprov-ui-1	/docker-entrypoint.…	0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 80/tcp, 0.0.0.0:8443->8443/tcp, :::8443->8443/tcp
openwifi-owgw-ui-1	/docker-entrypoint.…	0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp
openwifi-owfms-1	/docker-entrypoint.…	0.0.0.0:16004->16004/tcp, :::16004->16004/tcp, 0.0.0.0:16104->16104/tcp, :::16104->16104/tcp, 17004/tcp
openwifi-owsec-1	/docker-entrypoint.…	0.0.0.0:16001->16001/tcp, :::16001->16001/tcp, 0.0.0.0:16101->16101/tcp, :::16101->16101/tcp, 17001/tcp
openwifi-owanalytics-1	/docker-entrypoint.…	0.0.0.0:16009->16009/tcp, :::16009->16009/tcp, 0.0.0.0:16109->16109/tcp, :::16109->16109/tcp, 17009/tcp
openwifi-owgw-1	/docker-entrypoint.…	0.0.0.0:1812-1813->1812-1813/udp, :::1812-1813->1812-1813/udp, 0.0.0.0:5912-5913->5912-5913/tcp, :::5912-5913->5912-5913/tcp, 0.0.0.0:15002->15002/tcp, :::15002->15002/tcp, 0.0.0.0:16002-16003->16002-

		16003/tcp, :::16002-16003->16002-16003/tcp, 0.0.0.0:3799->3799/udp, :::3799->3799/udp, 0.0.0.0:16102->16102/tcp, :::16102->16102/tcp, 17002/tcp
openwifi-owsub-1	/docker-entrypoint.…	0.0.0.0:16006->16006/tcp, :::16006->16006/tcp, 0.0.0.0:16106->16106/tcp, :::16106->16106/tcp, 17006/tcp
openwifi-owprov-1	/docker-entrypoint.…	0.0.0.0:16005->16005/tcp, :::16005->16005/tcp, 0.0.0.0:16105->16105/tcp, :::16105->16105/tcp, 17005/tcp
openwifi-kafka-1	/opt/bitnami/script…	
openwifi-zookeeper-1	/docker-entrypoint.…	2181/tcp, 2888/tcp, 3888/tcp, 8080/tcp

The purpose of these containers is as follow:

1. owprov : The OWPROV service is designed for the TIP OpenWiFi CloudSDK (OWSDK), responsible for handling access points in groups using entities and venues. Just like other microservices within OWSDK, OWPROV is specified using an OpenAPI definition and communicates with access points using the ucentral communication protocol.
2. owgw : OWGW (OpenWiFi Gateway) is responsible for managing Access Points that support the OpenWiFi uCentral protocol. Like all the other microservices in OWSDK, OWGW is described using an OpenAPI definition and utilizes the ucentral communication protocol to communicate with the Access Points.
3. owfms : OWFMS (OpenWiFi Firmware Management Service) is a micro-service within the OpenWiFi ecosystem. It facilitates the tasks of upgrading and maintaining the appropriate firmware for all devices used in your OpenWiFi solution.
4. owsec : OWSEC (OpenWiFi Security) serves as the Authentication and Resource Policy Access service within the TIP OpenWiFi Cloud SDK (OWSDK). Just like all the other microservices in OWSDK, OWSEC is specified using an OpenAPI definition and communicates with Access Points through the ucentral communication protocol.
5. owanalytics: OWANALYTICS is responsible for collecting statistics about devices used in OpenWiFi and categorizing them based on their provisioning entities or venues from OWPROV. Similar to all other microservices in OWSDK, OWANALYTICS is defined using an OpenAPI definition and communicates with Access Points through the ucentral communication protocol.
6. owsub : OWSUB (OpenWiFi Subscriber Management Service) offers a subscriber management service for the Subscriber App and communicates with the other components of the Cloud SDK. Like all the other microservices within OWSDK, OWSUB is specified using an OpenAPI definition and utilizes the ucentral communication protocol to interact with Access Points.
7. kafka : The mentioned system is utilized for the purpose of both monitoring and facilitating inter-service communication within the network architecture.
8. Zookeeper : ZooKeeper serves as a coordination service and distributed configuration management system. It enables cluster coordination, stores configuration data, facilitates service discovery, and supports distributed locking, playing a crucial role in ensuring the reliability, scalability, and coordination of the OpenWiFi ecosystem.

c. *iPerf3*

Installing iperf3 is pretty straight forward. The code that we used is.

```
sudo apt -y update
```

To verify installation just run this line of code.

