



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

PROFESOR: RANGEL GONZALEZ JOSUE

APLICACIONES PARA COMUNICACIONES DE RED

3CV7

PRÁCTICA TFTP Y BROADCAST

ALUMNOS:

JOSUE ARTURO ALFARO BRACHO
VLADIMIR AZPEITIA HERNANDEZ
ALDO SUAREZ CRUZ

Índice

Índice	2
Introducción	3
Conceptos	3
TFTP	5
Objetivo de la práctica	5
Desarrollo	5
Activar telnet en router cisco	5
Configuración del servidor TFTP en linux	7
Enviar y recibir running-config del router R1, a través de TFTP	9
Elaborando el script en Python	11
Explicación del algoritmo	11
Código de python	11
Capturas de pantalla	14
Broadcast	15
Objetivo de la práctica	15
Desarrollo	16
Capturas de pantalla	21

1. Introducción

El presente trabajo TFTP y Broadcast es parte de los temas más complejos que vamos abordando durante este curso. En esta práctica abordaremos como temas principales TFTP para envío de archivos y el envío de estos archivos será en broadcast.

Es importante saber qué broadcast depende de si es para informática o para telecomunicaciones se conceptúa un poco diferente, la definición en el caso de nosotros que se va a ocupar es de la informática, broadcast es la transferencia de información desde un nodo emisor a una multitud de nodos receptores.

TFTP es un protocolo que se utiliza para el envío de los archivos es importante mencionar que se necesita un servidor principal desde el cual se va a enviar la información y un cliente que es el que va a recibir.

Para esto utilizaremos un programa para simular varias computadoras en una red con routers y switches simulados el cual se llama GNS3 el cual tiene la ventaja también de permitirnos mediante una configuración especial conectar nuestro propio equipo a esta simulación.

2. Conceptos

TFTP

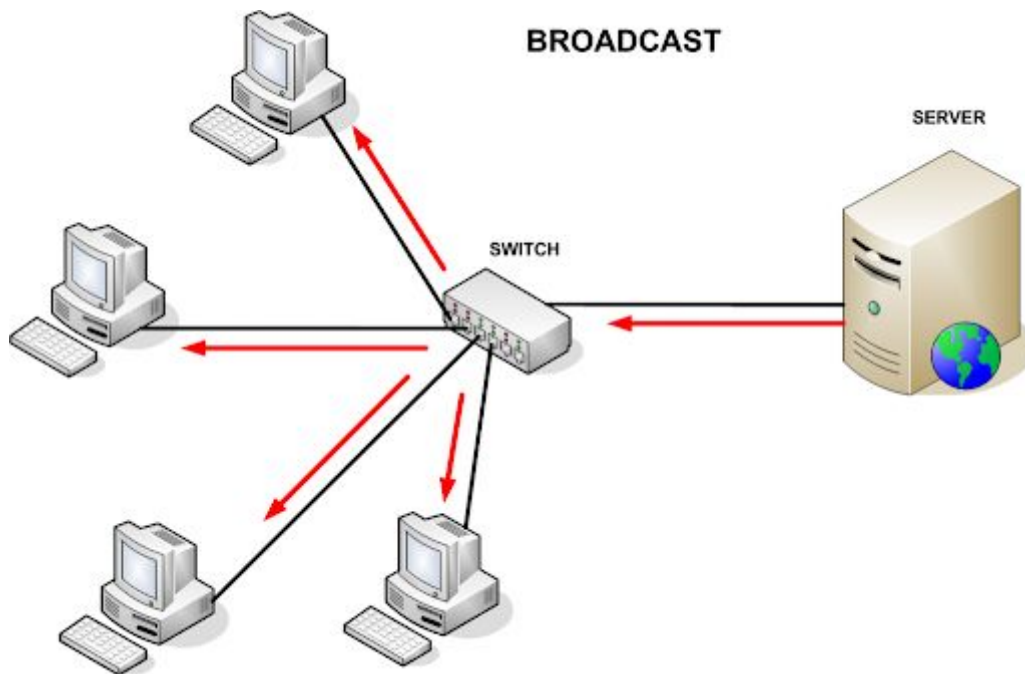
El protocolo Trivial File Transfer Protocol, en su forma abreviada TFTP, es un protocolo cliente-servidor muy simple que regula la transferencia de archivos en redes informáticas. Se definió originalmente en junio de 1981 en el RFC 783, si bien en la actualidad está vigente el estándar RFC 1350, publicado en 1992. Por defecto, el protocolo TFTP se basa en el protocolo mínimo de nivel de transporte UDP (User Datagram Protocol), que ofrece la posibilidad de transmitir datos sin necesidad de una conexión fija entre los miembros de la comunicación. No obstante, también es posible implementar el protocolo TFPT basándose en otros protocolos diferentes.

Cada archivo transferido vía TFTP constituye un intercambio independiente de paquetes, y existe una relación cliente-servidor informal entre la máquina que inicia la comunicación y la que responde.

- La máquina A, que inicia la comunicación, envía un paquete RRQ (read request/petición de lectura) o WRQ (write request/petición de escritura) a la máquina B, conteniendo el nombre del archivo y el modo de transferencia.
- B responde con un paquete ACK (acknowledgement/confirmación), que también sirve para informar a "A" del puerto de la máquina B al que tendrá que enviar los paquetes restantes.
- La máquina origen envía paquetes de datos numerados a la máquina destino, todos excepto el último conteniendo 512 bytes de datos. La máquina destino responde con paquetes ACK numerados para todos los paquetes de datos.
- El paquete de datos final debe contener menos de 512 bytes de datos para indicar que es el último. Si el tamaño del archivo transferido es un múltiplo exacto de 512 bytes, el origen envía un paquete final que contiene 0 bytes de datos.

Broadcast

Como ya se mencionó antes, la difusión amplia, difusión ancha o broadcast, es una forma de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.



Switch

La función básica de un switch es la de unir o conectar dispositivos en red. Es importante tener claro que un switch no proporciona por sí solo conectividad con otras redes, y obviamente, tampoco proporciona conectividad con Internet. Para ello es necesario un router.

Running-config

En los Routers Cisco la configuración que estos tienen por defecto se guarda en el archivo running-config en cuanto se inicia el router cisco voltea a ver dicho archivo para cargar la configuración de los puestos y protocolos que tenga guardados en este archivo.

