AI BASED REALISTIC MULTI-HOP WIRELESS SIMULATION

by

Tagore Pothuneedi

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Computer Science

Charlotte

2021

Approved by:

Dr. Pu Wang

Dr. Minwoo Jake Lee

Dr. Mohsen Dorodchi

ABSTRACT

TAGORE POTHUNEEDI. AI Based Realistic Multi-Hop Wireless Simulation.
(Under the direction of DR. PU WANG)

Wireless multi-hop networks and software-defined networking (SDN) are emerging technologies in wireless communications for deploying cost-efficient programmable network backbones[1]. However, the physical testbeds do not provide scalable environments, accelerated development, and testing. Simulations are a cost-effective approach to overcome the bottlenecks of the physical testbed. Furthermore, Wireless multi-hop networks tend to have a high environmental impact which leads to significant performance issues such as low SNR and throughput. Adopting a multi-channel multi-radio(MCMR) setup can reduce signal noise and increase overall channel utilization for wireless multi-hop networks. Existing wireless simulators fail to simulate dominant factors like co-channel and adjust channel interference while simulating a realistic physical layer to close the gap between the actual physical layer and multi-channel multi-radio topology simulation.

This thesis attempts to reduce the gap between the physical layer from the real-world testbed and simulators. We propose a high fidelity physical layer simulator (FedEdge Simulator) which uses dynamic link scheduling and trace-based channel modeling to simulate a realistic physical layer for wireless multi-hop networks. The simulator supports the integration of custom-built machine learning model's to model the channel accurately. In addition, the simulator can act as a learning environment for Reinforcement learning in wireless multi-hop networks and transfer knowledge from simulation to reality. Finally, To illustrate the reduced reality gap between simulation and reality, we set up our experiment to integrate our FedEdge simulator with the existing framework of FedEdge[2][3]. We also show that our design of realistic simulations could help in knowledge transfer in reinforcement networking.

ACKNOWLEDGEMENTS

Throughout my journey in the master's program, I received enormous support and assistance. I want to thank my advisor Dr. Pu Wang, who has given valuable support and guidance with his expertise and knowledge. Dr. Wang has a unique way of looking at the problem and making things simpler, which helped me understand the problem. He was always ready whenever help was required and drove me to learn and achieve the impossible.

I would like to acknowledge my thesis committee members Dr. Minwoo Jake Lee and Dr. Mohsen Dorodchi, for being a part of this study and providing their valuable comments for this thesis.

I would also like to extend my thanks to Pinyarash Pinyoanuntapong and my lab members for their support and helpful suggestions for research.

Finally, I would like to convey my profound gratitude to my parents and friends who always believed and encouraged me to work hard and try things differently.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API Application Programming Interface

DSSS Direct-sequence spread spectrum

FL Federated Learning

GAN Generative adversarial network

HT High Throughput

IP Internet protocol

IPC Inter Process communication

MCMR Multi Channel Multi Radio

MIMO Multiple Input Multiple output

ML Machine Learning

MLME Media Access Control Sublayer Management Entity

OFDM Orthogonal frequency-division multiplexing

QoS Quality of Service

RSSI Received Signal Strength Indicator

SDN Software Defined Networking

SDWMN Software Defined Wireless Multi-hop Networks

SDWN Software Defined Wireless Networking

SNR Signal to Noise Ratio

SUMO Simulation of Urban Mobility

TC Traffic control

UAV Unmanned aerial vehicle

Wifi Wireless Fidelity

CHAPTER 1: INTRODUCTION

1.1 Motivation

Software-defined wireless mesh network (SDWMN) is widely exploited in the research of wireless communications[1]. The wireless mesh network is a rich interconnection of mesh clients, mesh routers, and mesh gateways. Each node in the mesh wirelessly relays traffic towards the gateway nodes, thereby forming a wireless backbone. A traditional network device consists of tightly coupled packet forwarding called data plane, and forwarding tables called control plane. This makes traditional networks complex to maintain and difficult to build new network features. On the other hand, software-defined networks(SDN) address this problem by decoupling the control plane from the data plane and allowing the control plane to operate from a centralized location or a server. SDN gives programming flexibility for network devices. This allows quick real-time insights for wireless telemetry and traffic engineering to optimize the network and improve the wireless network's quality of service(QoS).

The high dynamic nature of wireless multi-hop networks makes them prone to interference and noise, impacting the network's overall performance. In particular, multi-hop networks have a significant impact serving as core backbone infrastructure. Using a multi-channel and multi-radio based wireless setup can reduce the interference and channel contention between the nodes in the network, thereby providing a better signal-to-noise ratio(SNR) and high throughput.

Machine learning plays a significant role towards the goal of achieving true next-generation wireless networks, which are self-driving, cost-efficient, and adaptive by nature[5]. ML with programmable wireless multi-hop networks can be used in the wireless backbone to enhance the network's performance by dynamically adapting to

network conditions where traditional networks require constant interoperation from the network providers. This kind of network can scale dynamically per user demand and adapt to real-time security threats and performance issues. However, physical deployments have severe limitations in terms of scalability and cost-efficiency. Realistic software simulations are cheap and scalable solutions to alleviate the problems with physical testbeds, reducing deployment costs and time.

The majority of wireless SDN simulators like mininet-wifi drive the simulations in wireless networking research and academics. Nevertheless, the major impediment is that most of the simulators(Table 1.1), including mininet-wifi, ignore the possibility of a multi-hop network operating in a multi-channel multi-radio scenario. The lack of support for MCMR channel operations leads to a reality gap between simulation and real networks which work in wireless multi-hop backbone networks. In this study, we tried to analyze and reduce the gap between the physical testbed and the simulation. Towards this goal, FedEdge Simulator is an attempt to build realistic multi-hop wireless networks simulations operating in multi-channel multi-radio and validate the simulation to real performance gap between FedEdge Simulator and physical testbed. To the best of our knowledge, there are no existing simulators that are flexible to integrate custom machine learning models to simulate the wireless channel.

Moreover, The physical layer FedEdge simulator works on top of Linux tools and open-flow environments, Making it flexible and straightforward to integrate with any existing frameworks like mininet-wifi. We integrate FedEdge simulator to FedEdge Framework[3][2] which is used for Federated Edge computing for Reinforcement networking.

1.2 Problems & Challenges

The importance of realistic simulations is evident to build robust wireless multi-hop networks. As shown in Table 1.1 The current works in wireless simulations are lacking in either or support for Multi-channel Multi-radio, Running simulations on

real network protocol stack.

Table 1.1: Comparison of wireless network simulators

Software & Support	802.11s	MCMR	Real Protocol stack	Open source
Mininet-Wifi	Yes	No	Yes	Yes
wmediumd	Yes	No	Yes	Yes
Netorium	Yes	No	Yes	Yes
ViPMesh	Yes	Yes	Yes	No
NS-3 GYM	Yes	Yes	No	Yes

Overall, we believe the following challenges are hindering the progress of simulation driven developments of wireless multi-hop networks.

- *Existence of reality gap*: Even with wide range of applications of current network simulations, when used for training in machine learning environments leads to poor performance due to reality gap between the simulations and physical environments.

- *Lack of reliable training environment*: Machine learning application performance depend on the huge amounts of training data and a realistic training environment. In general, researchers usually tend to work directly on the physical testbed and spend a large amount of training time to develop models due to lack of reliable environment for wireless multi-hop simulations.

- *Need for realistic wireless multi-hop simulator*: The current simulators setting of single-radio single-channel links limits the bandwidth available for communication between the nodes. However this is not a realistic usage settings for wireless mesh backbone networks. And the use of MCMR will tremendously reduce the interference from other nodes and increase the overall performance.

1.3 Research Objectives & Contributions

With current challenges in existing simulations we believe that a novel approach to accurately model multi-hop environment is essential. In this study we focused on realizing the performance drawbacks and build a simulation tool which can accurately validate wireless multi-hop networks.

We believe that a novel approach to model a multi-hop environment accurately is essential with current challenges in existing simulations. This study focused on realizing the performance drawbacks and building a simulation tool that can accurately validate wireless multi-hop networks. The objectives of this thesis are summarized as follows:

- **Analyze and reduce the reality gap**: To validate the reality gap between simulation and reality, we proposed realistic but straightforward approach to simulate a wireless environment by using netlink protocol and tc for managing link parameters. This allows us to assess the limitations in each process.

- **Realistic wireless multi-hop simulations**: MCMR wireless networks are becoming increasingly more realistic usage setting for core backbone networks due to their advantages in low interference and high bandwidth availability. To achieve advantage for simulations, we built FedEdge simulator to work on top of batman-adv[6] and integrated a dynamic link scheduler for a realistic performance in wireless mesh operations.

- **AI-Based operations in simulations**: We built a flexible simulation system for a wireless multi-hop network that can easily facilitate the ML and RL operations. Towards this goal, we ran an unmodified Federated edge computing system[3] on top of our FedEdge simulator. In addition, we integrated a GAN channel model[4] which can be used to model UAV mesh backbone networks.

To the best of our knowledge, this is the first attempt to realistically model a multi-hop network simulator that can integrate AI operations. This helps in understanding and studying the real-time wireless characteristics of wireless multi-hop backbone networks. Also, we believe that the outcomes of this research will help academia and researchers to build and test a wide range of ML driven wireless network applications.

1.4 Organizations of Thesis

The rest of the thesis is organized as follows. Chapter 2 is the literature review of existing works . In chapter 3 we presents simulator components and architecture. And in sub sections we describe the workflow of simulator , type of modes in which simulator can support , packet processing frame in netlink mode , working of dynamic link scheduler in tclink mode and applications, Implementation of channel models and Interference modelling in the simulator. Finally we present the results and future work in subsequent chapters 5 and 6 followed by reference and appendix.

