

KLE Technological University



KLE Technological
University

Creating Value
Leveraging Knowledge

COMPUTER NETWORKS-2

Project on Virtual Private Network

*Submitted in partial fulfillment of the requirement for
the degree of*

Bachelor of Engineering in

Computer Science and Engineering

Submitted By: Team A2-4

Apoorva Jinde	:	01FE18BCS044
Ashutosh Chauhan	:	01FE18BCS051
Ashutosh Pratap Singh	:	01FE18BCS053
Atharv Paramane	:	01FE18BCS055

Under the guidance of

Ms Vijaylakshmi Mam

**SCHOOL OF COMPUTER SCIENCE & ENGINEERING
HUBLI – 580 031 (India).**

Academic year 2020-2021

S.NO	Content
I	Introduction
	i. Problem Statement
	ii. Objectives
II	Component Requirements
	i. Hardware component Requirements
	ii. Software component Requirements
III	Topology
	i. Miniedit Visualization of Topology
	ii. Understanding of Topology
IV	Implementation
	i. Configuring the PPTP server
	ii. Configuring the PPTP clients
V	Screenshots
VI	Result Analysis
VII	Conclusions
VIII	References
IX	Drive Link

I. INTRODUCTION

i. Problem Statement

“The goal is to demonstrate and design the VPN connection so that a client connected to a private network and a client connected to a public network can communicate to a server connected to a private network. Here the type of VPN connection taken into consideration is PPTP.”

A VPN is a private network connection between two points over the internet. VPN stands for virtual private network, which is pretty self explanatory. It's creating a virtual dedicated connection over the internet instead of requiring a real dedicated connection for your network.

When you connect through a VPN, the data packets you send are first encrypted before the public network even sees them. The encrypted data is sent to the VPN server, which then forwards it to its destination. The VPN server ensures your anonymity because your IP address is hidden and the transmission transmits the address of the VPN server instead.

ii. Objectives

- a. Establish the connection between a client connected to private network and a server connected to a private network
- b. Establish the connection between a client connected to public network and server connected to a private network
- c. Provide a secure communication between the server and the clients using authentication.