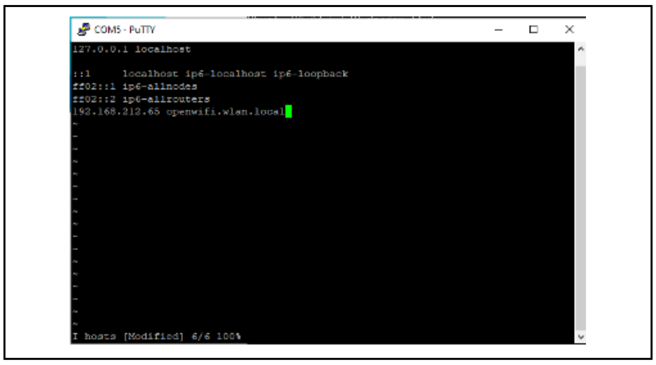
```
iperf3 -v
```

2. *Access Point*

Before stepping forward we need to make sure that the access point supports openwrt after making sure we can move forward. To set up the access point we can access it by two ways: the first one is via SSH and the second one is by console cable. The method I choose to access it is by console cable and accessing it on my pc via an app called MobaXTerm. The first prompt that is going to come is about is login prompt to make sure that the credentials are correct then we can move forward. After successfully login in we can move to the /etc directory because that is where we would mainly be. The first configuration is going to the hosts that are located in /etc. I edit the file using `vim` command that would look like this.

```
vim etc/hosts/
```

We need to add the IP address of the server that we use for OpenWiFi. The reason we do this so that the access point would be able to recognize the IP address as OpenWiFi domain. The example configuration would be like the image below.



```
127.0.0.1 localhost

::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.212.65 openwifi.wlan.local
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
I hosts [Modified] 6/6 100%
```

Fig. 3. Example of lastest docker compose

After doing so we can move to the next step, that is to load the certificate to the access point. The way I do that is by using a command on windows called secure copy or `scp`. After making sure that the certificate is compatible is compatible with the access point, I start a cmd on the folder where the certificate is located and run this command.

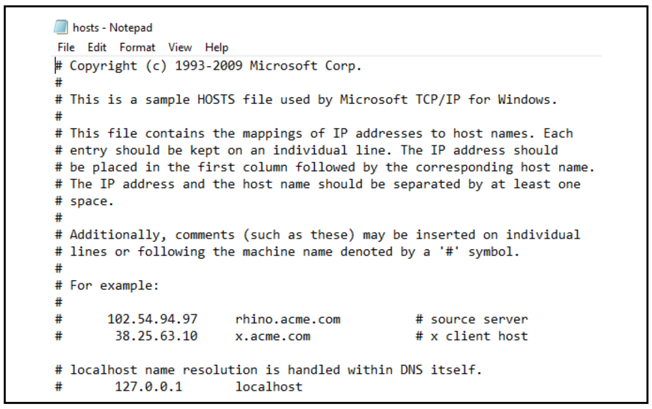
```
scp ./*
root@ip:/etc/ucentral/
```

Change the "ip" into the access point IP address. If you aren't sure what the IP of the access point is there's a command line you can use to check and look for. `ip a`

After doing so edit the IP address to the address of OpenWifi in the ucentral file that is located in /etc/config-shadow/. Then restart the network and ucentral service. If correctly done the access point is going to be available on the OpenWifi site.

3. *User Equipment*

The equipment is use is an old HP laptop with Windows 10 Home version 22H2 as operating system. The first step is to acquire the certificate so that the operating system would be able to recognize OpenWiFi and let the connection through. To get the certificate just open the official GitHub and locate the restapi-ca.pem and save it locally and add it to your machine. The machine I use is windows 10 so I use Microsoft Management Console. The next step is to change the hosts on local machine because I use windows so it's located in C:\Windows\System32\drivers\etc after locating the file open it using notepad with administrator rights and add the OpenWiFi IP address and the custom address that is going to be used. The one I used is openwifi.wlan.local for reference the hosts configuration is as follow in the image below.



```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
```

Fig. 4. Example of windows hosts file

Then download iPerf3 that is going to be used for testing purposes. Acquire the app from the official website which is iperf.fr and then extract it into your chosen destination. To test if the iPerf3 is running perfectly just run a test connection. The first step is activating the server side by using this command.

```
iperf3 -s
```

After making sure that the server side is running open the destination of iPerf3 extracted on your local machine and open CMD and just run this code.

```
Iperf3 -c IP
```

IV. Results and Discussion

A. *Physical Layer*

After making sure that every indicator is correct and user equipment is able to communicate with server via ping then the configuration on physical layer is done.