CHAPTER 2: Literature review

2.1 Mininet-wifi

Mininet-Wifi[7] focuses mainly on leveraging the software-defined networking paradigm in the context of wireless networks. It is a fork of mininet and built upon mininet's codebase. Like mininet, it is a container-based simulator extended to work with virtualized wifi stations and AP's. Wireless channel emulation is supported by a softmac layer in linux called mac80211. Stations in the network use Media Access Control Sublayer Management Entity(MLME). Hostapd[8] is a user-space demon that will take care of the user-space operations of AP's. Figure 2.1 illustrates core components of mininet-wifi.

Linux wireless command tools like iw, iwconfig, and wpa-supplicant are used to configure various station and AP parameters. Mininet-wifi supports infrastructure mode and Ad-hoc modes in wireless communications. Traffic control(tc) is a user-space utility program that mininet-wifi uses to configure the Linux kernel packet scheduler to control rate, delay, and loss, to emulate the behavior of the wireless environment. Mobility and propagation models are applied based on user-specified models. Mininet-wifi supports various models for mobility such as Random waypoint, Gauss Markov, Reference point & time-variant, and signal propagation. It uses Log distance, Friis, Lognormal Shadowing, etc. These models will help mininet-wifi to calculate path loss using log distance models as follows.

$$PathLoss = PL_{d_0} + 10nlog\left(\frac{d}{d_0}\right)$$

$$SignalStrength = pT + gT + gR - PathLoss$$

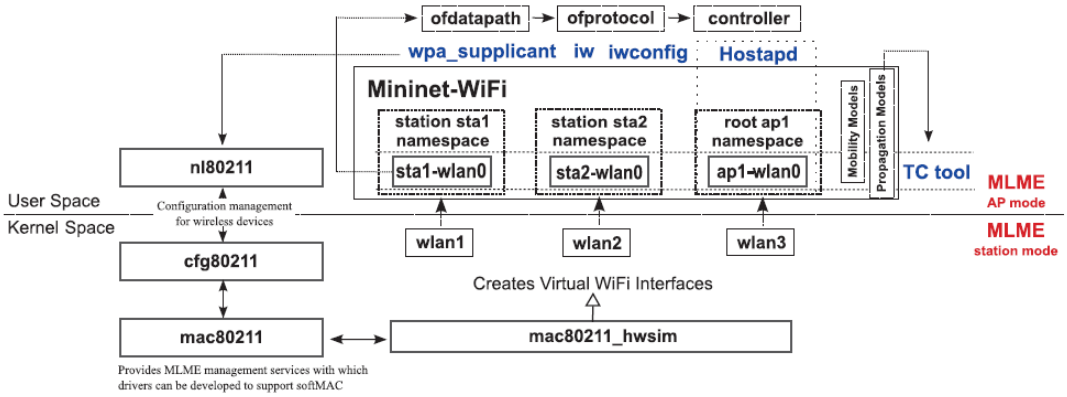


Figure 2.1: Mininet-Wifi

Where pT is Transmission power, gT is Transmitter antenna gain, gR is Receiver antenna gain. The calculated signal strength is the Received Signal Strength Indicator(RSSI) is used to calculate the rate of sending the data. The initial works of mininet-wifi have considered the distance between transmitter and receiver, which R2lab to calculate loss and rate. Mininet-wifi has been in active development, and the latest version added an implementation of supporting various ratesets, wmediumd, and support for Vehicular Adhoc Network scenarios through Simulation of Urban Mobility (SUMO)

The limitation of mininet wifi concerning wireless mesh networks is the inability to simulate multi-channel characteristics in mesh or Adhoc modes. Low throughputs are observed under stress tests of mininet-wifi(Table A.1, FigureA.1) in simple linear topologies like 2 nodes and 3 nodes wireless mesh networks on different channels, the distance between nodes varying from 40-105m and tested using all available propagation models. The performance further deteriorates by increasing the number of nodes in the simulated network.

2.2 Wmediumd

Wmediumd[9] is a userspace application written in C language, Developed by a united states company called cozybit. Figure 2.2 shows the design of wmediumd.

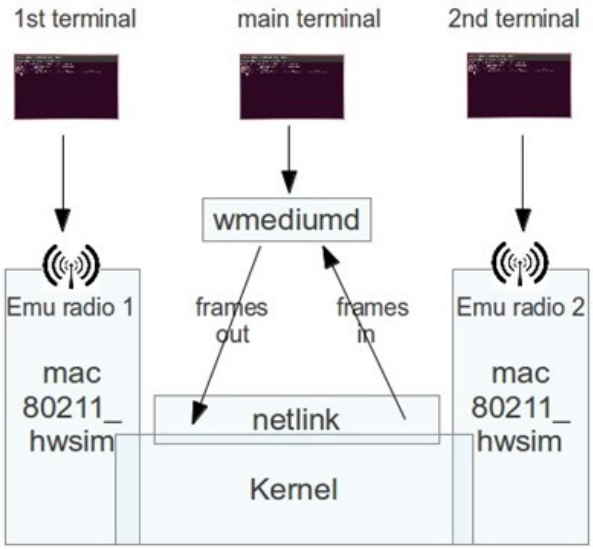


Figure 2.2: Wmediumd

In the latest version of mininet-wifi, integrates wmediumd for wireless mesh network simulations. Wmediumd leverages the capabilities of the virtual wireless interface by using mac80211_hwsim kernel module. The simulator will communicates with the mac80211_hwsim kernel module and generates probabilistic errors in packet transmission between the virtual wireless interfaces. Using a real protocol stack reduces the physical layer difference between simulation and actual testbeds and produces reliable outcomes.

The initial version of wmediumd is a simple implementation of creating the wireless interface nodes and configuring the nodes to communicate over a mesh network on the same network, SSID and Channel frequency. Later it was updated to support interference and backoff timers for channel access mechanisms. For kernel and userspace communication, wmediumd uses Netlink API and connects to the mac80211_hwsim family at runtime. Once the driver connects to the user space socket, it will forward the frames to the registered socket. wmediumd will work with the help of a user-specified probability configuration file which contains the loss probabilities parameters of each link.

The limitation of wmediumd is that it only considered that all of the mesh net-

work nodes are in the same channel and does not extend for a multi-channel environment. So the interference calculated by the model is similar to single-channel single-radio links, which are unrealistic in recent wireless multi-hop networks. The same is reflected in the tests(Figure A.2, Figure A.3) of wmediumd, where we see the throughput reduced by half with the number of nodes.

2.3 Netorium

Netorium [10] is a virtual wireless simulator that incorporates a radio propagation simulator called Meteor and a virtual wireless network application called Asteroid. The software was implemented in a layered approach as shown in Figure 2.3. The radio propagation environment in netorium uses multiple computer systems connected over a wired network and transmits wireless frames over as if they are wired frames. This extends the scalability of virtual wireless interfaces with less overhead and can emulate the wireless network features, network delay, bandwidth availability, and more. The paper mentions the drawback of using hardware emulator mac80211_hwsim[11] tends to require extensive computation resources, and running large-scale wireless networks on a single computer has substantial computational costs. Meteor can also reproduce the mobility and geometric behavior of the nodes in the network. As meteor has frame forwarding, it can work with data link layer protocols, e.g., B.A.T.M.A.N-adv[6]. The wireless packets generated by wireless interfaces created by mac80211_hwsim driver are encapsulated, as show in Figure 2.4 to transmit over the wired network. Asteroid use UDP encapsulation to avoid the difficulty of handling TCP connections. Geneve[12] is used for encapsulation, and it sets 0xFF01 to protocol field and wireless frame to options. When asteroid receives a frame, it will check the options field, send an acknowledgment frame to the transmitter, and forward it to the destination.

Asteroid will stop processing frame until the lock flag is set to true, which indicates some wireless node is transmitting and releasing it when the transmission is

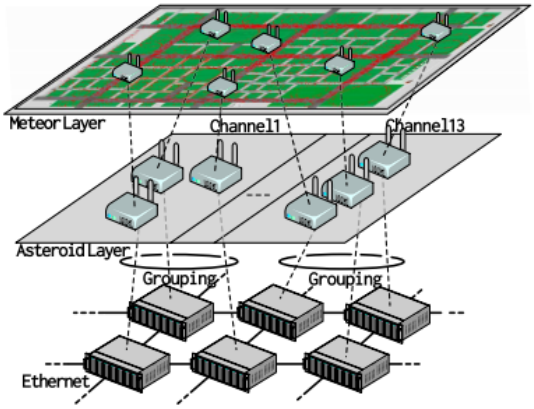


Figure 2.3: Netorium Architecture

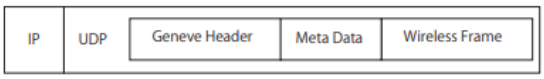


Figure 2.4: Encapsulation of wireless frame

done. The asteroid simulates the wireless medium behavior by the rate modulation of 802.11b/a/g and handles multiple DSSS and OFDM transmission parameters. This will benefit the simulation to know the contention window and data transmission times.

The limitation of netorium is that it requires more physical computer systems to scale well. Furthermore it does not support multi-channel operations for wireless mesh networks and OpenFlow-based programmability.

2.4 ViPmesh

Virtual Prototyping Mesh or ViPMesh[13] is a discrete event simulation tool that simulates wireless environments for mesh networks. ViPmesh is implemented using virtual wlan interfaces of linux mac80211_hwsim module and virtualization using QEMU and Linux containers. The architecture of vipmesh is shown in Figure 2.5. As mentioned, this software has multiple levels of virtualization. QEMU was used to

virtualize the guestOS and run the topology simulation which contains all the mesh nodes separated by Linux IP namespaces. This will help the nodes act as if they are working on their unmodified protocol stack. The frames will be transferred from guest to host using the VirtIO framework. An additional interface is used to assist in time synchronization between the guest os and simulation.

