IoT Networking

2023-04-08

Agenda

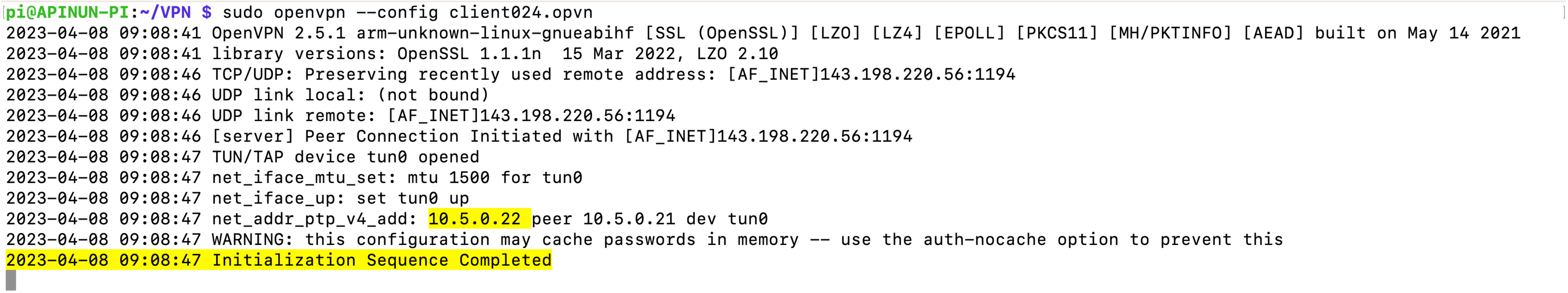
- Run OpenVPN on your Raspberry Pi (or on your VirtualBox VM)
- Create a configuration that allows OpenVPN to start automatically at boot time.
- Explore the features and characteristics of OpenVPN.
- Experiment with an 'overlay' network.
- Experiment with OpenVPN on mobile devices (e.g. iPhone and Android phones)

Raspberry Pi: Step by step

- Get the ovpn file from iotdev's /opt/scratch/ovpn/ file
    - ssh to iotdev.smartsensedesign.net
    - cd /opt/scratch/ovpn/
    - ls -l
    - cat FILENAME.ovpn
    - Copy the content and paste on to a file on your Raspberry Pi

Run OpenVPN on your Raspberry Pi

```
sudo openvpn --config [YOURVPNFILENAME].opvn
```

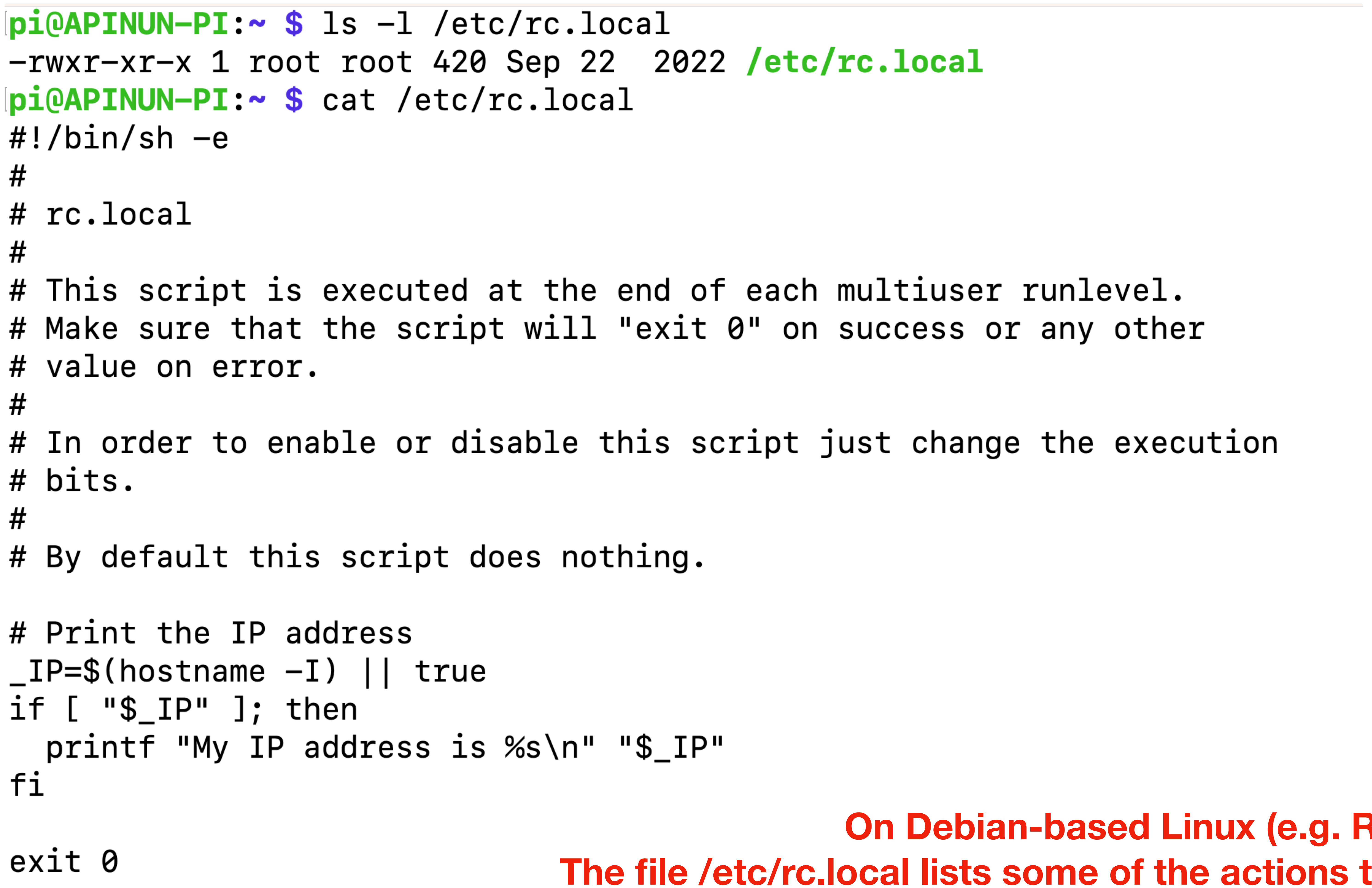


```
pi@APINUN-PI:~/VPN $ sudo openvpn --config client024.opvn
2023-04-08 09:08:41 OpenVPN 2.5.1 arm-unknown-linux-gnueabihf [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on May 14 2021
2023-04-08 09:08:41 library versions: OpenSSL 1.1.1n  15 Mar 2022, LZO 2.10
2023-04-08 09:08:46 TCP/UDP: Preserving recently used remote address: [AF_INET]143.198.220.56:1194
2023-04-08 09:08:46 UDP link local: (not bound)
2023-04-08 09:08:46 UDP link remote: [AF_INET]143.198.220.56:1194
2023-04-08 09:08:46 [server] Peer Connection Initiated with [AF_INET]143.198.220.56:1194
2023-04-08 09:08:47 TUN/TAP device tun0 opened
2023-04-08 09:08:47 net_iface_mtu_set: mtu 1500 for tun0
2023-04-08 09:08:47 net_iface_up: set tun0 up
2023-04-08 09:08:47 net_addr_ptp_v4_add: 10.5.0.22 peer 10.5.0.21 dev tun0
2023-04-08 09:08:47 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to prevent this
2023-04-08 09:08:47 Initialization Sequence Completed
```

