rCarleton University

SYSC 4701 Communications Systems

Fall 2023

Enhancing Content Delivery (CD) in Vehicular Ad-Hoc Networks (VANETs) using Software-Defined Networking (SDN)

Final Project Report

By:Ahmed Abdelaziz and Akuei Minyang

Department of Systems and Computer Engineering Faculty of Engineering Carleton University

Table of Contents

Table of figures

Abstract

The handover process is crucial for maintaining seamless data transfer in Vehicular Ad-Hoc Networks (VANETs) as mobile devices move between cells or base stations. In this study, we define VANETs as mobile devices navigating through predefined Access Points (APs) within a traffic block. While Access Point (AP) to AP communication ensures high reliability, Vehicle to Infrastructure (V2I) communication often faces reliability challenges.

This paper explores the dynamic nature of the handover process in VANETs, where vehicles serve as intermediaries for the seamless transfer of data between mobile devices and APs. Specifically, we demonstrate the feasibility of utilizing other vehicles as relay points in the handover process, introducing a dynamic approach that contributes to more reliable content delivery.

Acknowledgements

We would like to take this section to thank Professor Chung Horng-Lung and Masters student Ethan Fettes for their constant support throughout the semester. They have been very helpful and none of this research provided could have been done without their guidance.

SECTION 1.0 Introduction

Vehicular Ad-Hoc Networks (VANETs), comprising wirelessly connected mobile stations, stand at the forefront of this report. The configuration of these VANETs is executed using Mininet-wifi, a platform designed for realistic wireless network emulation. It's essential to note a distinct difference between Mininet-Wifi and Mininet: the former lacks wireless links and connectivity. Consequently, any code presented in the appendix or subsequent chapters accounts for this distinction.

While Mininet-Wifi accurately models Wi-Fi networks, it falls short in replicating other wireless technologies like 5G and LTE. The primary differentiator lies in the network strength, often quantified by the Received Signal Strength Indicator (RSSI).

The project's primary objective revolves around establishing connectivity between vehicles and Access Points (APs), as well as between APs and servers. This connectivity must persist, unaffected by the mobility of individual cars. To achieve this, we leverage IEEE802.11r, a specific Wi-Fi protocol uniquely suited for dynamic handovers. The term 'dynamic handover' signifies the ability to maintain continuous connectivity between a moving vehicle (Car X) and a specific Access Point (AP Y), even as the vehicle moves through a dynamic traffic bloc.

This introduction sets the stage for a detailed exploration of VANETs, Mininet-wifi configurations, and the pivotal role of IEEE802.11r in achieving seamless connectivity within dynamic vehicular environments.

1.1 Problem Solution:

To address content delivery in a wireless network, an SDN-based solution is used. It is key to note the use for SDN, to relay logs such as connectivity within AP1 compared to connectivity of AP4, a controller is needed. This is why an SDN-based solution is needed. The controller has a complete overview of the network topology such as:

- Neighboring APs
- Neighboring Vehicles
- Vehicle RSSI to AP connection
- Number of Vehicles connected.

Beyond this, once a vehicle is connected within the network, an attempt at file transfer is done through Infrastructure to Vehicle (V2I) and Vehicle to Vehicle (V2V).

SECTION 2 Methodology

The following section details the way in which the project was carried out. To start the project, it was debated which controller is best. The only two choices which were seriously considered, RYU and POX controllers. The installation of RYU controller was deemed as the ideal because it optimized the use of background scanning. Background scanning is a continuous, non-disruptive search for available networks

and access points while maintaining constant communication which are a priority. Background scanning is accessed to APs only as they check which APs are in area of connectivity and the same for ongoing cars. The point of background scanning to pick up or drop connectivity with mobile stations that are in constant need of updates. However, the installation of RYU controller proved hard, and the project was attempted with a POX controller. The handover was proven successful even though there were warnings for the use of a POX controller. The efficiency of the handover is later discussed within this report.

2.1 Software Defined Networking

It is important to acknowledge that without the use of SDN, then handover would be impossible. The attempt at which a mobile station can be configured and moved this could be done without the use of an SDN. This additionally goes for a car pinging another car if they both can create a wireless link, meaning they are both very close to each other. However, if car A moved ahead of car B it could not connect due to the fact that there is no handover, there is no absolute relation. Once car A moves on, it can only ping AP1, and car 2 is connected to AP2. They are not within reach and therefore cannot connect.

Through the introduction of SDN Controller, all APs have an absolute relation of each other, this means that AP1 understands where AP7 is, for example. Therefore, regardless of the mobility of car A relative to car B, they will always connect if the APs both cars are connected to are within the same network. We modelled the following for handover address to be a specific link, this will be later addressed in section 3

2.2 Content Delivery Network (CDN)

An aim of this project was to send files between cars while moving. To do this first we had the cars ping each other to see if there was an internet connection between car to car. The same would be done between car to server. The server can communicate with the car and vice versa. Following a ping, we would set up a simple HTTP server, this server would be for the car, the one where info would be retrieved from. Then the car would use wget to get this server from the car, think of the car being a client to the server. If a simple HTTP server and wget command could be done successfully, then the server can create files which can be retrieved by the car. In our project, simple text files were created but ideally, we could create large HD video files to be transmitted just as well due to the fact that constant connection would be between server and vehicle. However, this is more spoken about in Section 4 to discuss improvements.

2.3 Fast Roaming re-association Process (IEEE 802.11r)

IEEE 802.11r introduces Fast Basic Service Set (BSS) Transition, streamlining the handover process for users moving between different base stations. This transition involves disconnecting from one base station and swiftly connecting to another, significantly reducing interruption time during handovers. By eliminating much of the handshake overhead and optimizing key exchange, IEEE 802.11r aims to enhance the user experience for real-time applications, particularly in scenarios where uninterrupted connectivity is crucial, such as in voice and video applications. This improvement is particularly beneficial for users moving across various access points within the same mobility domain.