The implementation decouples the wall clock and simulation time to reduce the effect of computing performance of the host system, which is the work adapted from Werthmann et al.[14]. This patch to QEMU will provide different clocks for guest simulation and host. The time for guest and simulation are discrete in time steps, and the host system clock will be continuous. Blocking external I/O communication of simulation until the guest process completes and communicates the simulation time in discrete events will reduce the effect of computation and processing times. The frame flow starts from the emulated interface and is forwarded to the host OS with data rate and current wireless parameters.

Underlying simulation of ViPmesh is wmediumd with extension to support Medium Access techniques carrier sense, random backoff, frame re-transmissions. IEEE802.11e Enhanced Distributed channel access added, which is a channel access mechanism for 802.11s networks. The EDCA model also considers the multi-channel operation where nodes on the same frequency will form the same collision domain. Overall, the ViPmesh project can be considered for real-time wireless mesh simulations and support for different current wireless standards like multi-channel operation, Support for MIMO and HT. However, the work is not open source and is not available for general academic research purposes.

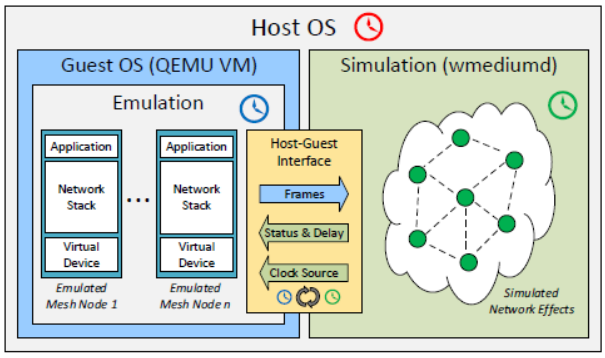


Figure 2.5: ViPMesh Architecture

CHAPTER 3: SYSTEM ARCHITECTURE

3.1 COMPONENTS

3.1.1 MAC80211_HWSIM

mac80211_hwsim[11] is a Linux kernel module used to simulate the IEEE 802.11 virtual radios for mac80211(Figure **??**). The module is built to test the functional features of mac80211 and other userspace tools which use the softmac layer of mac80211. The virtual radios emulated by mac80211_hwsim closely match the physical radio interfaces and operating system sees that all the radios emulated by the mac80211_hwsim as yet another hardware radio. The emulated radio interfaces do not have any limitations which helps researchers and developers emulate the real hardware to build and test new features to the various wireless drivers and tools independent of regulatory rules. Each radio in the mac80211_hwsim module works directly by copying frames to all the enabled radio interfaces which are on the same channel. The default driver initialization will create two radio interfaces by default if no ”radios” parameter is passed unless the ”radios” parameter is passed with the number of radios to create. The driver can generate up to a hundred radio interfaces. The module can be loaded after loading the operating using the following command.

```
#!/bin/bash
$modprobe mac80211_hwsim  #creates two radios
$modprobe mac80211_hwsim radios=100 #creates a hundred radios
$modprobe -r mac80211_hwsim #unload the kernel module
```

3.1.2 Linux IP Namespaces

The concept of Linux namespaces is to provide an abstraction to the system resources and make the resources appear to be owned by the process itself, isolating them from the global namespace. Any modification done to the resources resides exclusively inside the namespaces. Docker containers are implemented using namespaces. Linux supports different namespace containers like cgroup, IPC, Network, Mount, PID, Time, User, UTS. In our work, we use Network namespaces to build the nodes (AP, Stations) in our network. Network namespaces provide an abstraction for network resources that are available in the system as shown in figure 3.1. Any physical or virtual network resources available in the root namespace can be moved to the network namespace. When the network namespace is destroyed, the resources in the namespace will be moved back to the root namespace.

Each network namespace created has its own view of the network protocol stack, network devices, IP routing tables, firewall rules, sockets, etc. Network namespaces can be created by using the following command.

```
#!/bin/bash
$ip netns add <ns_name>    #create namespace
$ip netns delete <ns_name>  #delete namespace
$ip netns exec <ns_name> <command>  #execute <command> on namespace
```

3.1.3 Netlink Protocol

The Netlink[15] protocol is an application programming interface used for kernel and userspace communications. The protocol was developed to replace ioctl for network-related configurations and monitoring. The libnl suite contains various libraries to support libnl-route for routing, libnl-nf for monitoring, and libnl-genl, a generic library to extend and support custom functionality. This study specifically uses libnl-genl to handle the socket, parsing, deparsing, sending, and receiving the

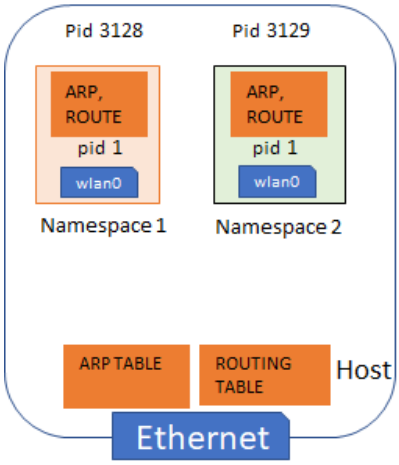


Figure 3.1: Linux IP Namespaces

packets from the radio interfaces created by mac80211_hwsim. The whole protocol suite is illustrated as shown in Figure 3.2

3.1.4 Traffic Control(tc)

Traffic control or tc is a Linux tool used to configure the traffic settings in the Linux kernel. Traffic control consists of shaping to control the traffic flow on an egress interface. In addition to shaping, tc can also support the following: (1) control the packet bursts, (2) Scheduling to ensure the quality of service by organizing the transmission of packets to ensure the high priority packets are guaranteed to transmit successfully, (3) policing occurs on ingress interface or port to filter or police the arriving traffic, (4)Dropping, traffic which is exceeding can be dropped on ingress and egress interfaces. Tc uses qdisc to enqueue the packets and is implemented as a 'pfifo' queue. The queue does not perform any processing on the packet. Instead, it holds on to the packet when the interface is not able to handle it.

3.1.5 hostapd & wpa_supplicant

Hostapd is a userspace daemon that implements IEEE802.11 access point management system. It runs in the background and controls authentications from clients. Uses the following command to configure hostapd by taking input, a configuration

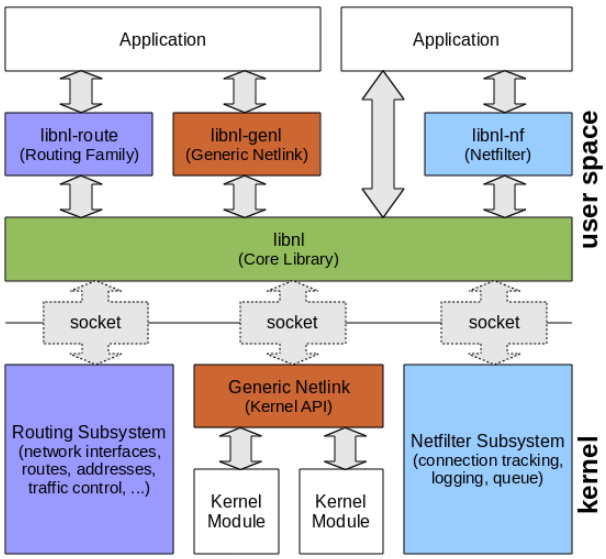


Figure 3.2: Netlink protocol suite

file which contains interface, channel, SSID name, encryption type, etc.

```
#!/bin/bash
$hostapd <file_name> -B    #starts an access point in background
```

On the contrary, wpa_supplicant is a daemon that runs in a client-side system. It is used to control the wireless connections on any client device. wpa_supplicant needs a network device and its driver loaded. Otherwise, the program will terminate. The steps wpa_supplicant uses to connect to the wireless access point is as follows.

- Requests the kernel driver to scan available BSSes
- Select a BSS based on its configuration
- Request the driver for association
- Use WPA-EAP, WPA-PSK for authentication & configure encryptions for communication
- Send and receive normal data packets

Uses following command to configure wpa_supplicant. '-I' is the parameter for interface and -c to pass configuration file which contains BSS, Encryption type, Password, etc.

```
#!/bin/bash
$wpa_supplicant -i <interface> -c <file_name.conf> # manage client network
```

3.2 FedEdge Simulator Architecture

FedEdge simulator enables the ability to mimic electronic hardware wireless radio and its operations to model an environment inside a computer system. It uses virtualized hardware and Linux-based software tools to build and maintain wireless networks. Most of the tools used are Linux inbuilt tools, namely IP namespaces[16] for containerization, mac80211_hwsim[11] to create virtual radio interfaces, iw tools[17] to manage the radio interfaces, and batman_adv [6] for shortest path routing. The simulator is built using the python language and enables open-flow Softswitch[18] for software-defined networking. In this section, we describe the framework of the Fededge simulator, the modes of operation, workflow, and the models used for channel emulation. The simulator operates in Netlink mode and TCLink mode.

3.2.1 Netlink Mode

FedEdge simulator architecture is illustrated in Figure 3.3. Network simulation in typical netlink mode is similar to wmediumd without the multi-channel multi-radio support. The userspace application registers with the driver using Netlink protocol and libnl libraries to manage and process the packets received from kernel space (mac80211_hwsim). The simulator will emulate the channel medium by introducing the error probabilities while transmitting the packets based on its channel and other stations on the same channel.