Issues

- How can we start openvpn client (on Raspberry Pi or Ubuntu VM) automatically when you boot your Pi/VM ?
    - Answers: systemd or /etc/rc.local
- How can we fix the IP address assigned to an openvpn client on our Pi or Ubuntu VM ?
    - Answer: use the OpenVPN client config directory or ccd feature, on the server side

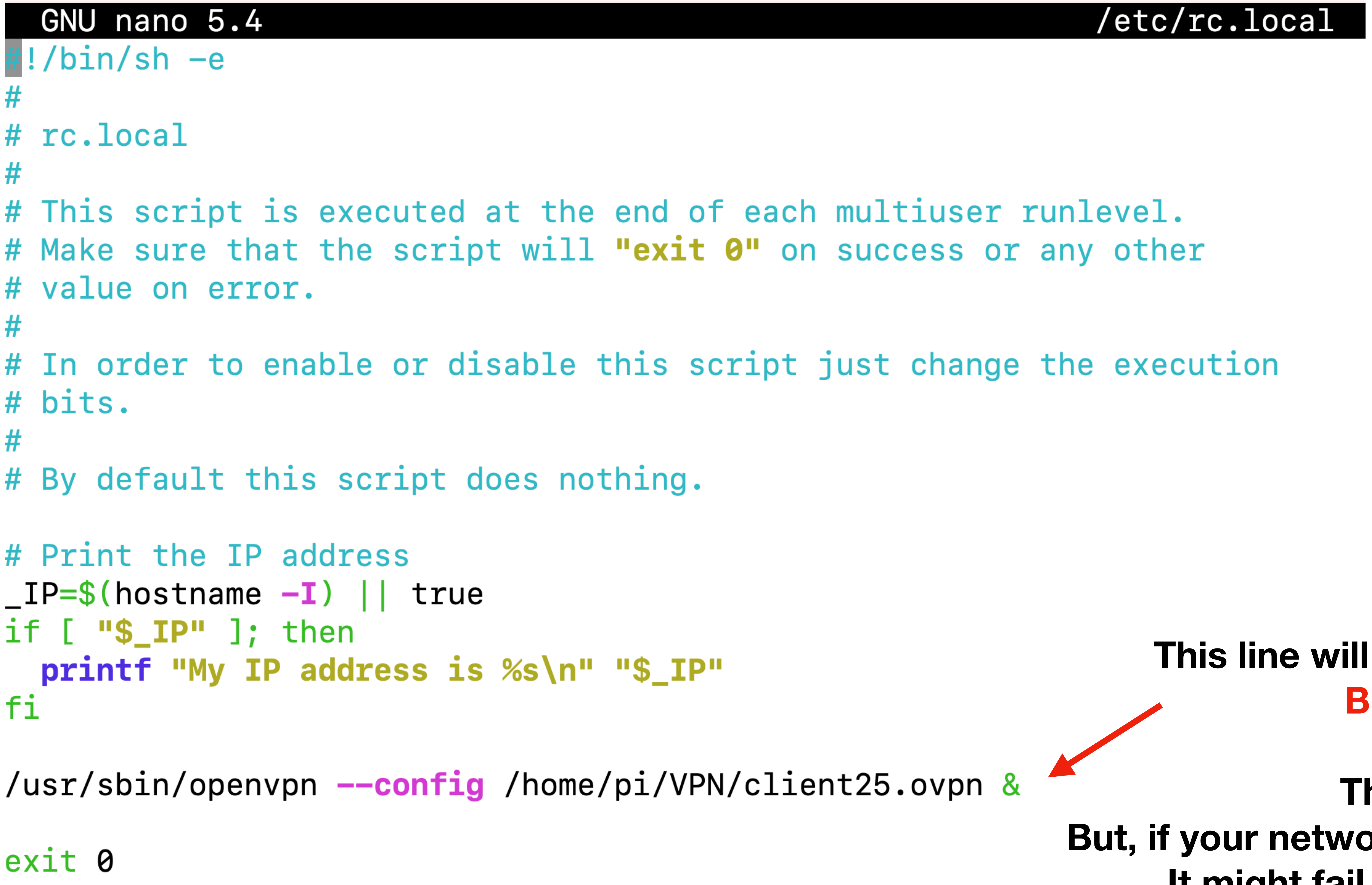
The /etc/rc.local file on your Raspberry Pi or VirtualBox VM



```
pi@APINUN-PI:~ $ ls -l /etc/rc.local
-rwxr-xr-x 1 root root 420 Sep 22  2022 /etc/rc.local
pi@APINUN-PI:~ $ cat /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

exit 0
```

On Debian-based Linux (e.g. Raspberry Pi and Ubuntu)
The file /etc/rc.local lists some of the actions that will be taken after booting the Pi/VM.
We can place commands or scripts here to run at every boot time.

Our first attempt:
add a line to /etc/rc.local to start OpenVPN once



```
  GNU nano 5.4                                         /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

/usr/sbin/openvpn --config /home/pi/VPN/client25.ovpn &

exit 0
```

This line will run openvpn in the background (&).
But it attempts only once !