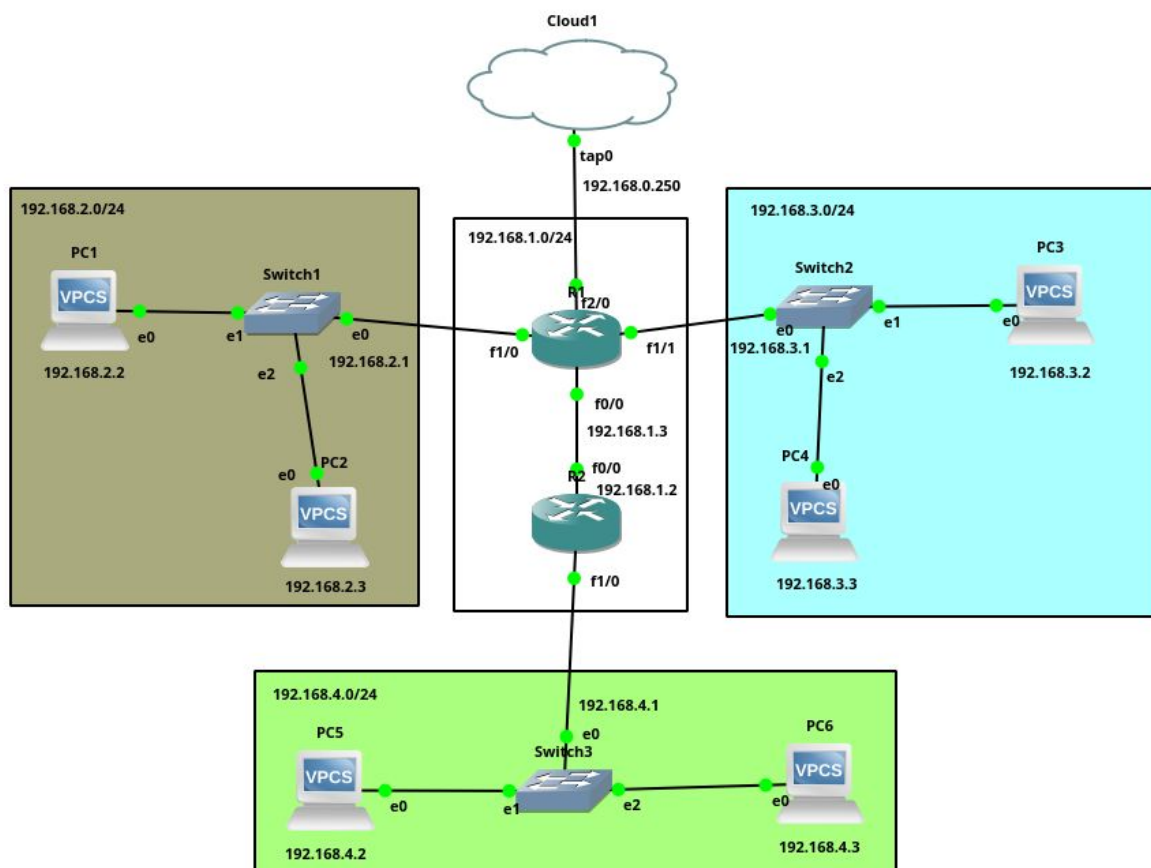
3. TFTP

Objetivo de la práctica

Obtener archivo running-config, de un router en gns3, mediante TFTP desde una máquina linux para modificarlo y enviarlo al router por TFTP

Desarrollo

Tenemos la siguiente topología en GNS3



Activar telnet en router cisco

Para realizar el script, primero debemos activar telnet en los dos routers(R1 y R2). Para ello abrimos la consola del router desde GNS3 y ejecutamos los siguientes comandos:

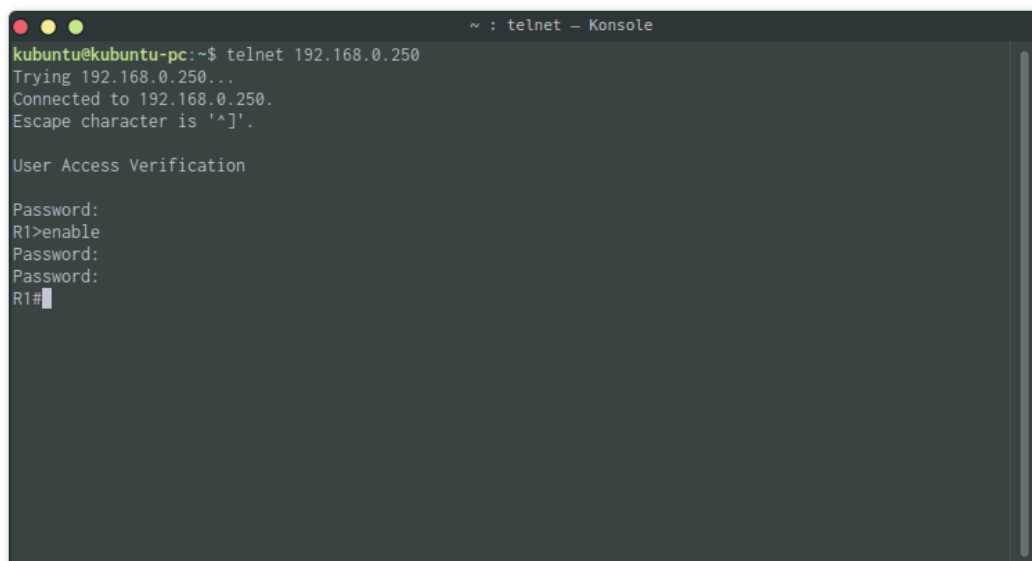
```
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#line vty 0
R1(config-line)#password Pass123
R1(config-line)#login
R1(config-line)#exit
R1(config)#
```

- ❑ El comando "**line vty**" habilita el telnet, y el parámetro "**0**" solo habilita una sesión al router.
- ❑ El comando "**password**" establece a "Pass123" como contraseña para telnet.
- ❑ El comando "**login**" autentica y le pide la contraseña de telnet

Ahora debemos establecer la contraseña de **enable**, del router, para poder controlarlo de forma remota. Ejecutamos el siguiente comando:

```
R1(config)#enable password Password
```

Para comprobar el funcionamiento, abrimos una terminal en nuestra máquina Linux. Y ejecutamos el comando **telnet 192.168.0.250**, para acceder al router R1.



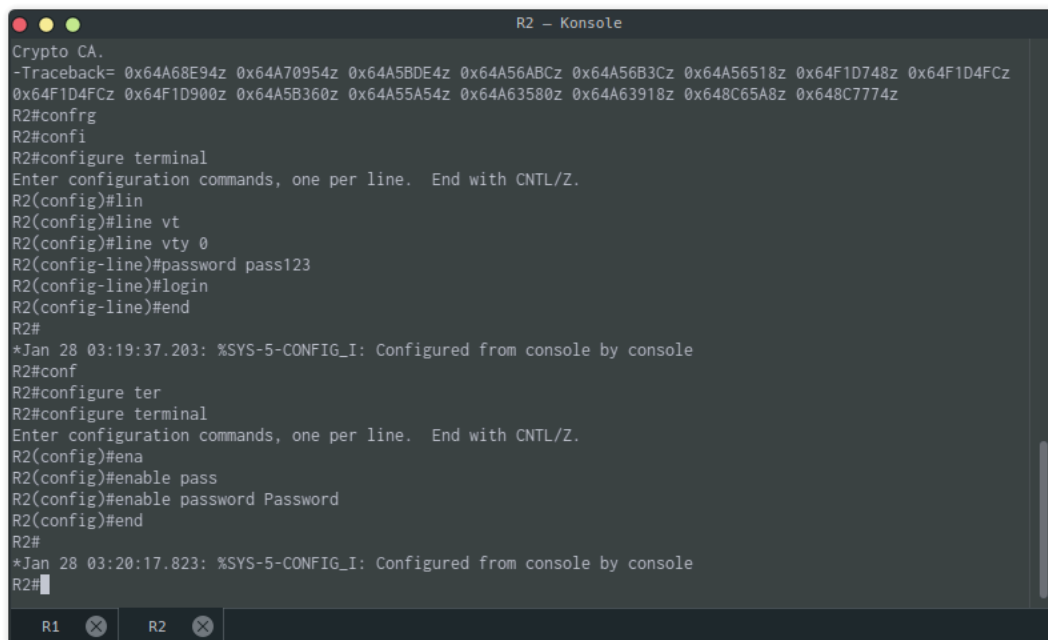
```
~ : telnet - Konsole
kubuntu@kubuntu-pc:~$ telnet 192.168.0.250
Trying 192.168.0.250...
Connected to 192.168.0.250.
Escape character is '^J'.

User Access Verification

Password:
R1>enable
Password:
Password:
R1#
```

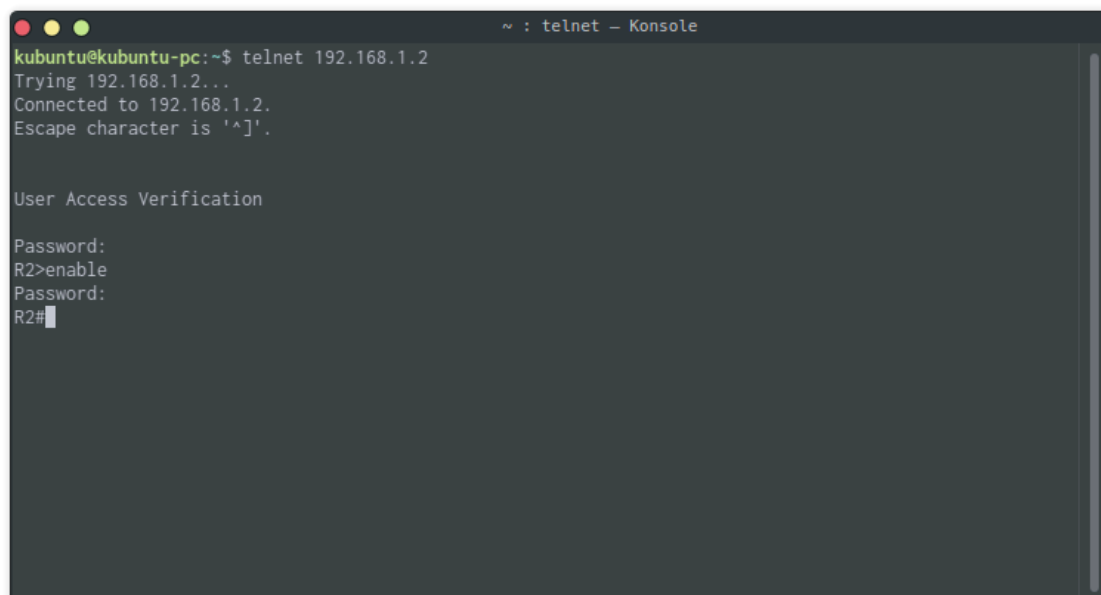
Ahora podemos controlar el router R1 de manera remota, con telnet.