B. *Application Layer*

To verify if application layer is done correctly, I open the OpenWiFi site and push a test configuration and after verifying that the configuration I input and the output that access point gave is same then I can assume that the application layer is run correctly.

V. Conclusion

In this paper, we presented the steps involved in setting up and configuring OpenWiFi on a server. We first described the physical layer configuration steps, which involved checking the physical connection, network connection, and physical condition of the AP. We then described the application layer configuration steps, which involved installing Docker and Docker Compose, cloning the OpenWiFi repository, deploying OpenWiFi on the server, installing iPerf3, configuring the AP and user equipment, and running iPerf3 to test the network performance. Overall, this paper provides a detailed overview of the steps involved in setting up and configuring OpenWiFi. The information presented in this paper can be used by network administrators to deploy OpenWiFi in their own networks.

Acknowledgments

I would like to express my heartfelt gratitude to all the individuals and institutions who have contributed to the completion of this research paper. Their support, guidance, and encouragement have been instrumental in shaping the outcome of this study.

First and foremost, I extend my sincere appreciation to my esteemed supervisors, DHONI PUTRA SETIAWAN, S.T., M.T., Ph.D. and PRANANTO BAYU HERLAMBANG, S.Kom. Their combined expertise, mentorship, and valuable feedback have been invaluable throughout the research process. Their unwavering support and dedication have motivated me to excel and pursue academic excellence.

I extend my thanks to the faculty and staff of Telecom Infra Project Community Lab at Telkom University for providing a conducive environment for research and facilitating access to resources that aided my investigation.

My gratitude extends to my colleagues and friends who have provided valuable discussions, constructive criticism, and moral support throughout this journey. Their camaraderie has made this academic pursuit an enjoyable and fulfilling experience.

Last but not least, I would like to thank my family for their unwavering support and understanding during the time spent on this project. Their love and encouragement have been the cornerstone of my perseverance.

To all those who have played a role, big or small, in the realization of this paper, I extend my heartfelt thanks. Your contributions have been vital in shaping the quality and depth of my research, and I am truly grateful for your involvement.

Thank you.

Gangan Hamamminata Jamalullail

Prodi S1 Telecommunication

9/8/2023

REFERENCES

[1] S. Chauhan, A. Sharma, S. Pandey, K. N. Rao, and P. Kumar, "IEEE 802.11be: A Review on Wi-Fi 7 Use Cases," *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), ICRITO 2021*, 2021, doi: 10.1109/ICRITO51393.2021.9596344.

[2] T. Mazhar *et al.*, "Quality of Service (QoS) Performance Analysis in a Traffic Engineering Model for Next-Generation Wireless Sensor Networks," *Symmetry 2023, Vol. 15, Page 513*, vol. 15, no. 2, p. 513, Feb. 2023, doi: 10.3390/SYM15020513.

[3] "OpenWiFi - Telecom Infra Project." https://telecominfraproject.com/openwifi/ (accessed Aug. 06, 2023).

[4] G. Bharani Dharan and S. Jayalakshmi, "Energy efficient next-gen of virtualization for cloud-native applications in modern data centres," *Proceedings of the 4th International Conference on IoT in Social, Mobile, Analytics and Cloud, ISMAC 2020*, pp. 203–210, Oct. 2020, doi: 10.1109/I-SMAC49090.2020.9243497.

[5] N. S. Tarkaa, P. I. Iannah, and I. T. Iber, "Design and simulation of local area network using cisco packet tracer," *Int J Eng Sci (Ghaziabad)*, vol. 6, no. 10, pp. 63–77, 2017.

[6] W. Stallings, "IEEE 802.11: Wireless LANs from a to n," *IT Prof*, vol. 6, no. 5, pp. 32–37, Sep. 2004, doi: 10.1109/MITP.2004.62.

[7] "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool." https://iperf.fr/ (accessed Aug. 07, 2023).