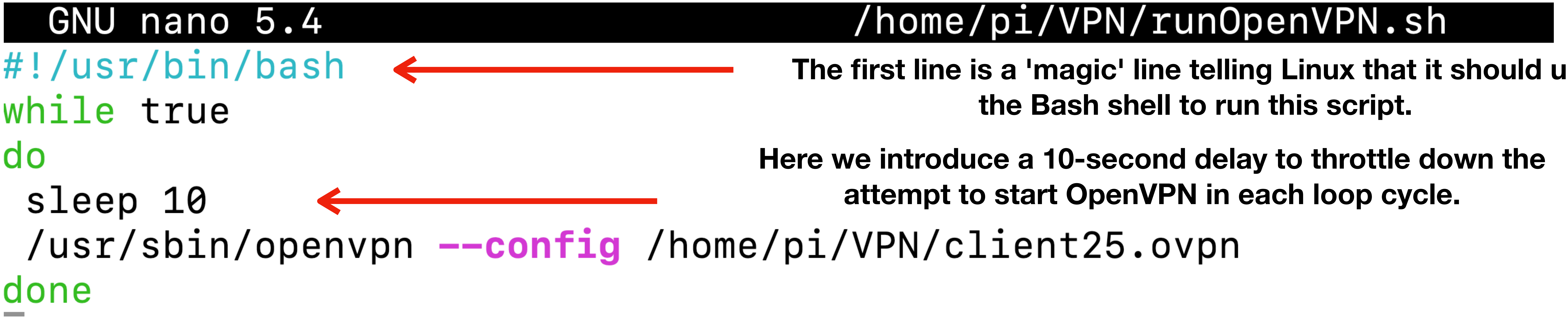
This may work sometimes.
But, if your network interface is not ready at that moment,
It might fail to connect to the OpenVPN server.

What might be a better approach ?
We create a Bash shell script to start OpenVPN and loop on that

This is a shell script that runs openvpn in a forever loop.

The script waits 10 seconds before starting an openvpn process

Important note: do not put & after the openvpn command because we want one process per loop.



```
  GNU nano 5.4                                /home/pi/VPN/runOpenVPN.sh
#!/usr/bin/bash
while true
do
 sleep 10
 /usr/sbin/openvpn --config /home/pi/VPN/client25.ovpn
done
```

The first line is a 'magic' line telling Linux that it should use the Bash shell to run this script.

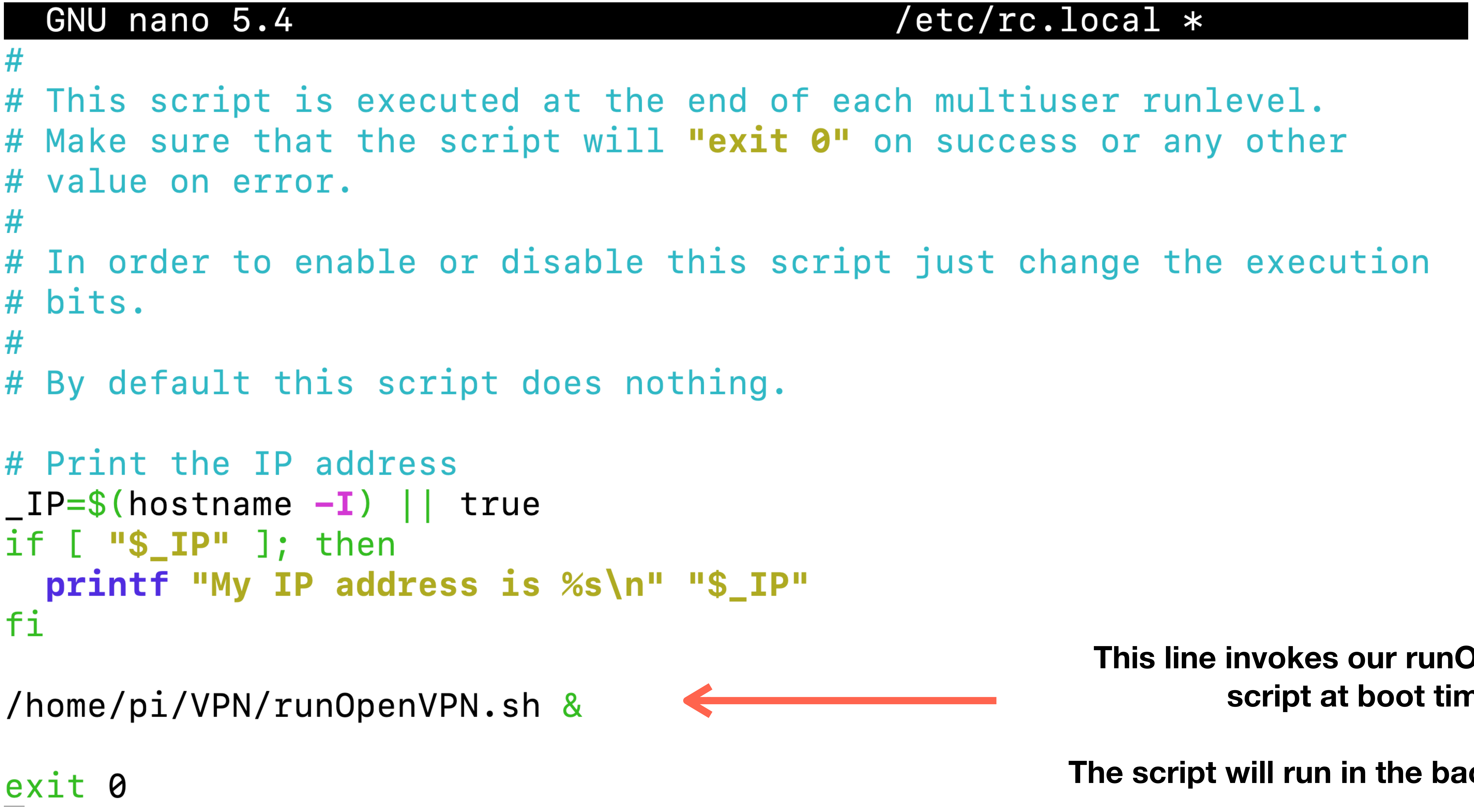
Here we introduce a 10-second delay to throttle down the attempt to start OpenVPN in each loop cycle.