Hacemos lo mismo para el router R2:



```
R2 - Konsole
Crypto CA.
-Traceback= 0x64A68E94z 0x64A70954z 0x64A5BDE4z 0x64A56ABCz 0x64A56B3Cz 0x64A56518z 0x64F1D748z 0x64F1D4FCz
0x64F1D4FCz 0x64F1D900z 0x64A5B360z 0x64A55A54z 0x64A63580z 0x64A63918z 0x648C65A8z 0x648C7774z
R2#conf
R2#confi
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#lin
R2(config)#line vt
R2(config)#line vty 0
R2(config-line)#password pass123
R2(config-line)#login
R2(config-line)#end
R2#
*Jan 28 03:19:37.203: %SYS-5-CONFIG_I: Configured from console by console
R2#conf
R2#configure ter
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#ena
R2(config)#enable pass
R2(config)#enable password Password
R2(config)#end
R2#
*Jan 28 03:20:17.823: %SYS-5-CONFIG_I: Configured from console by console
R2#
```

Para comprobar nos conectándonos en linux, vía telnet, al router R2:



```
~ : telnet - Konsole
kubuntu@kubuntu-pc:~$ telnet 192.168.1.2
Trying 192.168.1.2...
Connected to 192.168.1.2.
Escape character is '^['.

User Access Verification

Password:
R2>enable
Password:
R2#
```

Configuración del servidor TFTP en linux

Primero instalamos el paquete tftp-hpa :

```
$ sudo apt install tftpd-hpa
```

Abrimos el archivo de configuración `/etc/default/tftp-hpa` del programa TFTP y cambiamos la ruta de la carpeta TFTP, en mi caso sera `/home/kubuntu/tftp`

```
- : sudo nvim -- Console
```

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/home/kubuntu/tftp"
TFTP_ADDRESS="::69"
TFTP_OPTIONS="--secure --create"

...


/etc/default/tftpd-hpa
```

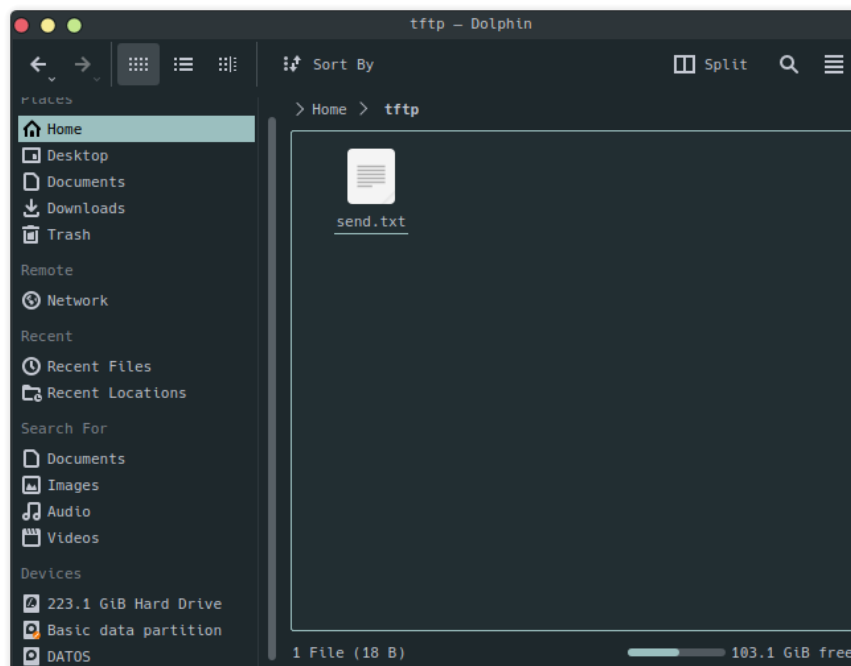
4,15

Todo

Comprobamos que funciona correctamente enviando un archivo desde un cliente tftp:

```
linuxlite ~ > Desktop 1 tftp 192.168.0.133
tftp> put send.txt
tftp>
```

Recibimos correctamente el archivo



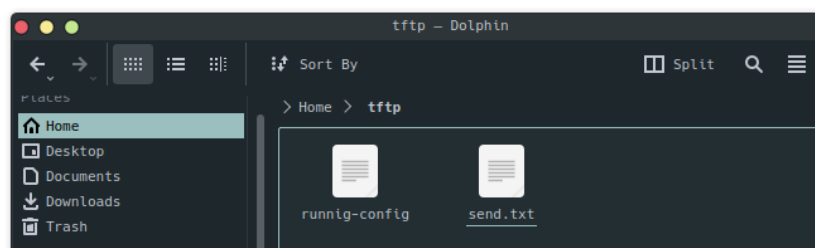
Enviar y recibir running-config del router R1, a través de TFTP

Ahora probamos enviando el archivo de configuración “**running-config**” del router R1 al servidor TFTP en linux.

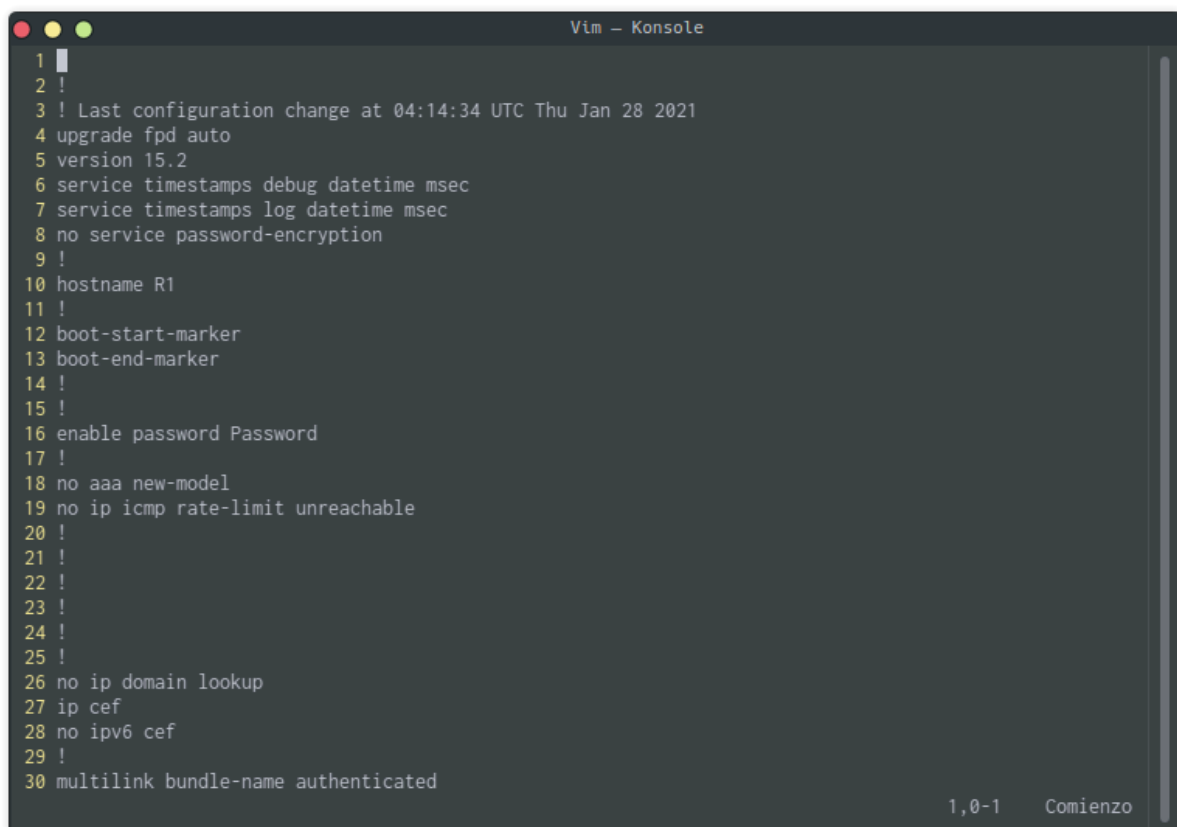
Ejecutamos el siguiente comando en el router R1:

R1#copy running-config tftp:

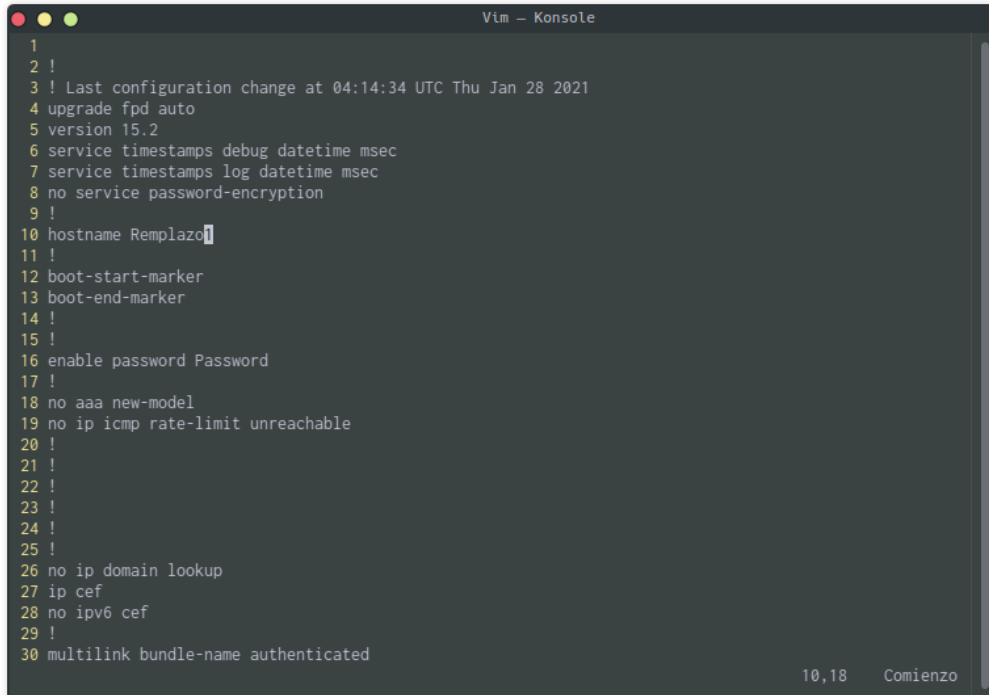
Podemos observar que el archivo running-config llegó con éxito al servidor TFTP en linux



Abrimos el archivo “running-config”, con nuestro editor de texto:

A screenshot of a Vim editor window titled 'Vim - Konsole'. The editor displays the contents of the 'running-config' file, which is a Cisco IOS configuration. The configuration includes commands for upgrading the firmware, setting service timestamps, enabling password encryption, setting the hostname to 'R1', and configuring various network settings. The text is displayed in a monospaced font with line numbers on the left. The status bar at the bottom right shows '1,0-1' and 'Comienzo'.

Modificamos la línea de 10, y sustituimos el nombre del hostname **R1** a **Remplazo1**



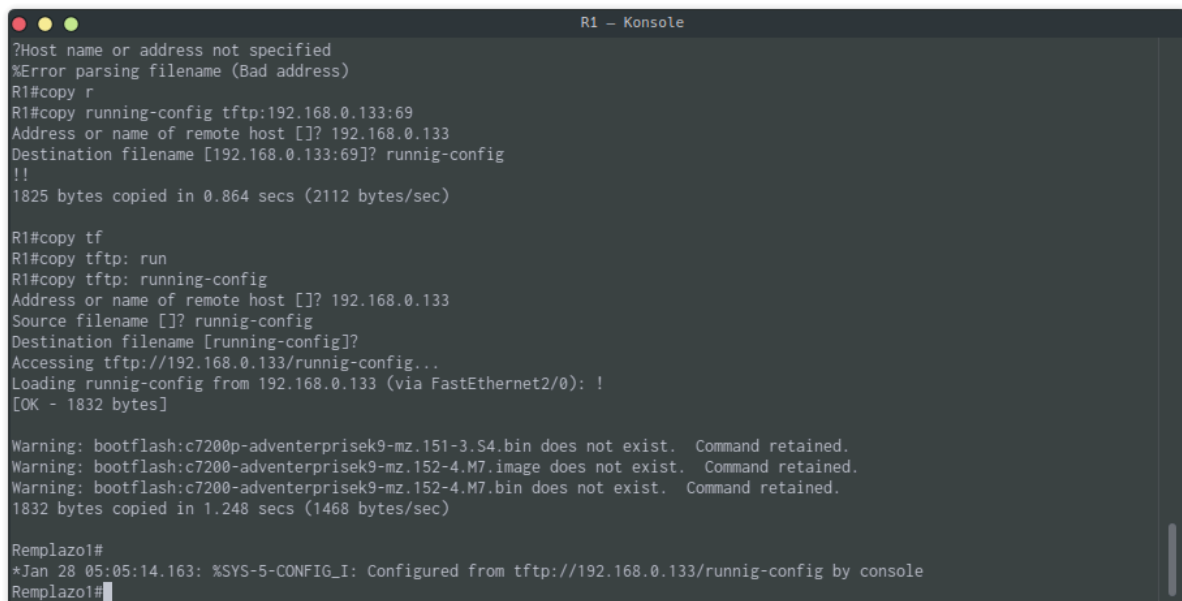
```
1
2 !
3 ! Last configuration change at 04:14:34 UTC Thu Jan 28 2021
4 upgrade fpd auto
5 version 15.2
6 service timestamps debug datetime msec
7 service timestamps log datetime msec
8 no service password-encryption
9 !
10 hostname Remplazo1
11 !
12 boot-start-marker
13 boot-end-marker
14 !
15 !
16 enable password Password
17 !
18 no aaa new-model
19 no ip icmp rate-limit unreachable
20 !
21 !
22 !
23 !
24 !
25 !
26 no ip domain lookup
27 ip cef
28 no ipv6 cef
29 !
30 multilink bundle-name authenticated
```

10,18 Comienzo

Ahora enviamos el archivo de configuración del servidor TFTP en linux al router, con el comando:

R1#copy tftp: running-config

Como podemos observar inmediatamente cambió el nombre del hostname a **Reemplazo1**



```
?Host name or address not specified
%Error parsing filename (Bad address)
R1#copy r
R1#copy running-config tftp:192.168.0.133:69
Address or name of remote host []? 192.168.0.133
Destination filename [192.168.0.133:69]? runnig-config
!!
1825 bytes copied in 0.864 secs (2112 bytes/sec)

R1#copy tf
R1#copy tftp: run
R1#copy tftp: running-config
Address or name of remote host []? 192.168.0.133
Source filename []? runnig-config
Destination filename [running-config]?
Accessing tftp://192.168.0.133/runnig-config...
Loading runnig-config from 192.168.0.133 (via FastEthernet2/0): !
[OK - 1832 bytes]

Warning: bootflash:c7200p-adventerprisek9-mz.151-3.S4.bin does not exist. Command retained.
Warning: bootflash:c7200-adventerprisek9-mz.152-4.M7.image does not exist. Command retained.
Warning: bootflash:c7200-adventerprisek9-mz.152-4.M7.bin does not exist. Command retained.
1832 bytes copied in 1.248 secs (1468 bytes/sec)