II. COMPONENTS REQUIREMENTS

i. Hardware component requirements

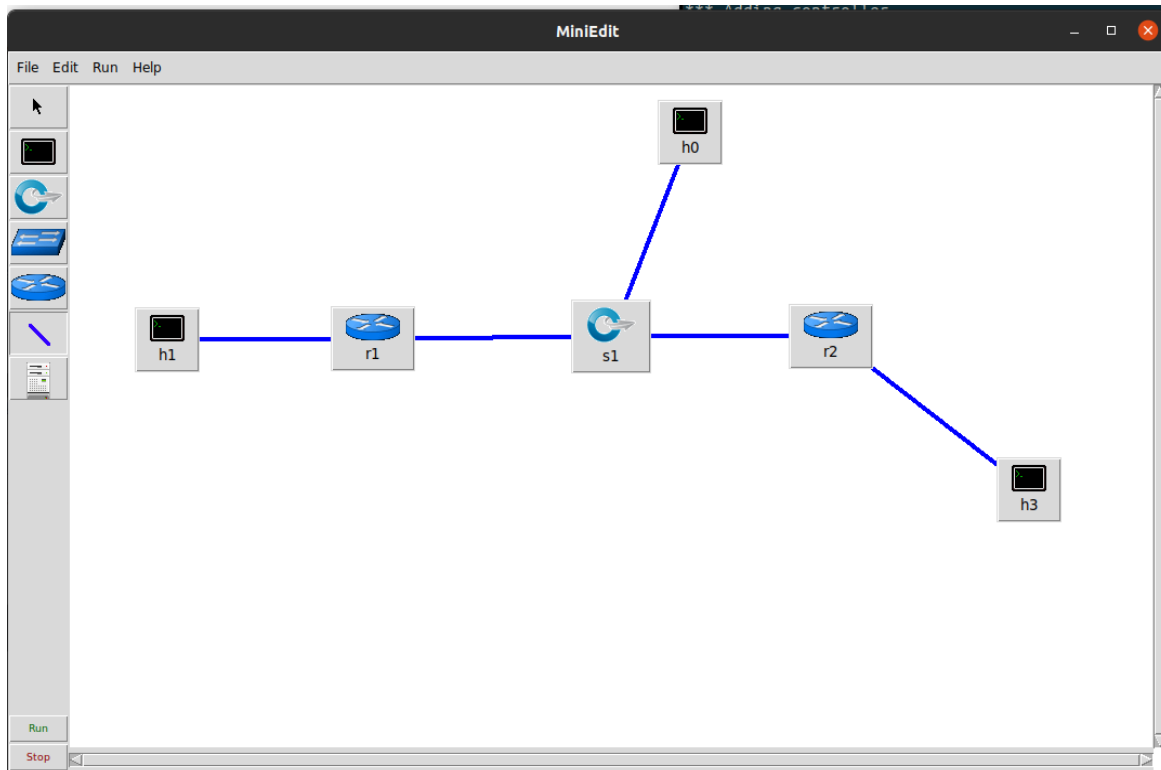
- a. RAM : 2 GB
- b. Hard Disk : 20 GB

ii. Software component requirements

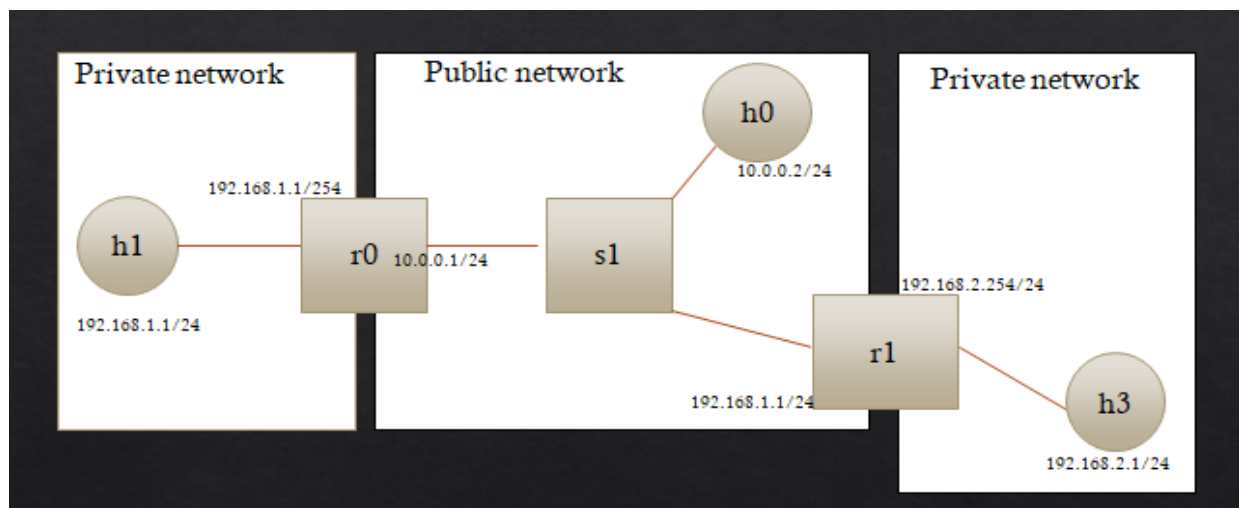
- a. Operating system: Ubuntu 16.04
- b. Mininet
- c. Mininet Dependencies + Core files
- d. PPTP server installation
- e. PPTP client installation

III. TOPOLOGY

i. Miniedit Visualization of Topology



ii. Understanding of Topology



We have created a topology consisting of two private networks and one public network. There are a total of two routers, one switch and three hosts. IP addresses are assigned according to the network.

