

Instituto Tecnológico de Cancún

Ingeniería en Sistemas Computacionales

Fundamentos de Telecomunicaciones

Unidad 1: Sistemas de comunicación

“Proyecto: Sistema de comunicación”

Docente: Ing. Ismael Jiménez Sánchez

Alumno: Uc Uc César Enrique

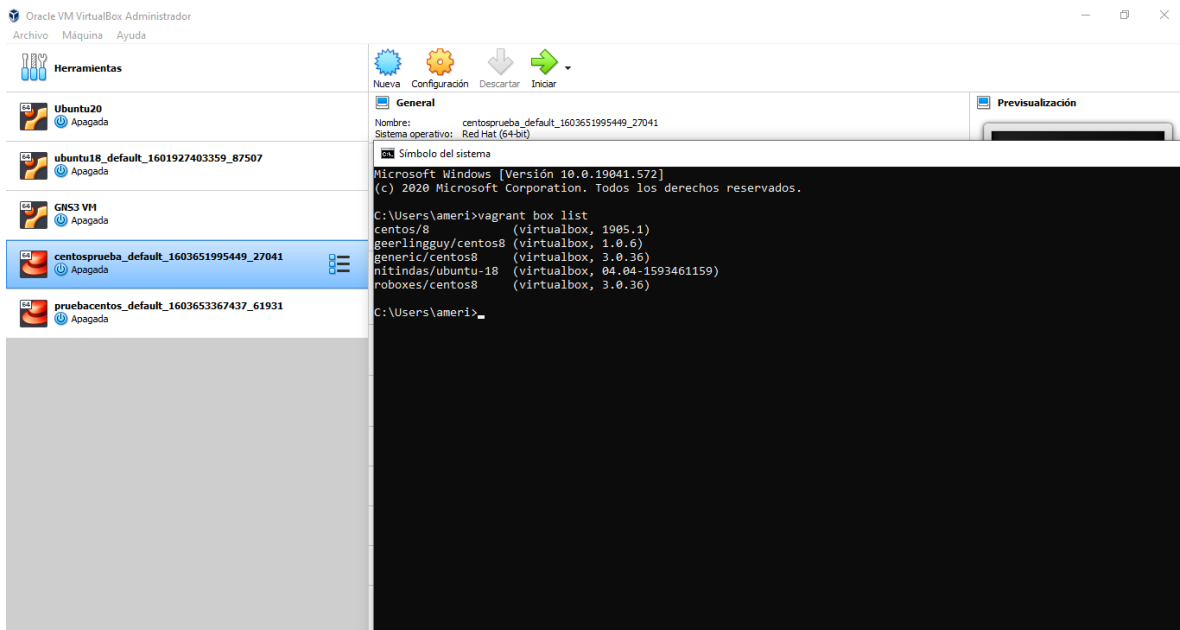
Requisitos:

- Scripts en python
- GNS3
- VBox
- Vagrant

Fase 1

Instalar 2 centos8 en virtualbox usando vagrant.

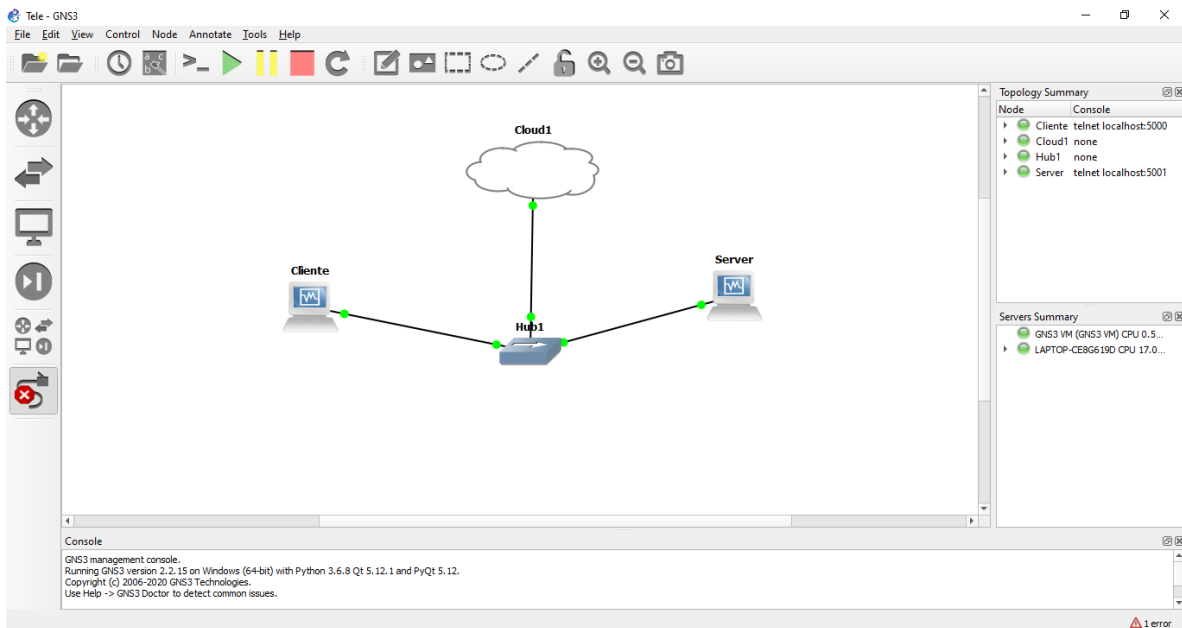
- En este proyecto utilizamos el centos8
Link: <https://app.vagrantup.com/centos/boxes/8>