In netlink mode the packet processing works as shown in Figure 3.4, the simulator has an active netlink socket to the driver to send and receive frames. Once the

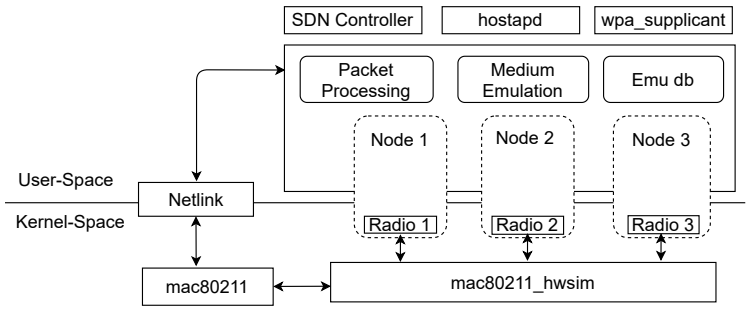


Figure 3.3: FedEdge Simulator in Netlink mode

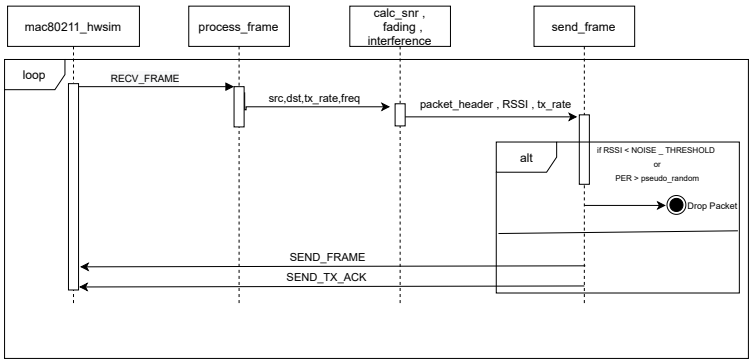


Figure 3.4: Packet processing in netlink mode

simulator registers to receive frames, a function to process the frame is associated with the callbacks. When the callback function receives the frame, it will unpack the frame to read source mac address, destination mac address, channel, transmit rates, payload, cookies, etc. All the parameters from the frame will be used to calculate the signal-to-noise ratio(SNR), signal fading, interference, transmission probability, and compare with the obtained signal with the noise threshold. Suppose the calculated RSSI is significantly higher than noise. In that case, the simulator will compare the packet error rate(per), obtained from the signal probability matrix for the given MCS index and RSSI to the randomly generated number between 0 and 1. If the transmission conditions satisfy, the simulator process the packet and send it back to the driver by constructing a new frame. Once the frame is successfully sent, the simulator acknowledges the source node by sending an acknowledgment frame.

However, we found this approach has drawbacks in terms of performance even

when the simulated topology operates in multi-channel. Performance analysis using cprofile on the simulator process, revealed that the userspace simulator is spending 69.09% of the time in receiving the frames from the kernel driver and only 30.15% in processing the callbacks. As the number of stations increases, these number are going down further and reducing the performance and actual throughput as revealed in iperf tests. To overcome this problem, we implemented dynamic link scheduling, which works in tclink mode.

3.2.2 TCLink Mode

The simulator architecture in tclink mode remains symmetric to netlink mode. However, in place of netlink, the simulator operates using traffic control(tc). tc is used to shape the traffic on the egress interfaces of each node based on the signal-to-noise ratio(SNR) obtained from the channel model and interference model. There is no switching of frames from kernel to user space, so zero copies are required, thereby reducing the overhead. Link scheduler comes into the play in tclink mode to dynamically schedule the links between nodes. Link scheduler runs for every 5secs and interacts with medium emulation to introduce signal fading and interference between the nodes. The user-space simulator will not receive any frames from the driver, so it does not know active transmissions. This is the trade-off in the tclink mode, where the interference model is restricted to static interference. The simulator assumes the worst-case scenario as all the nodes in the same channel interfere all the time. If the network topology nodes use non-overlapping channels, there will be no interference emulation thus not affecting the multi-channel operations. TClink mode exclusively uses link scheduler for timely update of link parameters. Figure 3.6 describe the flow of the dynamic link scheduler. The simulator starts with building the necessary wireless medium parameters like signal fading, interference, propagation loss., several threads are created for each node in the topology build stage and based on the number of interfaces each node contains. Each thread iteratively waits for an interval of 5

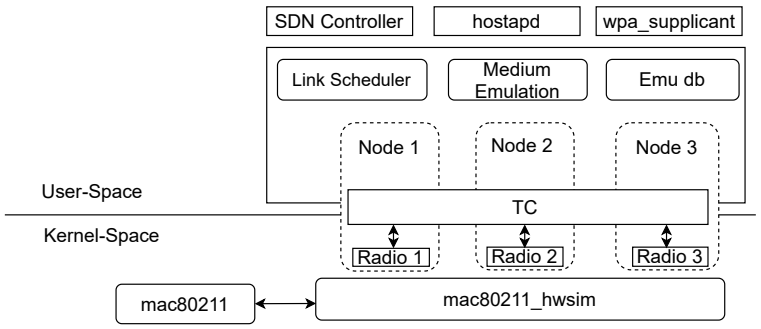


Figure 3.5: FedEdge Simulator in tclink mode

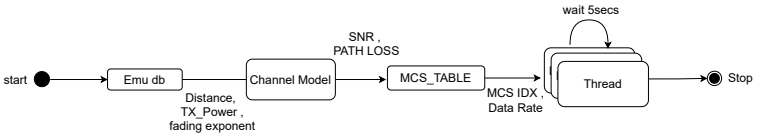


Figure 3.6: Dynamic Link Scheduling

seconds before updating the link parameters.

3.3 Simulator Workflow

3.3.1 Input Configuration file

The simulator starts receiving the input configuration file to build a wireless network. code listing at 3.1, is the JSON format configuration, which provides context to the simulator to build the topology. This file contains the following structures (1) topology type to determine if the simulation is mesh or UAV, (2) trace_sim is a boolean value to start the simulator in trace based channel modelling,(3) link_type contains the mode of the simulator.(4) controller type determines the usage of software-defined switch , (5)controller_ip will be used if the controller is set to remote and it contins the ip address of the remote controller,(6) noofnodes maintains the total count of nodes in the simulation, (7) node_type is dictionary map which maps the type of each node in the network,(8)radio_per_node is a dictionary map which maps number of interfaces in each node of the network,(9)node_mac is dictionary map which contains the mac address of each interface mapped to the nodes,(10)channel is a dictionary map which contain the mapping from nodes to channel of each in-

Listing 3.1: Sample JSON Input Configuration file to Simulator

```
{
"topology": <"uav" or "mesh">
"trace_sim": < true or false>
"link_type": <"netlink" or "tclink">
"controller":<"remote"or "local" or "none">
"controller_ip": <"n1":"192.168.1.1">
"noofnodes":<2>
"node_type":<"n1":mesh,"n2":uav>
"radio_per_node":<"n1":2>
"node_mac":<"n1":["33:33:00:00:00:01","33:33:00:00:00:02"]>
"channel":<"n1":[1,6]>
"tx_power":<"n1":[15,15]>
"position":<"n1":[x,y,z]>
"links":<"n1":["n2",n3"]>
"trace_generate":<True or False>
"trace_log":<"n1":["file_loc1","file_loc2"]>
"propagation_model":<log_distance,custom_model>
}
```

terface,(11)tx_power is dictionary map which maps each interface of node to the transmit power of the interface ,(12)position is a map which map between the location of each node to (x,y,z) co-ordinates in 3d space,(13)links is dictionary map which determines the active links of a particular node ,(14)trace_generate is a boolean value which work to create the trace files for a simulated topology,(15)trace_log is a list contains the file location of trace files or the interface name for which the trace is generated,(16)propagation_model is path loss model choosen to calculate the loss between the source and destination nodes.

3.3.2 Stages

This section describes the general workflow for FedEdge simulator and its input and output. The input to the simulator is a JSON file that contains context related to the topology that needs to be emulated. The simulator works in two stages. Firstly, as shown in Figure 3.7 the simulator parses the input configuration file to build the wireless medium parameters, save them to a database, and construct the topology. Secondly, the simulator works either in netlink mode or tclink mode(Figure 3.8. In

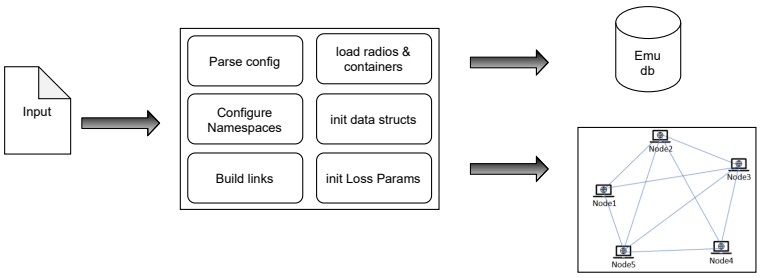


Figure 3.7: Stage 1 Build configuration step in FedEdge Simulation

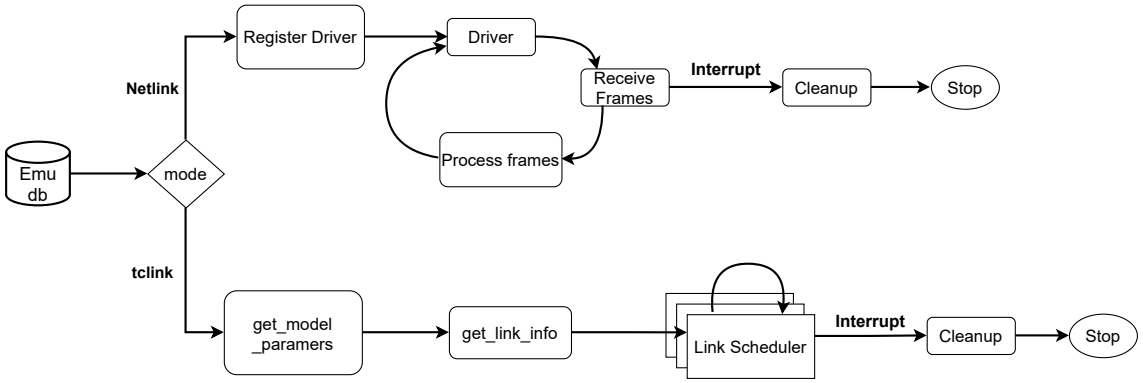


Figure 3.8: Stage 2 in FedEdge Simulation

netlink mode, the simulator will process frames until an interrupt occurs. In tclink mode, the simulator uses a link scheduler to update link parameters of each interface until the interrupt. Finally, in either of the modes, the topology is cleaned up by deleting the nodes, unloading the drivers, killing threads, cleaning up softswitch processes, etc.

