P3: Basic Network Applications Practices

Exercise 1.1– In this exercise you must answer some basic questions about networking configuration.

//from terminal
simctl netapps-basic sh
start
get virt1 0
get virt2 0 #(virt2.0)
…….

1. Open all the consoles: virt1.0 (console 0 in virt1), virt1.1, virt2.0 and virt2.1. Then, figure out which is the port number of the service daytime.
Tip. Take a look at the file /etc/services.

//from virt1
cat /etc/services | grep daytime

```
root@virt1:~# cat /etc/services | grep daytime
daytime         13/tcp
daytime         13/udp
```

2. In a windows-like OS, which port number you expect for the daytime service?
13

3. Annotate the MAC and IP addresses of each interface on virt1 and virt2. Which command have you used?
we used ifconfig, but there could use
ifconfig| grep Hwaddr ->for MAC
ifconfig| grep inet -> for ips

virt1.0= fe:fd:00:00:01:00
inet addr:10.1.1.1 /24
virt2.0 = fe:fd:00:00:02:00
inet addr:10.1.1.2 /24

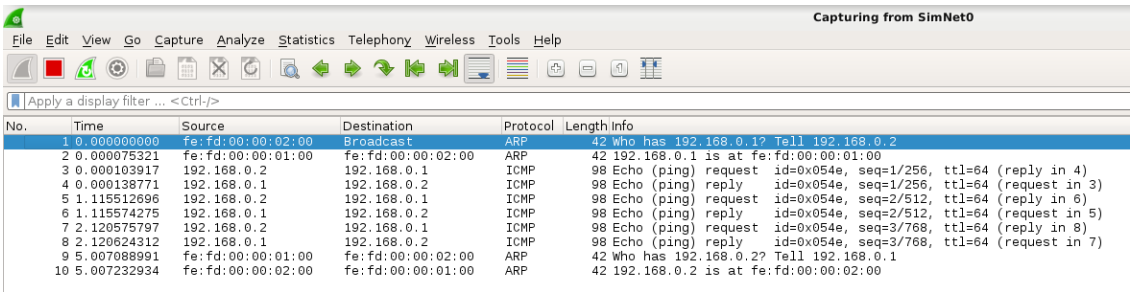
4. Assign the IP addresses 192.168.0.1 and 192.168.0.2 to the Ethernet interfaces of virt1 to virt2, respectively.
//from virt1
ifconfig eth0 192.168.0.1

//from virt2
ifconfig eth0 192.168.0.2

5. Send 3 ICMP echo-requests from virt2 to virt1 with the ping command.

//from virt2
ping -c 3 192.168.0.1

```
root@virt2:~# ping -c 3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=0.436 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=0.223 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=64 time=0.340 ms

--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.223/0.333/0.436/0.087 ms
```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe:fd:00:00:02:00	Broadcast	ARP	42	Who has 192.168.0.1? Tell 192.168.0.2
2	0.000075321	fe:fd:00:00:01:00	fe:fd:00:00:02:00	ARP	42	192.168.0.1 is at fe:fd:00:00:01:00
3	0.000103917	192.168.0.2	192.168.0.1	ICMP	98	Echo (ping) request id=0x054e, seq=1/256, ttl=64 (reply in 4)
4	0.000138771	192.168.0.1	192.168.0.2	ICMP	98	Echo (ping) reply id=0x054e, seq=1/256, ttl=64 (request in 3)
5	1.115512696	192.168.0.2	192.168.0.1	ICMP	98	Echo (ping) request id=0x054e, seq=2/512, ttl=64 (reply in 6)
6	1.115574275	192.168.0.1	192.168.0.2	ICMP	98	Echo (ping) reply id=0x054e, seq=2/512, ttl=64 (request in 5)
7	2.120575797	192.168.0.2	192.168.0.1	ICMP	98	Echo (ping) request id=0x054e, seq=3/768, ttl=64 (reply in 8)
8	2.120624312	192.168.0.1	192.168.0.2	ICMP	98	Echo (ping) reply id=0x054e, seq=3/768, ttl=64 (request in 7)
9	5.007088991	fe:fd:00:00:01:00	fe:fd:00:00:02:00	ARP	42	Who has 192.168.0.2? Tell 192.168.0.1
10	5.007232934	fe:fd:00:00:02:00	fe:fd:00:00:01:00	ARP	42	192.168.0.2 is at fe:fd:00:00:02:00

virt2 sent 3 ICMP echo request-reply successfully.

6. Find information and explain what is and for what can be used the loopback (lo) interface. Why lo does not have a MAC address?

Because it is the same machine, it is used to identify and communicate itself. Therefore, there is no MAC address. The lo is used to go directly to the same machine, without using the Link Level.

7. Configure the original IP addresses on each virtual machine.

//from virt1
ifconfig eth0 10.1.1.1
//from virt2
ifconfig eth0 10.1.1.2

Exercise 1.2– In this exercise you will start, stop and configure some network daemons (services).

1. Using the netstat command, list the TCP services that are active in virt1 under inetd. Describe the service names and ports used and check that the results are consistent with the configuration of inetd (/etc/inetd.conf).

//virt1

netstat -tnlp | grep inetd #shows ports number

netstat -tlp | grep inetd #shows services name

```
root@virt1:~# netstat -tnlp | grep inetd
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN      1262/inetd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      1262/inetd
root@virt1:~# netstat -tlp | grep inetd
tcp        0      0 *:auth                  *:*                     LISTEN      1262/inetd
tcp        0      0 *:ftp                   *:*                     LISTEN      1262/inetd
```

port 113, service= auth,
port 21, service=ftp

cat /etc/inetd.conf | grep ftp

cat /etc/inetd.conf | grep ident

```
root@virt1:~# cat /etc/inetd.conf | grep ftp
ftp             stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
root@virt1:~# cat /etc/inetd.conf | grep ident
ident           stream  tcp     wait    identd  /usr/sbin/identd        identd
```

Comparing with the configuration of inetd we observe that the both port are working

2. In virt1, edit the configuration of inetd and activate the service daytime. Restart the super-daemon and check that the port of daytime is being listened by inetd. Use nc to connect to this service from virt2 and describe what you observe.

//from virt1

nano /etc/inetd.conf

```
#daytime                stream  tcp     nowait  root    internal
```

#We take out the comment (#) to activate the line and make a reload

```
daytime         stream  tcp     nowait  root    internal
```

//from virt1

/etc/init.d/openbsd-inetd restart #we need to restart after modifying in .conf

cat /etc/services | grep daytime

```
root@virt1:~# cat /etc/services | grep daytime
daytime         13/tcp
daytime         13/udp
```

netstat -tnlp | grep inetd

```
tcp        0      0 0.0.0.0:13              0.0.0.0:*               LISTEN      1345/inetd
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN      1345/inetd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      1345/inetd
```

As we cat see, the port 13 is now listening with the PID 1345/inetd
//from virt1
/etc/init.d/openbsd-inetd reload
nc -l -p 8888 #creates a TCP socket that listens to the port 8888 (server)

-l=listens, -p=port

//from virt2
nc 10.1.1.1 8888 #client because isn't listening

```
root@virt1:~# /etc/init.d/openbsd-inetd reload
Reloading internet superserver: inetd.
root@virt1:~# /etc/init.d/openbsd-inetd reload
Reloading internet superserver: inetd.
root@virt1:~# nc -l -p 8888
a
holaaaaa
```

```
root@virt2:~# nc 10.1.1.1 8888
a
holaaaaa
```

we reload and set the netcat with port 8888 in virt1. we run virt1 as server and virt2 as client and the communication is established.

3. In virt1, check that the SSH daemon is listening to TCP port 22. Stop the SSH daemon and check that now the TCP port 22 is not being listened.

//virt1
cat /etc/services | grep ssh

```
root@virt1:~# cat /etc/services | grep ssh
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp
```

//from virt1
netstat -tnlp | grep ssh #1st netstat
/etc/init.d/ssh stop
netstat -tnlp | grep ssh #2n netstat

```
root@virt1:~# netstat -tnlp | grep ssh
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1348/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      1348/sshd
```

first netstat

```
root@virt1:~# /etc/init.d/ssh stop
Stopping OpenBSD Secure Shell server: sshd.
root@virt1:~# netstat -tnlp | grep ssh
root@virt1:~#
```

second netstat

cat /var/run/sshd.pid # (to see the id of the ssh process)

4. In virt1, change the configuration of the SSH daemon to listen to port 2222 instead of 22. Check your configuration. To finish this exercise, in virt1 change the configuration again of the SSH daemon to listen to its default port.

In the file config we change the line (Port 22) by (Port 2222) then we stop and start the service and look at the netstat.