Fase 2

Conectar en GNS3, las dos VMs de CentOS con un switch ethernet.

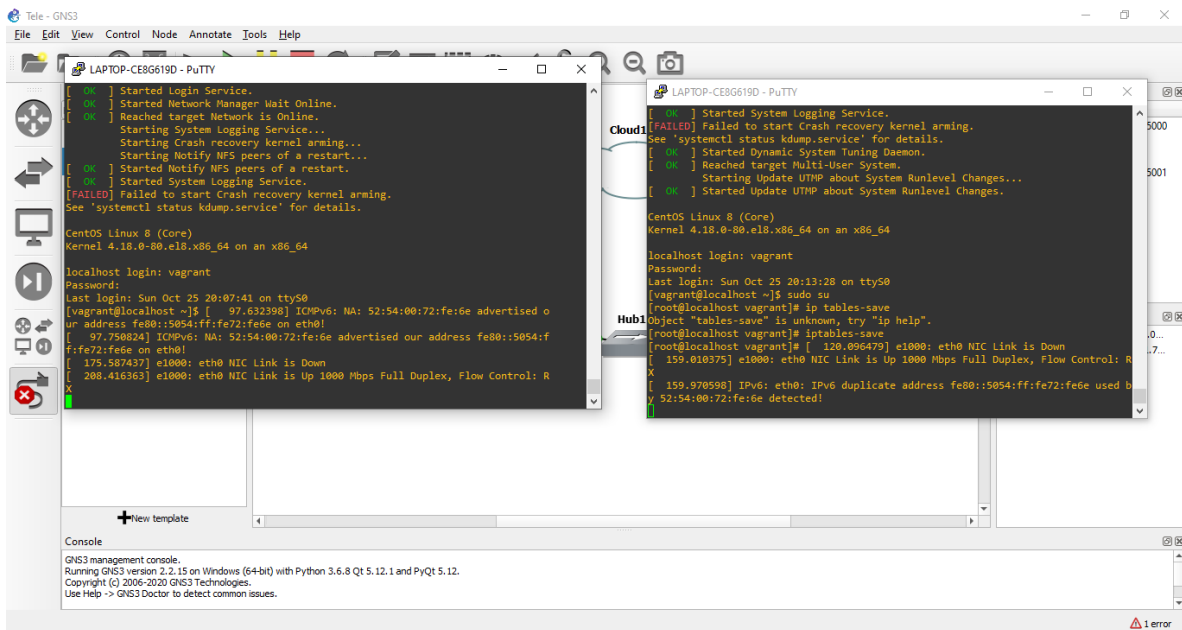
- Subimos las máquinas virtuales a la GNS3 y las conectamos de tal manera que queden así:



Fase3

Usar los scripts de python para conectar las dos VMs usando sockets.