3.4 Channel Modelling

Channel Modelling in wireless simulation involves the estimation of the signal-to-noise ratio from a given source to destination. The FedEdge Simulator estimates the channel based on Propagation loss models. In general FedEdge simulator has static and custom models, as shown in Figure 3.9. Static models are traditional channel models like Log Distance, Log-Normal Shadowing, etc. In addition, Custom models can be integrated to the simulator and not restricted to GAN Based Channel Model[4] and Trace based Channel model. In stage 1, propagation loss is calculated between

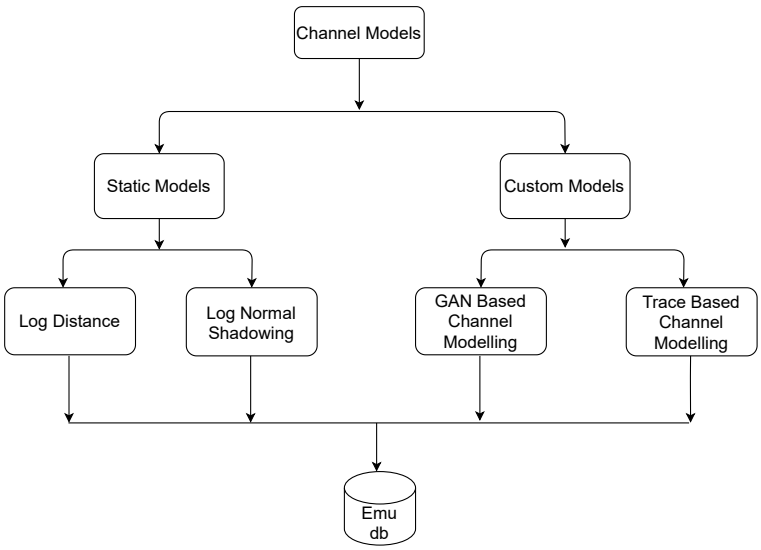


Figure 3.9: Channel Models

the nodes and stored in the database based on the user-selected channel model. In stage 2, the simulator uses the propagation loss data in the database to calculate the wireless parameters at runtime. This section discusses the propagation models used in the simulator.

3.4.1 Static Models

The log distance model is a traditional wireless propagation model used to calculate the propagation loss in various environments. However, the model is limited to line-of-sight signal propagation and does not consider obstructions like walls, trees, buildings, etc. The model is based on a simple estimation based on referencing the signal propagation with respect to distances. If path loss from transmitter to distance d0 is PL0, the path loss at distance d (d>d0) is given by the following equation.

$$PL_{d \to d_0} = PL_{d_0} + 10nlog\left(\frac{d}{d_0}\right)$$

PL_{d0}=Path Loss at distance d0 (dBm),

$PL_{d \to d_0}$= Path Loss at distance d (dBm),

n = path loss exponent depends on the environment.

Log-Normal shadowing is an extension to the log distance model. It accounts for random shadowing effects by adding a zero-gaussian distributed random variable with standard deviation Ï expressed in dB.

$$PL_{d \to d_0} = PL_0 + 10nlog\left(\frac{d}{d_0}\right) + \chi$$

PL_{d0}=Path Loss at distance d0 (dBm),

$PL_{d \to d_0}$= Path Loss at distance d (dBm),

n = path loss exponent depends on the environment,

χ = gaussian distributed random variable .

3.4.2 GAN Based Channel Model

The static models are traditional and instrumental, focusing only on path loss for simple line of sight signal propagations. However, for wireless networks which work on millimeter wave(mmwave) the traditional loss model does not capture delay, signal transmit and receive angles, path gains, etc. Significant research is done to develop wireless network models based on deep learning and generative networks to accurately model next-generation wireless networks. This study adopts a mmwave GAN Based channel model[4] to emulate UAV to the base station channel or, in other words, air to ground communications. This section describes the model and its integration with the FedEdge simulator.

The methodology demonstrated by the paper is characterized using unmanned aerial vehicles (UAV) and base stations located on rooftops(aerial) and street level(terrestrial) operating at 28Ghz channels. All the node orientation are calculated in 3d space w.r.t (x,y,z) axis. Training data was provided from a ray-tracing tool developed to provide datasets to train neural networks. The generative model used is a variational autoencoder as it avoids the min-max optimization. Problem formulation to develop this model is considered a link between the UAV as transmit-

ter and base station as a receiver or vice versa. Any link between the transmitter and receiver is formulated as following.

$$\mathbf{x} = \left\{ \left(L_k, \phi_k^{\mathrm{rx}}, \theta_k^{\mathrm{rx}}, \phi_k^{\mathrm{tx}}, \theta_k^{\mathrm{tx}}, \tau_k \right), k = 1, \ldots, K \right\}$$

At a given instance of time, the link condition vector denotes the distance vector connecting the UAV to the base station in 3d space with d=(dx,dy,dz) and type of the base station terrestrial, aerial. The condition vector is formulated as

$$u = (d, c)$$

The network architecture is of two stages, as shown in Figure 3.10. The first network predicts the link status based on the condition vector u and outputs the LOS, NLOS, NoLink. The second stage is path generator takes link state 's' condition vector 'u' and latent variable 'Z' as inputs and produces 'x' path vector. The path vector contains loss, angle of arrival and departure, the azimuth angle of arrival and departure for 'K' different paths.

$$\mathbf{x} = \left\{ \left(L_k, \phi_k^{\mathrm{rx}}, \theta_k^{\mathrm{rx}}, \phi_k^{\mathrm{tx}}, \theta_k^{\mathrm{tx}}, \tau_k \right), k = 1, \ldots, K \right\}$$

The encoder network is trained to map data based on 'x', 'u', 's' and produces a latent variable space with conditional distribution relevant to the data. The simulator builts and passes the conditional vector 'u' to the model and gets the vector 'x' of 'K' different path loss. Finally, it takes the median to get the propagation loss and SNR of the given source and destination nodes. This model was minimally customized to integrated with FedEdge simulator to emulate air to ground channels. uav to uav channel realizations in the simulator are done using static channel models and can easily be extend to any customized channel models.

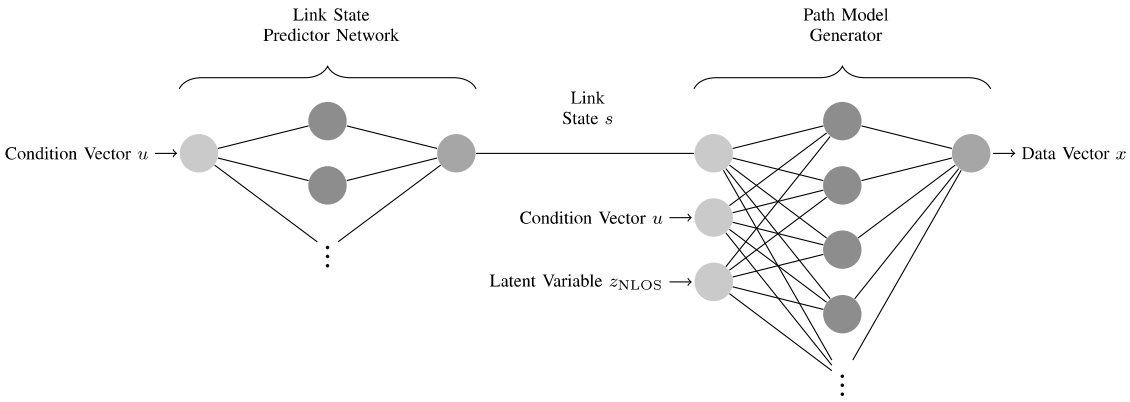


Figure 3.10: Two stage GAN Network[4]

3.4.3 Trace Based Channel Model

Verifying or evaluating a wireless network is the most challenging task given the random, unpredictable nature of the wireless medium. Trace-based channel modeling is an essential technique for replaying or reproducing the wireless scenarios from testbed to simulation or vice versa. Trace simulation in the FedEdge simulator works in two modes: replay an existing trace and trace generation for a given input topology. This approach will provide the required data to model complex wireless systems.

3.4.3.1 Trace Replay

The simulator can replay the trace based on a node's interface and entirely independent of the type of the node. As shown in Figure 3.11 the file location passed to the simulator with the input configuration file as dictionary key-value pair where key being the radio interface name and associated value is the trace log location. Trace files of each interface are processed, and threads are generated based on the number of replay files and passed to the link scheduler. Based on the signal level on each trace file link scheduler will run the traffic control(tc) and set the bandwidth on each interface of the replay nodes periodically until the trace files are complete. The node's interfaces, which are excluded from the trace replay, are manually taken care of by the simulator and link scheduler based on the input channel and interference models.