//from virt1
nano /etc/ssh/sshd_config

```
  GNU nano 2.2.4          Fichero: /etc/ssh/sshd_config

# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
```

```
# What ports, IPs and protocols we listen for
Port 2222
```

we change the ssh configuration, we opened the file and changed 22 to 2222

//virt1
/etc/init.d/ssh stop
/etc/init.d/ssh start
netstat -tnlp | grep ssh

```
root@virt1:~# /etc/init.d/ssh stop
Stopping OpenBSD Secure Shell server: sshd.
root@virt1:~# /etc/init.d/ssh start
Starting OpenBSD Secure Shell server: sshd.
root@virt1:~# netstat -tnlp | grep ssh
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN      1386/sshd
tcp6       0      0 :::2222                 :::*                    LISTEN      1386/sshd
```

the next step was to return to default port 22, so we edit again the nano file and put 22 instead of 2222 and reload the netstat and we obtain the previous values.

```
root@virt1:~# /etc/init.d/ssh stop
Stopping OpenBSD Secure Shell server: sshd.
root@virt1:~# /etc/init.d/ssh start
Starting OpenBSD Secure Shell server: sshd.
root@virt1:~# netstat -tnlp | grep ssh
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1470/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      1470/sshd
root@virt1:~# netstat -tlp | grep ssh
tcp        0      0 *:ssh                   *:*                     LISTEN      1470/sshd
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      1470/sshd
root@virt1:~#
```

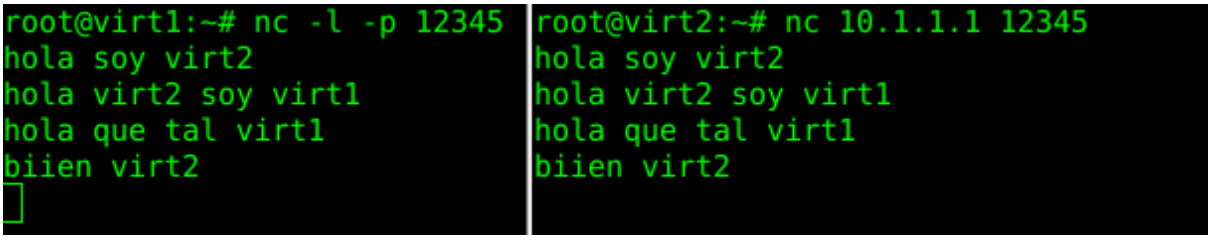
Exercise 1.3– In this exercise, you have to use netcat to create your own stand-alone servers.
1. Start a new capture on tap0 with wireshark in the phyhost and try the following command: tap0=SimNet

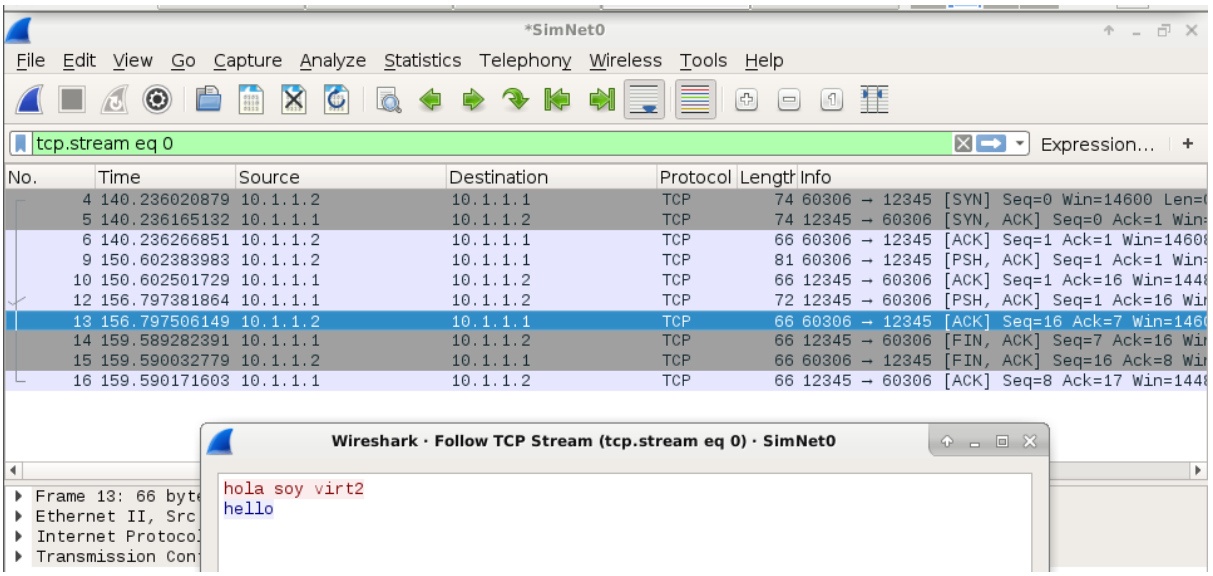
//from virt1
nc -l -p 12345

Describe what the previous command does. It is a client or a server? Describe how can you know the ports and open files related to this netcat process. Now try:
The -l flag sets the machine as a listener making it a server.
The -p flag indicates the port used.
so its a server

//from virt2
nc 10.1.1.1 12345

Describe the ports and the open files used by each netcat process. Send some text from each machine and terminate the connection. Describe the network parameters of the captured traffic. Use also the follow TCP stream option and describe how it works.





First virt2 send and SYN, then virt1 sends SYN and ACK, virt2 answers with an ACK . the connections is established, and the Follow TCP stream option shows the message we can choose the message in diferent encoding, and also decide the traffic of the message. To end the connection we send a FIN and ACK from virt1, virt2 responds with another FIN and ACK, finally virt1 sends the last ACK.

//from virt1
netstat -tnpl | grep nc
lsof -a -p 1382 -d0-10 #a=and, p=pid, d0-10=delimitations(show only 10 firsts)

```
root@virt1:~# netstat -tnpl | grep nc
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      1382/nc
tcp6       0      0 :::12345                ::::*                   LISTEN      1382/nc
root@virt1:~# lsof -a -p 1382 -d0-10
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nc      1382 root    0u   CHR    4,0      0t0  177 /dev/tty0
nc      1382 root    1u   CHR    4,0      0t0  177 /dev/tty0
nc      1382 root    2u   CHR    4,0      0t0  177 /dev/tty0
nc      1382 root    3u  IPv6   2375      0t0  TCP *:12345 (LISTEN)
nc      1382 root    4u  IPv4   2376      0t0  TCP *:12345 (LISTEN)
nc      1382 root    5u  IPv4   2377      0t0  TCP 10.1.1.1:12345->10.1.1.2:54149 (ESTABLISHED)
nc      1382 root    6u   CHR    4,0      0t0  177 /dev/tty0
nc      1382 root    7u   CHR    4,0      0t0  177 /dev/tty0
```

In this photo we can see the ports and the open files used by each netcat process.

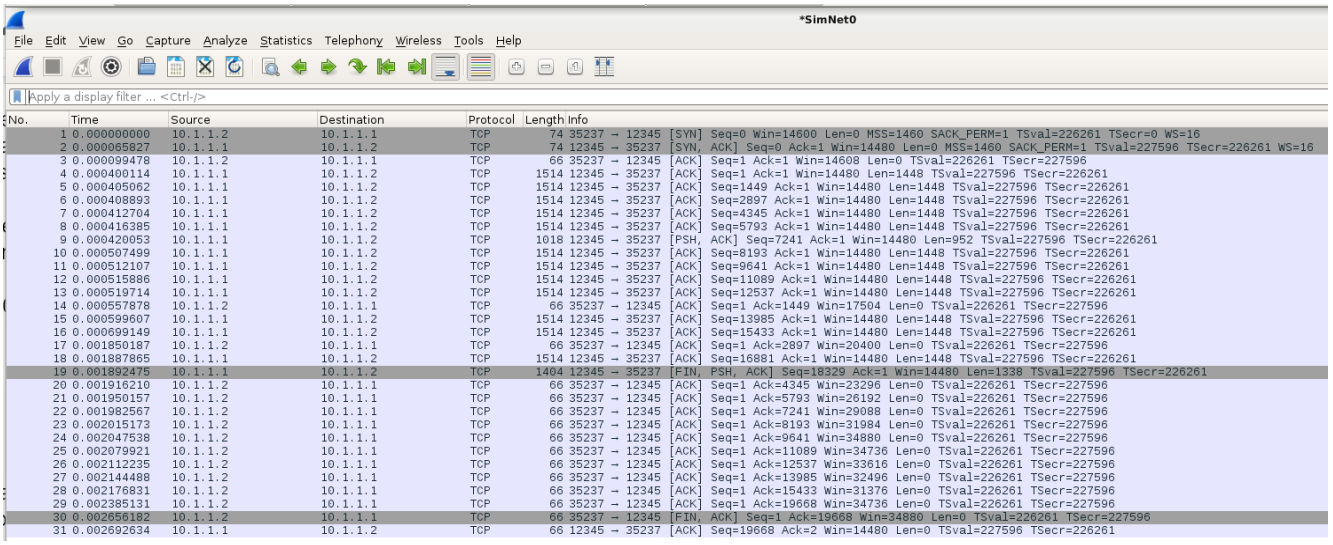
2. Start a new capture on tap0 with wireshark in the phyhost. In virt1, create a server with netcat that listens on port 23456 and transfers the file /etc/services.
Tip. Use the option -q 0 to quit after the transmission of the file.