- Desde putty conectamos los scripts de Python para conectar las maquinas usando sockets
- Socket para server:**
<https://github.com/tuxtter/pythonScripts/blob/master/tcpserver.py>
- Socket para cliente:**
<https://github.com/tuxtter/pythonScripts/blob/master/tcpclient.py>



```
LAPTOP-CE8G619D - PuTTY
[ OK ] Started Login Service.
[ OK ] Started Network Manager Wait Online.
[ OK ] Reached target Network is Online.
Starting System Logging Service...
Starting Crash recovery kernel arming...
Starting Notify NFS peers of a restart...
[ OK ] Started System Logging Service.
[ FAILED ] Failed to start Crash recovery kernel arming.
See 'systemctl status kdump.service' for details.

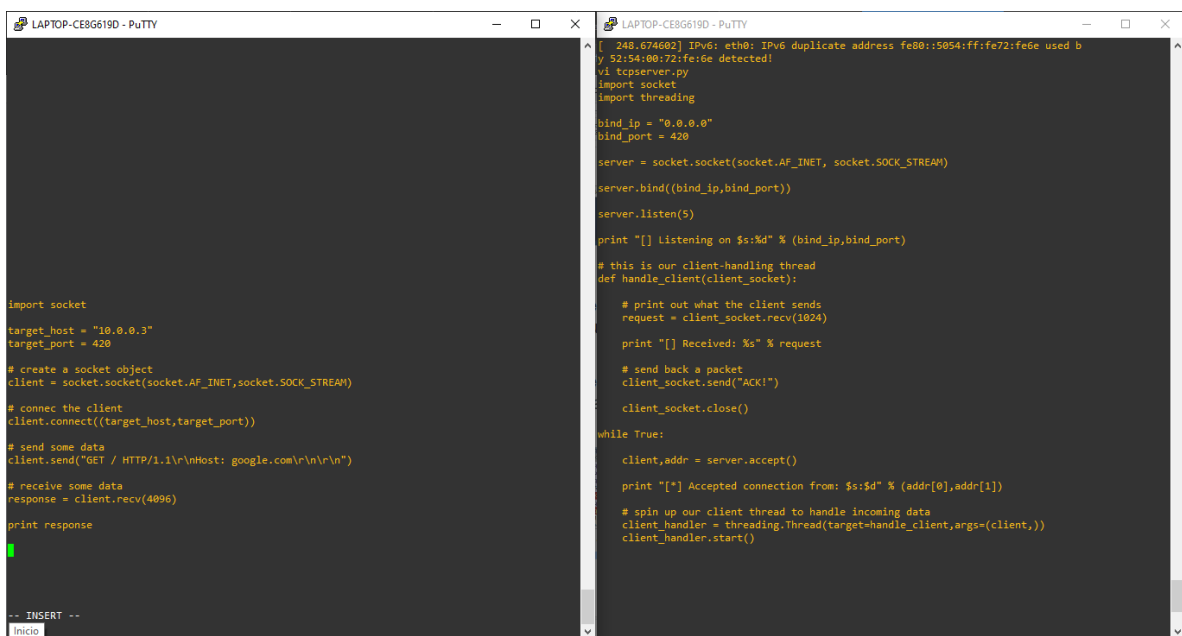
CentOS Linux 8 (Core)
Kernel 4.18.0-80.el8.x86_64 on an x86_64

localhost login: vagrant
Password:
Last login: Sun Oct 25 20:07:41 on tty50
[vagrant@localhost ~]$ [ 97.632398] ICMPv6: NA: 52:54:00:72:fe:6e advertised o
ur address fe80::5054:ff:fe72:fe6e on eth0!
[ 97.750824] ICMPv6: NA: 52:54:00:72:fe:6e advertised our address fe80::5054:f
f:fe72:fe6e on eth0!
[ 175.587437] e1000: eth0 NIC Link is Down
[ 208.416363] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: R
X

LAPTOP-CE8G619D - PuTTY
[ OK ] Started System Logging Service.
[ FAILED ] Failed to start Crash recovery kernel arming.
See 'systemctl status kdump.service' for details.
[ OK ] Started Dynamic System Tuning Daemon.
[ OK ] Reached target Multi-User System.
Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

CentOS Linux 8 (Core)
Kernel 4.18.0-80.el8.x86_64 on an x86_64

localhost login: vagrant
Password:
Last login: Sun Oct 25 20:13:28 on tty50
[vagrant@localhost ~]$ sudo su
[root@localhost vagrant]# ip netns exec
Object "tables-save" is unknown, try "ip help".
[root@localhost vagrant]# ip netns exec
[ 159.010375] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: R
X
[ 159.970598] IPv6: eth0: IPv6 duplicate address fe80::5054:ff:fe72:fe6e used b
y 52:54:00:72:fe:6e detected!
```



```
LAPTOP-CE8G619D - PuTTY
import socket

target_host = "10.0.0.3"
target_port = 420

# create a socket object
client = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

# connect the client
client.connect((target_host,target_port))

# send some data
client.send("GET / HTTP/1.1\r\nHost: google.com\r\n\r\n")

# receive some data
response = client.recv(4096)

print response

-- INSERT --
Inicio

LAPTOP-CE8G619D - PuTTY
[ 248.674602] IPv6: eth0: IPv6 duplicate address fe80::5054:ff:fe72:fe6e used b
y 52:54:00:72:fe:6e detected!
vi tcpserver.py
import socket
import threading

bind_ip = "0.0.0.0"
bind_port = 420

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server.bind((bind_ip,bind_port))

server.listen(5)

print "[*] Listening on %s:%d" % (bind_ip,bind_port)

# this is our client-handling thread
def handle_client(client_socket):

    # print out what the client sends
    request = client_socket.recv(1024)

    print "[*] Received: %s" % request

    # send back a packet
    client_socket.send("ACK!")

    client_socket.close()

while True:

    client,addr = server.accept()

    print "[*] Accepted connection from: %s:%d" % (addr[0],addr[1])

    # spin up our client thread to handle incoming data
    client_handler = threading.Thread(target=handle_client,args=(client,))
    client_handler.start()
```