Our topology extracted in form of python file.

```
#!/usr/bin/env python
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.link import Link,TCLink,Intf
if '__main__' == __name__:

    net = Mininet(link=TCLink)
    h1 = net.addHost('h1',ip="192.168.1.1/24")
    h2 = net.addHost('h2',ip="10.0.0.3/24")
    h3 = net.addHost('h3',ip="192.168.2.1/24")
    h4 = net.addHost('h4',ip="10.0.0.4/24")
    s0 = net.addHost('s0')
    r0 = net.addHost('r0')
    r1 = net.addHost('r1')
    net.addLink(h1, r0)
    net.addLink(r0, s0)
    net.addLink(s0, h2)
    net.addLink(s0, r1)
    net.addLink(r1, h3)
    net.addLink(s0, h4)
    net.build()
    r0.cmd("echo 1 > /proc/sys/net/ipv4/ip_forward")
    r1.cmd("echo 1 > /proc/sys/net/ipv4/ip_forward")
    s0.cmd("brctl addbr br0")
    s0.cmd("brctl addif br0 s0-eth0")
    s0.cmd("brctl addif br0 s0-eth1")
    s0.cmd("brctl addif br0 s0-eth2")
    s0.cmd("brctl addif br0 s0-eth3")
    s0.cmd("ifconfig br0 up")
    r0.cmd("ip addr add 192.168.1.254/24 brd + dev r0-eth0")
    r0.cmd("ip addr add 10.0.0.1/24 brd + dev r0-eth1")+
    h1.cmd("ip addr add 192.168.1.1/24 brd + dev h1-eth0")
    h1.cmd("ip route add default via 192.168.1.254")
    r0.cmd("iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o r0-eth1 -j MASQUERADE")
    h2.cmd("ip addr add 10.0.0.3/24 brd + dev h2-eth0")
    h4.cmd("ip addr add 10.0.0.4/24 brd + dev h4-eth0")
    r1.cmd("ip addr add 10.0.0.2/24 brd + dev r1-eth0")
    r1.cmd("ip addr add 192.168.2.254/24 brd + dev r1-eth1")
    r1.cmd("iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o r1-eth0 -j MASQUERADE")
    r1.cmd("modprobe ip_nat_pptp")
    h3.cmd("ip addr add 192.168.2.1/24 brd + dev h3-eth0")
    h3.cmd("ip route add default via 192.168.2.254")
    CLI(net)
    net.stop()
```

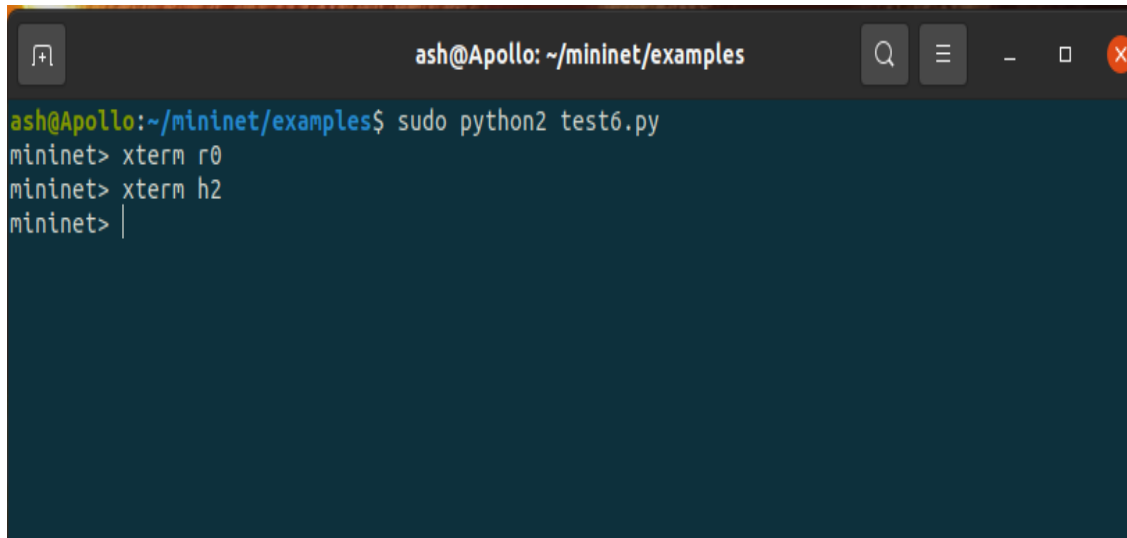
IV. IMPLEMENTATION

- i. Configuring the PPTP server:
 - 1. We have to run the constructed topology.
 - 2. From the mininet of that topology we open two xterms ro and h2.
 - 3. In ro we try to configure the ip address of the server .
 - 4. Now we will set up an account with username and password.
 - 5. Now start the pptpd server using the required commands.
 - 6. Now we need to set up the PPTP client.