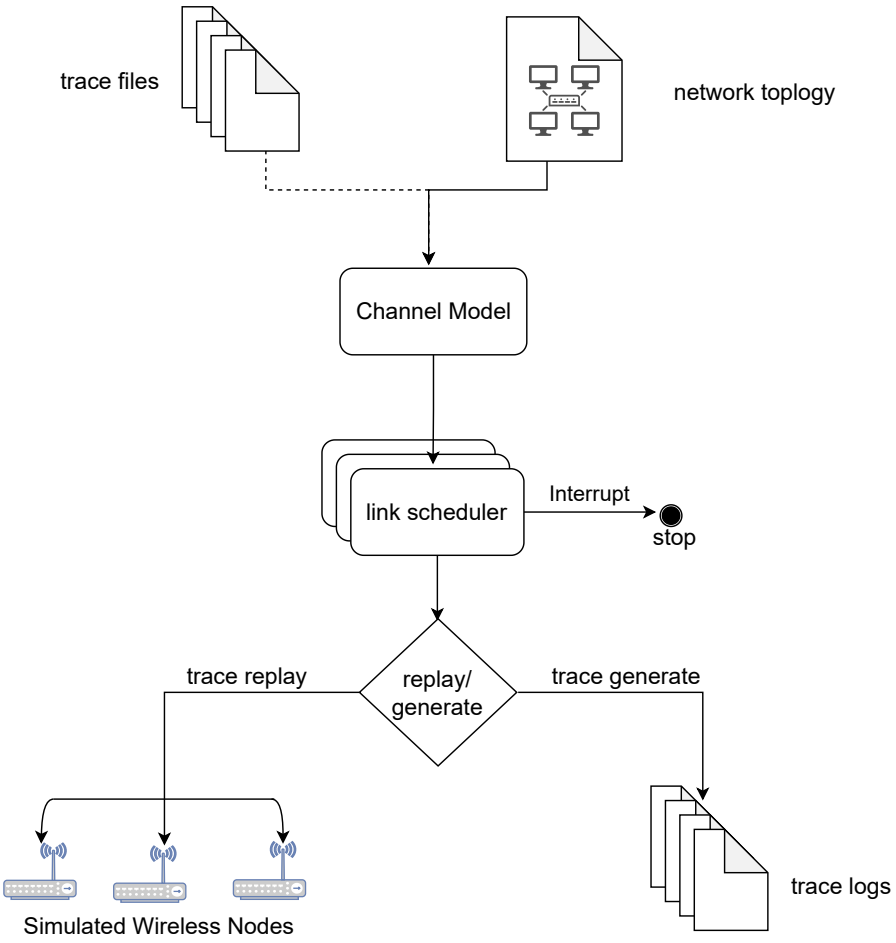


Figure 3.11: Trace Based Channel Modelling

3.4.3.2 Trace Generate

The second mode of the trace model operates to generate the trace logs, which can be used to replay on the testbed, visualize, or used as training data for machine learning. When the simulator runs in generate trace mode, it takes the input from the parsed input configuration and learns the interfaces. Besides, the program works in the opposite direction compared to replay(Figure 3.11), where it takes in the topology and produces the trace files in the form of a ".csv" which contains time, MCS index, RSSI, loss, traffic rate. The link scheduler will capture the data at an interval of 5 seconds until the interrupt. All the parameters recorded to the trace files are cal-

culations made by the simulator based on user-selected channel models, interference models, and network topology.

3.5 Interference Model

Medium access in wired networks is more straightforward when compared to wireless environments because of the ubiquity of unpredictable factors like interference, noise, signal loss, etc. Interference in wireless networks is seen when two or more nodes try to sense the channel and transmit the data simultaneously. This causes collisions and data corruption at the receiver. Upon detecting a collision, the transmitting nodes will back off a random amount of time before sending their data again. Interference is caused when more nodes are present in the same channel and trying to access the wireless medium. Ultimately this causes low throughput, high latency communications. In this study, to achieve a similar effect of interference, we consider the presence of interference as additive white gaussian noise and simulate lower SNR. By controlling the SNR, the simulator can change the throughput of nodes using link scheduling and traffic control(tc). The simulator can simulate the interference in two modes, static interference, and dynamic interference. The following sections describe the working of static and dynamic interference in the simulator.

3.5.1 Static Interference

The simulator builds the context from the topology file and builds functional data structures used at run time, and one such structure is 'channel_radios'. The 'channel_radios' contains a map of channel number and interface mac addresses using that channel. When queried with a channel number, the map returns the list of nodes in that channel. The static interference function uses this mapping to calculate the transmit power from all other stations in that channel, excluding the source and destination. This approach simulates the worst-case scenario by assuming that there will be constant interference from the nodes in the same channel all the time. The

following function is the formulation of the static interference in the simulator.

$$Interference\_power = \sum_{n=0}^{channel\_radios(channel)} (dbm\_to\_mW(signal[n][destination])$$

$$mW\_to\_dBm = 10 \cdot \log_{10}(mW)$$
$$dBm\_to\_mW = Watt \cdot 1000 = 10^{\frac{dBm}{10}}$$

Channel_radios (station_channel) – contains all stations in particular frequency or channel

3.5.2 Delayed Dynamic Interference

To model an accurate wireless interference, the simulator needs the real-time information of the frame transmissions, keeping track of the frames generated in the same channel, and simulating loss, latency between the frame transmissions. The FedEdge simulation can emulate the dynamic interference in Netlink mode in a delayed fashion. Delayed Dynamic interference is an imitation to achieve the dynamic interference in a single thread application. The approach is similar to static interference expect the simulator will maintain a channel array(s). if there are multiple channels, the simulator will maintain multiple arrays and keep track of the stations transmitting frames for a defined interval. In runtime, the interference calculations verify the channel arrays and add noise based on the actively transmitting stations present in the channel arrays. The channel arrays will be cleared as a function of time. If no stations are transmitting, the channel arrays will be empty. The following function is the formulation of delayed dynamic interference used in simulating dynamic interference.

$$Interference\_power = \sum_{n=0}^{node\_transmit(channel)} (dbm\_to\_mW(signal[n][destination])$$

$$mW\_to\_dBm = 10 \cdot \log_{10}(mW)$$

$$dBm\_to\_mW = Watt \cdot 1000 = 10^{\frac{dBm}{10}}$$

node_transmit (station_channel) – contains actively transmitting stations in particular frequency or channel

CHAPTER 4: EXPERIMIMENTAL EVALUATION

In this thesis, we present the experiments which are part of sim to real transfer in multi-agent reinforcement networking experiments[19] using FedEdge simulator. The FedEdge framework is built for the physical testbed to improve federated learning over multi-hop networks using multi-agent reinforcement learning [3]. To evaluate the fidelity of our simulator, we ran the FedEdge framework on top of the FedEdge simulator and physical testbed. The following scenarios are considered to realize the reality gap between the simulation and testbed:

- We assess the physical layer of the simulator by analyzing the federated learning experiments free of multi-agent RL on both the simulator and physical testbed.
- We analyze the reality gap difference by training MA-RL agents online in both environments.
- We examine the outcome of knowledge transfer of the multi-agent RL by assessing model performance on the physical testbed with the pre-trained Q-table from the simulator.

4.1 Experiment setup

To minimize the differences between simulation and physical testbed, we set up identical topologies with 10 routers and 9 workers and a server as shown in Figure 4.1. FedEdge framework works on top of the physical layer of the simulator using batman_adv shortest path and multi-agent RL for routing packets. A trace-based simulation was used on the simulator by inputting the logs from the testbed location in WiNSLAB at uncc. Based on the MCS index table of 802.11ac 20MHz channels

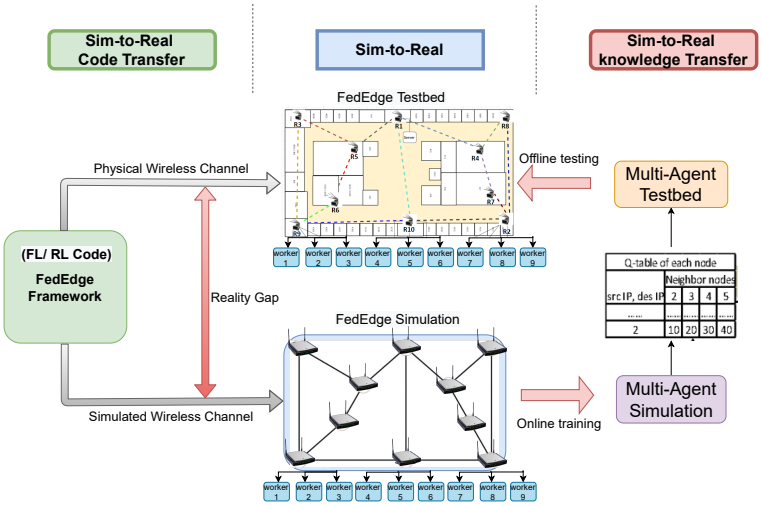
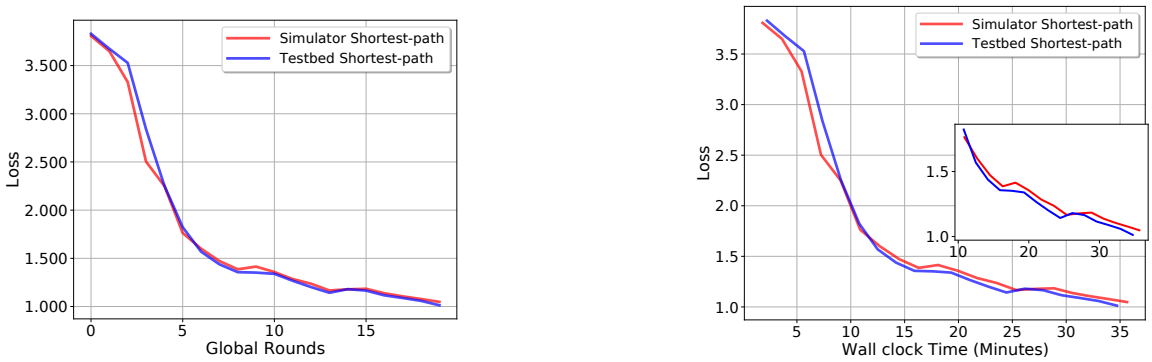


Figure 4.1: Experimental setup to in physical testbed and FedEdge simulator

specified at [20] the link scheduler converts the signal to traffic rate and sets the radio interface bandwidth. In FedEdge Framework, each worker node will participate in model training and distribute the model to the server for aggregation on each iteration. The CNN model in FedEdge uses two convolution layers with 32 and 64 filters respectively. Each layer has a 2x2 max-pooling layer. The convolutions are followed by a fully connected 128 unit ReLU activation. Finally, the output layer is connected with a softmax function. The learning rate used is 0.1 on all the worker nodes and the model is of size 5.8MB. The model was tested with widely known benchmark datasets such as FEMNIST and the extended federated version of MNIST[21] from the LEAF[22] which has 62 classes.