2.4 Mininet-Wi-Fi

Mininet-Wi-Fi is an addition of wireless network capacity to the Mininet network emulator. Mininet is an open-source network emulator that allows virtual networks to be hosted on a single workstation. It is widely used in network construction, education, and testing.
Mininet-Wi-Fi extends Mininet to include wireless devices and connectivity, allowing users to mimic and experiment with wireless networks in addition to traditional wired networks. Among the wireless capabilities it offers are stations (wireless clients), ad hoc networks, mesh networks, access points (APs), and mobility models for mobile nodes.

2.5 Background Scanning

Background module plays a crucial role in facilitating background scanning for seamless roaming within a single network where all Access Points (APs) share the same SSID. The background scanning process involves four distinct phases:

1. Discovery (Probing) Phase:

Purpose: During this phase, a station (STA) and an access point (AP) identify each other, agree on security capabilities, and establish an association for future communication.
Decisions made: The STA and AP determine specific techniques for:

- Confidentiality and MPDU integrity protocols to protect unicast traffic (communication between the STA and AP).
- Authentication method.
- Cryptography key management approach.

2. Authentication Phase:

Purpose: The STA and an authentication server (AS) mutually prove their identities. Non-authenticated traffic between the STA and AS is blocked until a successful authentication transaction is completed. The AP serves as a mediator, forwarding traffic between the STA and AS without participating in the authentication process.

3. Key Negotiation:

Purpose: The AP and STA perform operations leading to the generation and placement of cryptographic keys on both entities. Key exchange frames are exchanged exclusively between the AP and STA.

4. Association/Association:

Mechanism: This phase manages how stations associate with access points in the network.
Strategy: The association mechanism is based on selecting the AP with the strongest signal, ensuring a robust and reliable connection.
These four phases collectively contribute to the efficient and secure background scanning process, enhancing the roaming experience within the network, particularly in scenarios where uninterrupted connectivity is essential. The figure demonstrates how the process is done.

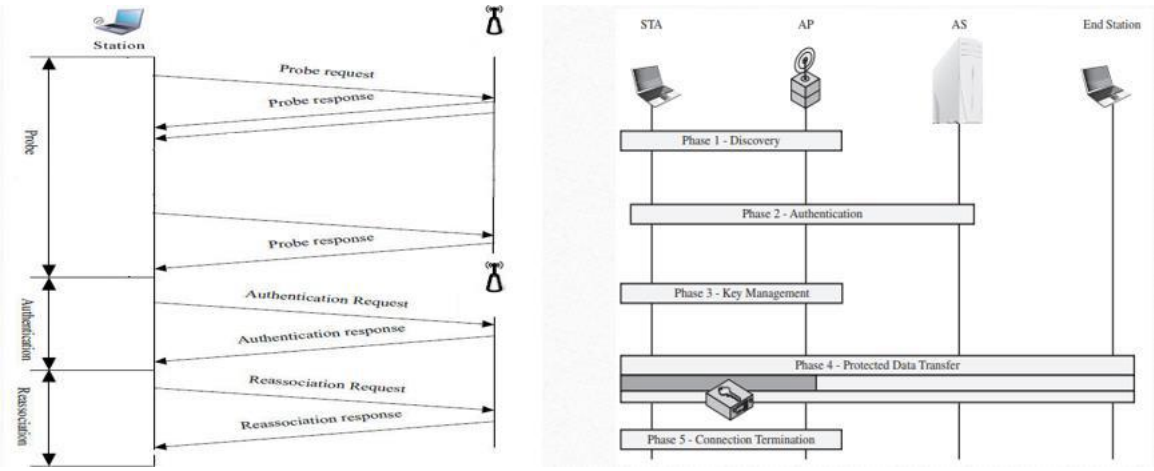


Figure 1: Illustration of background scanning

SECTION 3 System Design

The system was made up of the following components:

- 4 Access Points
- 5 Cars
- 2 Servers

The components were configured in

3.1 Configuration of V2V

A distinction had to be made between communication between car to AP (V2I) and car to car (V2V). The idea is that within a certain traffic block there are multiple APs, 2 servers on opposite sides of these physical base stations and multiple mobile cars. A car associates with another car using dependent on two situations: if a car is near another car, if a car is near an AP. If the cars can connect while being together then APs need not to be invoked. However, if a car is two or three streets down, the order of

communication is as follows: Car X->APX->APY->CarY. This is firstly set up by the configuration of the car.

