LAB 1: OpenVPN

Network Architecture

Universitat Pompeu Fabra

Martí Esquius Arnau - 267444

Yihan Jin - 253297

Zihao Zhou - 285879

2024-2025

Table of Contents

1. Introduction

This lab focuses on the practical implementation and analysis of OpenVPN, a tool for creating secure virtual private networks (VPNs). Building on the setup from Seminar 2, the lab explores the key processes involved in VPN operation, including connection establishment, data transfer, and client-to-client communication.

The work is structured into two parts:

Part 1 involves configuring an OpenVPN server and client, analyzing the handshake process using tcpdump, and verifying connectivity through virtual IP addresses.

Part 2 extends the setup to include a second client, addressing connectivity challenges between clients and examining real versus virtual network traffic.

Through this lab, we aim to develop a deeper understanding of VPN functionality, network troubleshooting, and secure communication protocols. The results will be documented in this report following the specified guidelines.

2. Part 1: Questions

In Part 1, we will determine IP addresses, validate ARP connectivity, capture traffic with tcpdump to check the OpenVPN handshake (TLS negotiation, key exchange), and verify virtual IP assignment.

2.1. Client and server IPs

The IP addresses for both the server and the client have been found using the command "ip a". For the server:

```
mininet@arqxxs:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:df:09:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.3/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s3
       valid_lft 307sec preferred_lft 307sec
    inet6 fe80::ff55:d222:dab1:85c8/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:41:26:47:63 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

And for the client:

```
mininet@arqxxs:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:97:cb:3d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.4/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s3
       valid_lft 480sec preferred_lft 480sec
    inet6 fe80::7161:9197:899a:d6b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:8f:8b:20:35 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::a999:1412:296f:1094/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

From those responses, the server IP is 192.168.56.3/24, and the client IP is 192.168.56.4/24

2.2. Client-Server connection

To verify server-client reachability, we pinged each machine and inspected their ARP tables using arp -n. The server's ARP table listed the client's MAC, and the client's table listed the server's MAC, confirming bidirectional communication at Layer 2.

Server:

```
mininet@arqxxs:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask
192.168.56.4             ether   08:00:27:97:cb:3d   C
192.168.56.2             ether   08:00:27:f8:3e:85   C
```

Client:

```
mininet@arqxxs:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask
192.168.56.2             ether   08:00:27:f8:3e:85   C
192.168.56.3             ether   08:00:27:df:09:05   C
```

2.3. Traffic filtering using tcpdump

To capture and filter traffic between the OpenVPN server and the client, we used the tcpdump tool on the server VM.

First, we identified the active network interface by running *ip a* and noting the interface associated with the server's IP address. In our case, this interface was *enp0s3*. With this information, we executed the following command:

sudo tcpdump -n -i enp0s3 host 192.168.56.4

```
mininet@arqxxs:~$ sudo tcpdump -n -i enp0s3 host 192.168.56.4
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
```

The *-n* flag option helps making the output easier to read, *-i enp0s3* chooses the right network connection, and host 192.168.56.4 makes sure we only see traffic between the server and the client. This let us watch how the server and client first started talking before the VPN connection was set up.

2.4. Handshake procedure

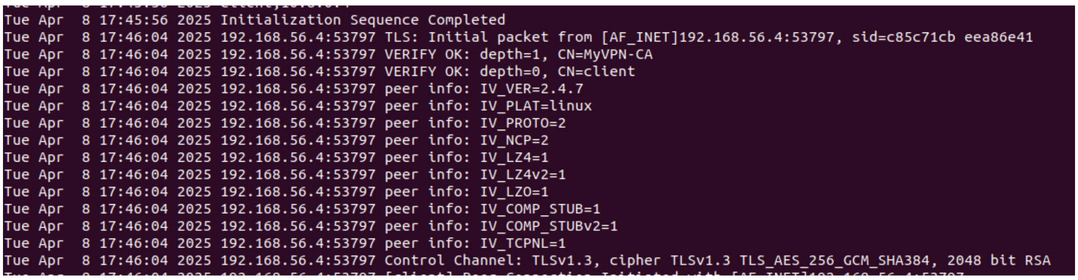
To observe the handshake process between the server and the client, we started both OpenVPN services. On the server, we ran *sudo openvpn --config server.conf*, and on the client, we ran *sudo openvpn --config client.conf*.

At the same time, we monitored the traffic using *tcpdump*, which showed packets being exchanged on UDP port 1194 — the default port used by OpenVPN. The OpenVPN logs on both sides confirmed the connection setup.

They showed the TLS handshake steps, including certificate verification and key exchange. We saw messages such as "VERIFY OK" (confirming the client's certificate was valid) and "Control Channel: TLSv1.3" (indicating a secure communication channel was established). Once the handshake was complete, the client logs displayed the message "Initialization Sequence Completed," confirming that the VPN tunnel was successfully established. The server logs also displayed the message "Peer Connection Initiated with [AF_INET]192.168.56.4" (the IP address of the client).