4.2 Experiment Results

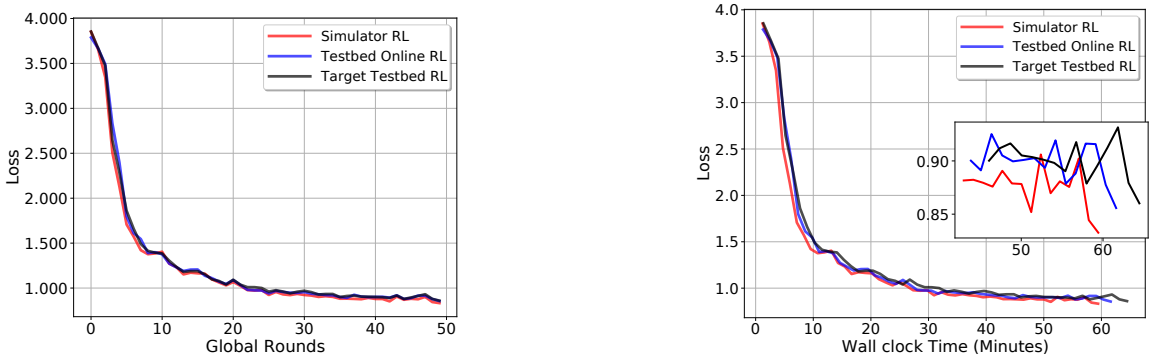
Initially, we evaluate the gap between the shortest path routing in both simulated environment and physical testbed. Referring Figure 4.2 the closeness of both the results, which lead to the similar iteration convergence achieving the same loss after performing the same number of epochs as shown in Figure 4.2(a) and roughly similar wall clock convergence times Figure 4.2(b). Hence, we verify the results as nearly identical in both simulation and reality. For the following experiment, we train



(a) Iteration loss convergence

(b) Wall-clock time loss convergence

Figure 4.2: Test after 20 epochs of shortest-path routing in simulator (red) and testbed (blue)

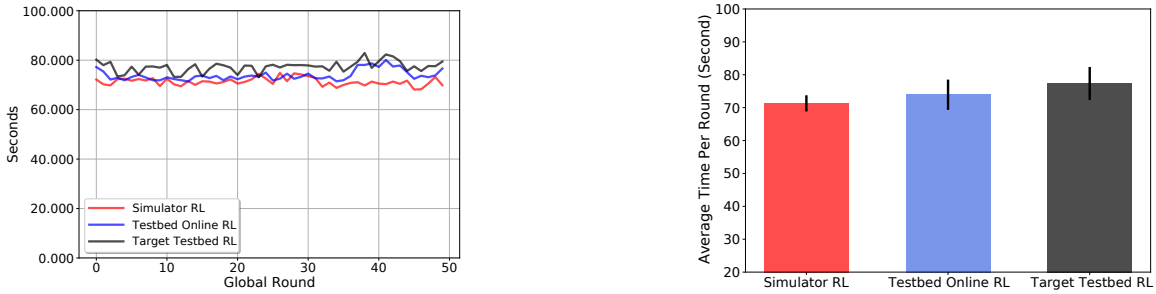


(a) Iteration loss convergence

(b) Wall-clock time loss convergence

Figure 4.3: Test after 50 epochs of on-policy softmax on simulator (red), on-policy softmax on testbed online learning (blue), on-policy softmax on testbed target testing (black)

the RL agent in an online manner on both the simulator physical layer and physical testbed. As illustrated in Figure 4.3, the iteration loss convergence are similar in terms of global rounds. However, in the simulation online RL training, we achieve a somewhat better wall-clock time than testbed with a difference of 2 minutes. Given the results and identical physical layer setup confirms that the FedEdge simulator can effectively reduce the reality gap between simulations and physical testbed. Finally, to show that simulation-driven development can improve real-world deployments. We had trained the multi-agent RL in the simulated topology on the worker nodes and



(a) Time per round

(b) Average time per round

Figure 4.4: Test after 50 rounds of FL communications: seconds per round(a) and mean seconds per round(b) of on-policy softmax on simulator (red), on-policy softmax on testbed online learning (blue), on-policy softmax on testbed target testing (black) over

transferred the pre-trained knowledge to each worker node in the physical testbed where we froze the Q-table update and allowed the RL agent to use only the pre-trained softmax policy. Figure 4.4 shows that the pre-trained testbed RL achieved better results in terms of wall-clock time convergence compared to the target physical testbed because of suspending the q-table update and pausing the dynamic adaptability to the wireless environment. Still, from Figure 4.4(a), we can see that the difference between the target testbed and online training is minimal, and both follow similar trends.Figure 4.4(b) shows the deviation and mean of 50 rounds of FL communications.

CHAPTER 5: CONCLUSION & FUTURE WORK

The challenges with a physical testbed to practice machine learning in wireless multi-hop networks are evident. However, it is impractical to extend the experiments with more physical testbeds. On the other hand, a realistic simulation for such an application can help in the accelerated development and prototyping of ML for wireless networks. In this study, we analyze the physical layer gap between the simulation and physical testbed scenarios, presented a functional simulation tool and an approach for democratizing AI for realistic wireless network simulations. we believe this approach will motivate a lot of researchers and academia to build and test robust machine learning-based models for a wide range of wireless applications.

In the future, the FedEdge simulator can be extended to support dynamic wireless environment update to support physical location changes to the nodes without restarting the simulation. In addition, support for RF heat map generation for the multi-channel topology simulation can help the users have high visibility of the node position and topology setup with per-channel signal coverage. Furthermore, P4 switches are increasingly being used in research to leverage the hardware level programmability support. This aids in testing new independent protocols and introducing custom in-band telemetry to improve communication between the nodes. Having realistic simulation support for p4 based software switch can help test the physical testbed topology on p4 switch architecture.

REFERENCES

[1] R. G. Clegg, J. Spencer, R. Landa, M. Thakur, J. Mitchell, and M. Rio, "Pushing software defined networking to the access," in *2014 Third European Workshop on Software Defined Networks*, pp. 31–36, 2014.

[2] P. Pinyoanuntapong, P. Janakaraj, P. Wang, M. Lee, and C. Chen, "Fedair: Towards multi-hop federated learning over-the-air," in *Proceedings of IEEE SPAWC*, 2020.

[3] P. Pinyoanuntapong, P. Janakaraj, R. Balakrishnan, M. Lee, C. Chen, and P. Wang, "Edgeml:towards network-accelerated federated learning over wireless edge," in *arXiv pre-print*, 2021.

[4] W. Xia, S. Rangan, M. Mezzavillla, A. Lozano, G. Geraci, V. Semkin, and G. Loianno, "Generative neural network channel modeling for millimeter-wave uav communication," 12 2020.

[5] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *ArXiv*, vol. abs/1710.02913, 2017.

[6] S. W. Marek Lindner, "batman_adv," 2011.

[7] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-wifi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, pp. 384–389, 2015.

[8] "Linux hostapd," 2004.

[9] c. bcopeland, "wmeidumd." https://github.com/cozybit/wmediumd.git, 2010.

[10] K. Akashi, T. Inoue, S. Yasuda, Y. Takano, and Y. Shinoda, "Netorium: High-fidelity scalable wireless network emulator," in *Proceedings of the 12th Asian Internet Engineering Conference*, AINTEC '16, (New York, NY, USA), p. 25â32, Association for Computing Machinery, 2016.

[11] J. Malinen, "mac80211_hwsim," 2008.

[12] J. Gross, T. Sridhar, P. Garg, C. Wright, and I. Ganga, "Geneve: Generic network virtualization encapsulation. ietf draft," 2016.

[13] M. Rethfeldt, H. Raddatz, B. Beichler, B. Konieczek, D. Timmermann, C. Haubelt, and P. Danielis, "Vipmesh: A virtual prototyping framework for ieee 802.11s wireless mesh networks," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–7, 2016.

[14] T. Werthmann, M. Kaschub, M. Kühlewind, S. Scholz, and D. P. Wagner, "Vmsimint: a network simulation tool supporting integration of arbitrary kernels and applications.," in *SimuTools*, pp. 56–65, 2014.

[15] "Netlink protocol library," 2012.

[16] "Linux namespaces," 2002.

[17] "Linux wireless iw."

