INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Aplicaciones para comunicaciones en Red

Practica Broadcast

Enviar imágenes

Alumno:

Cortez Enriquez Jovanny Wilver

González López Emiliano

3CV6

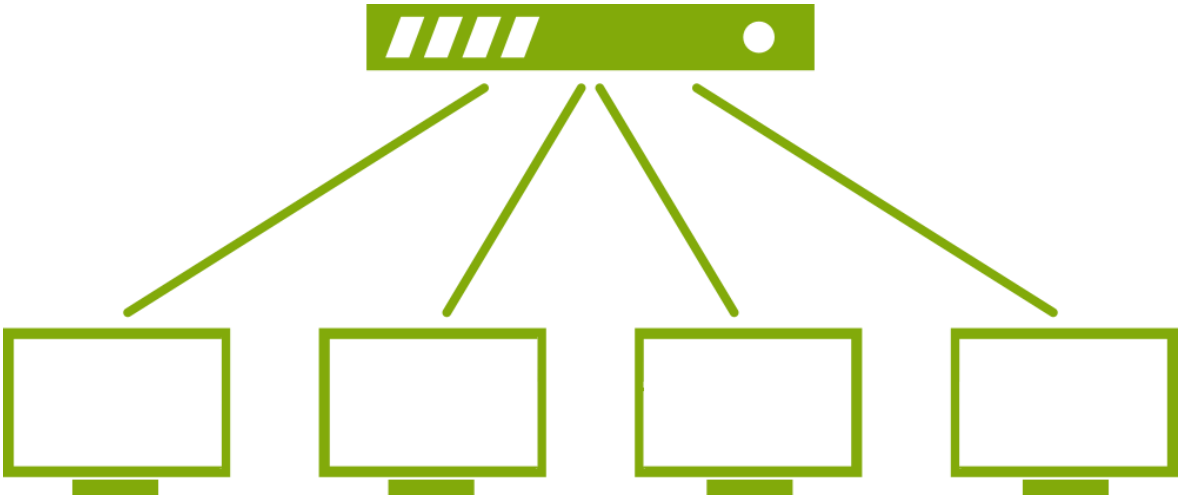
Contenido

Introducción

Es un modo de transmisión de información (difusión) 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.

Las redes de área local también se basan en el uso de un medio de transmisión compartido. Por lo tanto, es posible la difusión de cualquier trama de datos a todas las estaciones que se encuentren en el mismo segmento de la red. Para ello, se utiliza una dirección MAC especial. Todas las estaciones procesan las tramas con dicha dirección. Por ejemplo, la tecnología Ethernet realiza la difusión enviando tramas con dirección MAC de destino FF.FF.FF.FF.FF.FF.

Este paquete solamente alcanzará a los nodos que se encuentran dentro de la misma red física subyacente. En general, la red subyacente será una red de área local (LAN) o un segmento de ésta.



Broadcast dirigido

Un broadcast dirigido se envía a todos los hosts de una red específica. Este tipo de broadcast es útil para enviar un broadcast a todos los hosts de una red local. Por ejemplo, para que un host fuera de la red 172.16.4.0/24 se comunique con todos los hosts dentro de esa red, la dirección de destino del paquete sería 172.16.4.255. Aunque los routers no reenvían broadcasts dirigidos de manera predeterminada, se les puede configurar para que lo hagan.

Broadcast limitado

El broadcast limitado se usa para la comunicación que está limitada a los hosts en la red local. Estos paquetes siempre utilizan la dirección IPv4 de destino 255.255.255.255. Los routers no reenvían broadcasts limitados. Por esta razón, también se hace referencia a una red IPv4 como un dominio de broadcast. Los routers son dispositivos fronterizos para un dominio de broadcast.

Planteamiento del Problema

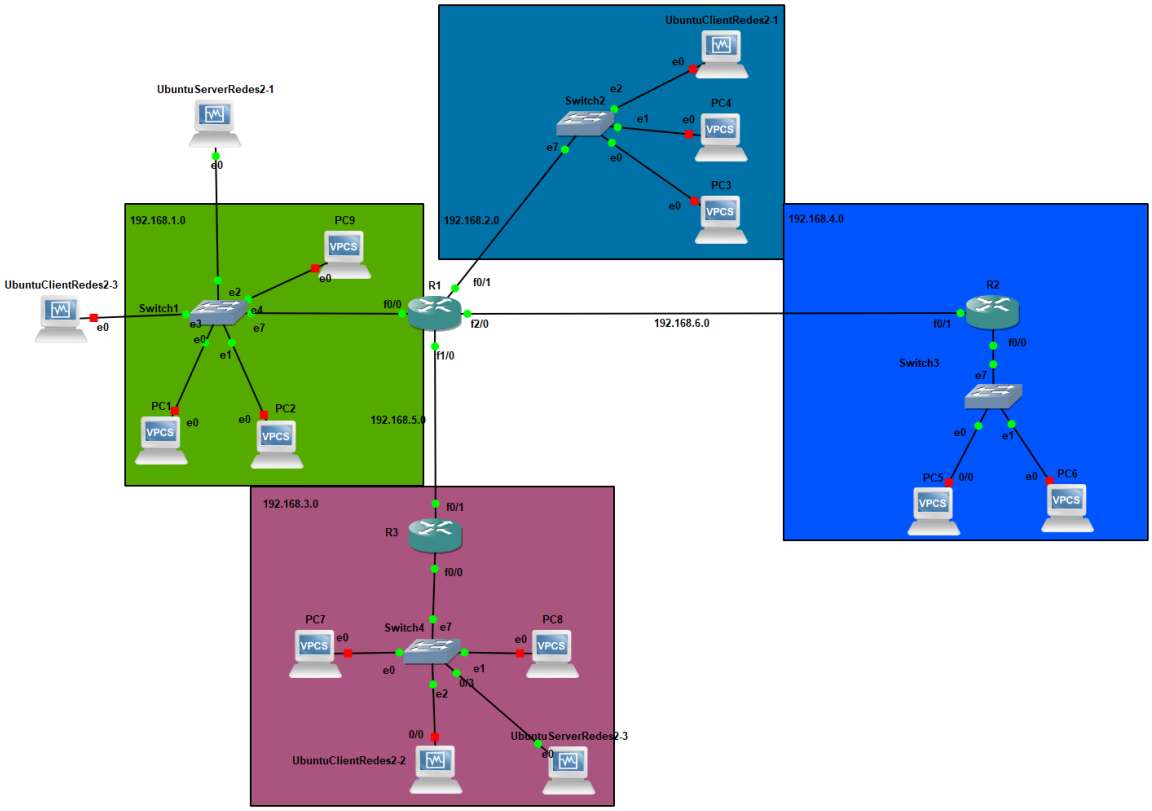
Diseñar y programar una aplicación que permita enviar múltiples imágenes a través de la arquitectura planteada en gns3.