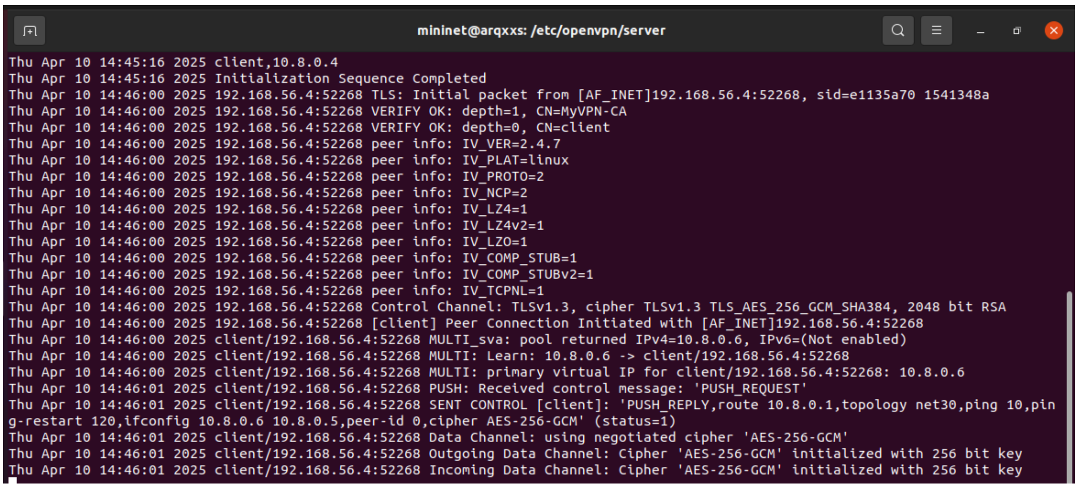
For the Server:

Tcp:



```
Tue Apr  8 17:45:56 2025 Initialization Sequence Completed
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 TLS: Initial packet from [AF_INET]192.168.56.4:53797, sid=c85c71cb eea86e41
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 VERIFY OK: depth=1, CN=MyVPN-CA
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 VERIFY OK: depth=0, CN=client
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_VER=2.4.7
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_PLAT=linux
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_PROTO=2
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_NCP=2
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_LZ4=1
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_LZ4v2=1
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_LZO=1
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_COMP_STUB=1
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_COMP_STUBv2=1
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 peer info: IV_TCPNL=1
Tue Apr  8 17:46:04 2025 192.168.56.4:53797 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, 2048 bit RSA
```

Vpn:



```
mininet@arqxxs: /etc/openvpn/server
Thu Apr 10 14:45:16 2025 client,10.8.0.4
Thu Apr 10 14:45:16 2025 Initialization Sequence Completed
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 TLS: Initial packet from [AF_INET]192.168.56.4:52268, sid=e1135a70 1541348a
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 VERIFY OK: depth=1, CN=MyVPN-CA
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 VERIFY OK: depth=0, CN=client
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_VER=2.4.7
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_PLAT=linux
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_PROTO=2
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_NCP=2
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_LZ4=1
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_LZ4v2=1
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_LZO=1
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_COMP_STUB=1
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_COMP_STUBv2=1
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 peer info: IV_TCPNL=1
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, 2048 bit RSA
Thu Apr 10 14:46:00 2025 192.168.56.4:52268 [client] Peer Connection Initiated with [AF_INET]192.168.56.4:52268
Thu Apr 10 14:46:00 2025 client/192.168.56.4:52268 MULTI_sva: pool returned IPv4=10.8.0.6, IPv6=(Not enabled)
Thu Apr 10 14:46:00 2025 client/192.168.56.4:52268 MULTI: Learn: 10.8.0.6 -> client/192.168.56.4:52268
Thu Apr 10 14:46:00 2025 client/192.168.56.4:52268 MULTI: primary virtual IP for client/192.168.56.4:52268: 10.8.0.6
Thu Apr 10 14:46:01 2025 client/192.168.56.4:52268 PUSH: Received control message: 'PUSH_REQUEST'
Thu Apr 10 14:46:01 2025 client/192.168.56.4:52268 SENT CONTROL [client]: 'PUSH_REPLY,route 10.8.0.1,topology net30,ping 10,pin
g-restart 120,ifconfig 10.8.0.6 10.8.0.5,peer-id 0,cipher AES-256-GCM' (status=1)
Thu Apr 10 14:46:01 2025 client/192.168.56.4:52268 Data Channel: using negotiated cipher 'AES-256-GCM'
Thu Apr 10 14:46:01 2025 client/192.168.56.4:52268 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit key
Thu Apr 10 14:46:01 2025 client/192.168.56.4:52268 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 256 bit key
```

For the Client:

Specific process to build the connection:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
2	5.091404	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
3	10.151880	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
4	10.267403	PcsCompu_97:cb:3d	PcsCompu_df:09:05	ARP	60	Who has 192.168.56.3? Tell 192.168.56.4
5	10.267434	PcsCompu_df:09:05	PcsCompu_97:cb:3d	ARP	42	192.168.56.3 is at 08:00:27:df:09:05
6	15.395711	PcsCompu_df:09:05	PcsCompu_97:cb:3d	ARP	42	Who has 192.168.56.4? Tell 192.168.56.3
7	15.396180	PcsCompu_97:cb:3d	PcsCompu_df:09:05	ARP	60	192.168.56.4 is at 08:00:27:97:cb:3d
8	15.532475	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
9	19.485186	192.168.56.3	192.168.56.4	TLSv1.2	86	Application Data
10	24.488079	192.168.56.4	192.168.56.3	OpenVPN	60	MessageType: P_CONTROL_HARD_RESET_CLIENT_V2
11	24.488170	192.168.56.3	192.168.56.4	ICMP	84	Destination unreachable (Port unreachable)
12	38.933004	192.168.56.4	192.168.56.3	OpenVPN	60	MessageType: P_CONTROL_HARD_RESET_CLIENT_V2
13	38.933224	192.168.56.3	192.168.56.4	OpenVPN	68	MessageType: P_CONTROL_HARD_RESET_SERVER_V2
14	38.933858	192.168.56.4	192.168.56.3	OpenVPN	64	MessageType: P_ACK_V1
15	38.933858	192.168.56.4	192.168.56.3	TLSv1.3	333	Client Hello
16	38.935035	192.168.56.3	192.168.56.4	TLSv1.3	1242	Server Hello, Change Cipher Spec, Application Data, Applicati…
17	38.935047	192.168.56.3	192.168.56.4	TLSv1.3	1230	Continuation Data
18	38.935167	192.168.56.3	192.168.56.4	TLSv1.3	77	Continuation Data
19	38.936798	192.168.56.4	192.168.56.3	OpenVPN	64	MessageType: P_ACK_V1
20	38.937204	192.168.56.4	192.168.56.3	OpenVPN	64	MessageType: P_ACK_V1
21	38.938440	192.168.56.4	192.168.56.3	TLSv1.3	1242	Change Cipher Spec
22	38.938441	192.168.56.4	192.168.56.3	TLSv1.3	1230	Continuation Data
23	38.938441	192.168.56.4	192.168.56.3	TLSv1.3	190	Continuation Data
24	38.938477	192.168.56.3	192.168.56.4	OpenVPN	64	MessageType: P_ACK_V1
25	38.938962	192.168.56.3	192.168.56.4	TLSv1.3	226	Application Data, Application Data
26	38.939292	192.168.56.4	192.168.56.3	OpenVPN	64	MessageType: P_ACK_V1
27	38.939407	192.168.56.3	192.168.56.4	TLSv1.3	287	Application Data
28	38.939817	192.168.56.4	192.168.56.3	OpenVPN	64	MessageType: P_ACK_V1
29	40.173914	192.168.56.4	192.168.56.3	TLSv1.3	91	Application Data
30	40.174775	192.168.56.3	192.168.56.4	OpenVPN	64	MessageType: P_ACK_V1
31	40.174834	192.168.56.3	192.168.56.4	TLSv1.3	200	Application Data
32	40.269965	192.168.56.4	192.168.56.3	OpenVPN	64	MessageType: P_ACK_V1
33	40.269966	192.168.56.4	192.168.56.3	OpenVPN	114	MessageType: P_DATA_V2
34	44.058949	PcsCompu_97:cb:3d	PcsCompu_df:09:05	ARP	60	Who has 192.168.56.3? Tell 192.168.56.4
35	44.058949	192.168.56.4	192.168.56.3	OpenVPN	114	MessageType: P_DATA_V2
36	44.058957	PcsCompu_df:09:05	PcsCompu_97:cb:3d	ARP	42	192.168.56.3 is at 08:00:27:df:09:05
37	45.348156	PcsCompu_df:09:05	PcsCompu_97:cb:3d	ARP	42	Who has 192.168.56.4? Tell 192.168.56.3
38	45.348534	PcsCompu_97:cb:3d	PcsCompu_df:09:05	ARP	60	192.168.56.4 is at 08:00:27:97:cb:3d
39	46.115777	192.168.56.3	192.168.56.4	OpenVPN	114	MessageType: P_DATA_V2

Vpn:

```
mininet@arqxxs: /etc/openvpn/client
Thu Apr 10 14:46:01 2025 PUSH: Received control message: 'PUSH_REPLY,route 10.8.
0.1,topology net30,ping 10,ping-restart 120,ifconfig 10.8.0.6 10.8.0.5,peer-id 0
,cipher AES-256-GCM'
Thu Apr 10 14:46:01 2025 OPTIONS IMPORT: timers and/or timeouts modified
Thu Apr 10 14:46:01 2025 OPTIONS IMPORT: --ifconfig/up options modified
Thu Apr 10 14:46:01 2025 OPTIONS IMPORT: route options modified
Thu Apr 10 14:46:01 2025 OPTIONS IMPORT: peer-id set
Thu Apr 10 14:46:01 2025 OPTIONS IMPORT: adjusting link_mtu to 1624
Thu Apr 10 14:46:01 2025 OPTIONS IMPORT: data channel crypto options modified
Thu Apr 10 14:46:01 2025 Data Channel: using negotiated cipher 'AES-256-GCM'
Thu Apr 10 14:46:01 2025 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized
 with 256 bit key
Thu Apr 10 14:46:01 2025 Incoming Data Channel: Cipher 'AES-256-GCM' initialized
 with 256 bit key
Thu Apr 10 14:46:01 2025 ROUTE: default_gateway=UNDEF
Thu Apr 10 14:46:01 2025 TUN/TAP device tun0 opened
Thu Apr 10 14:46:01 2025 TUN/TAP TX queue length set to 100
Thu Apr 10 14:46:01 2025 /sbin/ip link set dev tun0 up mtu 1500
Thu Apr 10 14:46:02 2025 /sbin/ip addr add dev tun0 local 10.8.0.6 peer 10.8.0.5
Thu Apr 10 14:46:02 2025 /sbin/ip route add 10.8.0.1/32 via 10.8.0.5
Thu Apr 10 14:46:02 2025 WARNING: this configuration may cache passwords in memo
ry -- use the auth-nocache option to prevent this
Thu Apr 10 14:46:02 2025 Initialization Sequence Completed
```