Make the bash file executable (+x)

```
pi@APINUN-PI:~/VPN $ ls -al
total 48
drwxr-xr-x  2 pi pi 4096 Apr  8 09:47 .
drwxr-xr-x 16 pi pi 4096 Apr  8 09:47 ..
-rw-r--r--  1 pi pi 8522 Apr  1 12:06 client024.opvn
-rw-r--r--  1 pi pi 8526 Apr  1 11:45 client025.ovpn
-rw-r--r--  1 pi pi 8526 Apr  8 09:21 client25.ovpn
-rw-r--r--  1 pi pi  100 Apr  8 09:45 runOpenVPN.sh
pi@APINUN-PI:~/VPN $ chmod +x runOpenVPN.sh
pi@APINUN-PI:~/VPN $ ls -al
total 48
drwxr-xr-x  2 pi pi 4096 Apr  8 09:47 .
drwxr-xr-x 16 pi pi 4096 Apr  8 09:47 ..
-rw-r--r--  1 pi pi 8522 Apr  1 12:06 client024.opvn
-rw-r--r--  1 pi pi 8526 Apr  1 11:45 client025.ovpn
-rw-r--r--  1 pi pi 8526 Apr  8 09:21 client25.ovpn
-rwxr-xr-x  1 pi pi  100 Apr  8 09:45 runOpenVPN.sh
```

The executable (x) bit tells Linux that this file is a runnable program.

Modify /etc/rc.local to start our bash script

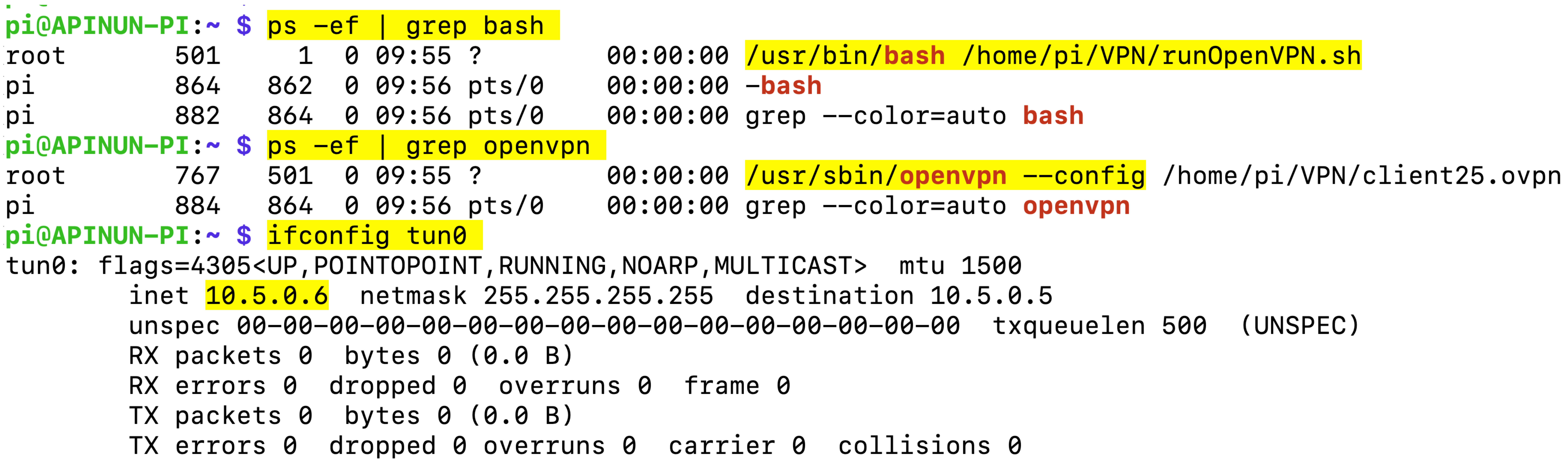


```
  GNU nano 5.4                                      /etc/rc.local *
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

/home/pi/VPN/runOpenVPN.sh &

exit 0
```

This line invokes our runOpenVPN.sh script at boot time.

The script will run in the background (&).

If you do it right, after rebooting your Pi or VM you should see these processes



```
pi@APINUN-PI:~ $ ps -ef | grep bash
root       501      1  0 09:55 ?        00:00:00 /usr/bin/bash /home/pi/VPN/runOpenVPN.sh
pi         864    862  0 09:56 pts/0    00:00:00 -bash
pi         882    864  0 09:56 pts/0    00:00:00 grep --color=auto bash
pi@APINUN-PI:~ $ ps -ef | grep openvpn
root       767    501  0 09:55 ?        00:00:00 /usr/sbin/openvpn --config /home/pi/VPN/client25.ovpn
pi         884    864  0 09:56 pts/0    00:00:00 grep --color=auto openvpn
pi@APINUN-PI:~ $ ifconfig tun0
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.5.0.6  netmask 255.255.255.255  destination 10.5.0.5
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        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
```

The tun0 interface is up when the OpenVPN client on Raspberry Pi connects to the OpenVPN server successfully.
This tun0 interface is a point-to-point 'secure' tunnel between our Raspberry Pi and the OpenVPN server.

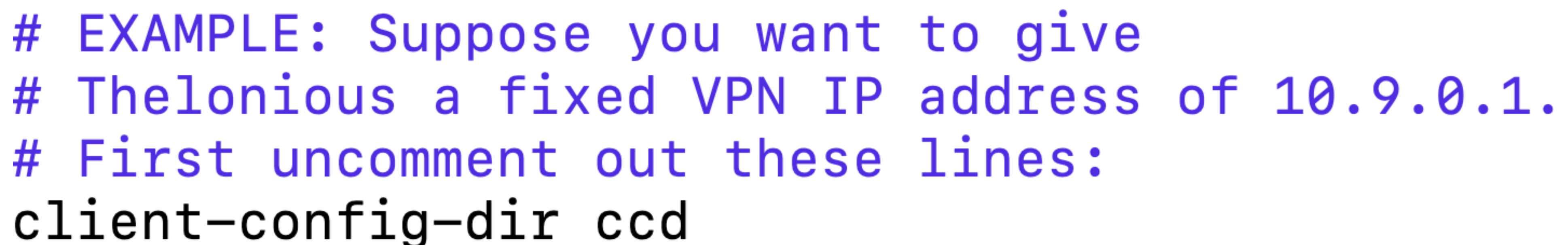
We will try to fix the OpenVPN IP address being assigned each VPN client

This is a proposed clientID to VPN IP address mapping

Client ID	VPN IP address
client001	10.5.0.101
client002	10.5.0.102
client003	10.5.0.103
client004	10.5.0.104
client005	10.5.0.105
client006	10.5.0.106
client007	10.5.0.107
client008	10.5.0.108
client009	10.5.0.109
client001	10.5.0.110

OpenVPN client-config-directory (ccd)

Step 1: we need to specify the client-config-dir in our OpenVPN server configuration file.