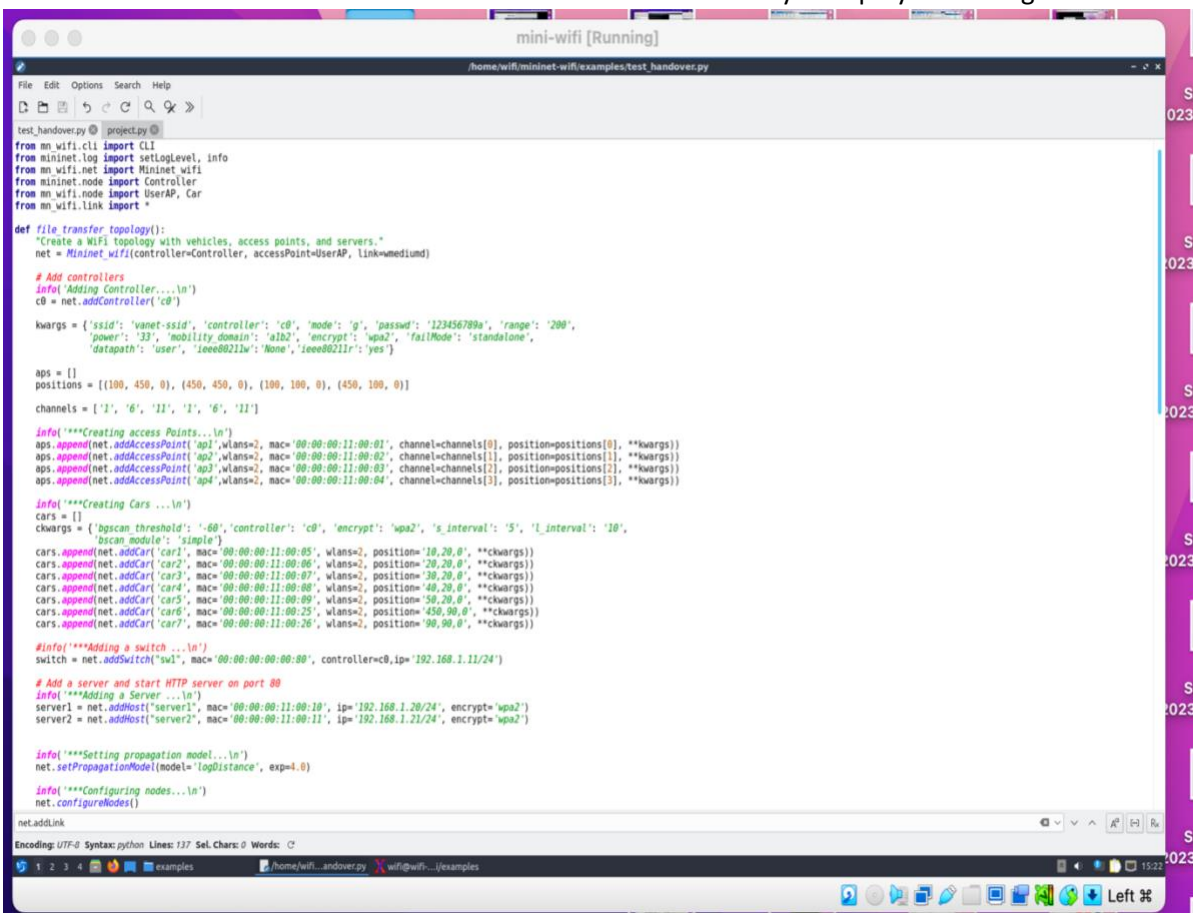


Figure 2: Cars and AP Configuration

This is how the following cars were set up. Every car had background scan settings which allow for handover. Cars had mac addresses which could be used for Layer 2 communication and additionally this could prove useful whenever rerouting.

```
net.plotGraph(max_x=600, max_y=600)

net.setMobilityModel(time=0, model='RandomDirection', nodes=cars, max_x=600, max_y=600)
net.startMobility(time=0, mob_rep=1, reverse=False,ac_method='ssf')

p = [{}, {}, {}, {}, {}, {}]
p[0] = {'position': '40,30,0'}
p[1] = {'position': '40,30,0'}
p[2] = {'position': '40,30,0'}
p[3] = {'position': '40,30,0'}
p[4] = {'position': '40,30,0'}
p[5] = {'position': '500,500,00'}
for i in range(5):
    net.mobility(cars[i], 'start', time=i, **p[i])
    net.mobility(cars[i], 'stop', time=(20 + int(i)), **p[5])
net.stopMobility(time=26)

for car in cars:
    net.addLink(car, intf= car.wintfs[0].name, cls=mesh, channel=6)
```

Figure 3: Setting cars up for mobility

Without Mobility, a test for handover would not be completed. The project started by having 4 stationary cars and testing pinging car to car. Following this, we set the cars to move at random speeds to move as that would introduce a new obstacle to our testing. Following this, the same objective of testing would be performed. As shown in figure 3, it shows that cars could ping each other during motion. Figure 4 represents a visualization of what the configuration is.

```
mininet-wifi> car1 ping car2
PING 192.168.0.6 (192.168.0.6) 56(84) bytes of data.
64 bytes from 192.168.0.6: icmp_seq=1 ttl=64 time=1012 ms
64 bytes from 192.168.0.6: icmp_seq=2 ttl=64 time=9.75 ms
64 bytes from 192.168.0.6: icmp_seq=3 ttl=64 time=15.1 ms
64 bytes from 192.168.0.6: icmp_seq=4 ttl=64 time=1.58 ms
64 bytes from 192.168.0.6: icmp_seq=5 ttl=64 time=0.930 ms
```