2.5. Client and server's VPN virtual IPs

After establishing the OpenVPN connection, we checked virtual IPs using *ip addr show tun0*. The server was assigned 10.8.0.1, and the client received 10.8.0.2. Pinging 10.8.0.1 from the client confirmed the VPN tunnel was functional.

```
mininet@arqxxs:~$ ip addr show tun0
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
 UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::84fd:cb17:b28b:1a9a/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

2.6. OpenVPN server's IP tcpdump capture

We use the command "*ip.addr == 192.168.56.3*" in the wireshark to filter the server's IP. In the wireshark it will only show the information that has either 192.168.56.3 for source or for destination, as we can see in the following figure:

No.	Time	Source	Destination	Protocol	Length	Info
2	1.061170	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
4	11.544658	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
6	21.891157	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
8	32.188878	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
14	41.621087	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
15	51.716641	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
17	61.784587	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
20	71.913537	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
24	82.057022	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
27	92.076571	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
29	102.216133	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
31	112.304489	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
36	122.401617	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
37	131.548631	192.168.56.4	192.168.56.3	OpenVPN	82	MessageType: P_DATA_V2
1	0.000000	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
3	10.467892	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
5	20.660215	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
7	31.016033	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
13	41.619960	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
16	51.716833	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
18	61.784875	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
19	71.912010	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
23	82.055080	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
28	92.077025	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
30	102.216403	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
32	112.304653	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
35	122.400933	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2
38	131.548916	192.168.56.3	192.168.56.4	OpenVPN	82	MessageType: P_DATA_V2

2.7. OpenVPN client ping to OpenVPN server

We captured VPN traffic using *sudo tcpdump -n -i tun0 host 10.8.0.1*. The output showed encrypted packets between the server (10.8.0.1) and client (10.8.0.6 for network layer), confirmed by ICMP echoes during a ping test. Physical interface (eth0) traffic was excluded, focusing only on the VPN tunnel.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=1/256, ttl=64 (reply in 2)
2	0.000058	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=1/256, ttl=64 (request in…
3	1.001549	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=2/512, ttl=64 (reply in 4)
4	1.001577	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=2/512, ttl=64 (request in…
5	2.003789	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=3/768, ttl=64 (reply in 6)
6	2.003817	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=3/768, ttl=64 (request in…
7	3.004410	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=4/1024, ttl=64 (reply in …
8	3.004442	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=4/1024, ttl=64 (request i…
9	4.006611	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=5/1280, ttl=64 (reply in …
10	4.006641	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=5/1280, ttl=64 (request i…
11	5.008087	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=6/1536, ttl=64 (reply in …
12	5.008116	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=6/1536, ttl=64 (request i…
13	6.008579	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=7/1792, ttl=64 (reply in …
14	6.008606	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=7/1792, ttl=64 (request i…
15	7.009761	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=8/2048, ttl=64 (reply in …
16	7.009789	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=8/2048, ttl=64 (request i…
17	8.010969	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=9/2304, ttl=64 (reply in …
18	8.010997	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=9/2304, ttl=64 (request i…
19	9.012609	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=10/2560, ttl=64 (reply in…
20	9.012648	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=10/2560, ttl=64 (request …
21	10.014900	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=11/2816, ttl=64 (reply in…
22	10.014928	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=11/2816, ttl=64 (request …
23	11.016255	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=12/3072, ttl=64 (reply in…
24	11.016286	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=12/3072, ttl=64 (request …
25	12.016500	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=13/3328, ttl=64 (reply in…
26	12.016528	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=13/3328, ttl=64 (request …
27	13.016598	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=14/3584, ttl=64 (reply in…
28	13.016626	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=14/3584, ttl=64 (request …
29	14.016371	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=15/3840, ttl=64 (reply in…
30	14.016399	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=15/3840, ttl=64 (request …
31	15.017702	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=16/4096, ttl=64 (reply in…
32	15.017732	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=16/4096, ttl=64 (request …
33	16.017794	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=17/4352, ttl=64 (reply in…
34	16.017821	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=17/4352, ttl=64 (request …
35	17.019802	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=18/4608, ttl=64 (reply in…
36	17.019831	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=18/4608, ttl=64 (request …
37	18.020565	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=19/4864, ttl=64 (reply in…
38	18.020596	10.8.0.1	10.8.0.6	ICMP	84	Echo (ping) reply id=0x0003, seq=19/4864, ttl=64 (request …
39	19.021861	10.8.0.6	10.8.0.1	ICMP	84	Echo (ping) request id=0x0003, seq=20/5120, ttl=64 (reply in…

3. Part 2: Client-to-client connection

Part 2 extends the setup to a multi-client VPN: we'll create a second client, troubleshoot client-to-client communication by enabling client-to-client in the server configuration, and use tcpdump to contrast encrypted traffic on physical interfaces (real IPs/ports) with unencrypted ICMP over the VPN's virtual interface (10.8.0.x addresses).

3.1-3.5. Set up PKI and configure a second client

We created a new client2 by cloning the server VM, updating client.conf, and starting all relevant VMs. We followed the instructions for seminar 2 and changed from "client" to "client2" for the names of the certificate and key files.

```
ca /etc/openvpn/easy-rsa/pki/ca.crt
cert /etc/openvpn/easy-rsa/pki/issued/client2.crt
key /etc/openvpn/easy-rsa/pki/private/client2.key
```

By adding client-to-client to the server's configuration (/etc/openvpn/server.conf) and restarting OpenVPN, we allowed direct communication between clients through the VPN server later.

3.6. Execution of OpenVPN server and client apps

Server:

```
mininet@arqxxs:/etc/openvpn/server$ sudo openvpn server.conf
Thu Apr 10 17:29:34 2025 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Mar 22 2022
Thu Apr 10 17:29:34 2025 library versions: OpenSSL 1.1.1f  31 Mar 2020, LZO 2.10
Thu Apr 10 17:29:34 2025 Diffie-Hellman initialized with 2048 bit key
Thu Apr 10 17:29:34 2025 ROUTE: default_gateway=UNDEF
Thu Apr 10 17:29:34 2025 TUN/TAP device tun0 opened
Thu Apr 10 17:29:34 2025 TUN/TAP TX queue length set to 100
Thu Apr 10 17:29:34 2025 /sbin/ip link set dev tun0 up mtu 1500
Thu Apr 10 17:29:34 2025 /sbin/ip addr add dev tun0 local 10.8.0.1 peer 10.8.0.2
Thu Apr 10 17:29:34 2025 /sbin/ip route add 10.8.0.0/24 via 10.8.0.2
Thu Apr 10 17:29:34 2025 Could not determine IPv4/IPv6 protocol. Using AF_INET
Thu Apr 10 17:29:34 2025 Socket Buffers: R=[212992->212992] S=[212992->212992]
Thu Apr 10 17:29:34 2025 UDPv4 link local (bound): [AF_INET][undef]:1194
Thu Apr 10 17:29:34 2025 UDPv4 link remote: [AF_UNSPEC]
Thu Apr 10 17:29:34 2025 MULTI: multi_init called, r=256 v=256
Thu Apr 10 17:29:34 2025 IFCONFIG POOL: base=10.8.0.4 size=62, ipv6=0
Thu Apr 10 17:29:34 2025 ifconfig_pool_read(), in='client,10.8.0.4', TODO: IPv6
Thu Apr 10 17:29:34 2025 succeeded -> ifconfig_pool_set()
Thu Apr 10 17:29:34 2025 IFCONFIG POOL LIST
Thu Apr 10 17:29:34 2025 client,10.8.0.4
Thu Apr 10 17:29:34 2025 Initialization Sequence Completed
```

Client1:

```
mininet@arqxxs: /etc/openvpn/client
Thu Apr 10 17:33:52 2025 PUSH: Received control message: 'PUSH_REPLY,route 10.8.
0.1,topology net30,ping 10,ping-restart 120,ifconfig 10.8.0.6 10.8.0.5,peer-id 1
,cipher AES-256-GCM'
Thu Apr 10 17:33:52 2025 OPTIONS IMPORT: timers and/or timeouts modified
Thu Apr 10 17:33:52 2025 OPTIONS IMPORT: --ifconfig/up options modified
Thu Apr 10 17:33:52 2025 OPTIONS IMPORT: route options modified
Thu Apr 10 17:33:52 2025 OPTIONS IMPORT: peer-id set
Thu Apr 10 17:33:52 2025 OPTIONS IMPORT: adjusting link_mtu to 1624
Thu Apr 10 17:33:52 2025 OPTIONS IMPORT: data channel crypto options modified
Thu Apr 10 17:33:52 2025 Data Channel: using negotiated cipher 'AES-256-GCM'
Thu Apr 10 17:33:52 2025 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized
 with 256 bit key
Thu Apr 10 17:33:52 2025 Incoming Data Channel: Cipher 'AES-256-GCM' initialized
 with 256 bit key
Thu Apr 10 17:33:52 2025 ROUTE: default_gateway=UNDEF
Thu Apr 10 17:33:52 2025 TUN/TAP device tun0 opened
Thu Apr 10 17:33:52 2025 TUN/TAP TX queue length set to 100
Thu Apr 10 17:33:52 2025 /sbin/ip link set dev tun0 up mtu 1500
Thu Apr 10 17:33:52 2025 /sbin/ip addr add dev tun0 local 10.8.0.6 peer 10.8.0.5
Thu Apr 10 17:33:52 2025 /sbin/ip route add 10.8.0.1/32 via 10.8.0.5
Thu Apr 10 17:33:52 2025 WARNING: this configuration may cache passwords in memo
ry -- use the auth-nocache option to prevent this
Thu Apr 10 17:33:52 2025 Initialization Sequence Completed
```

Client2:

```
0.1,topology net30,ping 10,ping-restart 120,ifconfig 10.8.0.10 10.8.0.9,peer-id
0,cipher AES-256-GCM'
Thu Apr 10 17:32:59 2025 OPTIONS IMPORT: timers and/or timeouts modified
Thu Apr 10 17:32:59 2025 OPTIONS IMPORT: --ifconfig/up options modified
Thu Apr 10 17:32:59 2025 OPTIONS IMPORT: route options modified
Thu Apr 10 17:32:59 2025 OPTIONS IMPORT: peer-id set
Thu Apr 10 17:32:59 2025 OPTIONS IMPORT: adjusting link_mtu to 1624
Thu Apr 10 17:32:59 2025 OPTIONS IMPORT: data channel crypto options modified
Thu Apr 10 17:32:59 2025 Data Channel: using negotiated cipher 'AES-256-GCM'
Thu Apr 10 17:32:59 2025 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized
 with 256 bit key
Thu Apr 10 17:32:59 2025 Incoming Data Channel: Cipher 'AES-256-GCM' initialized
 with 256 bit key
Thu Apr 10 17:32:59 2025 ROUTE: default_gateway=UNDEF
Thu Apr 10 17:32:59 2025 TUN/TAP device tun0 opened
Thu Apr 10 17:32:59 2025 TUN/TAP TX queue length set to 100
Thu Apr 10 17:32:59 2025 /sbin/ip link set dev tun0 up mtu 1500
Thu Apr 10 17:32:59 2025 /sbin/ip addr add dev tun0 local 10.8.0.10 peer 10.8.0.
9
Thu Apr 10 17:32:59 2025 /sbin/ip route add 10.8.0.1/32 via 10.8.0.9
Thu Apr 10 17:32:59 2025 WARNING: this configuration may cache passwords in memo
ry -- use the auth-nocache option to prevent this
Thu Apr 10 17:32:59 2025 Initialization Sequence Completed
```

OpenVPN's Default Topology: net30. We initialized the OpenVPN server and two clients, confirming successful execution via the ‘Initialization Sequence Completed’ message in all logs.

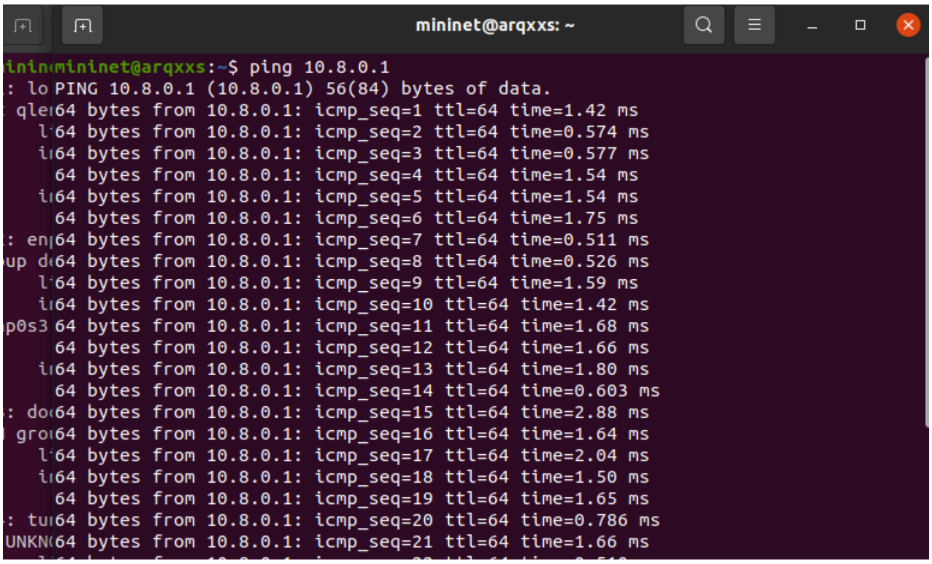
Virtual IP assignments were verified with *ip addr show tun0*: the server (10.8.0.1), Client1 (10.8.0.6), and Client2 (10.8.0.10). Ping tests confirmed bidirectional communication between all nodes, demonstrating a functional multi-client VPN.

3.7. Ping from first client to second client and vIP of server node

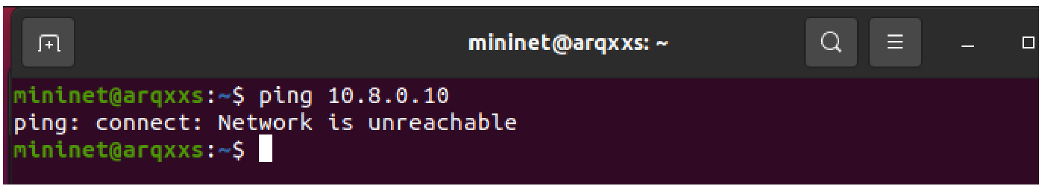
From 3.6 we know that:

- Client1’s virtual IP is 10.8.0.6
- Client2’s virtual IP is 10.8.0.10
- Server’s virtual IP is 10.8.0.1

To test the connection between server and client 1, we sent a ping from client 1 to server:



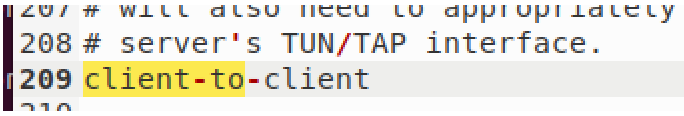
To test the connection between client 1 and client 2 we also tried to use the ping command. We have seen that when trying to ping the virtual IP address of client 2 from client 1 terminal we get the error “Network is unreachable”.



In conclusion, from Client1 we pinged both the server (10.8.0.1) and Client2 (10.8.0.10). While server pings succeeded immediately, Client2 only responded after enabling client-to-client in the server.conf.

3.8 - 3.9. Discussion on client-to-client connection

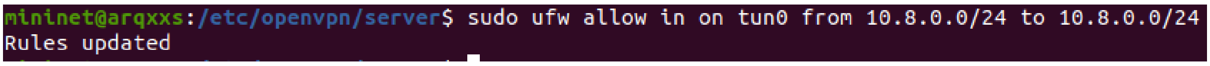
The client-to-client directive tells OpenVPN to allow traffic between VPN clients directly. Without this line (or if it's commented out with ";"), clients cannot communicate with each other through the VPN. We find the line and remove it in the server.conf:



```
208 # server's TUN/TAP interface.
209 client-to-client
```

Clients could not initially ping each other because OpenVPN defaults to client isolation. By adding client-to-client to server.conf, we instructed the server to route traffic between clients. Post-configuration, pings from Client1 (10.8.0.6) to Client2 (10.8.0.10) succeeded, confirming the change resolved the issue. This demonstrates OpenVPN's security-first design while allowing flexibility for trusted networks.

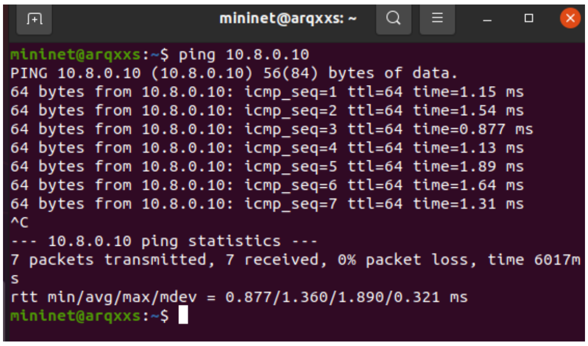
We also make sure that we close the firewall by using *sudo ufw allow in on tun0 from 10.8.0.0/24 to 10.8.0.0/24* in the server's terminal.



```
mininet@arqxxs:/etc/openvpn/server$ sudo ufw allow in on tun0 from 10.8.0.0/24 to 10.8.0.0/24
Rules updated
```

3.10. Verification ping from client to client.

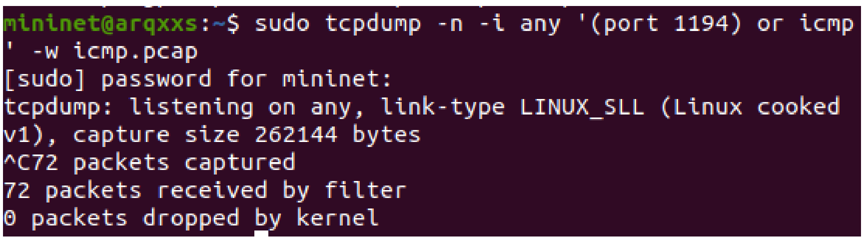
After the change of the configuration, pings are successful from client1 to client2.