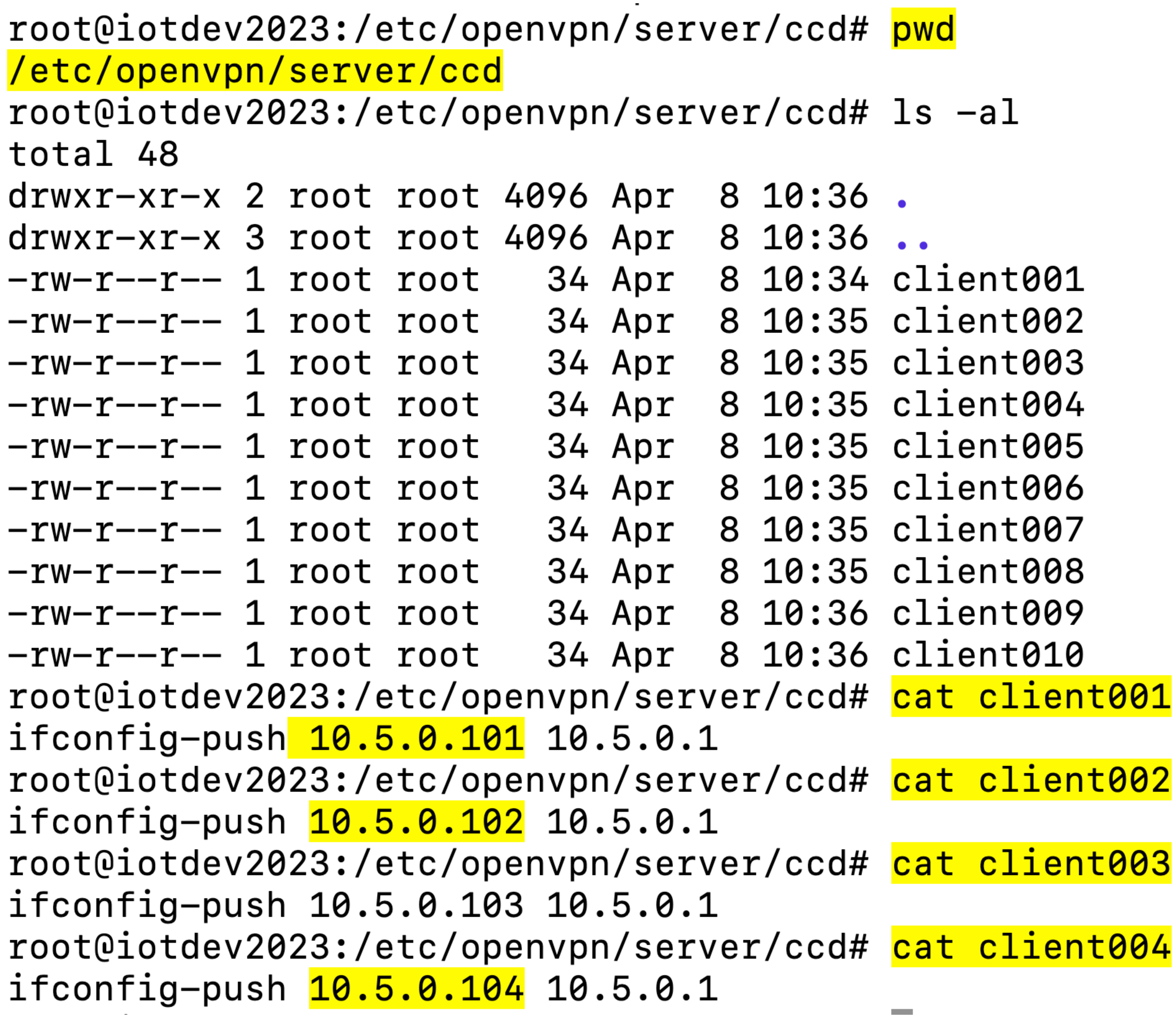


```
# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
client-config-dir ccd
```

Step 2: we need to create the ccd folder.

```
mkdir -p /etc/openvpn/server/ccd
```

Step3: we need to create a configuration file for each client ID.



```
root@iotdev2023:/etc/openvpn/server/ccd# pwd
/etc/openvpn/server/ccd
root@iotdev2023:/etc/openvpn/server/ccd# ls -al
total 48
drwxr-xr-x 2 root root 4096 Apr  8 10:36 .
drwxr-xr-x 3 root root 4096 Apr  8 10:36 ..
-rw-r--r-- 1 root root   34 Apr  8 10:34 client001
-rw-r--r-- 1 root root   34 Apr  8 10:35 client002
-rw-r--r-- 1 root root   34 Apr  8 10:35 client003
-rw-r--r-- 1 root root   34 Apr  8 10:35 client004
-rw-r--r-- 1 root root   34 Apr  8 10:35 client005
-rw-r--r-- 1 root root   34 Apr  8 10:35 client006
-rw-r--r-- 1 root root   34 Apr  8 10:35 client007
-rw-r--r-- 1 root root   34 Apr  8 10:35 client008
-rw-r--r-- 1 root root   34 Apr  8 10:36 client009
-rw-r--r-- 1 root root   34 Apr  8 10:36 client010
root@iotdev2023:/etc/openvpn/server/ccd# cat client001
ifconfig-push 10.5.0.101 10.5.0.1
root@iotdev2023:/etc/openvpn/server/ccd# cat client002
ifconfig-push 10.5.0.102 10.5.0.1
root@iotdev2023:/etc/openvpn/server/ccd# cat client003
ifconfig-push 10.5.0.103 10.5.0.1
root@iotdev2023:/etc/openvpn/server/ccd# cat client004
ifconfig-push 10.5.0.104 10.5.0.1
```