Connect to the previous server with a nc from virt2 and store the file sent by the server with the name “file.txt”. For captured traffic, describe what you observe using the option follow TCP stream.

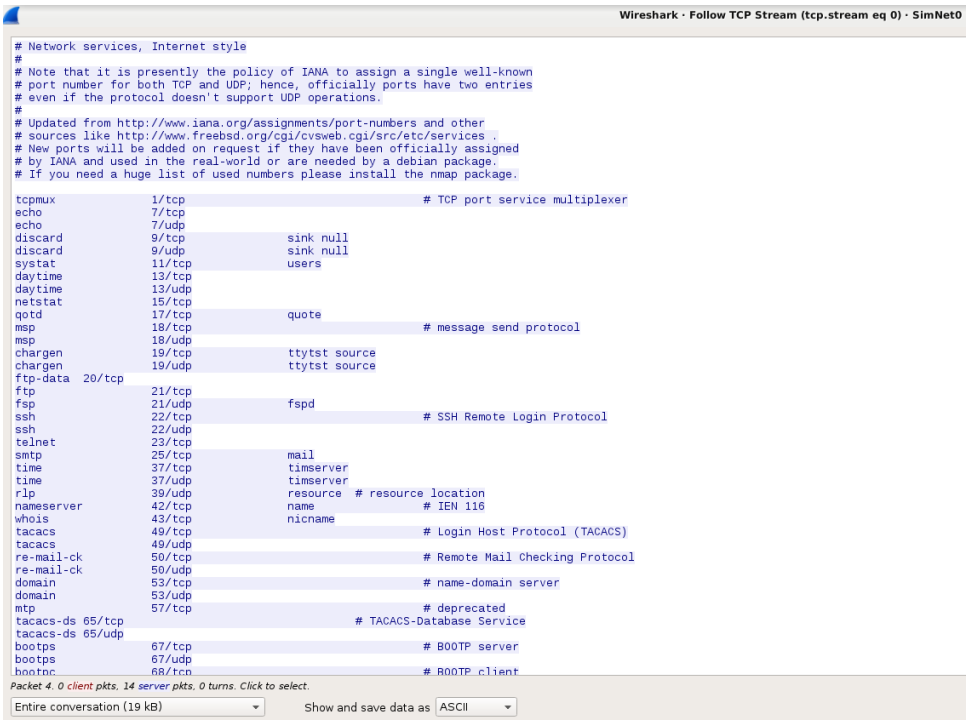
We send the file services from virt1 to virt2 creating a file named file.txt in virt2 with the content.

//from virt1
cat /etc/services | nc -l -p 23456 -q0
#-q, --hold-timeout

//from virt2
nc 10.1.1.1 23456 > file.txt



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.2	10.1.1.1	TCP	74	35237 → 12345 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=226261 TSecr=0 WS=16
2	0.000065827	10.1.1.1	10.1.1.2	TCP	74	12345 → 35237 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=227596 TSecr=226261 WS=16
3	0.000099478	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=226261 TSecr=227596
4	0.000400114	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=1 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
5	0.000405062	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=1449 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
6	0.000408893	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=2897 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
7	0.000412704	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=4345 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
8	0.000416385	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=5793 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
9	0.000420053	10.1.1.1	10.1.1.2	TCP	1018	12345 → 35237 [PSH, ACK] Seq=7241 Ack=1 Win=14480 Len=952 TSval=227596 TSecr=226261
10	0.000507499	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=8193 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
11	0.000512107	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=9641 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
12	0.000515886	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=11089 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
13	0.000519714	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=12537 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
14	0.000557878	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=1449 Win=17504 Len=0 TSval=226261 TSecr=227596
15	0.000599607	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=13985 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
16	0.000699149	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=15433 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
17	0.001850187	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=2897 Win=20400 Len=0 TSval=226261 TSecr=227596
18	0.001887865	10.1.1.1	10.1.1.2	TCP	1514	12345 → 35237 [ACK] Seq=16881 Ack=1 Win=14480 Len=1448 TSval=227596 TSecr=226261
19	0.001892475	10.1.1.1	10.1.1.2	TCP	1404	12345 → 35237 [FIN, PSH, ACK] Seq=18329 Ack=1 Win=14480 Len=1338 TSval=227596 TSecr=226261
20	0.001916210	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=4345 Win=23296 Len=0 TSval=226261 TSecr=227596
21	0.001950157	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=5793 Win=26192 Len=0 TSval=226261 TSecr=227596
22	0.001982567	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=7241 Win=29088 Len=0 TSval=226261 TSecr=227596
23	0.002015173	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=8193 Win=31984 Len=0 TSval=226261 TSecr=227596
24	0.002047538	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=9641 Win=34880 Len=0 TSval=226261 TSecr=227596
25	0.002079921	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=11089 Win=34736 Len=0 TSval=226261 TSecr=227596
26	0.002112235	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=12537 Win=33616 Len=0 TSval=226261 TSecr=227596
27	0.002144488	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=13985 Win=32496 Len=0 TSval=226261 TSecr=227596
28	0.002176831	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=15433 Win=31376 Len=0 TSval=226261 TSecr=227596
29	0.002385131	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [ACK] Seq=1 Ack=19668 Win=34736 Len=0 TSval=226261 TSecr=227596
30	0.002656182	10.1.1.2	10.1.1.1	TCP	66	35237 → 12345 [FIN, ACK] Seq=1 Ack=19668 Win=34880 Len=0 TSval=226261 TSecr=227596
31	0.002692634	10.1.1.1	10.1.1.2	TCP	66	12345 → 35237 [ACK] Seq=19668 Ack=2 Win=14480 Len=0 TSval=227596 TSecr=226261

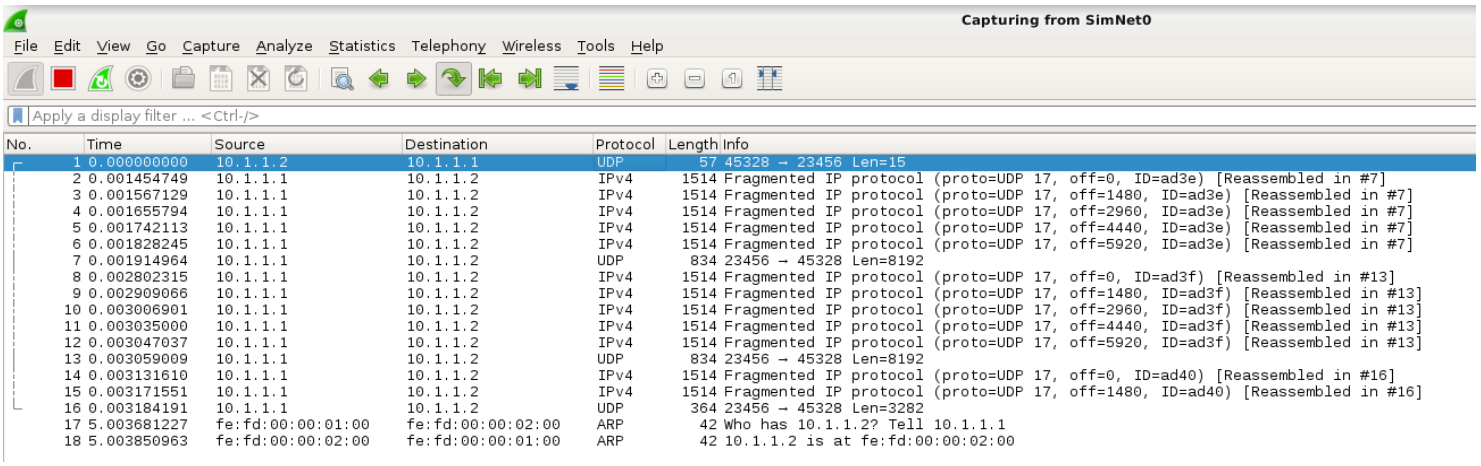


As we can observe, follow tcp stream shows the traffic of information in plain text.

3. Start a new capture on tap0 with wireshark in the phyhost. Repeat the steps of the previous exercise but this time using UDP.
Note. In the client you have to type two times “ENTER“ to start the data reception.
For captured traffic, describe what you observe and the differences between the previous capture. Notice that in this case you can use the option follow UDP stream.