- ii. Configuring the PPTP clients
 - 1. We have to create a VPN on the client side to connect to the PPTP server we created.
 - 2. After that the VPN will try to connect to the server and we will get the notification of the VPN connected.
 - 3. Set the routing table for host h2.
 - 4. Start the http server at h3. And h3 connects to h1. (h3 ping h1).

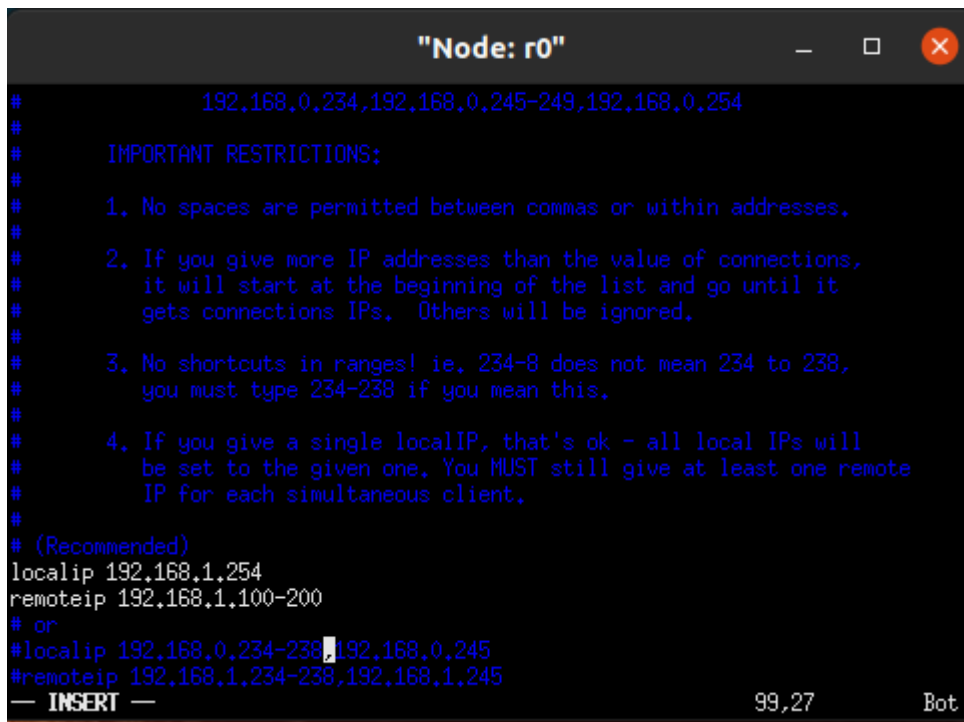
V. SCREENSHOTS

Step 1: Run the python file containing the topology created. And open the xterms r0 and h2 where r0 will be acting as the pptp server.



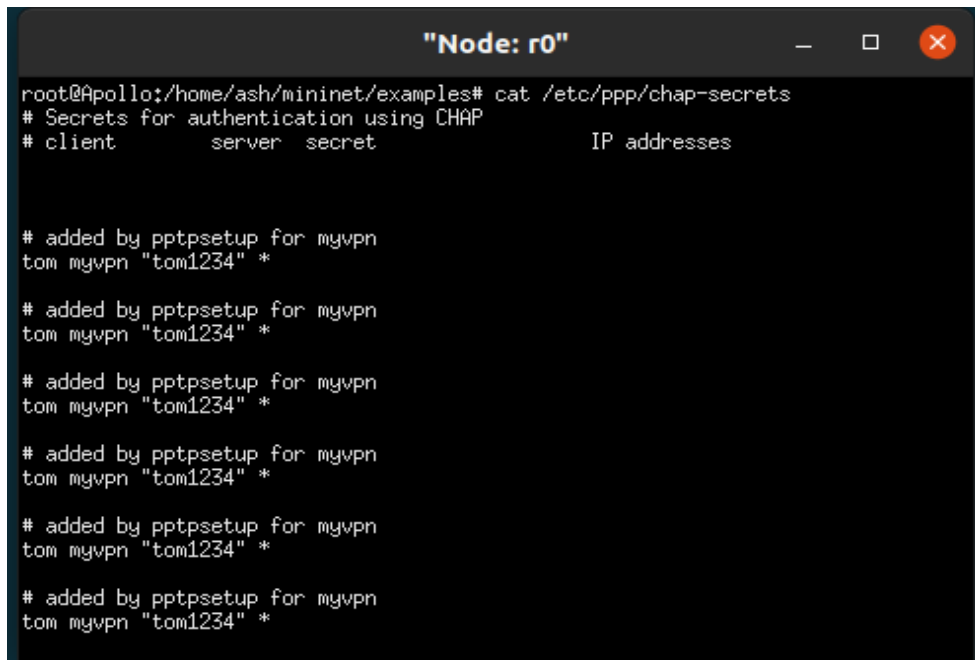
```
ash@Apollo: ~/mininet/examples
ash@Apollo:~/mininet/examples$ sudo python2 test6.py
mininet> xterm r0
mininet> xterm h2
mininet> |
```