Figure 4: Terminal of V2V communication

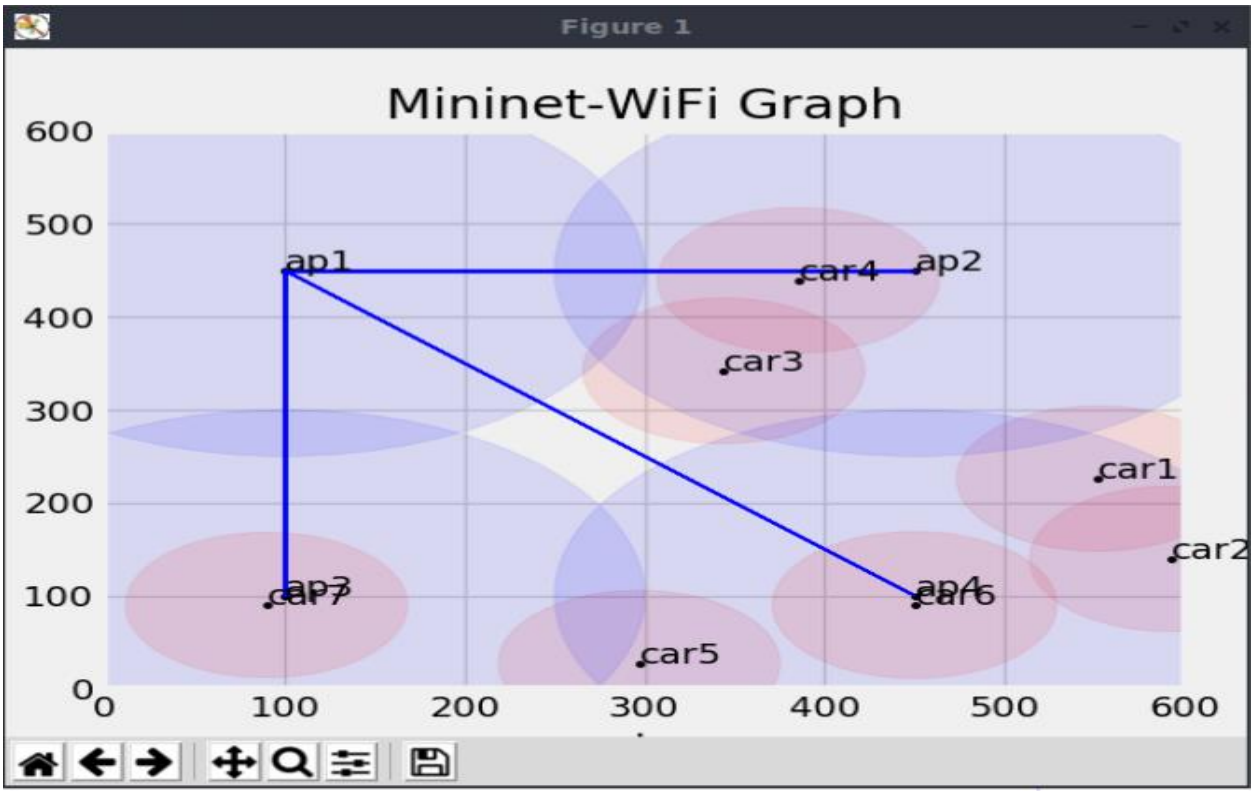


Figure 5: Visualization of testing

3.2 Configuration of V2I

The cars were set up to communicate with the infrastructure, in our case servers and Aps, along their path using the carx-mp1 wireless network of the cars. We aimed at using this configuration to allow for information exchange from the server to the cars hence allowing us to emulate content delivery network. The Aps are used as gateways that store no data; they always forward any data onto servers. This decision was made to help emulate a more difficult environment, if cars are automatically linked with servers, then a problem of mobility would be pointless. There would be no difficulty in the environment designed. Therefore it was designed to have Access Points connected to a switch that is linked to both servers. Whenever a car pings a server, it goes through the following scheme discussed. The AP a ping request would go through is dependent on which AP a car is associated with, based on previous reports it was felt best to use strongest signal first approach. This approach is reliant on car returning to controller RSSI values of APs and controller associating car with AP with highest RSSI value. The following figure shows the configuration of the other subnet, including APs, a switch, and server.

```
info('***Adding a switch ...\n')
switch = net.addSwitch("sw1",mac='00:00:00:00:00:88')

info('***Adding a Server ...\n')
server1 = net.addHost("server1",mac='00:00:00:00:01:01',ip='192.168.1.1/24')


info('***Setting propagation model...\n')
net.setPropagationModel(model='logDistance', exp=3.0)

info('***Configuring nodes...\n')
net.configureNodes()

net.addLink(switch,server1)

for i in range(len(aps)-1):
    net.addLink(aps[i], aps[i+1])
    net.addLink(aps[i],switch)

for car in cars:
    net.addLink(car, intf=car.params['wlan'][1], cls= mesh, ssid='meshV2V', channel='5')
```

Figure 6: Showing configuration of subnetwork

3.3 Content Distribution throughout Mobility

Once mobility is achieved and cars can still ping each other, content distribution is relatively easy because there is already internet connectivity. In the testing performed there is two tests:

i) Content Distribution between car to car
ii) Content Distribution between car to server.

For content to be created, it was only tested using simple text files, an evolution upon this will be discussed in section 6. Figure 6 shows an example of content distribution between car to car and figure 7 shows the same between car to server.

```
wifi@wifi-virtualbox...ninet-wifi/examples
        TX packets 18  bytes 2176 (2.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet-wifi> car1 wget -O -car5 192.168.1.5
--2023-12-08 15:48:40--  http://192.168.1.5/
Connecting to 192.168.1.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4810 (4.7K) [text/html]
Saving to: '-car5'

-car5               100%[===================>]   4.70K  --.-KB/s    in 0.01s

2023-12-08 15:48:40 (448 KB/s) - '-car5' saved [4810/4810]

mininet-wifi>
```

Figure 7: File transfer in V2V

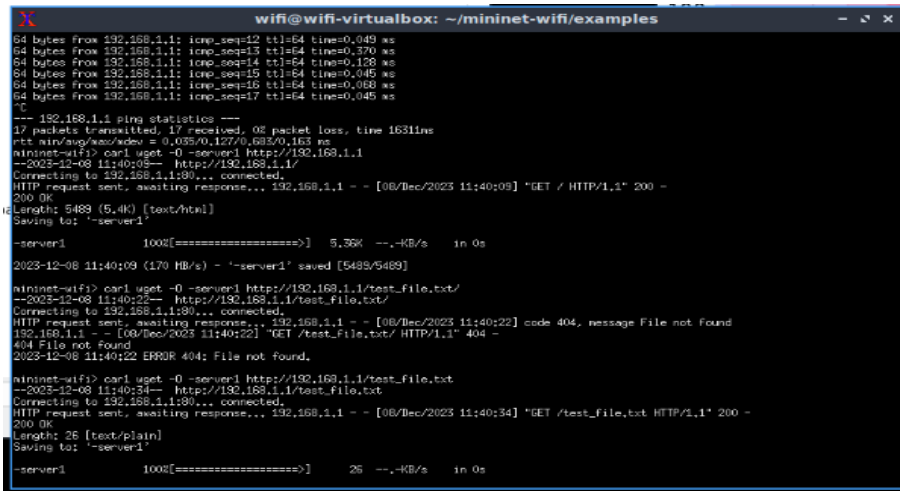


Figure 8: File transfer in V2I

SECTION 4.0 Results and Evaluation

The primary objective at the project's onset was to establish successful file transfer, a milestone that was notably achieved, particularly in the context of Vehicle to Vehicle (V2V) communication. However, during testing, there were challenges with the connectivity of car to server. Two program versions emerged: one ensuring constant security in V2V communication, and another prioritizing the security of Vehicle to Infrastructure (V2I) communication. Notably, achieving V2I security came at the expense of implementing absolute handover. In this context, absolute handover signifies that, irrespective of a car's location, if connected to an Access Point (AP), it can always ping another car, even when the two cars are connected to APs on opposite ends.