Fase 4

Capturar el tráfico de la comunicación entre las dos VMs al momento de utilizar los scripts.

The image shows a GNS3 environment with two VMs connected. A Wireshark packet capture window is open, showing traffic on the interface 'id 0'. The capture shows several DHCP Discover and Router Solicitation packets from the VMs to the broadcast address ff:ff:ff:ff:ff:ff.

Wireshark Packet Capture Data:

No.	Time	Source	Destination	Protocol	Length	Info
10	313.678051	0.0.0.0	255.255.255.255	DHCP	324	DHCP Discover - Transaction ID 0x92007124
11	314.859678	0.0.0.0	255.255.255.255	DHCP	324	DHCP Discover - Transaction ID 0x92007124
12	317.135079	fe80::5054:ff:fe72::...	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
13	318.772804	0.0.0.0	255.255.255.255	DHCP	324	DHCP Discover - Transaction ID 0x92007124
14	326.351660	fe80::5054:ff:fe72::...	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
15	326.756208	0.0.0.0	255.255.255.255	DHCP	324	DHCP Discover - Transaction ID 0x92007124
16	341.909070	0.0.0.0	255.255.255.255	DHCP	324	DHCP Discover - Transaction ID 0x92007124
17	343.757880	fe80::5054:ff:fe72::...	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
18	356.024411	0.0.0.0	255.255.255.255	DHCP	324	DHCP Request - Transaction ID 0x92007124
19	356.028336	::	ff02::1	ICMPv6	90	Multicast Listener Report Message v2
20	356.372285	::	ff02::1:ff72:fe6e	ICMPv6	86	Neighbor Solicitation for fe80::5054:ff:fe72:fe6e

Frame 1 Details:

- Frame 1: 324 bytes on wire (2592 bits), 324 bytes captured (2592 bits) on interface -, id 0
- Ethernet II, Src: RealtekU_72:fe:6e (52:54:00:72:fe:6e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
- User Datagram Protocol, Src Port: 68, Dst Port: 67
- Dynamic Host Configuration Protocol (Discover)

Hex Dump:

```
0000 ff ff ff ff ff ff 52 54 00 72 fe 6e 08 00 45 c0 .....RT..n..E:
0010 01 36 00 00 00 00 40 11 78 f8 00 00 00 ff ff ..6...@..x.....
0020 ff ff 00 44 00 43 01 22 d7 84 01 01 06 00 77 13 ...D.C...R.....
0030 5f 5d 00 20 00 00 00 00 00 00 00 00 00 00 00 00 ...].....
0040 00 00 00 00 00 00 52 54 00 72 fe 6e 08 00 00 00 .....RT..n..E:
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Toplogy Summary:

- Node Console
- Client telnet localhost:5000
- Cloud1 none
- Hub1 none
- Server telnet localhost:5001

Servers Summary:

- GNS3 VM (GNS3 VM) CPU 1.0...
- LAPTOP-CE806190 CPU 100.0...

Console:

```
Running GNS3 version 2.2.15 on Windows
Copyright (c) 2006-2020 GNS3 Technologies
Use Help -> GNS3 Doctor to detect and fix issues
```

Fase 5

Hacer reporte de conclusiones.

En conclusión podemos decir que al conectar las maquinas virtual mediante el GNS3 y unir las mediante putty y socket determinamos cual es el cliente y el servidor. Se instalaron las 2 máquinas de Python y configuramos la ip de cada servidor.

En la última fase pudimos observar la conexión telnet y Wireshark captura la interacción que hay del cliente con el servidor, se observa el manejo y control de datos y se asegura que los datos lleguen a su destino sin la perdida de algún paquete.