//from virt1
cat /etc/services | nc -l -u -p 23456 -q0
#u=udp

//from virt2
nc -u 10.1.1.1 23456 > file.txt



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.2	10.1.1.1	UDP	57	45328 → 23456 Len=15
2	0.001454749	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=ad3e) [Reassembled in #7]
3	0.001567129	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=ad3e) [Reassembled in #7]
4	0.001655794	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=ad3e) [Reassembled in #7]
5	0.001742113	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=4440, ID=ad3e) [Reassembled in #7]
6	0.001828245	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=5920, ID=ad3e) [Reassembled in #7]
7	0.001914964	10.1.1.1	10.1.1.2	UDP	834	23456 → 45328 Len=8192
8	0.002802315	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=ad3f) [Reassembled in #13]
9	0.002909066	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=ad3f) [Reassembled in #13]
10	0.003006901	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=ad3f) [Reassembled in #13]
11	0.003035000	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=4440, ID=ad3f) [Reassembled in #13]
12	0.003047037	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=5920, ID=ad3f) [Reassembled in #13]
13	0.003059009	10.1.1.1	10.1.1.2	UDP	834	23456 → 45328 Len=8192
14	0.003131610	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=ad40) [Reassembled in #16]
15	0.003171551	10.1.1.1	10.1.1.2	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=ad40) [Reassembled in #16]
16	0.003184191	10.1.1.1	10.1.1.2	UDP	364	23456 → 45328 Len=3282
17	5.003681227	fe:fd:00:00:01:00	fe:fd:00:00:02:00	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
18	5.003850963	fe:fd:00:00:02:00	fe:fd:00:00:01:00	ARP	42	10.1.1.2 is at fe:fd:00:00:02:00

The most important difference is now the ack is not used.
In UDP there is no handshake therefore we don't establish and end connection (no use of SYN and FIN frames). There is no flow control (no ACK).

4. In the phyhost, create a server with netcat that listens on port 12345 and emulates the daytime service.
Tip. Use the date command.
After several seconds, try the service from phyhost with nc. Which interface you should use to capture the traffic of the previous network exchange?

Which is the actual date (minutes and seconds) that you observe as output in the client nc? It is correct? Which do you think that is the cause of this behavior?

//from terminal
sudo ifconfig SimNet0 10.1.1.3/24
date | nc -l -p 12345 -q0

```
telem@debian:~$ sudo ifconfig SimNet0 10.1.1.3/24
telem@debian:~$ date | nc -l -p 12345 -q0
```

//from virt1 (after some min)
nc 10.1.1.3 12345

```
root@virt1:~# nc 10.1.1.3 12345
Mon Oct 10 20:51:33 CEST 2022
root@virt1:~# date
lun oct 10 20:54:20 CEST 2022
```

The command date gives the time when the connections were stablished.
In order to the interface, we configured the SinNet 0, so this interface captures the traffic.

5. In virt1, create a server with netcat that listens on port 22333 and provides to the client the amount of free disk in the machine. Tip. Use the df -h command.
(dh -f)=show the file system disk space statistics in "human-readable" format,

//virt1
df -h | nc -l -p 22333 -q0

//virt2
nc 10.1.1.1 22333

```
root@virt2:~# nc 10.1.1.1 22333
S.ficheros            Size  Used Avail Use% Montado en
/dev/ubda             2,0G  1,6G  385M  81% /
tmpfs                  30M     0   30M   0% /lib/init/rw
udev                   10M   16K   10M   1% /dev
tmpfs                  30M     0   30M   0% /dev/shm
tmpfs                  30M  1,5M   28M   5% /tmp
tmpfs                  30M     0   30M   0% /var/tmp
/dev/ubdb             352K  352K     0 100% /mnt/vnuml
none                   37G  8,1G   27G  24% /mnt/hostfs
```

Try the service from virt2 with nc.
//virt1
nc 10.1.1.2 22333

//virt2
df -h | nc -l -p 22333 -q0

```
root@virt1:~# nc 10.1.1.2 22333
S.ficheros            Size  Used Avail Use% Montado en
/dev/ubda             2,0G  1,6G  385M  81% /
tmpfs                  30M     0   30M   0% /lib/init/rw
udev                   10M   16K   10M   1% /dev
tmpfs                  30M     0   30M   0% /dev/shm
tmpfs                  30M     0   30M   0% /tmp
tmpfs                  30M     0   30M   0% /var/tmp
/dev/ubdb             352K  352K     0 100% /mnt/vnuml
none                   37G  8,1G   27G  24% /mnt/hostfs
```

As we can see, with the df command we can see the free amount of disk in the machine.

Exercise 1.4– In this exercise, you must create a service under inetd.
1. In virt1, implement a service using inetd (without using a netcat as server) that listens on port 22333 and that provides the amount of free disk in the system to the client. Explain the configuration that you have to make.
//from virt1
/etc/init.d/openbsd-inetd start
nano /etc/inetd.conf # (here we add this new server, writing this)
diskfree stream tcp nowait root /root/diskfree.sh

```
#:STANDARD: These are standard services.
ftp             stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ft$
#telnet         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.te$
diskfree        stream  tcp     nowait  root    /root/diskfree.sh
#:BSD: Shell, login, exec and talk are BSD protocols.
```

nano /etc/services #(we add the port by editing in /etc/services)

```
wnn6            22273/tcp
wnn6            22273/udp
diskfree        22333/tcp
#
```

/etc/init.d/openbsd-inetd restart # (necessary to do, after modifying .sh files)

nano diskfree.sh # (we edited the script, and save in diskfree.sh)

```
virt1.
File  Edit  View  Search  Terminal  Help
  GNU nano 2.2.4            Fichero: diskfree.s

#!/bin/bash
df -h
cat
pwd
ls
```

chmod u+x /root/diskfree.sh # we give the edit premission

//from virt2

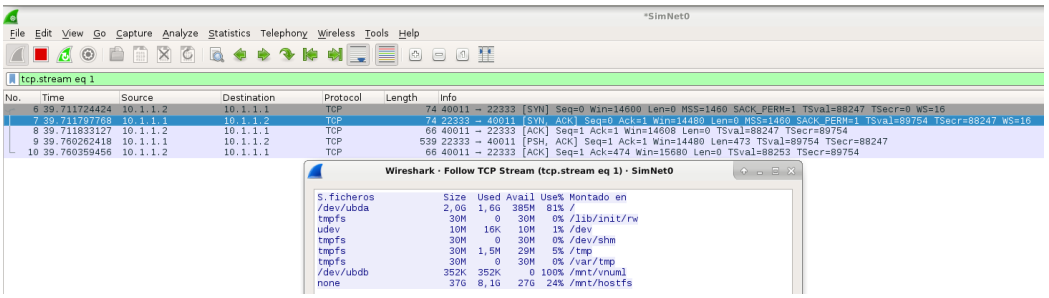
nc 10.1.1.1 22333

```
root@virt2:~# nc 10.1.1.1 22333
S.ficheros            Size  Used Avail Use% Montado en
/dev/ubda             2,0G  1,6G  385M  81% /
tmpfs                  30M     0   30M   0% /lib/init/rw
udev                   10M   16K   10M   1% /dev
tmpfs                  30M     0   30M   0% /dev/shm
tmpfs                  30M  1,5M   29M   5% /tmp
tmpfs                  30M     0   30M   0% /var/tmp
/dev/ubdb             352K  352K     0 100% /mnt/vnuml
none                   37G  8,1G   27G  24% /mnt/hostfs
```

In this case, can you connect to this service several times? To this respect, explain how this implementation works and the differences with respect to using a simple server with netcat. Tip. If you have problems, check /var/log/daemon.log which is the inetd's log file.

```
root@virt1:~# tail -f /var/log/daemon.log
Nov  6 13:24:13 vnx /usr/sbin/smcroute[1131]: Debu: clean handler called
Nov  6 13:24:13 vnx rpc.statd[546]: Caught signal 15, un-registering and exiting
Nov  6 13:25:00 vnx /usr/sbin/smcroute[1176]: Debu: buildIfVc: Interface lo Phy-Ix 1 Addr: 127.0.0.1, Flags: 0x0049
Nov  6 13:25:48 vnx init: Switching to runlevel: 0
Nov  6 13:25:50 vnx /usr/sbin/smcroute[1177]: Debu: readIpcServer, waiting for connection...
Nov  6 13:25:50 vnx /usr/sbin/smcroute[1177]: Debu: readIpcServer, accepted connection
Nov  6 13:25:50 vnx /usr/sbin/smcroute[1177]: Debu: readIpcServer, CmdPkt read (9)
Nov  6 13:25:50 vnx /usr/sbin/smcroute[1177]: Debu: clean handler called
Nov  6 13:25:51 vnx rpc.statd[602]: Caught signal 15, un-registering and exiting
Oct 10 22:03:18 vnx /usr/sbin/smcroute[1166]: Debu: buildIfVc: Interface lo Phy-Ix 1 Addr: 127.0.0.1, Flags: 0x0049
```

//from wireshark



//from virt1
nano /etc/inetd.conf
nano /etc/services
nano diskfree.sh
chmod +x diskfree.sh
/etc/init.d/openbsd-inetd start
/etc/init.d/openbsd-inetd stop
/etc/init.d/openbsd-inetd start

//from virt2
nc 10.1.1.1 22333

Exercise 1.5– In this exercise, we are going to analyze the Web service.
1. In virt1, start the apache2 WEB server. To do so, type the following command (notice that we avoid viewing
STDERR):