Observation #1: traceroute

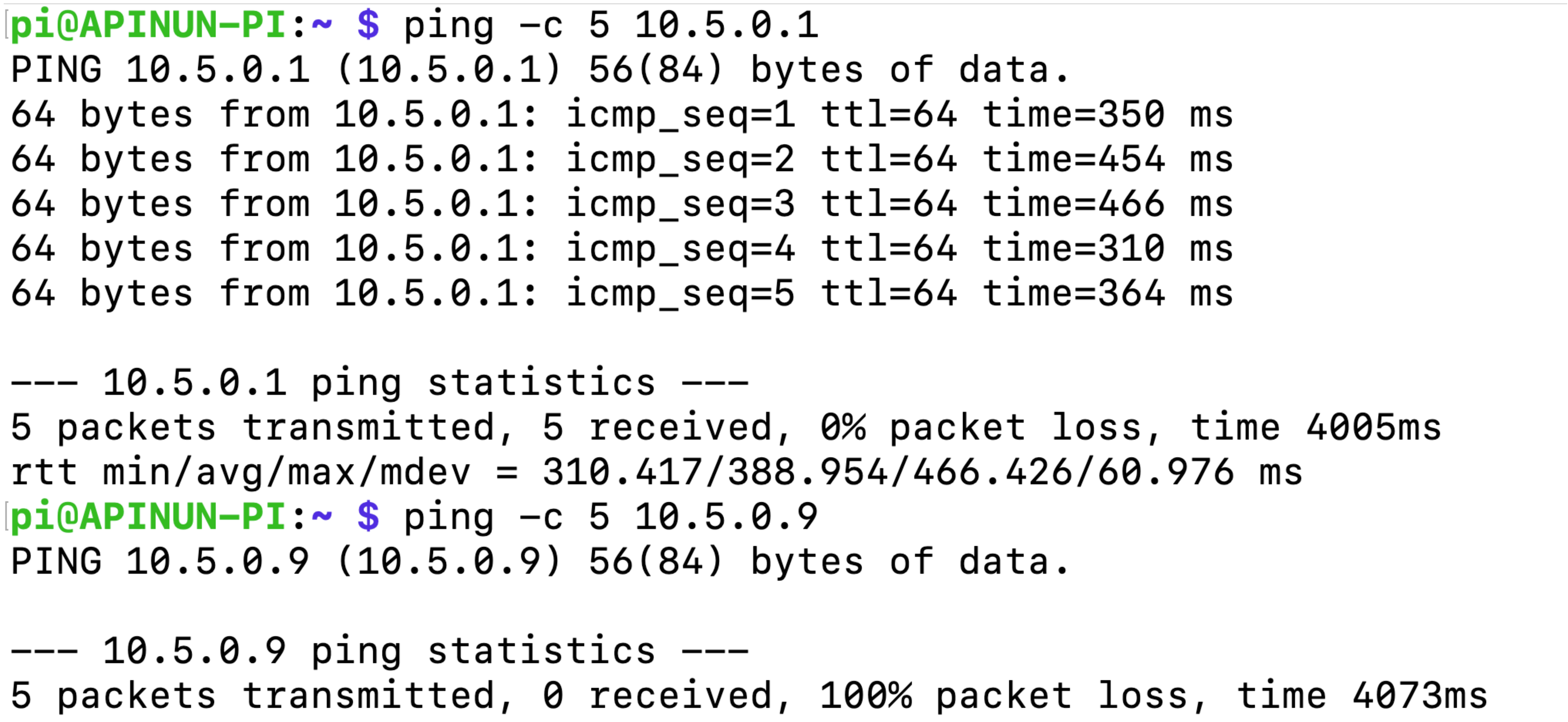
```
pi@APINUN-PI:~ $ traceroute iotdev.smartsensedesign.net
traceroute to iotdev.smartsensedesign.net (143.198.220.56), 30 hops max, 60 byte packets
 1  172.50.0.254 (172.50.0.254)  0.360 ms  0.246 ms  0.238 ms
 2  * * *
 3  10.118.136.13 (10.118.136.13)  25.428 ms  25.297 ms  25.184 ms
 4  182.232.253.169 (182.232.253.169)  38.175 ms  37.944 ms  38.369 ms
 5  et-0-3-0-518.iig-sila-pe02.ais-idc.com (182.232.253.57)  44.282 ms et-1-3-0-517.iig-sila-pe01.ais-idc.com (182.232.253.55
  44.166 ms et-0-3-0-515.iig-sila-pe01.ais-idc.com (182.232.253.51)  43.886 ms
 6  49.231.45.207 (49.231.45.207)  73.737 ms  52.763 ms 49.231.70.55 (49.231.70.55)  59.595 ms
 7  14061.sgw.equinix.com (27.111.229.164)  52.447 ms  51.762 ms  52.218 ms
 8  143.244.224.206 (143.244.224.206)  52.108 ms  58.180 ms  58.106 ms
 9  143.244.224.231 (143.244.224.231)  57.985 ms  57.748 ms  50.399 ms
10  * * *
11  * * *
12  * * *
13  * * *
14  143.198.220.56 (143.198.220.56)  67.796 ms  67.760 ms  67.627 ms
pi@APINUN-PI:~ $ traceroute 10.5.0.1
traceroute to 10.5.0.1 (10.5.0.1), 30 hops max, 60 byte packets
 1  10.5.0.1 (10.5.0.1)  50.340 ms  54.782 ms  57.438 ms
```

If we do a traceroute from our Pi to the iotdev VM, it takes around 14 hops to get there.