Remplazo1#
*Jan 28 05:05:14.163: %SYS-5-CONFIG_I: Configured from tftp://192.168.0.133/runnig-config by console
Remplazo1#
```

Elaborando el script en Python

Ahora vamos a hacer un script en Python para automatizar este proceso:

Explicación del algoritmo

Para esta parte como podemos ver el script en Python 3, para utilizar Telnet utilizaremos la librería **telnetlib**. Primero, lo que hacemos es mandar directamente las instrucciones a los routers desde el script para que vaya realizando las cosas, gracias al telnet que configuramos previamente, cabe mencionar que para las contraseñas hay que codificarlas en ascii, el comando que utilizamos para traernos el archivo de configuración es el **copy running-config tftp**: el cual nos permite traer vía el tftp el archivo especificado. Cabe mencionar que se le metió un sleep para poder esperar a que el archivo se pase de forma completa ya que en ocasiones si no se mete el sleep puede que el archivo no se termine de pasar y a la hora de editarlo marque el error el programa de que no está completo o está siendo utilizado, una vez que llego el archivo aplicamos los cambios que se necesiten, para esto abrimos el archivo que tenemos en el servidor tftp y aplicamos el reemplazo y lo cerramos, una vez que los hemos hecho, procedemos a enviar el archivo con los cambios con el comando **copy tftp: running-config**.

Código de python

```
import telnetlib
import os
import time

TFTP_DIR = '/home/kubuntu/tftp'

HOST_R1 = "192.168.0.250" # ip addr router r1
HOST_R2 = "192.168.1.2" # ip addr router r2
IP_TFTP = "192.168.0.133" # ip tftp server

password = 'pass123' # password telnet
password_enable = 'Password' # password for enable mode

'''
    SEND AND RECIVE RUNNIG CONFIG FOR ROUTER R1
'''

tn_r1 = telnetlib.Telnet(HOST_R1) # connecto to r1
```

```
tn_r1.read_until(b"Password: ")
tn_r1.write(password.encode('ascii') + b"\r") # write password telnet

print("You are connected to {}".format(HOST_R1))

tn_r1.read_until(b"R1>")
tn_r1.write('enable'.encode('ascii') + b"\r") # write enable command

tn_r1.read_until(b"Password: ")
tn_r1.write(password_enable.encode('ascii') + b"\r") # write password
enable

print('R1: "enable" mode activated')

tn_r1.read_until(b"R1#")
tn_r1.write('copy running-config tftp:'.encode('ascii') + b"\r") # send
runnig config to tftp server in linux

tn_r1.read_until(b"Address or name of remote host []? ")
tn_r1.write(IP_TFTP.encode('ascii') + b"\r") # set server tftp address

tn_r1.read_until(b"Destination filename [r1-config]? ")
tn_r1.write('backup'.encode('ascii') + b"\r") # set file name

time.sleep(6) # wait to send file

print('R1: runnig-config send to tftp server')

path = os.path.join(TFTP_DIR, 'backup')
file = open(path, 'r');

# Read operation
lines = []
for line in file.readlines():
    if 'R1' in line:
        lines.append(line.replace('R1', 'Remplazol'))
    else:
        lines.append(line)
file.close()

# Write operation
```

```
new_file = open(path, 'w+')
new_file.writelines(lines)
new_file.close()

print('R1: hostname R1 is update to Remplazol1')

tn_r1.read_until(b"R1#")
tn_r1.write('copy tftp: running-config'.encode('ascii') + b"\r") # send
runnig config to router in gns3

tn_r1.read_until(b"Address or name of remote host []? ")
tn_r1.write(IP_TFTP.encode('ascii') + b"\r") # set tftp server ip
address

tn_r1.read_until(b"Source filename []? ")
tn_r1.write('backup'.encode('ascii') + b"\r") # set filename of backup

tn_r1.read_until(b"Destination filename [running-config]? ")
tn_r1.write(b"\r") # set return key to continue

print('R1: running-config modified send to router R1')

tn_r1.read_until(b"Remplazol1#")
tn_r1.write('write'.encode('ascii') + b"\r") # save config

print('R1: configuration is saved')

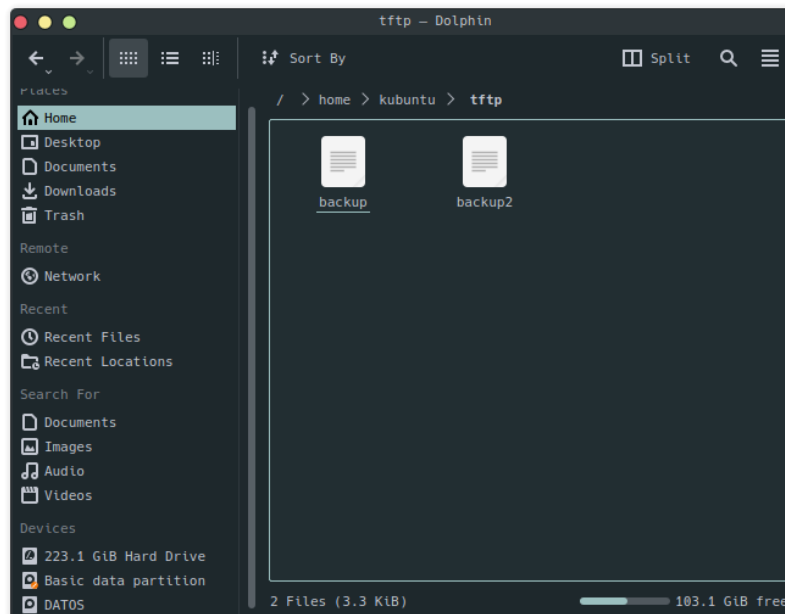
tn_r1.read_until(b"Remplazol1#")
tn_r1.close()
print('R1: telnet connection is close')
```

Capturas de pantalla

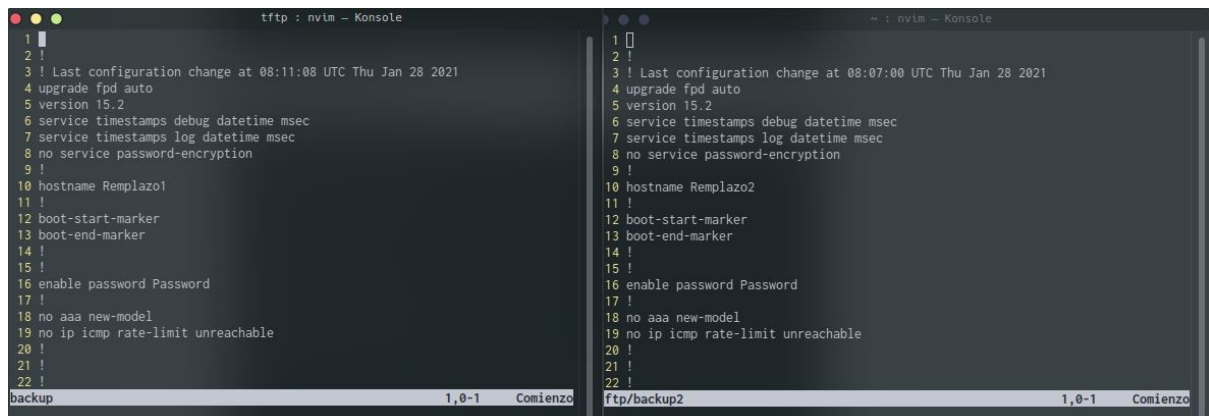
Ejecutando el script de python:

```
tftp_script : bash – Konsole
kubuntu@kubuntu-pc:~/Desktop/tftp_script$ python3 telscript.py
You are connected to 192.168.0.250
R1: "enable" mode activated
R1: runnig-config send to tftp server
R1: hostname R1 is update to Remplazo1
R1: running-config modified send to router R1
R1: configuration is saved
R1: telnet connection is close
You are connected to 192.168.1.2
R2: "enable" mode activated
R2: runnig-config send to tftp server
R2: hostname R2 is update to Remplazo2
R2: running-config modified send to router R2
R2: configuration is saved
R2: telnet connection is close
```

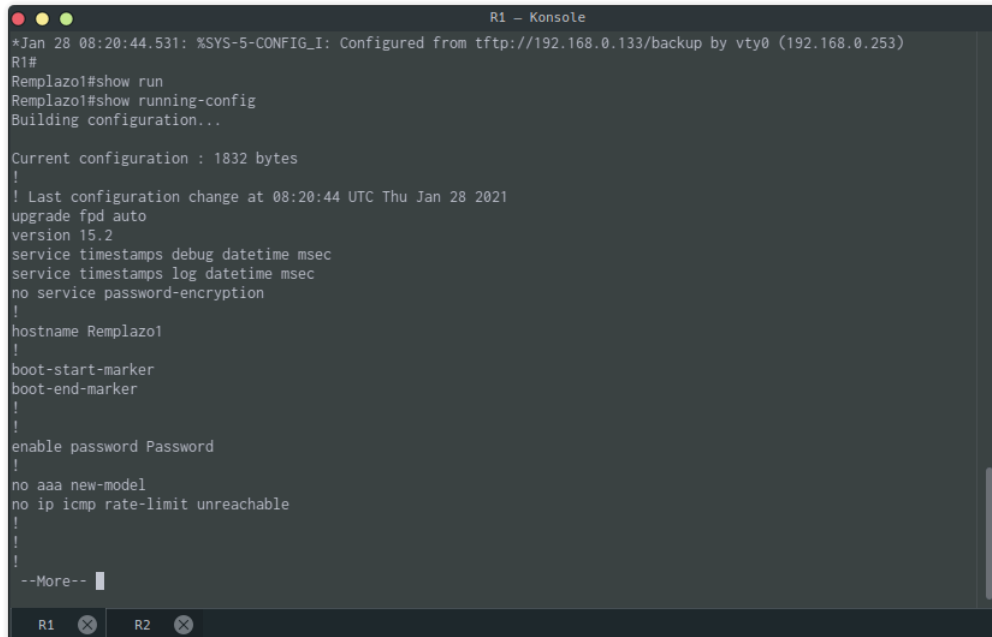
Archivos de configuración en servidor TFTP agregados:



Abrimos los archivos de configuración recibidos y podemos comprobar que corresponde al router **R1** y **R2**. Además, observamos que el hostname ya ha sido actualizado a **Remplazo1** y **Reemplazo2**, respectivamente.

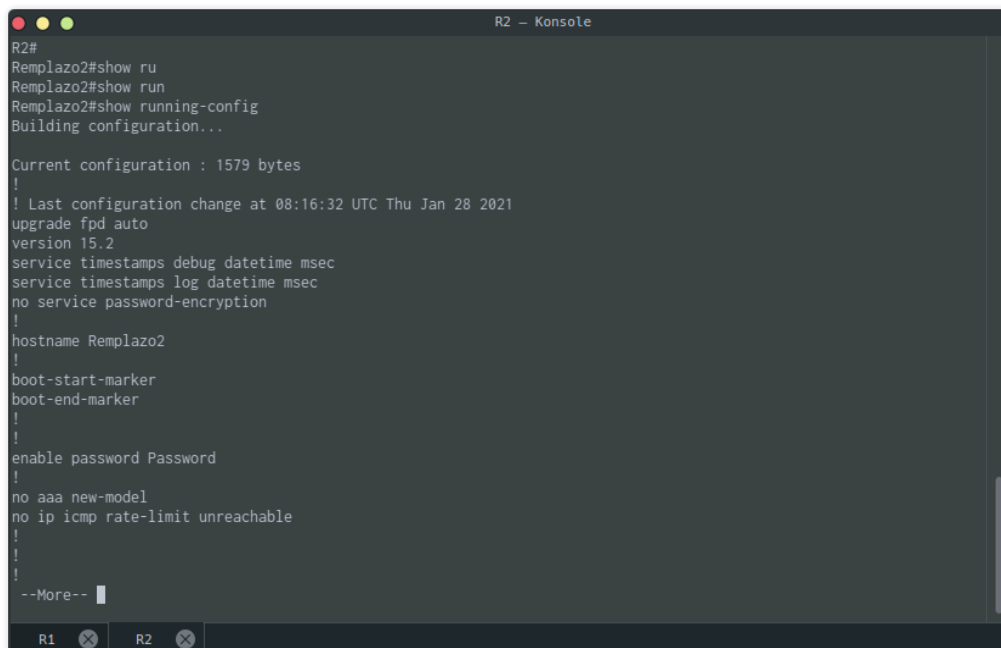


Finalmente comprobamos en GNS3, que la configuración de los routers R1 y R2 ha sido actualizada a través de tftp:



```
R1 - Konsole
*Jan 28 08:20:44.531: %SYS-5-CONFIG_I: Configured from tftp://192.168.0.133/backup by vty0 (192.168.0.253)
R1#
Remplazo1#show run
Remplazo1#show running-config
Building configuration...

Current configuration : 1832 bytes
!
! Last configuration change at 08:20:44 UTC Thu Jan 28 2021
upgrade fpd auto
version 15.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Remplazo1
!
boot-start-marker
boot-end-marker
!
!
enable password Password
!
no aaa new-model
no ip icmp rate-limit unreachable
!
!
--More--
```



```
R2 - Konsole
R2#
Remplazo2#show ru
Remplazo2#show run
Remplazo2#show running-config
Building configuration...