Step 2: Open the vim editor in r0 and set up the pptp server ip address.



```
"Node: r0"
# 192.168,0,234,192,168,0,245-249,192,168,0,254
#
# IMPORTANT RESTRICTIONS:
#
# 1. No spaces are permitted between commas or within addresses.
#
# 2. If you give more IP addresses than the value of connections,
#    it will start at the beginning of the list and go until it
#    gets connections IPs. Others will be ignored.
#
# 3. No shortcuts in ranges! ie. 234-8 does not mean 234 to 238,
#    you must type 234-238 if you mean this.
#
# 4. If you give a single localIP, that's ok - all local IPs will
#    be set to the given one. You MUST still give at least one remote
#    IP for each simultaneous client.
#
# (Recommended)
localip 192.168,1,254
remoteip 192,168,1,100-200
# or
#localip 192,168,0,234-238,192,168,0,245
#remoteip 192,168,1,234-238,192,168,1,245
— INSERT — 99,27 Bot
```


Step 3: Setup an account with username and password.



```
"Node: r0"
root@Apollo:/home/ash/mininet/examples# cat /etc/ppp/chap-secrets
# Secrets for authentication using CHAP
# client        server  secret          IP addresses

# added by pptpsetup for myvpn
tom myvpn "tom1234" *

# added by pptpsetup for myvpn
tom myvpn "tom1234" *

# added by pptpsetup for myvpn
tom myvpn "tom1234" *

# added by pptpsetup for myvpn
tom myvpn "tom1234" *

# added by pptpsetup for myvpn
tom myvpn "tom1234" *
```

Step 4: Start the PPTP server in ro using the following commands shown.

`/usr/sbin/pptpd -d -f -l 10.0.0.1`

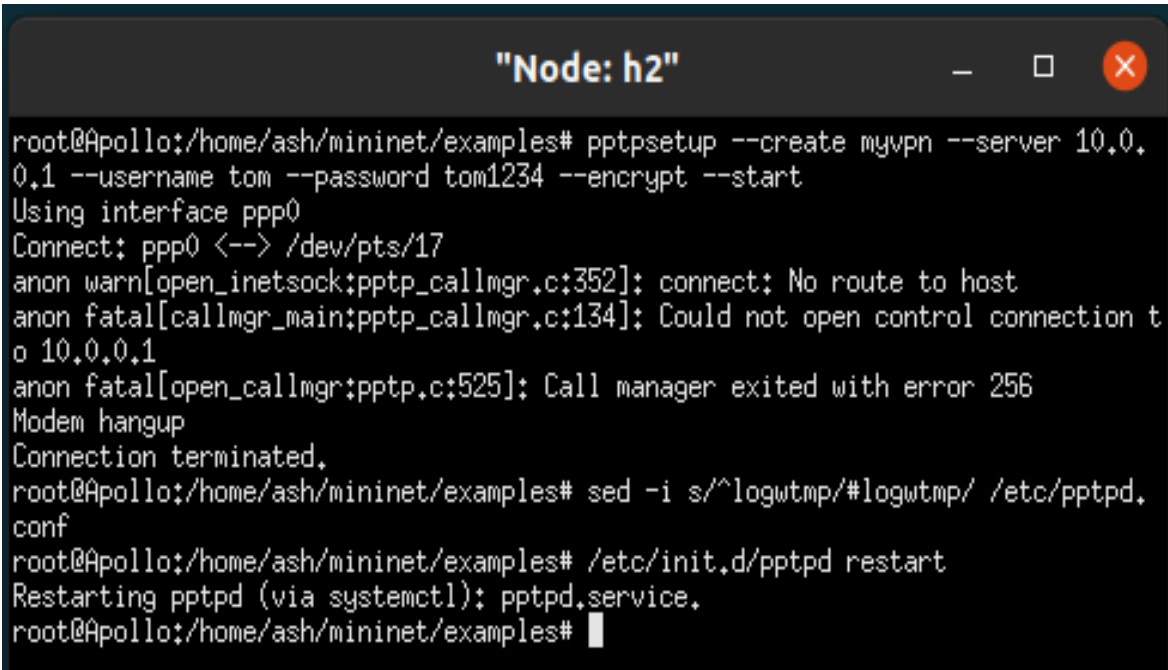


```
"Node: r0"
root@Apollo:/home/ash/mininet/examples# /usr/sbin/pptpd -d -f -l 10.0.0.1
```