//from virt1
ls /etc/apache2

```
root@virt1:~# ls /etc/apache2
apache2.conf  envvars     magic           mods-enabled  sites-available
conf.d        httpd.conf  mods-available  ports.conf    sites-enabled
root@virt1:~#
```

la config está en /etc/apache2

//from virt1
/etc/init.d/apache2 start 2> /dev/null

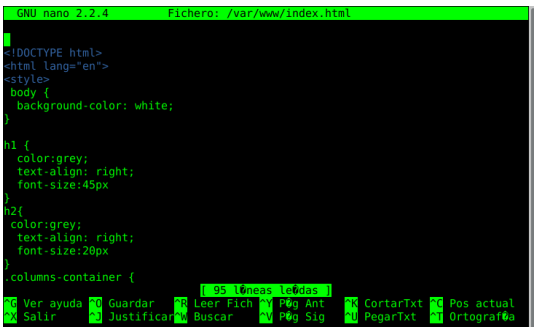
```
root@virt1:~# /etc/init.d/apache2 start 2> /dev/null
Starting web server: apache2httpd (pid 859) already running
.
root@virt1:~#
```

Do yo see a configuration line in inetd for apache2 why? Find out the PID of the process that is listening on port 80 in virt1. Is this process inetd? Describe how you do it.
//from vrit1
netstat -tlnp | grep apache

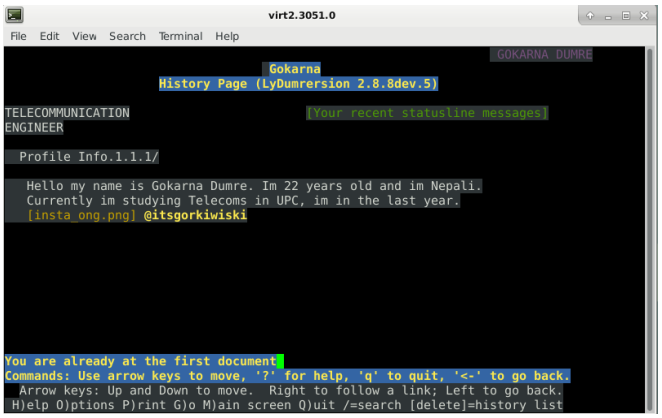
```
root@virt1:~# netstat -tlnp | grep apache
tcp6       0      0 :::80                   :::*                    LISTEN      751/apache2
```

PID= 751. There is not any inetd configuration

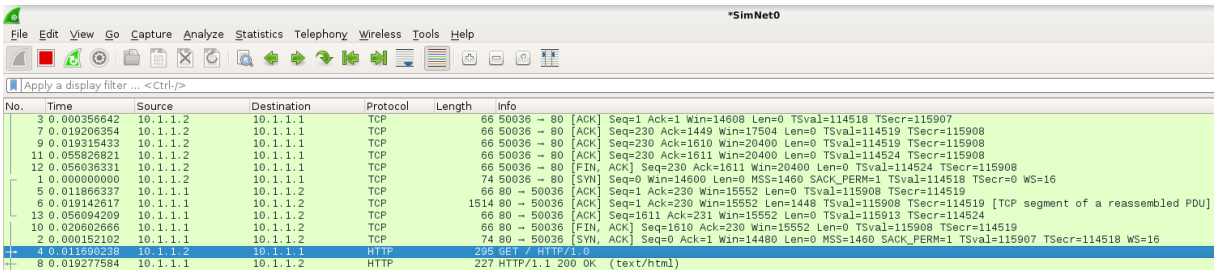
2. In virt1, edit the file /var/www/index.html with vi and without modifying the HTML tags, change some of the text of It works!. Using a console in virt2 and the lynx WEB browser, display the web page of virt1.
//from virt1
nano /var/www/index.html



//from virt2
lynx http://10.1.1.1



Here first at virt1 we edited /var/www/index. html with our html code and then at virt2 we acces to the web server and we can see the result in the console.

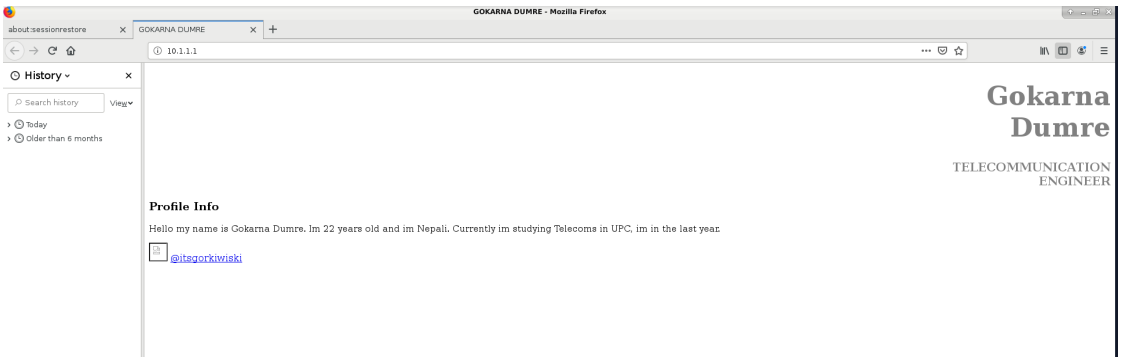


This is the traffic captured by wireshark.

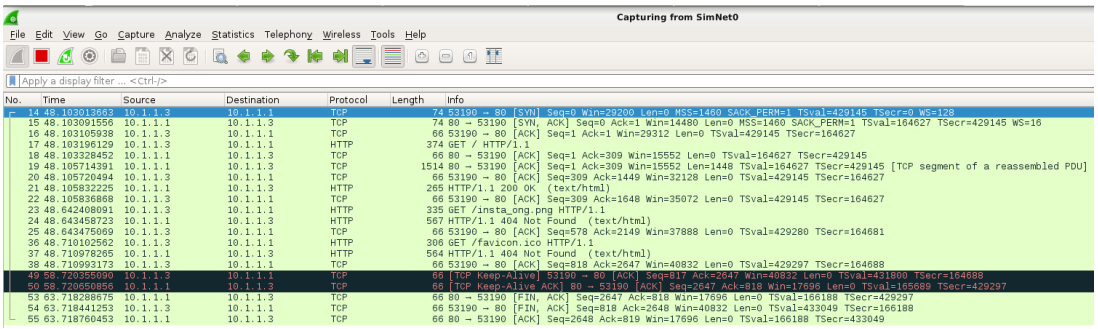
3. Start a new capture on tap0 with wireshark in the phyhost. Also in the phyhost, give the IP address 10.1.1.3/24 to the tap0 interface. Open a firefox in the phyhost and use it to see the web page served by virt1. For captured traffic, use the follow TCP stream option of wireshark and roughly comment the protocols that you see.

//from terminal
sudo ifconfig SimNet0 10.1.1.3/24
/usr/bin/firefox http://10.1.1.1

//from virt1
/etc/init.d/apache2 stop
#the page

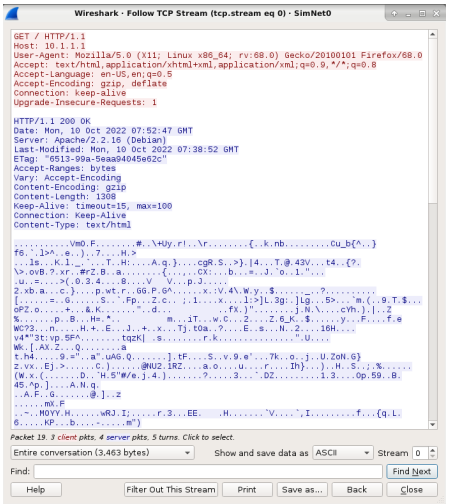


#the traffic



No.	Time	Source	Destination	Protocol	Length	Info
14	48.103013663	10.1.1.3	10.1.1.1	TCP	74	53190 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=429145 TSecr=0 WS=128
15	48.103091556	10.1.1.1	10.1.1.3	TCP	74	80 → 53190 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=164627 TSecr=429145 WS=16
16	48.103105938	10.1.1.3	10.1.1.1	TCP	66	53190 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=429145 TSecr=164627
17	48.103196129	10.1.1.3	10.1.1.1	HTTP	374	GET / HTTP/1.1
18	48.103328452	10.1.1.1	10.1.1.3	TCP	66	80 → 53190 [ACK] Seq=1 Ack=309 Win=15552 Len=0 TSval=164627 TSecr=429145
19	48.105714391	10.1.1.1	10.1.1.3	TCP	1514	80 → 53190 [ACK] Seq=1 Ack=309 Win=15552 Len=1448 TSval=164627 TSecr=429145 [TCP segment of a reassembled PDU]
20	48.105720494	10.1.1.3	10.1.1.1	TCP	66	53190 → 80 [ACK] Seq=309 Ack=1449 Win=32128 Len=0 TSval=429145 TSecr=164627
21	48.105832225	10.1.1.1	10.1.1.3	HTTP	265	HTTP/1.1 200 OK (text/html)
22	48.105836868	10.1.1.3	10.1.1.1	TCP	66	53190 → 80 [ACK] Seq=309 Ack=1648 Win=35072 Len=0 TSval=429145 TSecr=164627
23	48.642408091	10.1.1.3	10.1.1.1	HTTP	335	GET /insta_ong.png HTTP/1.1
24	48.643458723	10.1.1.1	10.1.1.3	HTTP	567	HTTP/1.1 404 Not Found (text/html)
25	48.643475069	10.1.1.3	10.1.1.1	TCP	66	53190 → 80 [ACK] Seq=578 Ack=2149 Win=37888 Len=0 TSval=429280 TSecr=164681
36	48.710102562	10.1.1.3	10.1.1.1	HTTP	306	GET /favicon.ico HTTP/1.1
37	48.710978265	10.1.1.1	10.1.1.3	HTTP	564	HTTP/1.1 404 Not Found (text/html)
38	48.710993173	10.1.1.3	10.1.1.1	TCP	66	53190 → 80 [ACK] Seq=818 Ack=2647 Win=40832 Len=0 TSval=429297 TSecr=164688
49	58.720355090	10.1.1.3	10.1.1.1	TCP	66	[TCP Keep-Alive] 53190 → 80 [ACK] Seq=817 Ack=2647 Win=40832 Len=0 TSval=431800 TSecr=164688
50	58.720650856	10.1.1.1	10.1.1.3	TCP	66	[TCP Keep-Alive ACK] 80 → 53190 [ACK] Seq=2647 Ack=818 Win=17696 Len=0 TSval=165689 TSecr=429297
53	63.718288675	10.1.1.1	10.1.1.3	TCP	66	80 → 53190 [FIN, ACK] Seq=2647 Ack=818 Win=17696 Len=0 TSval=166188 TSecr=429297
54	63.718441253	10.1.1.3	10.1.1.1	TCP	66	53190 → 80 [FIN, ACK] Seq=818 Ack=2648 Win=40832 Len=0 TSval=433049 TSecr=166188
55	63.718760453	10.1.1.1	10.1.1.3	TCP	66	80 → 53190 [ACK] Seq=2648 Ack=819 Win=17696 Len=0 TSval=166188 TSecr=433049