Current configuration : 1579 bytes
!
! Last configuration change at 08:16:32 UTC Thu Jan 28 2021
upgrade fpd auto
version 15.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Remplazo2
!
boot-start-marker
boot-end-marker
!
!
enable password Password
!
no aaa new-model
no ip icmp rate-limit unreachable
!
!
--More--
```

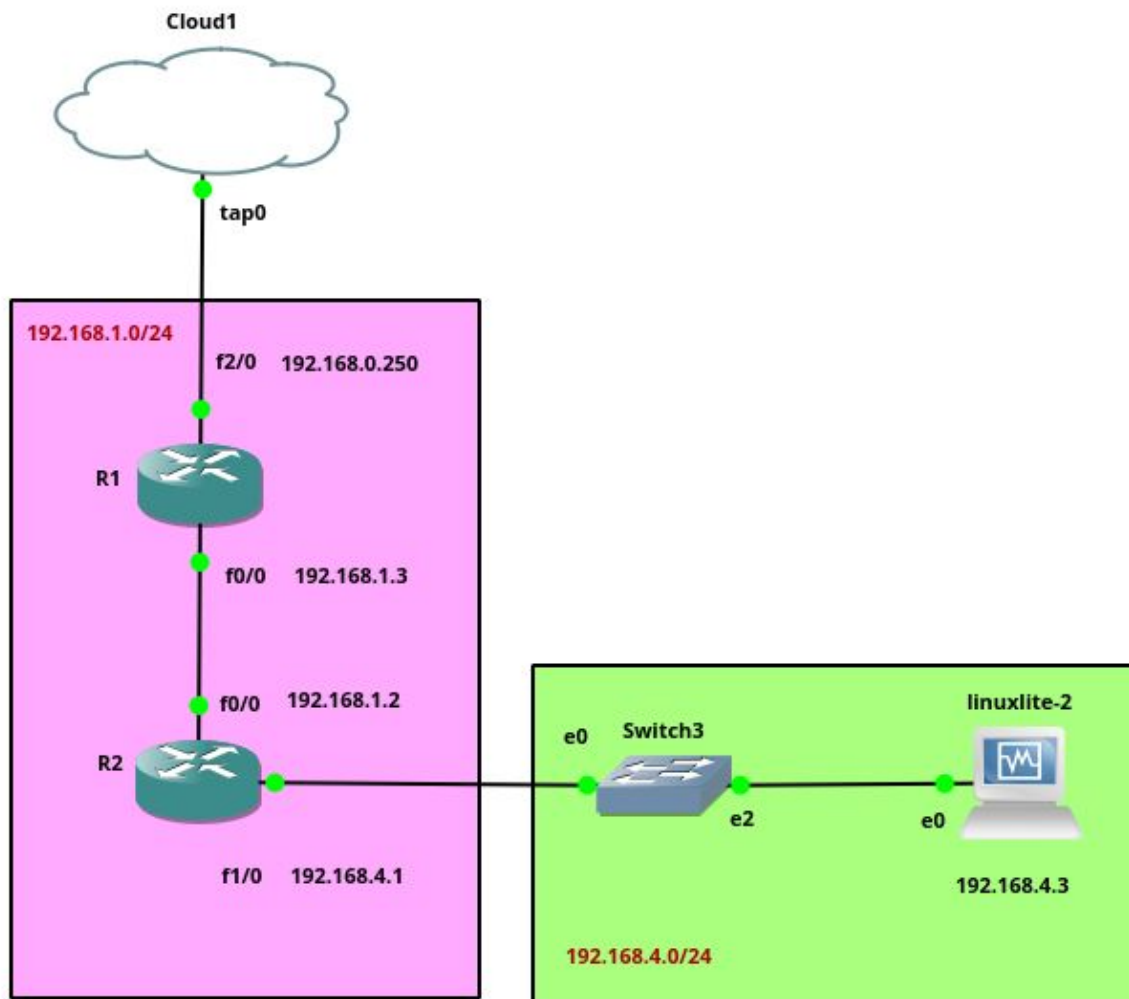
4. Broadcast

Objetivo de la práctica

Lograr que la transmisión broadcast llegue desde la máquina host hasta todas las máquinas que estén en el broadcast

Desarrollo

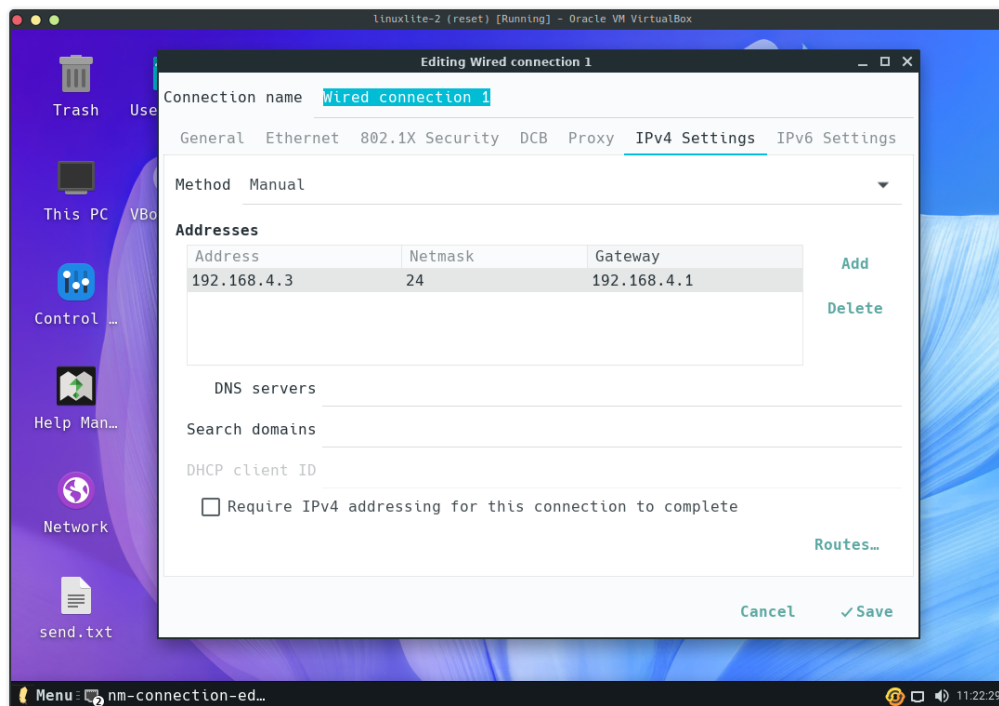
Tenemos esta topología en gns3:



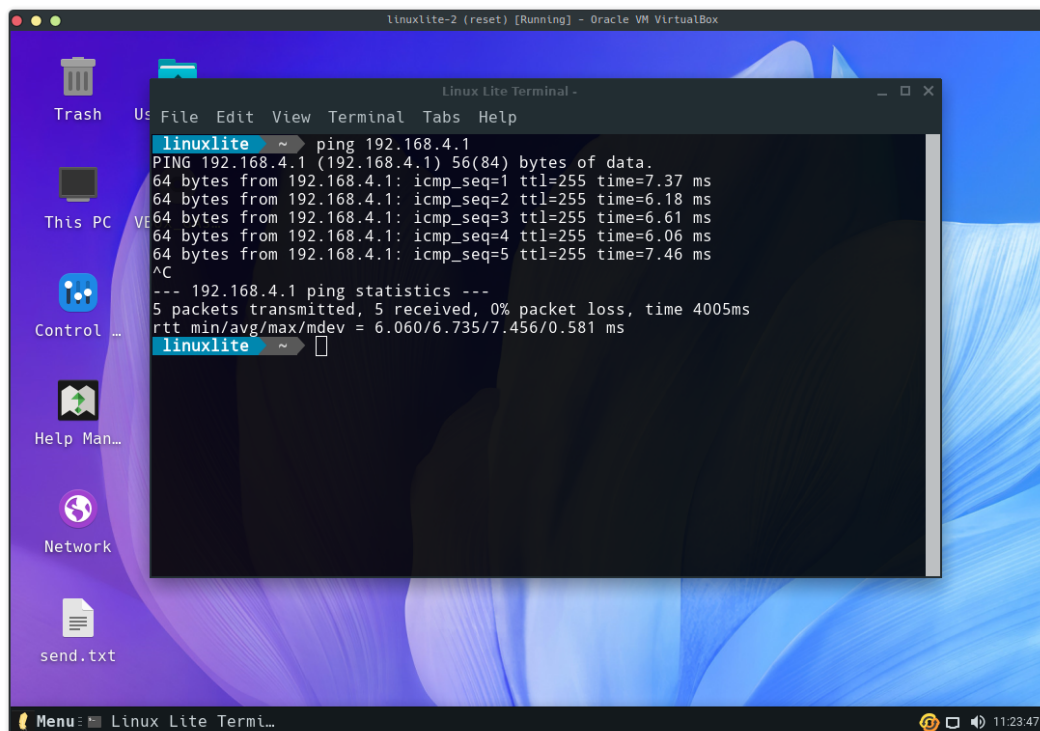
Ahora contamos con una máquina virtual linux, en mi caso con kubuntu. Esto para poder ejecutar el cliente broadcast implementado en lenguaje C.

Los routers están configurados con RIPV2 y además se ha configurado el cloud para poder utilizarlo con el adaptador virtual TAP0

Primero debemos encender nuestra máquina virtual **linuxlite-2** y asignarle una ip estática de manera manual, también le especificamos la puerta de enlace:



Hacemos ping a la puerta de enlace para comprobar:



Después, para habilitar la transmisión de paquetes broadcast a través del router utilizamos los siguientes comandos:

1. **enable**
2. **configure terminal**: Se habilita la terminal para entrar a modo de configuración del router
3. **ip forward-protocol udp**: Se especifica que debe aceptar el protocolo UDP
4. **interface type number**: Se indica la interfaz se hará la configuración siguiente
5. **ip address address mask**: Se asigna la ip con máscara a la interfaz actual
6. **ip helper-address address**: Habilita una dirección IP auxiliar para la interfaz que recibe los paquetes de difusión UDP, cabe mencionar que esto a menudo da como resultado que el origen de los paquetes de difusión UDP reciba respuestas de dos o más hosts. En la mayoría de las circunstancias, la fuente de los paquetes de difusión UDP acepta la primera respuesta e ignora las respuestas posteriores. En algunas situaciones, la fuente de los paquetes de difusión UDP no puede manejar respuestas duplicadas y reacciona recargando u otro comportamiento inesperado.
7. **exit**: Se sale de la configuración de la interfaz actual
8. **interface type number**
9. **ip address address mask**
10. **ip directed-broadcast**: Habilita el envío broadcast por IP en la interfaz que transmite las transmisiones UDP.
11. **end**

Configurando router R1:



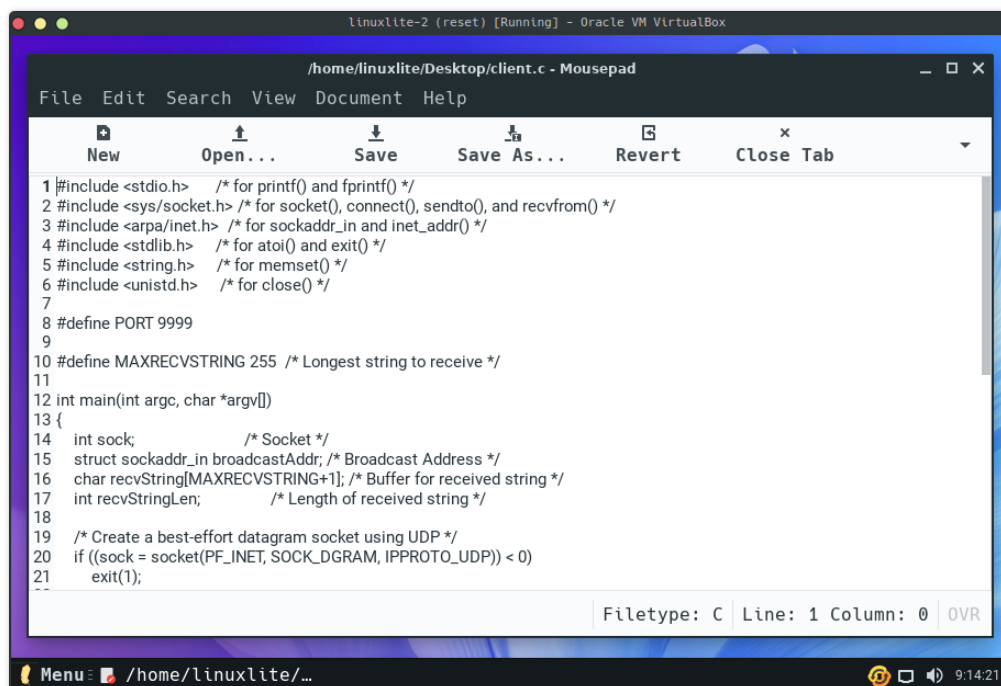
```
R2#
R2#
R2#
R2#
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#ip forw
R2(config)#ip forward-protocol udp
R2(config)#interface f2/0
R2(config-if)#ip helpe
R2(config-if)#ip helper-address 192.168.0.255
R2(config-if)#exit
R2(config)#interface f0/0
R2(config-if)#ip ad
R2(config-if)#ip dir
R2(config-if)#ip directed-broadcast
R2(config-if)#end
R2#
*Jan 29 03:28:11.503: %SYS-5-CONFIG_I: Configured from console by console
R2#
```

Configurando router R2P:



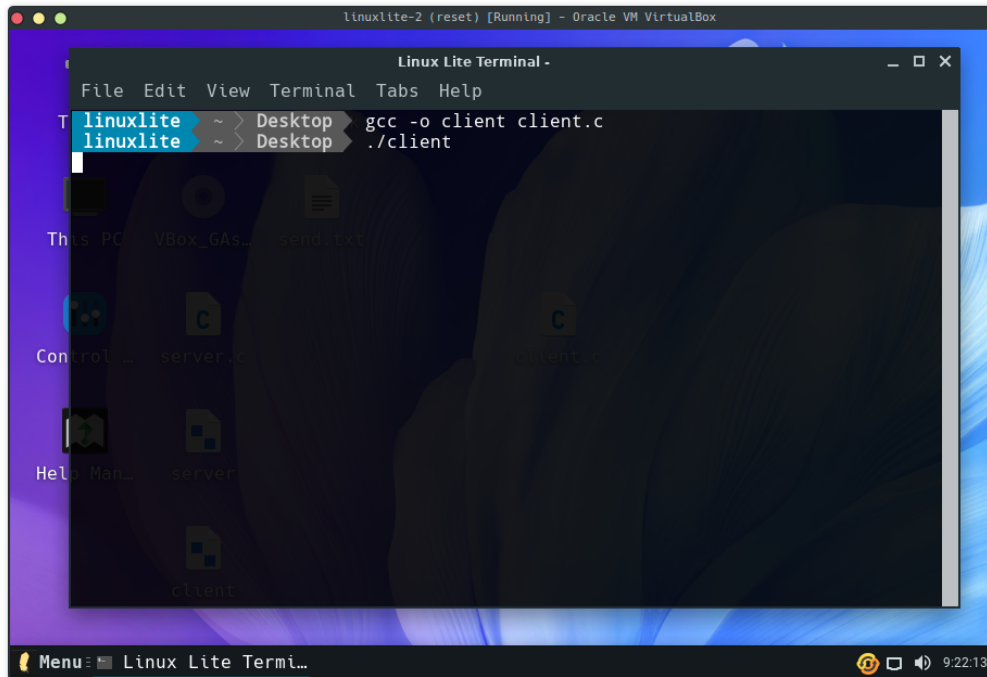
```
R2#
R2#configura termin
R2#configura termin
R2#configure terminal
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#ip fow
R2(config)#ip forw
R2(config)#ip forward-protocol udp
R2(config)#interface f0/0
R2(config-if)#ip help
R2(config-if)#ip helper-address 192.168.4.255
R2(config-if)#exit
R2(config)#interface f1/0
R2(config-if)#ip dir
R2(config-if)#ip directed-broadcast
R2(config-if)#end
R2#
*Jan 29 03:31:03.591: %SYS-5-CONFIG_I: Configured from console by console
R2#
```

Después cargamos el cliente broadcast en la máquina virtual, para esto nos podemos ayudar de las carpetas compartidas de virtualbox.

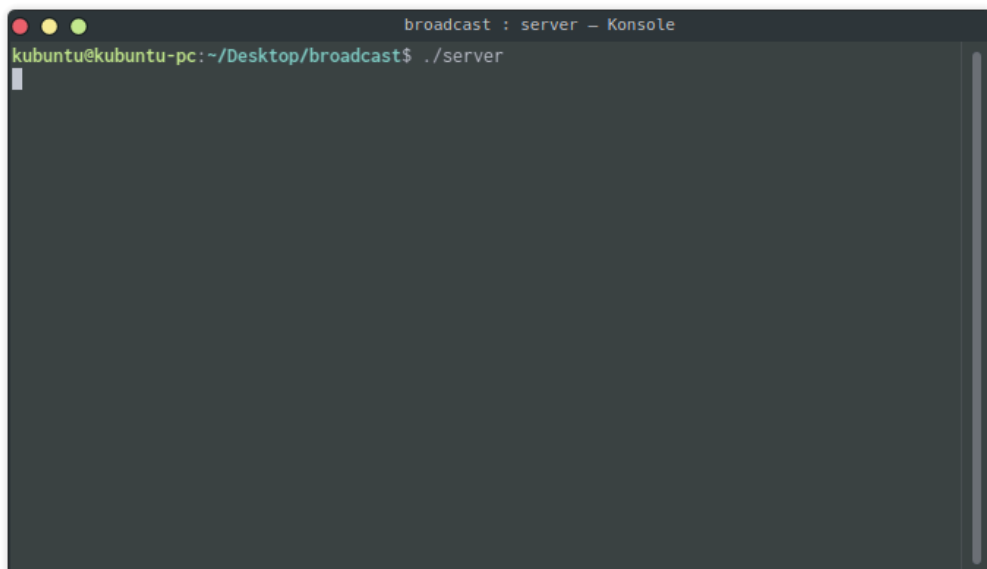