Conversely, when successful server-to-car communication was established, V2V communication suffered, primarily in proximity. This approach, favoring reliance on the server, involved cars requesting files from servers rather than attempting direct connections to out-of-reach cars. This shift toward server-centric communication is viewed as a positive development, introducing an added layer of protection. With this approach, cars are restricted from accessing one another, enhancing security, while maintaining the ability to access public files hosted on servers. This strategic balance reinforces the network's security posture and aligns with the project's overarching goals. An implementation of file caching is not necessary since the server has saved what the car sends, which is an instantaneous way of representing caching. The following figure shows the difference between requesting a file from server that is there comparatively to a request for file that is not there.

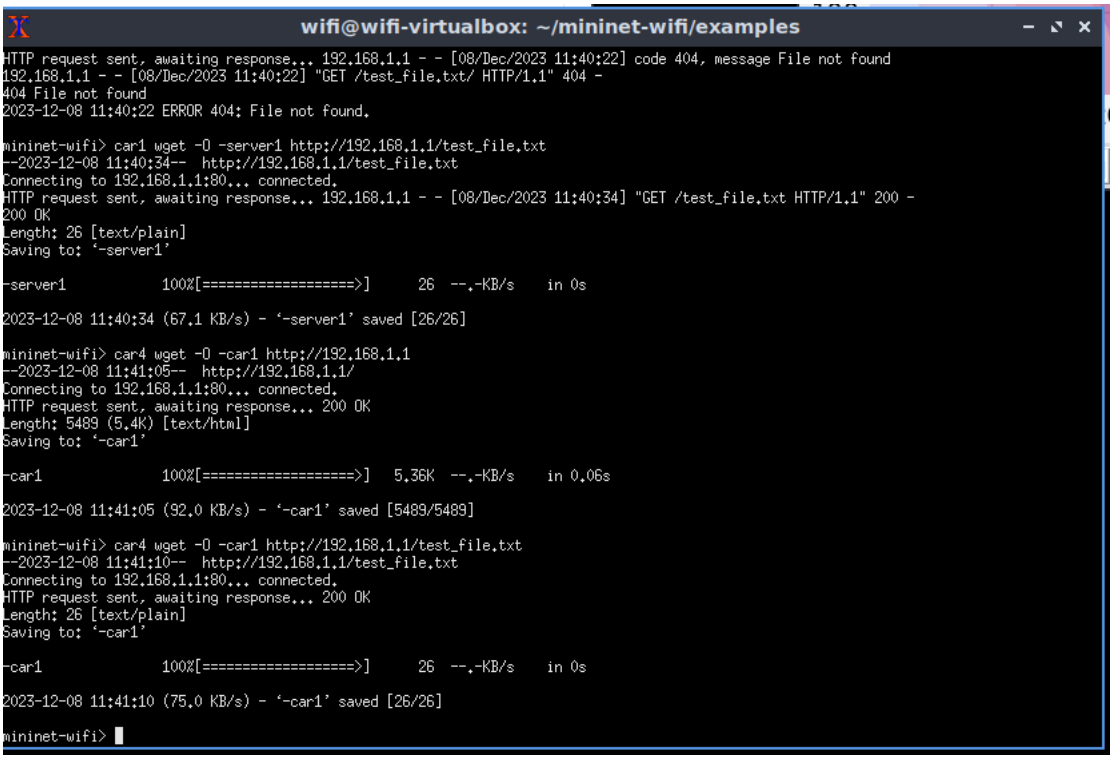


Figure 9: Demonstration of server caching

SECTION 5 TECHNICAL PROBLEMS ENCOUNTERED 5.1 Mininet-WiFi

5.1 Mininet-WiFi

To achieve SDN-CD VANETs network, we decided to use Mininet-wifi because it has wireless capabilities that would allow for file exchange and communication between cars while the cars were in motion. The simulator proved to be a steep learning curve as it is very different from regular Mininet. We were able to figure out how to make it work and, in the end, use it to implement the system.

5.2 SDN Controller

We had a choice to use ODL, POX or RYU controllers to implement the SDN capabilities of the network. The ODL controller did not work as we wanted. Our virtual machines could not connect to ODL server and therefore we could not view the work we were doing.
We tried to use RYU controller, but we were not able to download it on one of our laptops. Since the project was to be done by 2 students, we opted to use another controller that will allow collaboration. Due to the ODL controller not working as expected and the inability to download the RYU controller on what of the laptops, we unanimously decided to use the POX controller. The POX controller was efficient enough for the implementation of our network topology and therefore we used it.

5.3 Topology Creation

We had several topologies that we tried to implement for the network. We decided early on to use 2 servers, 5 cars and 4 Access Points (Aps). The cars were to move between the APs while communicating with other devices. We debated having switches in the topology and first decided to implement the topology without the switches, but it proved futile as the cars could not communicate with the servers but only with themselves. Due to this problem, we added the switch to the network and the cars could now communicate with the servers hence enabling us to emulate SDN-CD.

We also had a dilemma on whether to manually start the servers or initialize them in the code. We tried both and, in the end, decided to use both. This allowed us to set up cars as servers and by observing the mobility, we request content from the nearest car to the server. The servers were started together with the network and manually to change the port number we were trying to use. The ability to manually set up the code or dynamically start the servers was very helpful in the sense that it allowed us to test and see where the network was not working right and therefore make necessary changes.