#the follow tcp stream



Firstly the terminal (pyhost) and virt1 stablished connection with SYN and ACK frames.Secondly, pyhost sends HTTP request and virt1 responds with a HTTP response, then we stop the the web server.

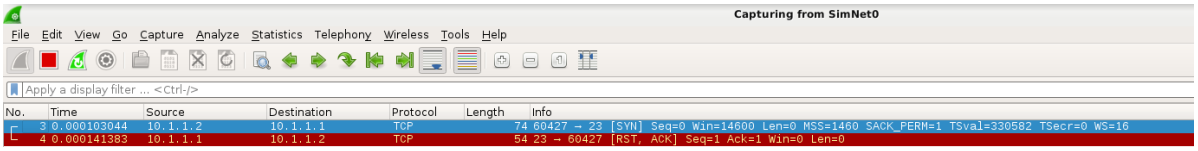
Exercise 1.6– In this exercise, we are going to analyze the remote terminal service which TELNET and SSH.

1. Start a new capture on tap0 with wireshark in the phyhost. Try to establish a remote terminal on virt1 using telnet from virt2. Did it work? Explain the output of the telnet command and the captured traffic. Which do you think that is cause of the behavior observed?

//from virt2

telnet 10.1.1.1

```
root@virt2:~# telnet 10.1.1.1
Trying 10.1.1.1...
telnet: Unable to connect to remote host: Connection refused
```



It doesn't work because first we have to activate the Telnet service in virt1.
(the RST flag is sended by a receptor indicating that the frame received was unexpected).

/from virt1

```
root@virt1:~# netstat -tlnp | grep telnet
root@virt1:~#
```

As we can see in virt1, telnet is not "listening".

2. Activate the TELNET service under inetd in virt1.

Note. Be careful to not to leave any space at the beginning of inetd configuration lines.

Check your configuration with netstat and start a capture on tap0 with wireshark in the phyhost. Try to establish a remote terminal from virt2 to virt1 using telnet. Did it work? explain what you observe. Take a look at the /var/log/daemon.log and describe the messages in that file.

//from virt1

nano /etc/inetd.conf #here we discomment the telnet line, so its active

/etc/init.d/openbsd-inetd reload #then reload the server

//from virt2

telnet 10.1.1.1

```
root@virt2:~# telnet 10.1.1.1
Trying 10.1.1.1...
Connected to 10.1.1.1.
Escape character is '^]'.
Debian GNU/Linux 6.0
virt1 login: root

Login incorrect
virt1 login: Connection closed by foreign host.
```

//from virt1

tail -f /var/log/daemon.log #to see the error

```
root@virt1:~# tail -f /var/log/daemon.log
Oct 10 10:30:33 vnx /usr/sbin/smcroute[1199]: Debu: clean handler called
Oct 10 10:30:34 vnx rpc.statd[603]: Caught signal 15, un-registering and exiting
Oct 10 10:32:22 vnx inetd[1251]: stream/nowait: *: ai_socktype not supported
Oct 10 10:32:22 vnx /usr/sbin/smcroute[1271]: Debu: buildIfVc: Interface lo Phy-Ix 1 Addr: 127.0.0.1, Flags: 0x0049
Oct 10 10:40:33 vnx in.telnetd[1395]: connect from 10.1.1.2 (10.1.1.2)
Oct 10 10:40:33 vnx telnetd[1395]: doit: getnameinfo: Success
Oct 10 10:40:33 vnx telnetd[1395]: doit: getaddrinfo: Name or service not known
```

The connection is stablished but the credentials are wrong, because the root user cannot access with TELNET to a remote machine. In the wireshark we can see the flags changing between virt1 and virt2 and if we follow the tcp stream we can see the messages.

3. In general, the root user cannot access with TELNET to a remote machine. To enable to this possibility, you have to enable one or more TTYs in the file /etc/securetty. Each line with pts/X enables a TTY or possible TELNET connection. Start a new capture on tap0 with wireshark in the phyhost, enable a TTY for the root user in virt1 (uncommenting one this lines at the beginning of /etc/securetty) and try again to establish a remote terminal from virt2 to virt1 using telnet from virt2.0. Using virt2.1 and the TELNET session in virt2.0, type a netstat command with the appropriate parameters to check the ports used by the TELNET connection and the processes that have registered these ports. Explain the differences of what you see in virt2.0 and virt2.1. Check the file descriptors used by the telnet client and the telnet server.

//from virt1

nano /etc/securetty #here we disconnect the pts/o line

```
  GNU nano 2.2.4          Fichero: /etc/securetty

# /etc/securetty: list of terminals on which root is allowe
# See securetty(5) and login(1).

console

# For accessing with root with a TELNET (by JOSE)
pts/0
pts/1
```

//from virt1 netstat -tnlp | grep inetd

```
root@virt1:~# netstat -tlnp | grep inetd
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN      1252/inetd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      1252/inetd
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      1252/inetd
root@virt1:~#
```

```
root@virt2:~# netstat -tlnp | grep inetd
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN      1233/inetd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      1233/inetd
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      1233/inetd
root@virt2:~#
```

We can also notice that the PID is different for the same proces TELNET in virt1 and virt2.

Now the Telnet works!

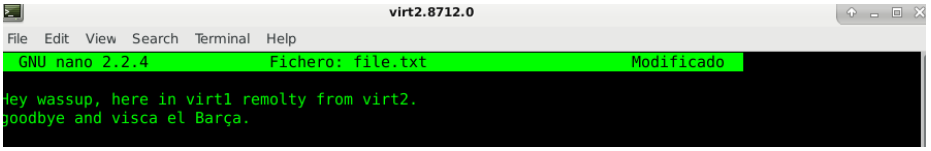
```
root@virt2:~# telnet 10.1.1.1
Trying 10.1.1.1...
Connected to 10.1.1.1.
Escape character is '^]'.
Debian GNU/Linux 6.0
virt1 login: root
Password:
Last login: Mon Oct 10 11:07:36 CEST 2022 on pts/0
Linux vnx 3.3.8 #1 Sun Nov 6 04:59:42 MST 2016 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@virt1:~#
```

4. Under the TELNET session, create an empty file called file.txt in the home directory of the root user in virt1 and exit. With virt1.0 check the creation of the file.

//from virt2

Telnet 10.1.1.1

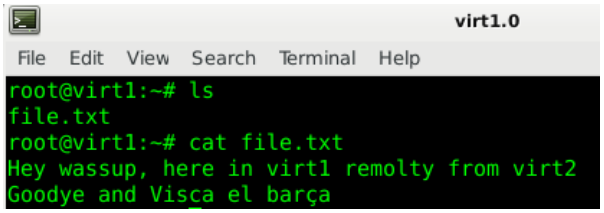
nano file.txt



//from virt1.0

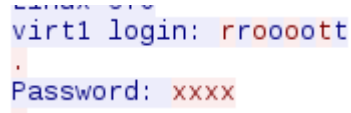
ls

cat file.txt



5. Use the follow TCP stream option of wireshark and comment the TELNET protocol. What do you think about the security of this protocol?

//in wireshark

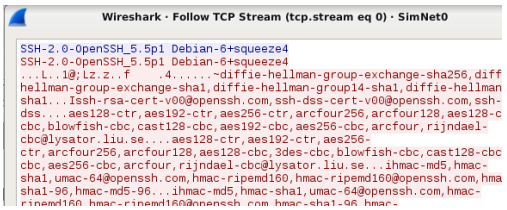


we can obtain the credentials of virt1, the reason is because Telnet isn't a encrypted protocol and the data through these protocol is not secure.