```
/home/linuxlite/Desktop/client.c - Mousepad
File Edit Search View Document Help
New Open... Save Save As... Revert Close Tab
1 #include <stdio.h> /* for printf() and fprintf() */
2 #include <sys/socket.h> /* for socket(), connect(), sendto(), and recvfrom() */
3 #include <arpa/inet.h> /* for sockaddr_in and inet_addr() */
4 #include <stdlib.h> /* for atoi() and exit() */
5 #include <string.h> /* for memset() */
6 #include <unistd.h> /* for close() */
7
8 #define PORT 9999
9
10 #define MAXRECVSTRING 255 /* Longest string to receive */
11
12 int main(int argc, char *argv[])
13 {
14     int sock; /* Socket */
15     struct sockaddr_in broadcastAddr; /* Broadcast Address */
16     char recvString[MAXRECVSTRING+1]; /* Buffer for received string */
17     int recvStringLength; /* Length of received string */
18
19     /* Create a best-effort datagram socket using UDP */
20     if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
21         exit(1);
22 }
```

Compilamos y ejecutamos el cliente broadcast en la máquina virtual **linuxlite-2**:



Ejecutamos el servidor broadcast desde la nuestra máquina padre, donde se está emulando la subred:



Capturas de pantalla

Finalmente podemos observar que el cliente broadcast recibe correctamente los paquetes enviados desde la máquina linux, a través del servidor broadcast, y estos pasan correctamente a través de la subred y los routers R1 y R2