**But if we do a traceroute from our Pi/VM to tun0 of our VPN server, it appears only 1 hop but with a higher delay.
In a sense, OpenVPN creates a virtual direct link between our Raspberry Pi and the VPN server.**

Observation #2: pinging the other guys

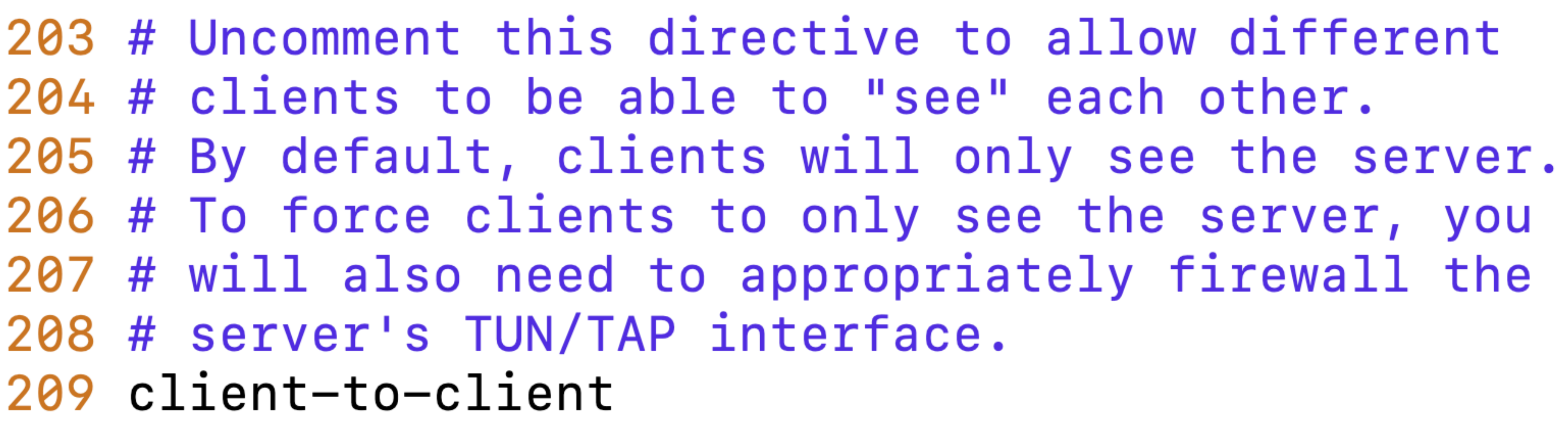
**In a default configuration, the client can ping to the server (10.5.0.1) only.
Each client cannot ping the other clients.**



```
pi@APINUN-PI:~ $ ping -c 5 10.5.0.1
PING 10.5.0.1 (10.5.0.1) 56(84) bytes of data.
64 bytes from 10.5.0.1: icmp_seq=1 ttl=64 time=350 ms
64 bytes from 10.5.0.1: icmp_seq=2 ttl=64 time=454 ms
64 bytes from 10.5.0.1: icmp_seq=3 ttl=64 time=466 ms
64 bytes from 10.5.0.1: icmp_seq=4 ttl=64 time=310 ms
64 bytes from 10.5.0.1: icmp_seq=5 ttl=64 time=364 ms

--- 10.5.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 310.417/388.954/466.426/60.976 ms
pi@APINUN-PI:~ $ ping -c 5 10.5.0.9
PING 10.5.0.9 (10.5.0.9) 56(84) bytes of data.

--- 10.5.0.9 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4073ms
```

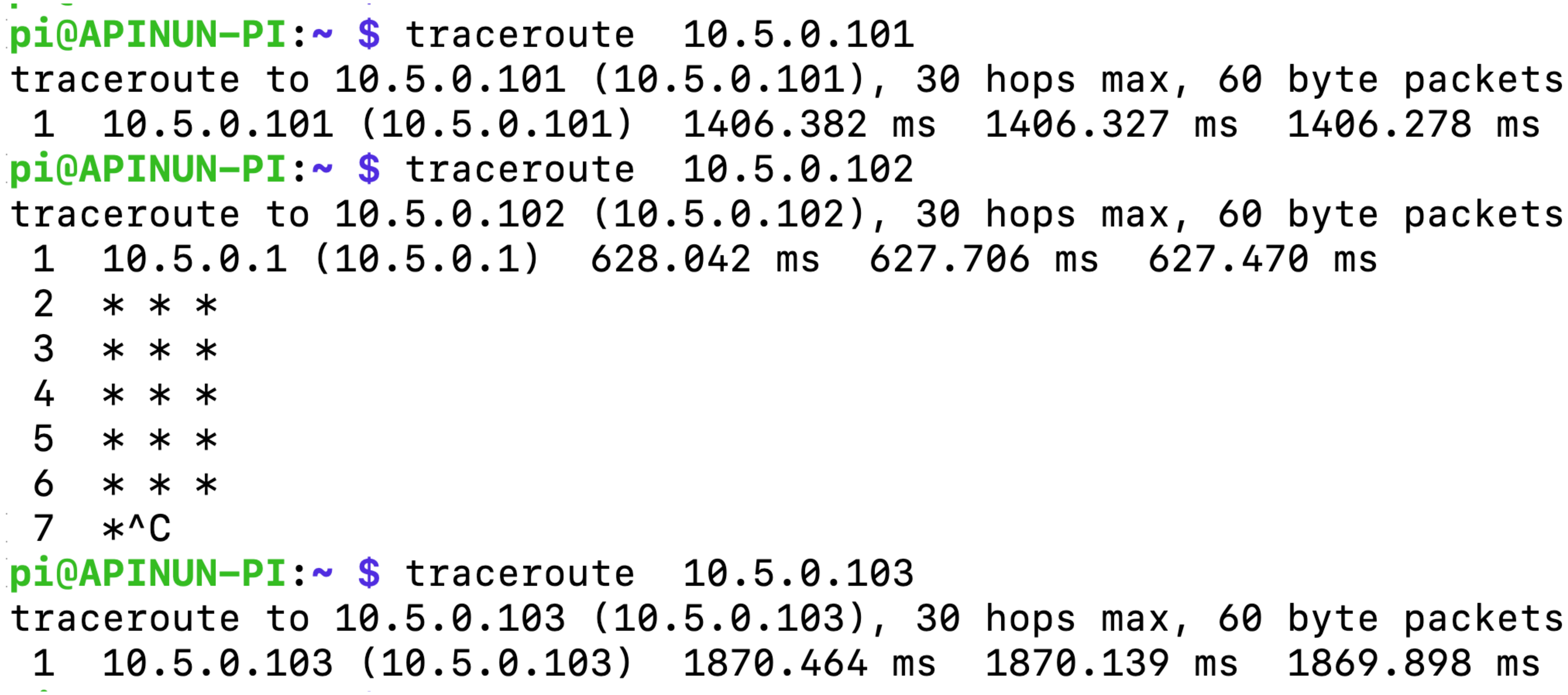
In the default configuration, OpenVPN does not allow each VPN client to directly communicate with each other.