6. Capture in tap0 and try an SSH session from virt2 to virt1. Use the follow TCP stream option of wireshark and comment the differences between SSH with TELNET about ports used and security.

//from virt2

ssh 10.1.1.1



In comparation with the Telnet protocol, SSH is encrypted and the date through these protocol is secure.

//from virt2

netstat -tnlp | grep telnet/ssh # to see the ports used by each protocol

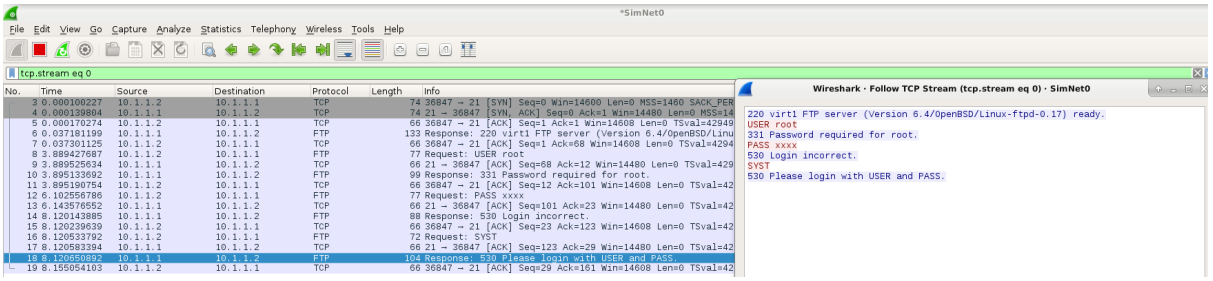
telnet port= 23, ssh port=22

Exercise 1.7– In this exercise, we are going to analyze the file transfer service with FTP, SCP and SFTP.

1. Start a new capture on tap0 with wireshark in the phyhost. Then, establish an FTP session from virt2 to virt1 using the root user and the console virt2.0. Did it work? Use the follow TCP stream option of wireshark and comment what you observe.

//from virt2

ftp 10.1.1.1

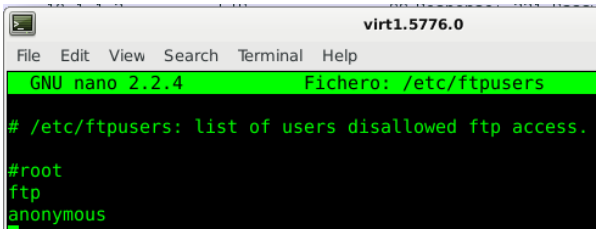


It doesn't work, according to the ftp the credentials are wrongs.

2. The FTP access for the root user is blocked by default as with TELNET. To enable it, you have to modify the configuration file /etc/ftpusers by removing or commenting (using the # symbol at the beginning of the line) the line for the root user. Using the console virt1.0 allow the FTP access for the root user.

//from virt1

nano /etc/ftpusers



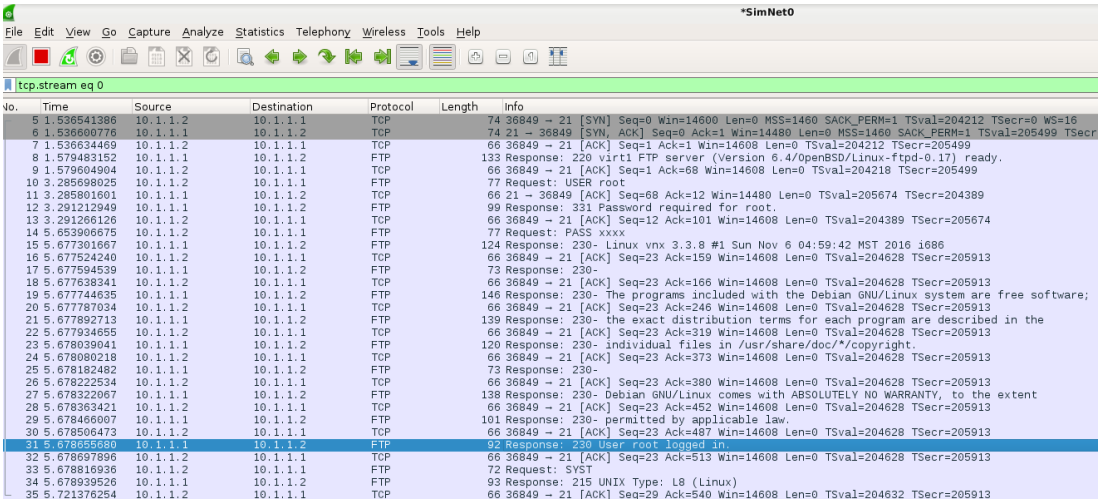
3. Start a new capture on tap0 with wireshark in the phyhost and establish an FTP session from virt2 to virt1 using the console virt2.0. Using the console virt2.1 check the ports and the file descriptors used in virt2. Using the console virt1.0 check the ports and the file descriptors used in virt1.

//from virt2

ftp 10.1.1.1

```
root@virt2:~#  ftp 10.1.1.1
Connected to 10.1.1.1.
220 virt1 FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) ready.
Name (10.1.1.1:root): root
331 Password required for root.
Password:
230- Linux vnx 3.3.8 #1 Sun Nov 6 04:59:42 MST 2016 i686
230-
230- The programs included with the Debian GNU/Linux system are free software;
230- the exact distribution terms for each program are described in the
230- individual files in /usr/share/doc/*/copyright.
230-
230- Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
230- permitted by applicable law.
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

The connections is stablished in binary mood.



//from virt1
netstat -tnlp | grep 21 #we obtain the pid=1171
lsof -a -p 1171 -d0-10

```
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
1171/inetd
root@virt1:~# lsof -a -p 1171 -d0-10
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
inetd    1171 root    0u   CHR    1,3      0t0   41 /dev/null
inetd    1171 root    1u   CHR    1,3      0t0   41 /dev/null
inetd    1171 root    2u   CHR    1,3      0t0   41 /dev/null
inetd    1171 root    4u  IPv4   1729      0t0  TCP *:ftp (LISTEN)
inetd    1171 root    6u  IPv4   1735      0t0  TCP *:auth (LISTEN)
```

//from virt2.1
netstat -tnlp | grep 21 #we obtain the pid=1178
lsof -a -p 1178 -d0-10

```
root@virt2:~# netstat -tnlp | grep 21
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
1178/inetd
root@virt2:~# lsof -a -p 1178 -d0-10
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
inetd    1178 root    0u   CHR    1,3      0t0   41 /dev/null
inetd    1178 root    1u   CHR    1,3      0t0   41 /dev/null
inetd    1178 root    2u   CHR    1,3      0t0   41 /dev/null
inetd    1178 root    4u  IPv4   1870      0t0  TCP *:ftp (LISTEN)
inetd    1178 root    5u  IPv4   1873      0t0  TCP *:telnet (LISTEN)
inetd    1178 root    6u  IPv4   1876      0t0  TCP *:auth (LISTEN)
```

In virt1 (server) we have 2 fd (no telnet) but in virt2(client) we have 3 fds(yes telnet).

4. Use the FTP session to get all the files in /usr/bin that start with “z” and exit. Which is the default data repre-sentation for these transmissions?

//from virt1
ls /usr/bin/z*

```
root@virt1:~# ls /usr/bin/z*
/usr/bin/zdump  /usr/bin/zipgrep  /usr/bin/zipinfo  /usr/bin/zsoelim
```

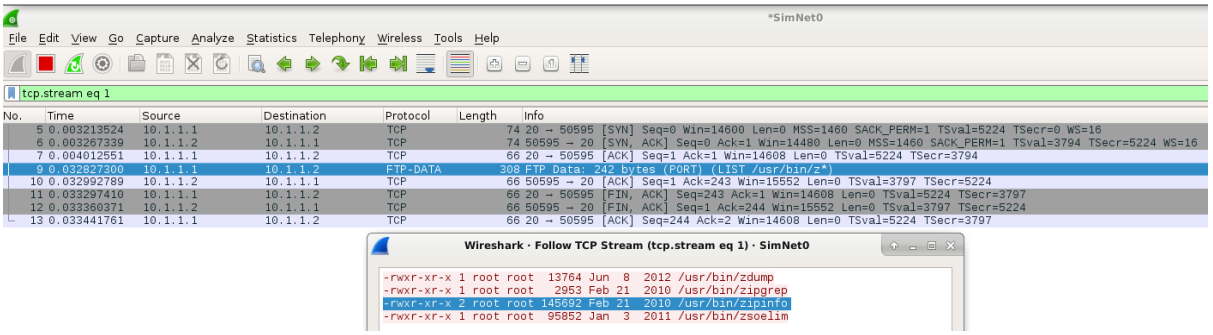
//from ftp sesion in virt2
prompt #to do interactive mood on
cd /usr/bin

mget z*