Step 5: We have to create a VPN on the client side to connect to the PPTP server we created. At the first attempt we get an error of connection terminated. We need to fix the error using the commands in ro:

```
sed -i s/^logwtmp/#logwtmp/ /etc/pptpd.conf
```

Then restart the pptp server again.



```
root@Apollo:/home/ash/mininet/examples# pptpsetup --create myvpn --server 10.0.0.1 --username tom --password tom1234 --encrypt --start
Using interface ppp0
Connect: ppp0 <--> /dev/pts/17
anon warn[open_inetsock:pptp_callmgr.c:352]: connect: No route to host
anon fatal[callmgr_main:pptp_callmgr.c:134]: Could not open control connection to 10.0.0.1
anon fatal[open_callmgr:pptp.c:525]: Call manager exited with error 256
Modem hangup
Connection terminated.
root@Apollo:/home/ash/mininet/examples# sed -i s/^logwtmp/#logwtmp/ /etc/pptpd.conf
root@Apollo:/home/ash/mininet/examples# /etc/init.d/pptpd restart
Restarting pptpd (via systemctl): pptpd.service.
root@Apollo:/home/ash/mininet/examples#
```

Step 6: Now the VPN has been connected, we can try pinging the hosts from private to private or private to public network.

h1 from private network1 and h3 from private network2.

```
ash@Apollo: ~/mininet/examples
mininet> h1 ping h3
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.090 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.072 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.067 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.070 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.069 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.064 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.076 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.082 ms
64 bytes from 127.0.0.1: icmp_seq=12 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=13 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=14 ttl=64 time=0.075 ms
64 bytes from 127.0.0.1: icmp_seq=15 ttl=64 time=0.063 ms
64 bytes from 127.0.0.1: icmp_seq=16 ttl=64 time=0.065 ms
64 bytes from 127.0.0.1: icmp_seq=17 ttl=64 time=0.067 ms
64 bytes from 127.0.0.1: icmp_seq=18 ttl=64 time=0.092 ms
64 bytes from 127.0.0.1: icmp_seq=19 ttl=64 time=0.084 ms
64 bytes from 127.0.0.1: icmp_seq=20 ttl=64 time=0.021 ms
64 bytes from 127.0.0.1: icmp_seq=21 ttl=64 time=0.070 ms
```

```
ash@Apollo: ~/mininet/examples
mininet> h2 ping h3
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.038 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.026 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.111 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.077 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.079 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.094 ms
64 bytes from 127.0.0.1: icmp_seq=12 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=13 ttl=64 time=0.098 ms
64 bytes from 127.0.0.1: icmp_seq=14 ttl=64 time=0.071 ms
64 bytes from 127.0.0.1: icmp_seq=15 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=16 ttl=64 time=0.040 ms
64 bytes from 127.0.0.1: icmp_seq=17 ttl=64 time=0.071 ms
64 bytes from 127.0.0.1: icmp_seq=18 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=19 ttl=64 time=0.072 ms
64 bytes from 127.0.0.1: icmp_seq=20 ttl=64 time=0.021 ms
64 bytes from 127.0.0.1: icmp_seq=21 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=22 ttl=64 time=0.079 ms
64 bytes from 127.0.0.1: icmp_seq=23 ttl=64 time=0.031 ms
64 bytes from 127.0.0.1: icmp_seq=24 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=25 ttl=64 time=0.030 ms
```

We see that the hosts are communicating and that the VPN server is working.