SECTION 6 FUTURE WORK

The following are some of topics future students can work on:

1. Achievement of absolute handover between vehicle to vehicle
2. The use of multiple servers to increase difficulty.
3. Load balancing association between cars and APs
4. File transfer of more complex files

SECTION 7: CONCLUSION

In conclusion, our project aimed to address the critical challenges associated with content delivery in Vehicular Ad-Hoc Networks (VANETs) by leveraging Software-Defined Networking (SDN) principles. Through a comprehensive exploration of Mininet-wifi and the integration of IEEE 802.11r for efficient handovers, we sought to establish seamless connectivity between mobile vehicles, Access Points (APs), and servers.

The project successfully demonstrated the feasibility of utilizing other vehicles as relay points in the handover process, introducing a dynamic approach to enhance reliable content delivery. By implementing a robust SDN-based solution, we achieved a comprehensive network overview, including neighboring APs, neighboring vehicles, RSSI values for vehicle-to-AP connections, and the number of connected vehicles.

The distinction between Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication was carefully considered, providing flexibility based on proximity and ensuring continuous connectivity even in dynamic traffic scenarios. The implementation of IEEE 802.11r's Fast BSS Transition streamlined handovers, reducing interruption time, and enhancing the user experience, especially in real-time applications.

Despite encountering challenges with SDN controllers and topology creation, the project successfully addressed technical problems, leading to an effective and practical SDN-based VANET system. The results highlighted successful file transfer between vehicles (V2V) and between vehicles and servers (V2I), contributing to the project's overall goals.

Looking ahead, future students could focus on achieving absolute handover between vehicles, exploring load balancing strategies between cars and APs, and expanding file transfer capabilities to handle more complex data types. Our project provides a foundational framework for advancing research in SDN-enabled VANETs, aiming to enhance connectivity, reliability, and security in dynamic vehicular environments.

SECTION 7 References

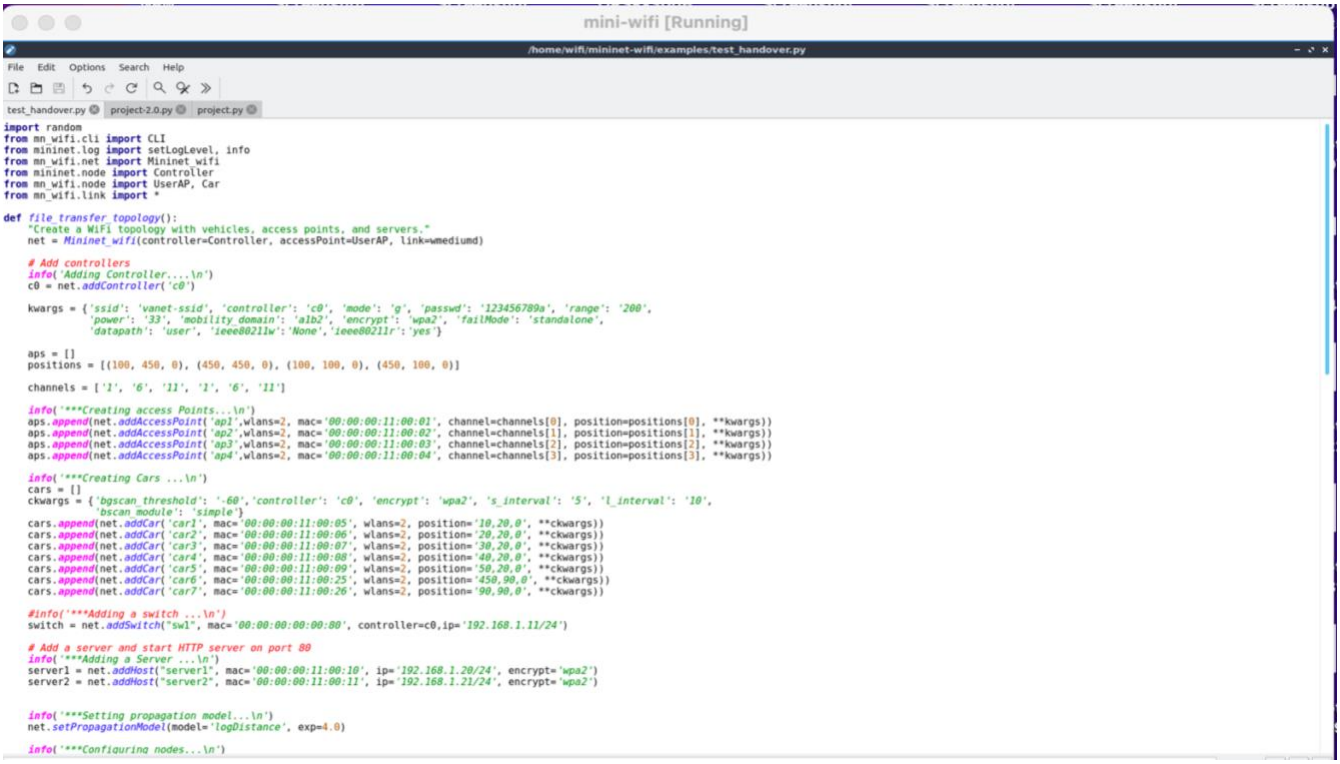
[1] Zidan.M,Ahmed.M, ‘Handover in Software-Defined Wireless Networking’ April 15,2019