```
ftp> prompt
Interactive mode off.
ftp> cd /usr/bin
250 CWD command successful.
ftp> mget z*
local: zdump remote: zdump
200 PORT command successful.
150 Opening BINARY mode data connection for 'zdump' (13764 bytes).
226 Transfer complete.
13764 bytes received in 0.00 secs (45410.2 kB/s)
local: zipgrep remote: zipgrep
200 PORT command successful.
150 Opening BINARY mode data connection for 'zipgrep' (2953 bytes).
226 Transfer complete.
2953 bytes received in 0.01 secs (256.2 kB/s)
local: zipinfo remote: zipinfo
200 PORT command successful.
150 Opening BINARY mode data connection for 'zipinfo' (145692 bytes).
226 Transfer complete.
145692 bytes received in 0.01 secs (26554.2 kB/s)
local: zsoelim remote: zsoelim
200 PORT command successful.
150 Opening BINARY mode data connection for 'zsoelim' (95852 bytes).
226 Transfer complete.
95852 bytes received in 0.00 secs (21693.0 kB/s)
```

Using binary mode to transfer files, as we saw in exercise 3.

5. Use the follow TCP stream option of wireshark to comment the FTP dialogue previously generated. Figure out also how the data (files) are transferred and which ports are used.



6. Look at the files you have downloaded in virt2. Check the permissions. Are these permissions the same as in the server? When you finish, remove these files in the client and exit the FTP session.

/from virt1

ls -la /usr/bin/z*

```
root@virt1:~# ls -la /usr/bin/z*
-rwxr-xr-x 1 root root  13764 jun  8  2012 /usr/bin/zdump
-rwxr-xr-x 1 root root   2953 feb 21  2010 /usr/bin/zipgrep
-rwxr-xr-x 2 root root 145692 feb 21  2010 /usr/bin/zipinfo
-rwxr-xr-x 1 root root  95852 ene  3  2011 /usr/bin/zsoelim
```

//from virt2.1
ls -la ./z*

```
root@virt2:~# ls -la ./z*
-rw-r--r-- 1 root root    417 oct 11 00:50 ./z*
-rw-r--r-- 1 root root  13764 oct 11 00:56 ./zdump
-rw-r--r-- 1 root root   2953 oct 11 00:56 ./zipgrep
-rw-r--r-- 1 root root 145692 oct 11 00:56 ./zipinfo
-rw-r--r-- 1 root root  95852 oct 11 00:56 ./zsoelim
```

in server :
-rwxr-xr-x —>The user has read, write and execute permissions; the group and others can only read and execute.
in client:
-rw-r--r—> the owner permissions are rw- , indicating that the owner can read and write to the file but can't execute it as a program

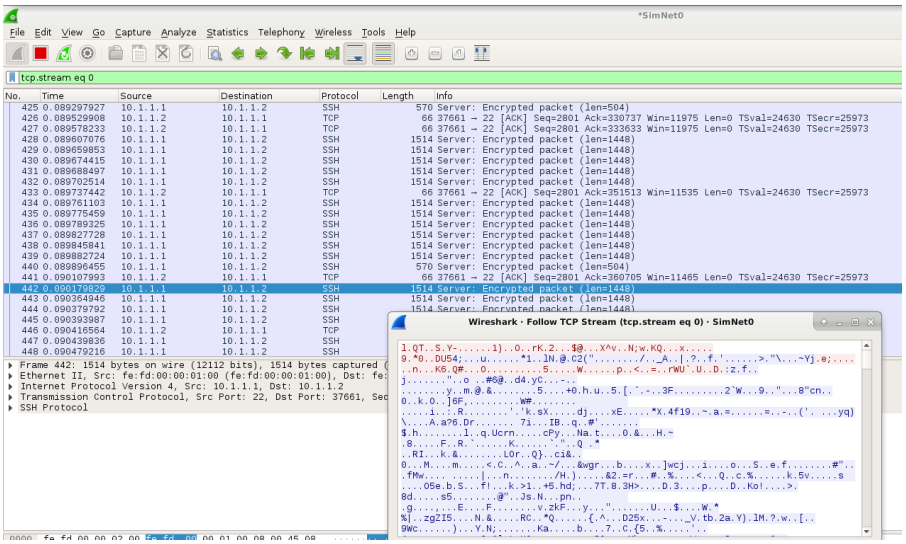
7. Start a new capture on tap0 with wireshark in the phyhost and establish an SFTP session from virt2 to virt1 using the console virt2.0. Use the SFTP session to get all the files in /usr/bin that start with “z”. Use the follow TCP stream option of wireshark and roughly comment the SFTP dialogue.
//from virt2
sftp 10.1.1.1

```
root@virt2:~# sftp 10.1.1.1
root@10.1.1.1's password:
Connected to 10.1.1.1.
sftp>
```

//from sftp session
get -r /usr/bin/z*

```
sftp> get -r /usr/bin/z*
Fetching /usr/bin/zdump to zdump
/usr/bin/zdump                                100%   13KB  13.4KB/s   00:00
Fetching /usr/bin/zipgrep to zipgrep
/usr/bin/zipgrep                              100% 2953     2.9KB/s   00:00
Fetching /usr/bin/zipinfo to zipinfo
/usr/bin/zipinfo                              100%  142KB 142.3KB/s   00:00
Fetching /usr/bin/zsoelim to zsoelim
/usr/bin/zsoelim                              100%   94KB  93.6KB/s   00:00
sftp>
```

//from wireshark, #we can observer the encrypted packets



//from virt2.1
ls -la ./z*

```
root@virt2:~# ls -la ./z*
-rwxr-xr-x 1 root root  13764 oct 11 01:29 ./zdump
-rwxr-xr-x 1 root root   2953 oct 11 01:29 ./zipgrep
-rwxr-xr-x 1 root root 145692 oct 11 01:29 ./zipinfo
-rwxr-xr-x 1 root root  95852 oct 11 01:29 ./zsoelim
```

In order to sftp, packets encrypted, client has more permission, same as server.

8. Start a new capture on tap0 with wireshark in the phyhost. Also in the phyhost, give the IP address 10.1.1.3/24 to the tap0 interface and create a file called “file.txt” with the content “hello world”. Using scp and the root user, transfer this file to the directory /root of virt1. Use the follow TCP stream option of wireshark and roughly comment the SCP dialogue.

//from terminal
sudo ifconfig SimNet0 10.1.1.3/24
echo “hello world2 > file.txt

```
telem@debian:~$ sudo ifconfig SimNet0 10.1.1.3/24
telem@debian:~$ echo "hello world" > file.txt
telem@debian:~$ ls
Desktop    Downloads  muchotexto  Pictures  shared     Videos
Documents  file.txt   Music       Public    Templates
```

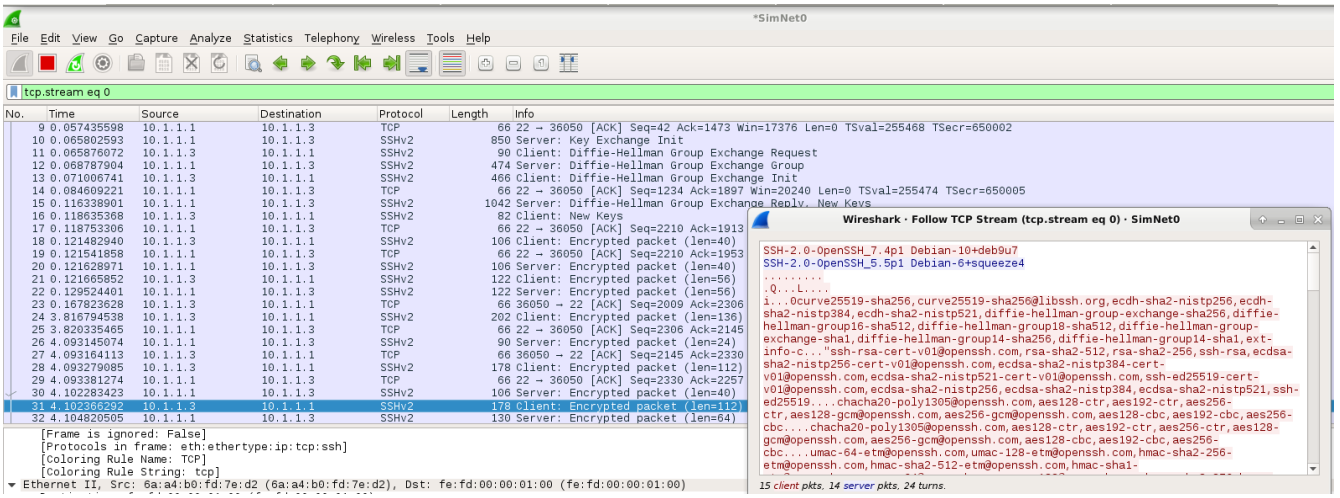
sudo scp file.txt root@10.1.1.1:/tmp

```
telem@debian:~/Desktop$ sudo scp file.txt root@10.1.1.1:/tmp
root@10.1.1.1's password:
file.txt                                      100%   44    58.0KB/s   00:00
```

//in virt1

pwd file.txt

```
root@virt1:~# pwd file.txt
/root
```



RSA encryption used, protocol Diffie-Hellman.

port ssh=22