VI. RESULT ANALYSIS

We see that the hosts are able to ping each other.

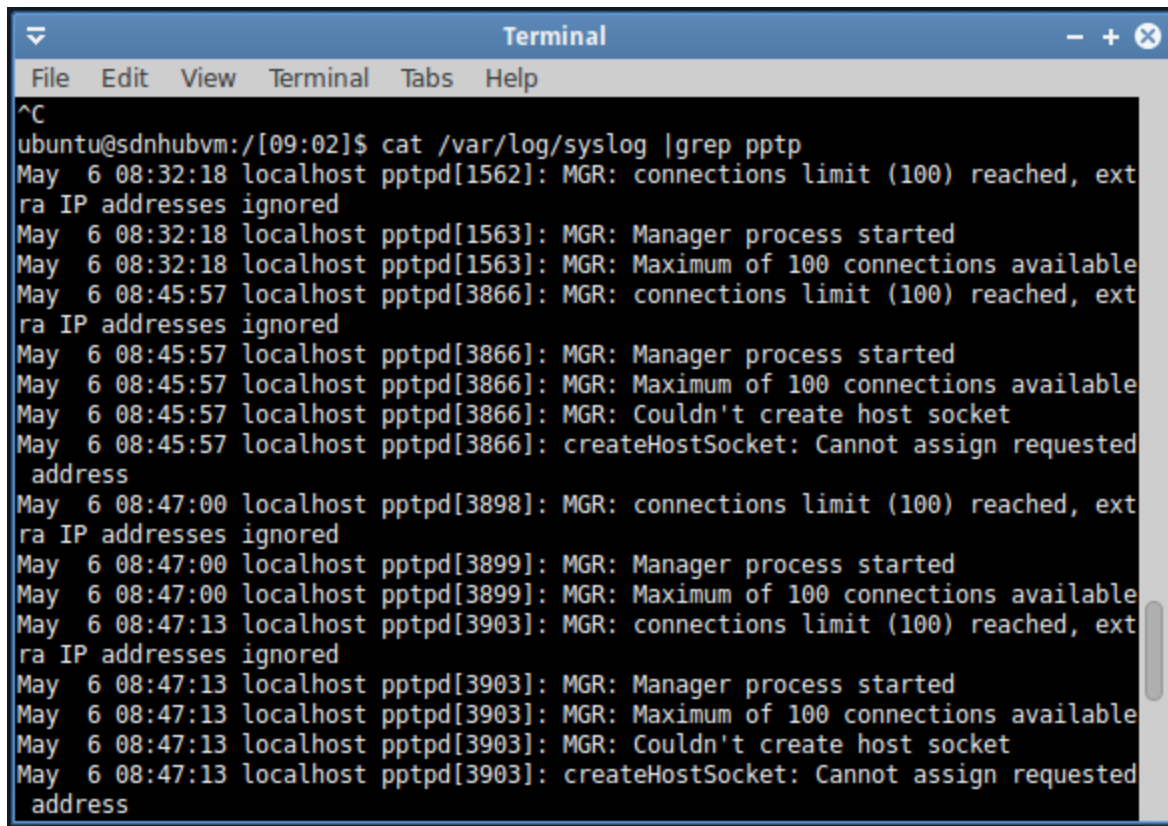
```
ash@Apollo: ~/mininet/examples

mininet> h1 ping h3
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.090 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.072 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.067 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.070 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.069 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.064 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.076 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.082 ms
64 bytes from 127.0.0.1: icmp_seq=12 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=13 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=14 ttl=64 time=0.075 ms
64 bytes from 127.0.0.1: icmp_seq=15 ttl=64 time=0.063 ms
64 bytes from 127.0.0.1: icmp_seq=16 ttl=64 time=0.065 ms
64 bytes from 127.0.0.1: icmp_seq=17 ttl=64 time=0.067 ms
64 bytes from 127.0.0.1: icmp_seq=18 ttl=64 time=0.092 ms
64 bytes from 127.0.0.1: icmp_seq=19 ttl=64 time=0.084 ms
64 bytes from 127.0.0.1: icmp_seq=20 ttl=64 time=0.021 ms
64 bytes from 127.0.0.1: icmp_seq=21 ttl=64 time=0.070 ms
```

```
ash@Apollo: ~/mininet/examples

5 packets transmitted, 0 received, 100% packet loss, time 200ms

mininet> h2 ping h3
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.038 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.026 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.111 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.077 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.079 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.094 ms
64 bytes from 127.0.0.1: icmp_seq=12 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=13 ttl=64 time=0.098 ms
64 bytes from 127.0.0.1: icmp_seq=14 ttl=64 time=0.071 ms
64 bytes from 127.0.0.1: icmp_seq=15 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=16 ttl=64 time=0.040 ms
64 bytes from 127.0.0.1: icmp_seq=17 ttl=64 time=0.071 ms
64 bytes from 127.0.0.1: icmp_seq=18 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=19 ttl=64 time=0.072 ms
64 bytes from 127.0.0.1: icmp_seq=20 ttl=64 time=0.021 ms
64 bytes from 127.0.0.1: icmp_seq=21 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=22 ttl=64 time=0.079 ms
64 bytes from 127.0.0.1: icmp_seq=23 ttl=64 time=0.031 ms
64 bytes from 127.0.0.1: icmp_seq=24 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=25 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=26 ttl=64 time=0.026 ms
```

