



# PROYECTO

Telecom

## Descripción breve

Una practica donde veremos la comunicación entre 2 máquinas virtuales

Héctor Emilio Cantellano Gómez  
Instituto tecnológico de Cancún

# Introducción

En este proyecto podremos ver como es el proceso para que 2 maquinas virtuales se comuniquen entre si, se va utilizar varios programas para poder lograr estos y estos son: VirtualBox, GENS3, PuTTY y WireShark, además de que se podrá observar como son necesarios todos estos programas además de su compatibilidad.

Recursos:

-VirtualBox

-Vagrant

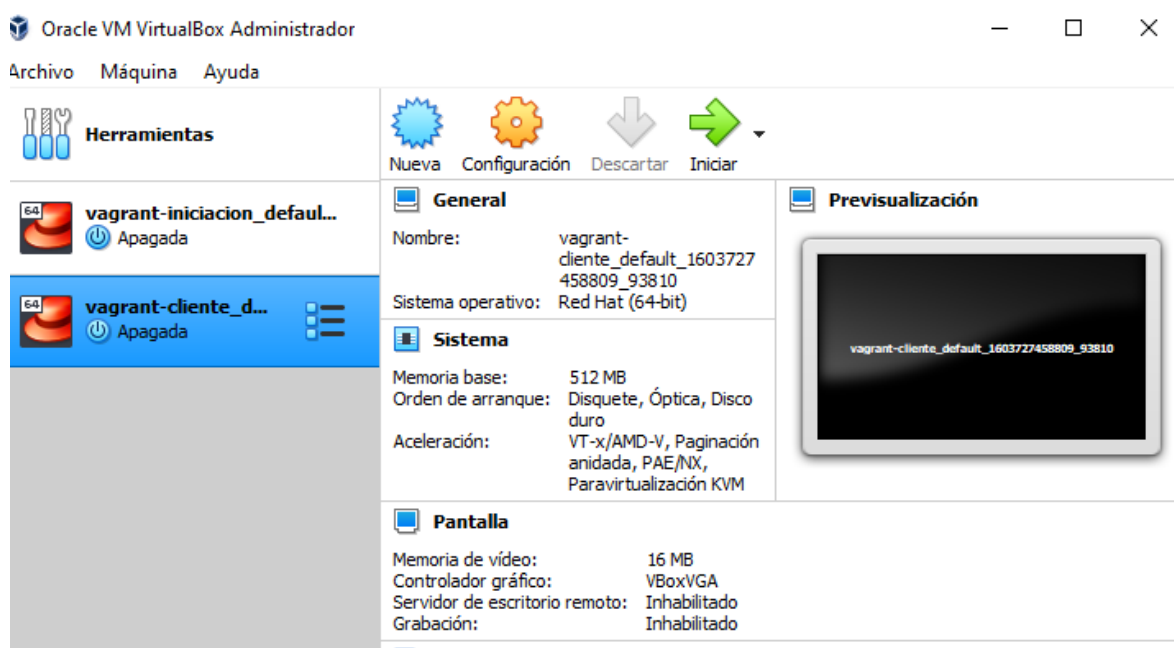
-GENS3

-PuTTY

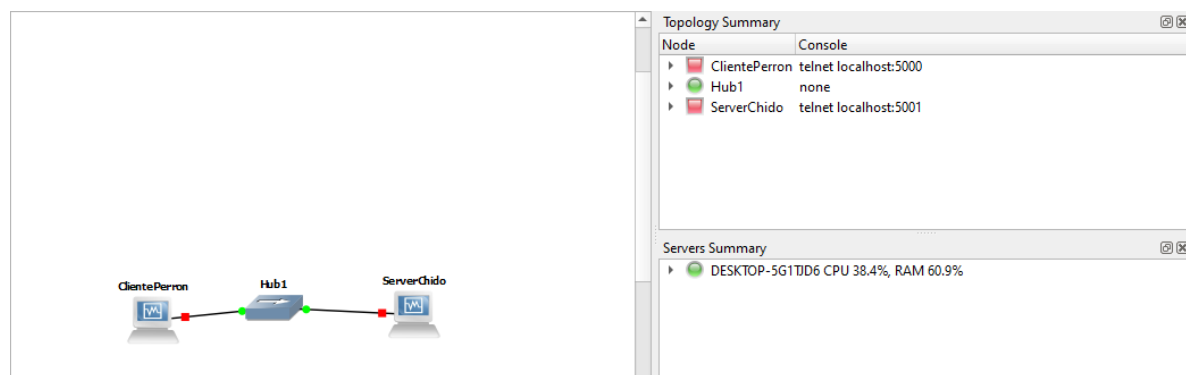
-WireShark

# Proceso

**Fase 1:** Primero se instaló el SO de CentOS la versión 8 en 2 máquinas virtuales utilizando Vagrant y Powershell de Windows y las maquinas fueron de VirtualBox.