```
mininet@arqxxs: ~
mininet@arqxxs:~$ ping 10.8.0.10
PING 10.8.0.10 (10.8.0.10) 56(84) bytes of data.
64 bytes from 10.8.0.10: icmp_seq=1 ttl=64 time=1.15 ms
64 bytes from 10.8.0.10: icmp_seq=2 ttl=64 time=1.54 ms
64 bytes from 10.8.0.10: icmp_seq=3 ttl=64 time=0.877 ms
64 bytes from 10.8.0.10: icmp_seq=4 ttl=64 time=1.13 ms
64 bytes from 10.8.0.10: icmp_seq=5 ttl=64 time=1.89 ms
64 bytes from 10.8.0.10: icmp_seq=6 ttl=64 time=1.64 ms
64 bytes from 10.8.0.10: icmp_seq=7 ttl=64 time=1.31 ms
^C
--- 10.8.0.10 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6017m
s
rtt min/avg/max/mdev = 0.877/1.360/1.890/0.321 ms
mininet@arqxxs:~$
```

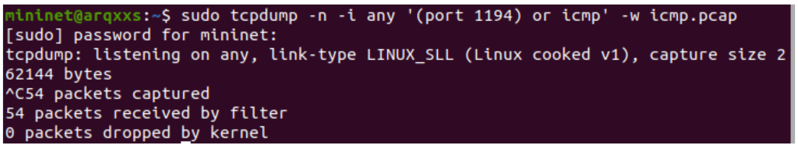
3.11 & 3.12. Network capture and "Virtual" vs "Real" IPs

When starting a network capture in the server node using *sudo tcpdump -n -i any port 1194 or icmp* and executing a continuous ping from client1 to client 2, we have seen:



```
mininet@arqxxs:~$ sudo tcpdump -n -i any '(port 1194) or icmp
' -w icmp.pcap
[sudo] password for mininet:
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked
v1), capture size 262144 bytes
^C72 packets captured
72 packets received by filter
0 packets dropped by kernel
```

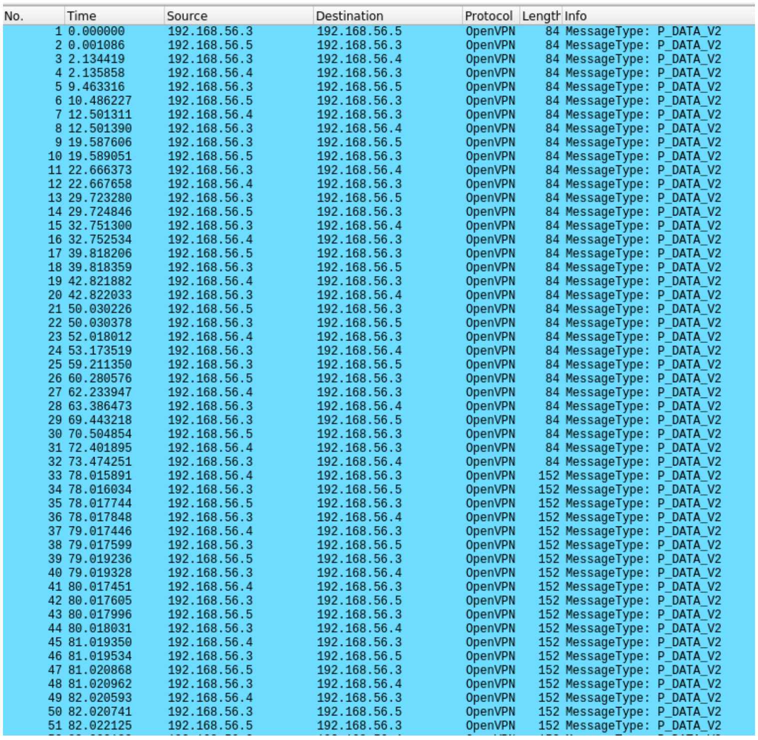
When doing the same in the terminal window for client2 node:



```
mininet@arqxxs:~$ sudo tcpdump -n -i any '(port 1194) or icmp' -w icmp.pcap
[sudo] password for mininet:
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 2
62144 bytes
^C54 packets captured
54 packets received by filter
0 packets dropped by kernel
```

The client-to-client configuration has been successfully implemented and verified through multiple testing methods. The VPN now properly routes traffic between all connected clients while maintaining server connectivity.

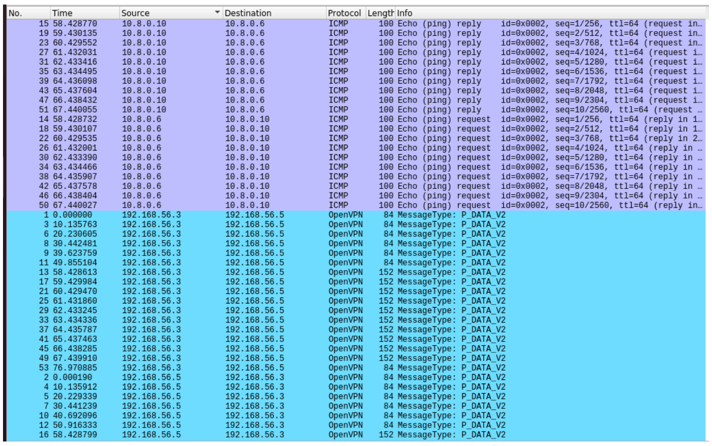
Server's capture:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
2	0.001086	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
3	2.134419	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
4	2.135858	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
5	9.463316	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
6	10.486227	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
7	12.501311	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
8	12.501390	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
9	19.587606	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
10	19.589051	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
11	22.666373	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
12	22.667658	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
13	29.723280	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
14	29.724846	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
15	32.751300	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
16	32.752534	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
17	39.818206	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
18	39.818359	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
19	42.821882	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
20	42.822033	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
21	50.030226	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
22	50.030378	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
23	52.018012	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
24	53.173519	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
25	59.211350	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
26	60.280576	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
27	62.233947	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
28	63.386473	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
29	69.443218	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
30	70.504854	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
31	72.401895	192.168.56.4	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
32	73.474251	192.168.56.3	192.168.56.4	OpenVPN	84	MessageType: P_DATA_V2
33	78.015891	192.168.56.4	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
34	78.016034	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
35	78.017744	192.168.56.5	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
36	78.017848	192.168.56.3	192.168.56.4	OpenVPN	152	MessageType: P_DATA_V2
37	79.017446	192.168.56.4	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
38	79.017599	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
39	79.019236	192.168.56.5	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
40	79.019328	192.168.56.3	192.168.56.4	OpenVPN	152	MessageType: P_DATA_V2
41	80.017451	192.168.56.4	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
42	80.017605	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
43	80.017996	192.168.56.5	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
44	80.018031	192.168.56.3	192.168.56.4	OpenVPN	152	MessageType: P_DATA_V2
45	81.019350	192.168.56.4	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
46	81.019534	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
47	81.020868	192.168.56.5	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
48	81.020962	192.168.56.3	192.168.56.4	OpenVPN	152	MessageType: P_DATA_V2
49	82.020593	192.168.56.4	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2
50	82.020741	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
51	82.022125	192.168.56.5	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2

(Only actual ip address)

Client2's capture:



No.	Time	Source	Destination	Protocol	Length	Info
15	58.428770	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=1/256, ttl=64 (request in…
19	59.430135	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=2/512, ttl=64 (request in…
23	60.429552	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=3/768, ttl=64 (request in…
27	61.432031	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=4/1024, ttl=64 (request i…
31	62.433416	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=5/1280, ttl=64 (request i…
35	63.434495	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=6/1536, ttl=64 (request i…
39	64.436098	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=7/1792, ttl=64 (request i…
43	65.437604	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=8/2048, ttl=64 (request i…
47	66.438432	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=9/2304, ttl=64 (request i…
51	67.440055	10.8.0.10	10.8.0.6	ICMP	100	Echo (ping) reply id=0x0002, seq=10/2560, ttl=64 (request …
14	58.428732	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=1/256, ttl=64 (reply in 1…
18	59.430107	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=2/512, ttl=64 (reply in 1…
22	60.429535	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=3/768, ttl=64 (reply in 2…
26	61.432001	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=4/1024, ttl=64 (reply in …
30	62.433390	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=5/1280, ttl=64 (reply in …
34	63.434466	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=6/1536, ttl=64 (reply in …
38	64.435907	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=7/1792, ttl=64 (reply in …
42	65.437578	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=8/2048, ttl=64 (reply in …
46	66.438404	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=9/2304, ttl=64 (reply in …
50	67.440027	10.8.0.6	10.8.0.10	ICMP	100	Echo (ping) request id=0x0002, seq=10/2560, ttl=64 (reply in…
1	0.000000	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
3	10.135763	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
6	20.230605	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
8	30.442481	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
9	39.623759	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
11	49.855104	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
13	58.428613	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
17	59.429984	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
21	60.429470	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
25	61.431860	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
29	62.433245	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
33	63.434336	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
37	64.435787	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
41	65.437463	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
45	66.438285	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
49	67.439910	192.168.56.3	192.168.56.5	OpenVPN	152	MessageType: P_DATA_V2
53	76.970885	192.168.56.3	192.168.56.5	OpenVPN	84	MessageType: P_DATA_V2
2	0.000190	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
4	10.135912	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
5	20.229339	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
7	30.441239	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
10	40.692096	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
12	50.916333	192.168.56.5	192.168.56.3	OpenVPN	84	MessageType: P_DATA_V2
16	58.428799	192.168.56.5	192.168.56.3	OpenVPN	152	MessageType: P_DATA_V2

(Contains real and virtual ip)

The server only sees scrambled data (UDP packets on port 1194) between real IP addresses—it's like passing sealed letters between clients.

The clients (like client2) open these sealed packets and see the actual ping messages (ICMP) using the fake VPN IPs. The server never sees the pings—it just moves the scrambled data. This shows how the VPN hides real network details but lets clients talk "secretly" using virtual addresses.

4. Conclusions

This lab successfully demonstrated the implementation and analysis of OpenVPN across server-client and multi-client configurations.

In Part 1, we established secure VPN connectivity, verified through IP assignments, ARP tables, and traffic captures using tcpdump. The handshake process revealed TLS negotiation and key exchange, while virtual IPs (e.g., 10.8.0.1 for the server) confirmed encrypted tunnel functionality.

In Part 2 extended this by introducing a second client, exposing OpenVPN's default client isolation. By enabling client-to-client in the server configuration, we resolved connectivity issues, allowing direct client communication via virtual IPs (10.8.0.x). Traffic analysis highlighted the separation between encrypted UDP traffic on physical interfaces (real IPs) and decrypted ICMP over virtual interfaces.

Overall, this lab provided a practical understanding of OpenVPN's connection establishment, tunneling behavior, and client communication policies, supported by systematic traffic analysis and network testing.