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the command `cat /var/log/syslog |grep ptpd` and its output. The output consists of several log entries from May 6, showing the Manager process (MGR) for ptpd starting, reaching connection limits, and attempting to create host sockets. The entries are for processes with PIDs 1562, 1563, 3866, 3898, 3899, 3903, and 3903. The log messages include: "connections limit (100) reached, extra IP addresses ignored", "Manager process started", "Maximum of 100 connections available", "Couldn't create host socket", and "createHostSocket: Cannot assign requested address".

```
^C
ubuntu@sdnhubvm:/[09:02]$ cat /var/log/syslog |grep ptpd
May  6 08:32:18 localhost ptpd[1562]: MGR: connections limit (100) reached, extra IP addresses ignored
May  6 08:32:18 localhost ptpd[1563]: MGR: Manager process started
May  6 08:32:18 localhost ptpd[1563]: MGR: Maximum of 100 connections available
May  6 08:45:57 localhost ptpd[3866]: MGR: connections limit (100) reached, extra IP addresses ignored
May  6 08:45:57 localhost ptpd[3866]: MGR: Manager process started
May  6 08:45:57 localhost ptpd[3866]: MGR: Maximum of 100 connections available
May  6 08:45:57 localhost ptpd[3866]: MGR: Couldn't create host socket
May  6 08:45:57 localhost ptpd[3866]: createHostSocket: Cannot assign requested address
May  6 08:47:00 localhost ptpd[3898]: MGR: connections limit (100) reached, extra IP addresses ignored
May  6 08:47:00 localhost ptpd[3899]: MGR: Manager process started
May  6 08:47:00 localhost ptpd[3899]: MGR: Maximum of 100 connections available
May  6 08:47:13 localhost ptpd[3903]: MGR: connections limit (100) reached, extra IP addresses ignored
May  6 08:47:13 localhost ptpd[3903]: MGR: Manager process started
May  6 08:47:13 localhost ptpd[3903]: MGR: Maximum of 100 connections available
May  6 08:47:13 localhost ptpd[3903]: MGR: Couldn't create host socket
May  6 08:47:13 localhost ptpd[3903]: createHostSocket: Cannot assign requested address
```

VII. CONCLUSION

We are able to extend the private network using a public network such as the internet. As there was no secure connection between the private and public networks but when we created VPN, hosts from different private and public networks were able to communicate securely. The hosts from the private and public networks were not able to communicate before, but connecting to the VPN the hosts are able to communicate securely.

VIII. REFERENCES

1. [§http://blog.fens.me/vpn-pptp-client-ubuntu/](http://blog.fens.me/vpn-pptp-client-ubuntu/)
2. [§https://bugs.launchpad.net/ubuntu/+source/pptpd/+bug/1451419](https://bugs.launchpad.net/ubuntu/+source/pptpd/+bug/1451419)

IX. Drive link:

https://drive.google.com/drive/folders/1JVzFkGADk_Y2CqCjvoilJuBloE43NuDA?usp=sharing