[18] E. L. Fernandes, E. Rojas, J. Alvarez-Horcajo, Z. L. Kis, D. Sanvito, N. Bonelli, C. Cascone, and C. E. Rothenberg, "The road to bofuss: The basic openflow userspace software switch," *Journal of Network and Computer Applications*, p. 102685, 2020.

[19] P. Pinyoanuntapong, T. Pothuneedi, R. Balakrishnan, M. Lee, C. Chen, and P. Wang, "Sim-to-real transfer in multi-agent reinforcement networking for federated edge computing," *CoRR*, vol. abs/2110.08952, 2021.

[20] "wlanprofessionals mcs snr rssi chart."

[21] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[22] S. Caldas, S. Meher Karthik Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A Benchmark for Federated Settings," *arXiv e-prints*, p. arXiv:1812.01097, Dec. 2018.

APPENDIX A: Tests on existing simulators

This section includes the tests that are done on the existing simulators which support mesh networks. As each simulator have different kind of architectures, The wireless standard evaluated will be changing. But the main focus remains to see how realistic a simulation is able to replicate the real world MCMR mesh environment.

1. Mininet Wifi

   Table A.1 shows the tests conducted on mininet-wifi with respect to wireless multi-hop network operating in MCMR only. The topology evaluated are as flows

   1.1 Single Hop testing

   $$Station1 <------> Station2$$

   1.2 Multi Hop testing

   $$Station1 <------> Station2 <------> Station3$$

   The above two topologies are test for 20,40,80Mhz bands on a, n, ac modes. The distance between stations are test at 40,52,80,105 meters. The propagation models tested are logDistance, Friis, LogNormalShadowing. The max throughput observed is around 40Mbps on single hop tests in 20Mhz channel width and 20Mbps on Multi-hop at the same channel width. By changing the channel width to 40Mhz or 80Mhz the throughput results are similar which are 10-15Mbps in all the bands a,n,ac. For convenience below visualization graph(Figure A.1) shows the average throughput of tests in each band and Table A.1 shows every test result in different tunable parameters.

Table A.1: Mininet-Wifi Tests

Topology under evaluation	Band	Channel Width(MHz)	Propagation model	Tx power	Node distance	latency(avg)ms	Throughput(mode=a)	Throughput(mode=n)	Throughput(mode=ac)
1	5Ghz	20	logDistance,exp=4	15dBm	40m	1.073	9.28Mbps	8.93Mbps	9.27Mbps
2	5Ghz	20	logDistance,exp=4	15dBm	80m	1.932	4.96Mbps	4.95Mbps	5Mbps
1	5Ghz	20	logDistance,exp=4	15dBm	52m	0.335	4.48Mbps	4.11Mbps	4.47Mbps
2	5Ghz	20	logDistance,exp=4	15dBm	105m	0.383	2.31Mbps	2.16Mbps	2.32Mbps
1	5Ghz	40	logDistance,exp=4	15dBm	40m	11.714	8.44Mbps	8.15Mbps	9.17Mbps
2	5Ghz	40	logDistance,exp=4	15dBm	80m	19.43	4.62Mbps	4.50Mbps	3.33Mbps
1	5Ghz	40	logDistance,exp=4	15dBm	52m	1.153	1.24Mbps	1.37Mbps	1.23Mbps
2	5Ghz	40	logDistance,exp=4	15dBm	105m	12.466	0.681Mbps	0.726Mbps	0.908Mbps
1	5Ghz	80	logDistance,exp=4	15dBm	40m	0.312	8.99Mbps	8.26Mbps	9.31Mbps
2	5Ghz	80	logDistance,exp=4	15dBm	80m	0.337	3.4Mbps	4.59Mbps	3.33Mbps
1	5Ghz	80	logDistance,exp=4	15dBm	52m	17.823	1.31Mbps	4.56Mbps	4.45Mbps
2	5Ghz	80	logDistance,exp=4	15dBm	105m	28.728	0.908Mbps	2.35Mbps	2.33Mbps
1	5Ghz	20	Friis,exp=4	15dBm	40m	0.785	17.8Mbps	30.5Mbps	28.4Mbps
2	5Ghz	20	Friis,exp=4	15dBm	80m	1.183	29.2Mbps	31Mbps	29.3Mbps
1	5Ghz	20	Friis,exp=4	15dBm	80m	0.779	17.8Mbps	30.7Mbps	28.7Mbps
2	5Ghz	20	Friis,exp=4	15dBm	140m	0.798	31.9Mbps	29.3Mbps	29.6Mbps
1	5Ghz	40	Friis,exp=4	15dBm	40m	8.809	6.06Mbps	5.26Mbps	10.2Mbps
2	5Ghz	40	Friis,exp=4	15dBm	80m	8.417	10.3Mbps	10.3Mbps	10.0Mbps
1	5Ghz	40	Friis,exp=4	15dBm	80m	8.632	11.2Mbps	12Mbps	11.8Mbps
2	5Ghz	40	Friis,exp=4	15dBm	140m	8.586	11.8Mbps	12Mbps	11.7Mbps
1	5Ghz	80	Friis,exp=4	15dBm	40m	8.751	11.5Mbps	11.6Mbps	11.9Mbps
2	5Ghz	80	Friis,exp=4	15dBm	80m	8.657	12Mbps	11.8Mbps	11.7Mbps
1	5Ghz	80	Friis,exp=4	15dBm	80m	10.686	11.6Mbps	11.6Mbps	11.3Mbps
2	5Ghz	80	Friis,exp=4	15dBm	140m	10.619	11.9Mbps	11.6Mbps	11.7Mbps
1	2.4Ghz	20	logDistance,exp=4	15dBm	40m	2.672	NA	12.4Mbps	13.1Mbps
2	2.4Ghz	20	logDistance,exp=4	15dBm	80m	1.885	NA	6.32Mbps	6.92Mbps
1	2.4Ghz	20	logDistance,exp=4	15dBm	15m	0.777	NA	31.6Mbps	31.1Mbps
2	2.4Ghz	20	logDistance,exp=4	15dBm	30m	1.093	NA	24.5Mbps	21.8Mbps
1	2.4Ghz	20	Friis,exp=4	15dBm	40m	0.732	NA	31.7Mbps	33.6Mbps
2	2.4Ghz	20	Friis,exp=4	15dBm	80m	0.754	NA	31.6Mbps	32Mbps
1	2.4Ghz	20	Friis,exp=4	15dBm	15m	0.745	NA	31.8Mbps	32.6Mbps
2	2.4Ghz	20	Friis,exp=4	15dBm	30m	0.722	NA	32.5Mbps	32.1Mbps
1	2.4Ghz	20	logNormalShadowing,exp=4	15dBm	40m	3.923	NA	18.8Mbps	19.3Mbps
2	2.4Ghz	20	logNormalShadowing,exp=4	15dBm	80m	0.802	NA	11.54Mbps	11.4Mbps

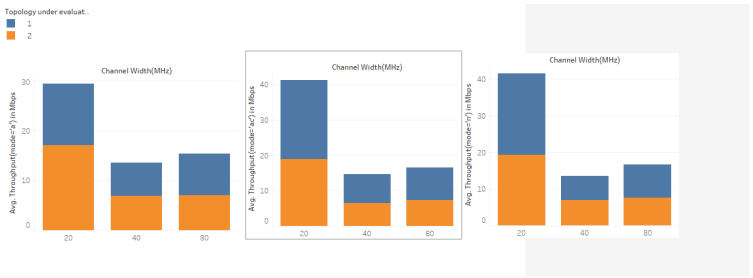


Figure A.1: Mininet-wifi test

2. Wmediumd

The test topology for wmediumd is similar to Mininet-wifi. The wmediumd has no options for different modes like mininet wifi. So the tests are limited and only single and multihop are done in this case as show below.

2.1 Single Hop testing

$$Station1 < - - - - - - > Station2$$

2.2 Multi Hop testing

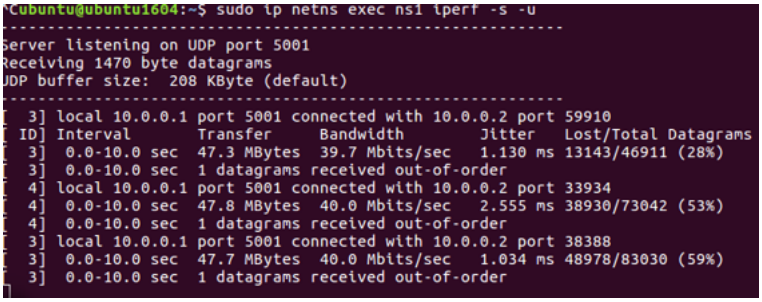


Figure A.2: Wmediumd Single Hop test

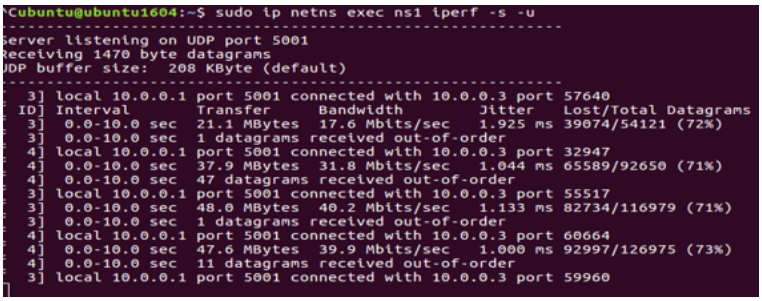


Figure A.3: Wmediumd Multi Hop test

$$Station1 < - - - - - > Station2 < - - - - - > Station3$$

The throughput results(Figure A.2 A.3) are similar to mininet-wifi which are around 40Mbps for single hop and for multi-hop its different ,wmediumd is able to get 35-40Mbps on multihop.