**Fase 2:** Des pues mediante GENS3 se conectaron las máquinas virtuales con un switch hub, pero antes se les asigno una ip a cada máquina y desde la VirtualBox configuramos que tendrían una conexión con cable



**Fase 3:** Ya al estar conectadas y puesto una ip a cada una de las maquinas, se le instalo a cada una Python para poder usar unos scripts y poder comunicarse entre sí, los comandos son algo complicados, pero con la practica se dominan enseguida.

```
ClientPerron - PuTTY
target_host = "192.168.60.102"
target_port = 3080

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

# connec 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

^

ServerChido - PuTTY
import thread

host = '192.168.60.102'
port = 3080

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print('Socket Ready...')

s.bind((host, port))
print('Bind Ready...')

print('Listening...')
s.listen(1)

def handle_client(client_socket):
    while True:
        data = client_socket.recv(1024)
        if not data: break
        print('Client says: ' + data)
        print('Sending: ' + data)
        client_socket.send(data)
        client_socket.close()

"tcpserverchido.py" [New] 30L, 635C written
[root@localhost vagrant]#
```

Nota: Para mayor comodidad y facilidad utilizamos PuTTY además de que con todo esto se conceto por vía Telnet.

ClientePerron - PuTTY

```
# 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

~
~
~
~
~

"tcpclientperron.py" 19L, 355C written
[root@localhost vagrant]# python2 tcpclientperron.py
bash: python2: command not found
[root@localhost vagrant]# python tcpclientperron.py
bash: python: command not found
[root@localhost vagrant]# python2 tcpclientperron.py
GET / HTTP/1.1
Host: google.com

[root@localhost vagrant]#
```

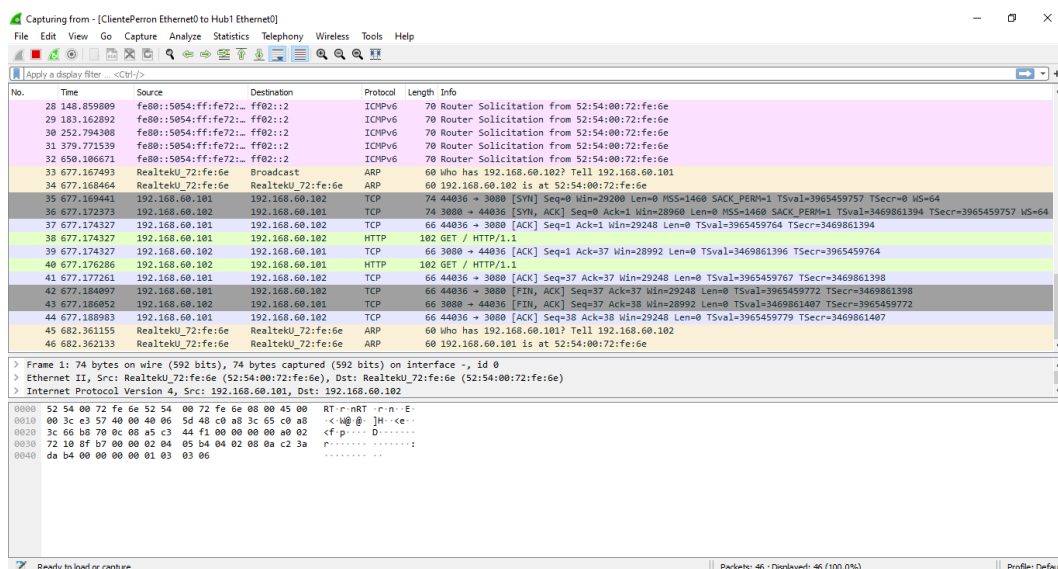
ServerChido - PuTTY

```
data = client_socket.recv(1024)
if not data: break
print('Client says: ' + data)
print('Sending: ' + data)
client_socket.send(data)
client_socket.close()

"tcpserverchido.py" [New] 30L, 635C written
[root@localhost vagrant]# python2 tcpserverchido.py
python2: can't open file 'tcpserverchido.py': [Errno 2] No such file or director
y
[root@localhost vagrant]# python2 tcpserverchido.py
Socket Ready...
Bind Ready...
Listening...
Conexion from: ('192.168.60.101', 47214)
Client says: GET / HTTP/1.1
Host: google.com

Sending: GET / HTTP/1.1
Host: google.com
```

**Fase 4:** Por último, para poder ver el proceso que tienen estas 2 máquinas virtuales se usa WireShark para ver el tráfico de paqueterías que se envían



# Conclusiones

Al concluir este proyecto se puede ver lo laborioso, pero no complicado proceso que el comunicarse entre dispositivos además de que se puede identificar como es el protocolo que se tiene que seguir, también te hace cuestionar de las posibilidades que puede tener todo esto, es impresionante ver como se usan variedad de programas para poder un mejor control y posibilidades para hacer las cosas, además que al principio se es difícil notar para que sirve cada cosa porque en si unas complementan a otras, también se pudo apreciar como por tener algo mal en una de estos programas puede afectar a todo lo demás o simplemente no funcione.