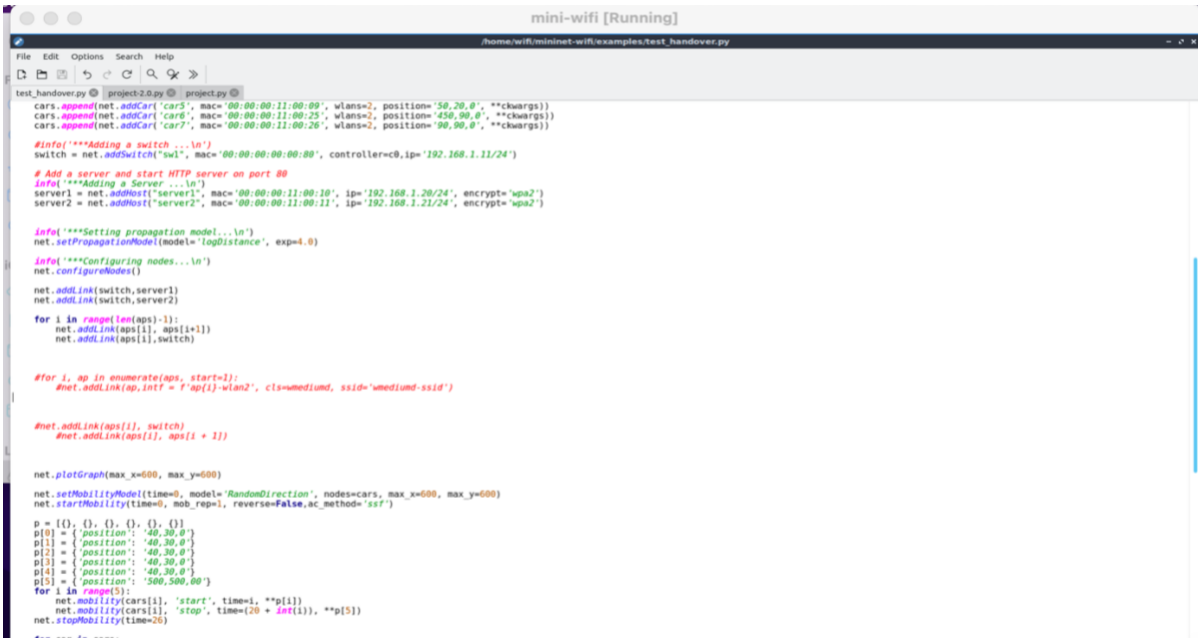
[2] Puttaswami,K,’ Enhanced Handover process for Reliable Video Streaming over SDN based Wireless Networking’ December,2019

[3] Fontes,R ‘The Mininet WiFi Book’ github.com,’ https://github.com/ramonfontes/mn-wifi-ebook’ December 9 2019

SECTION 8 Appendix

The following demonstrates both programs we had in section 4. The program which optimizes absolute handover between vehicles will be shown first.





Following this, the code that optimizes V2I communication is presented.

mini-wifi [Running]

/home/wifi/mininet-wifi/examples/project-2.0.py

File Edit Options Search Help

test_handover.py project-2.0.py project.py

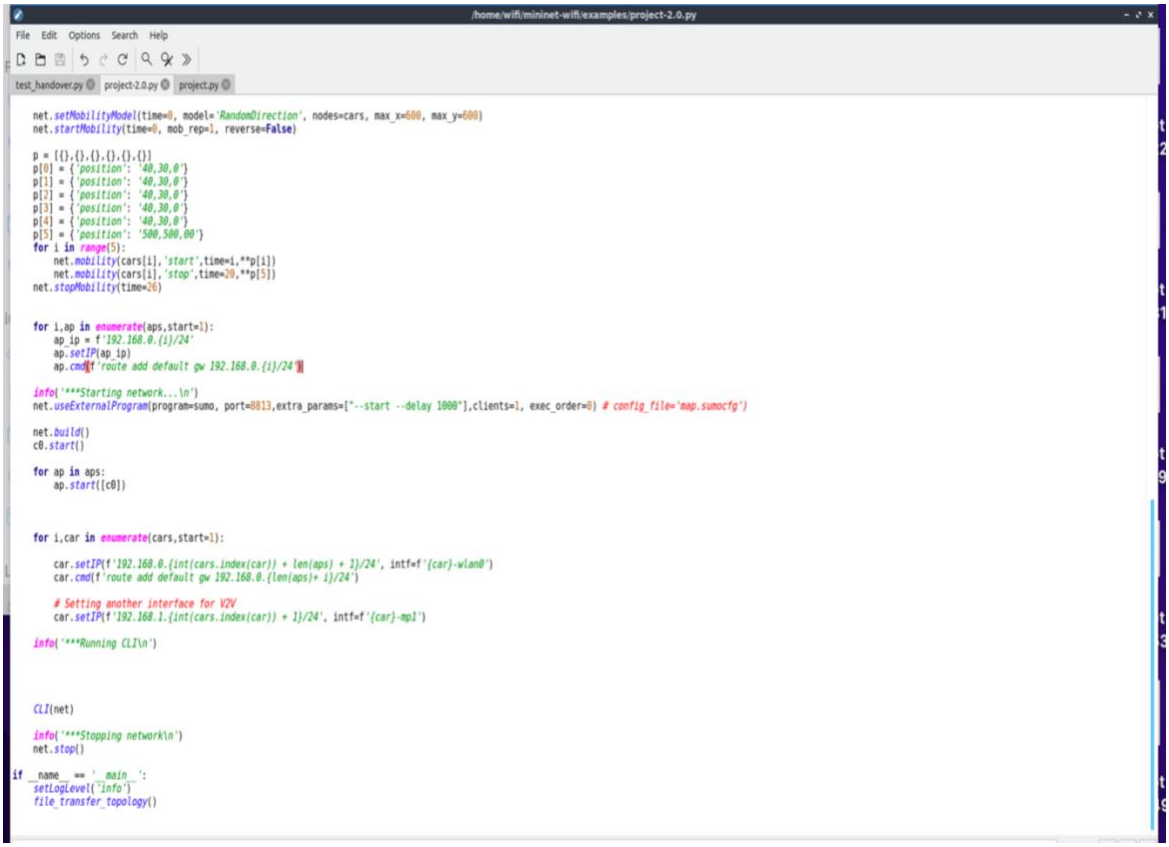
```
import random

from mn_wifi.cli import CLI
from mininet.log import setLogLevel, info
from mn_wifi.net import Mininet_wifi
from mininet.node import Controller
from mn_wifi.node import UserAP, Car, Station
from mn_wifi.link import wmediumd, ITSLink, mesh
from mn_wifi.wmediumdConnector import interference
from mn_wifi.vanet import vanet
from mn_wifi.sumo.runner import sumo

def file_transfer_topology():
    "Create a WiFi topology with vehicles, access points, and servers."
    net = Mininet_wifi(controller=Controller, accessPoint=UserAP, link=wmediumd, wmediumd_mode=interference)

    #### Add controllers ####
    c0 = net.addController('c0')

    kwargs = {'ssid': 'vanet-ssid', 'mode': 'g', 'passwd': '123456789a',
              'encrypt': 'wpa2', 'failMode': 'standalone', 'datapath': 'user', 'ieee80211r': 'yes'}
    aps = []
    positions = [(100, 450, 0), (500, 450, 0), (100, 150, 0), (500, 150, 0)]

    channels = ['1', '6', '11', '1', '6', '11']

    info('***Creating access Points...\n')
    for i in range(1, 5):
        ap = net.addAccessPoint(f'ap{i}', mac=f'00:00:00:11:00:0{i}', channel=channels[i-1],
                                position=positions[i-1],range = 300, **kwargs,mobility_domain = 'a1b2')
        aps.append(ap)

    info('***Creating Cars ...\n')
    cars = []
    for i in range(1, 6):
        car = net.addStation(f'car{i}', wlans=2, encrypt=['wpa2', ''],
                  bgscan_threshold=-60, s_interval=5, l_interval=10,bgscan_module = "simple")
        cars.append(car)

    info('***Adding a switch ...\n')
    switch = net.addSwitch("sw1",mac='00:00:00:00:00:80')

    info('***Adding a Server ...\n')
    server1 = net.addHost("server1",mac='00:00:00:00:01:01',ip='192.168.1.1/24')


    info('***Setting propagation model...\n')
    net.setPropagationModel(model='logDistance', exp=3.0)

    info('***Configuring nodes...\n')
    net.configureNodes()

    net.addLink(switch,server1)

    for i in range(len(aps)-1):
        net.addLink(aps[i], aps[i+1])
```