We can allow client-to-client communication



```
# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
client-to-client
```

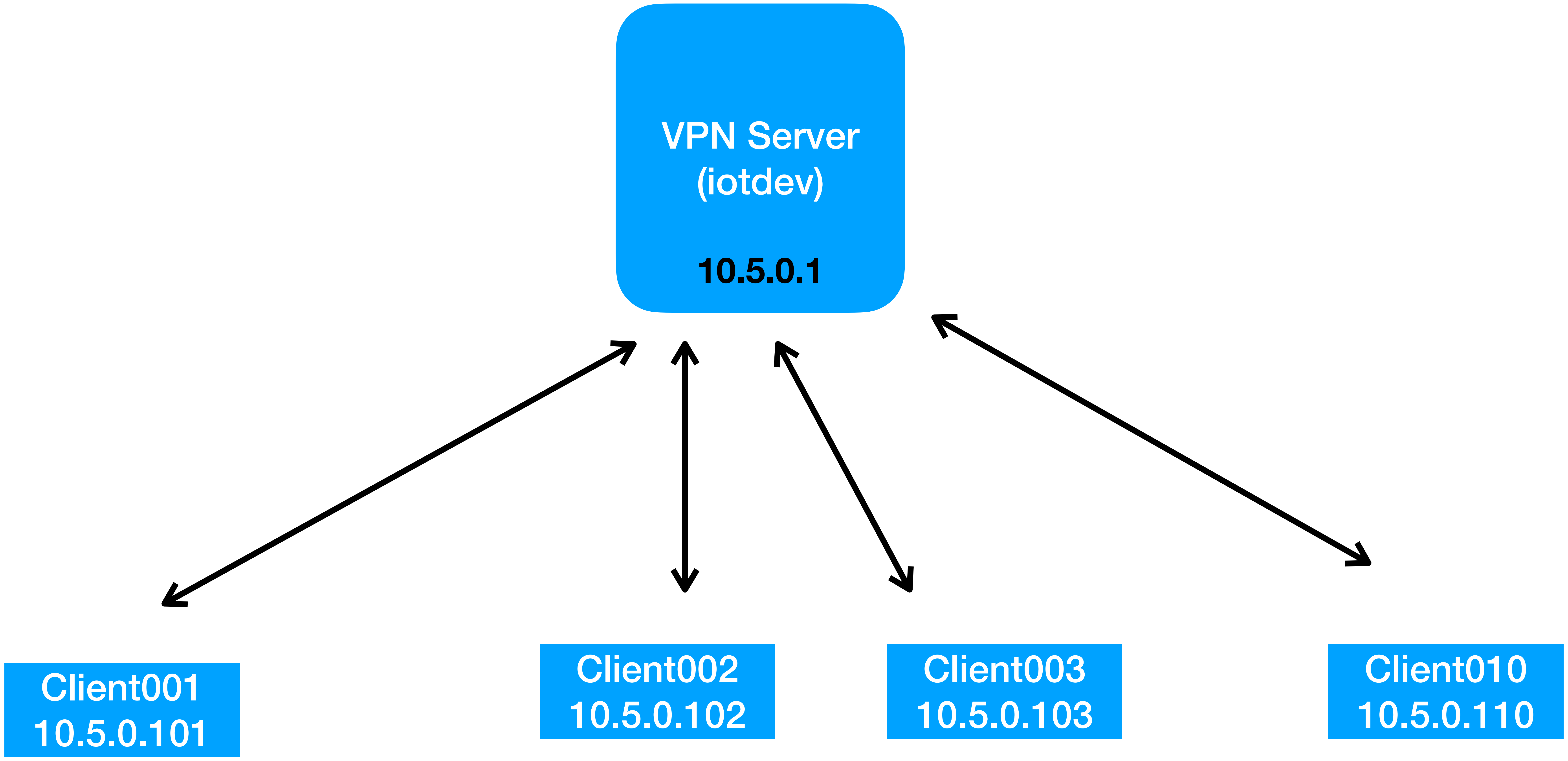
It looks like your Pi is just one hop away from the other Pi in the same VPN



```
pi@APINUN-PI:~ $ traceroute  10.5.0.101
traceroute to 10.5.0.101 (10.5.0.101), 30 hops max, 60 byte packets
 1  10.5.0.101 (10.5.0.101)  1406.382 ms  1406.327 ms  1406.278 ms
pi@APINUN-PI:~ $ traceroute  10.5.0.102
traceroute to 10.5.0.102 (10.5.0.102), 30 hops max, 60 byte packets
 1  10.5.0.1 (10.5.0.1)  628.042 ms  627.706 ms  627.470 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  *^C
pi@APINUN-PI:~ $ traceroute  10.5.0.103
traceroute to 10.5.0.103 (10.5.0.103), 30 hops max, 60 byte packets
 1  10.5.0.103 (10.5.0.103)  1870.464 ms  1870.139 ms  1869.898 ms
```

This is what it looks like from VPN perspective

This is an 'overlay' network



Exercises today

1. Bring your Pi home, connect your Pi to your local LAN (wire or wireless).
    1.1. See if you can ping your friend's Pi from home via OpenVPN.
    1.2. Try SSH from your Pi to your friend's Pi via OpenVPN and SSH public key.
        1.2.1. Create an SSH key pair on your Pi.
        1.2.2. Give your SSH public key to your friend
        1.2.3. Ask your friend to allow you to log on to his/her Pi via SSH public key
2. Try OpenVPN on your mobile phone, using OpenVPN app from the official app store
    2.1. Try the ovpn file with a fixed VPN IP (i.e. the one below client010)
    2.2. Try the ovpn file without a fixed VPN IP (i.e. the one from client011 and above)