/home/wifi/mininet-wifi/examples/project-2.0.py

File Edit Options Search Help

test_handover.py project-2.0.py project.py

```
    net.addLink(switch,server1)

    for i in range(len(aps)-1):
        net.addLink(aps[i], aps[i+1])
        net.addLink(aps[i],switch)

    for car in cars:
        net.addLink(car, intf=car.params['wlan'][1], cls= mesh, ssid='meshV2V', channel='5')


    net.plotGraph(max_x=600, max_y=600)

    for i in range(5):
        cars[i].coord = ['0,0,0','0,0,0','0,0,0']

    net.setMobilityModel(time=0, model='RandomDirection', nodes=cars, max_x=600, max_y=600)
    net.startMobility(time=0, mob_rep=1, reverse=False)

    p = [{},{},{},{},{},{}]
    p[0] = {'position': '40,30,0'}
    p[1] = {'position': '40,30,0'}
    p[2] = {'position': '40,30,0'}
    p[3] = {'position': '40,30,0'}
    p[4] = {'position': '40,30,0'}
    p[5] = {'position': '500,500,00'}
    for i in range(5):
        net.mobility(cars[i],'start',time=i,**p[i])
        net.mobility(cars[i],'stop',time=20,**p[5])
    net.stopMobility(time=26)


    for i,ap in enumerate(aps,start=1):
        ap_ip = f'192.168.0.{i}/24'
        ap.setIP(ap_ip)
        ap.cmd(f'route add default gw 192.168.0.{i}/24')

    info('***Starting network...\n')
    net.useExternalProgram(program=sumo, port=8813,extra_params=["--start --delay 1000"],clients=1, exec_order=0) # config_file='map.sumocfg')

    net.build()
    c0.start()

    for ap in aps:
        ap.start([c0])



    for i,car in enumerate(cars,start=1):

        car.setIP(f'192.168.0.{int(cars.index(car)) + len(aps) + 1}/24', intf=f'{car}-wlan0')
        car.cmd(f'route add default gw 192.168.0.{len(aps)+ 1}/24')

        # Setting another interface for V2V
        car.setIP(f'192.168.1.{int(cars.index(car)) + 1}/24', intf=f'{car}-mp1')
```


/home/wifi/mininet-wifi/examples/project-2.0.py
File Edit Options Search Help
test_handover.py
project-2.0.py
project.py
net.setMobilityModel(time=0, model='RandomDirection', nodes=cars, max_x=600, max_y=600)
net.startMobility(time=0, mob_rep=1, reverse=False)
p = [{},{},{},{},{},{}]
p[0] = {'position': '40,30,0'}
p[1] = {'position': '40,30,0'}
p[2] = {'position': '40,30,0'}
p[3] = {'position': '40,30,0'}
p[4] = {'position': '40,30,0'}
p[5] = {'position': '500,500,00'}
for i in range(5):
net.mobility(cars[i],'start',time=i,**p[i])
net.mobility(cars[i],'stop',time=20,**p[5])
net.stopMobility(time=26)
for i,ap in enumerate(aps,start=1):
ap_ip = f'192.168.0.{i}/24'
ap.setIP(ap_ip)
ap.cmd(f'route add default gw 192.168.0.{i}/24')
info('***Starting network...\n')
net.useExternalProgram(program=sumo, port=8813,extra_params=["--start --delay 1000"],clients=1, exec_order=0) # config_file='map.sumocfg')
net.build()
c0.start()
for ap in aps:
ap.start([c0])
for i,car in enumerate(cars,start=1):
car.setIP(f'192.168.0.{int(cars.index(car)) + len(aps) + 1}/24', intf=f'{car}-wlan0')
car.cmd(f'route add default gw 192.168.0.{len(aps)+ i}/24')
# Setting another interface for V2V
car.setIP(f'192.168.1.{int(cars.index(car)) + 1}/24', intf=f'{car}-mp1')
info('***Running CLI\n')
CLI(net)
info('***Stopping network\n')
net.stop()
if __name__ == '__main__':
setLogLevel('info')
file_transfer_topology()