Estas imágenes deben ser enviadas creando sockets de tipo Broadcast.

Requisitos

- Programa GNS3
- Arquitectura planteada por el profesor
- Sistema Operativo Linux Ubuntu Server 20
- Python 3
- VirtualBox

Desarrollo



Se realizan las configuraciones siguientes:

- Configurar la tabla de ruteo de manera dinámica con el protocolo RIP.
- Configurar con el comando ip helper los router para permitir la comunicación broadcast.

Configuración del Router

1- Modo configuración de terminal

>conf t

2- Entrar a la interfaz por donde escuchará la petición

>int fx/x

3- Asignar la IP del servidor DHCP

>ip helper-address 192.168.1.4

4- Habilitamos la interfaz

>no shutdown

5- Salir

>exit

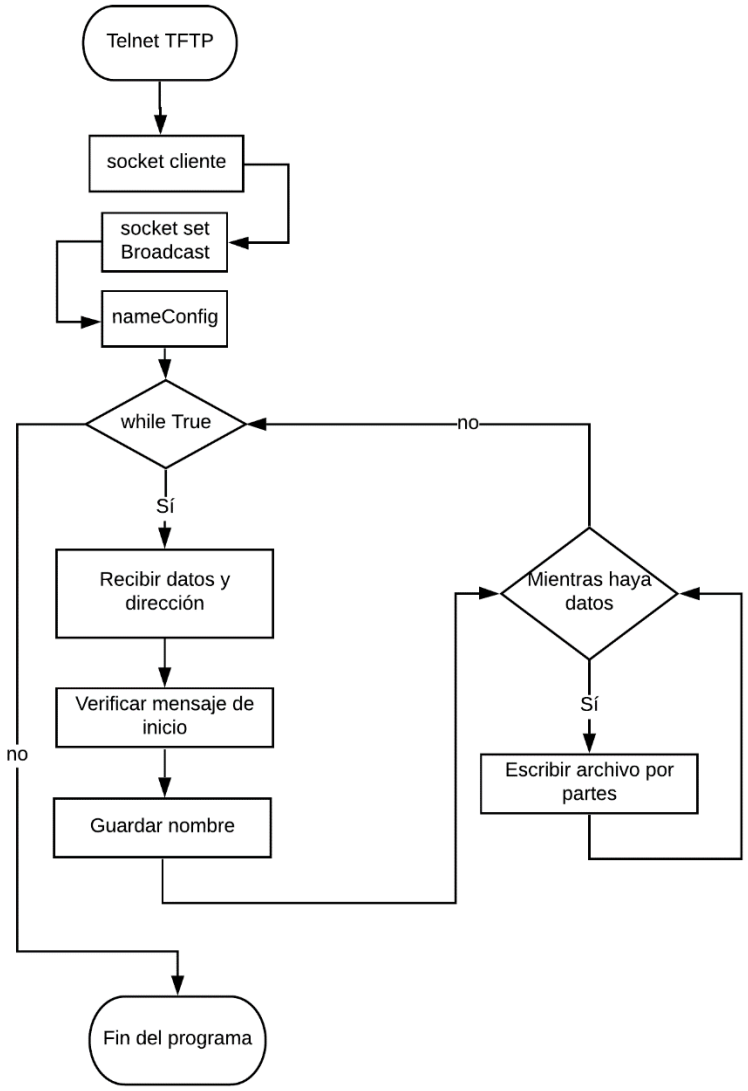
```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface f0/1
R1(config-if)#ip he
R1(config-if)#ip help
R1(config-if)#ip helper-address 192.168.1.4
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#exit
R1#
```

6- Verificar la que este habilitada la dirección de broadcast

>show ip interface fx/x

```
R1#show ip interface fa0/0
FastEthernet0/0 is up, line protocol is up
  Internet address is 192.168.2.1/24
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is 192.168.1.10
  Directed broadcast forwarding is disabled
  Multicast reserved groups joined: 224.0.0.9
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is enabled
```

Diagrama del programa del cliente.



En este algoritmo creamos primeramente declaramos un socket, para después habilitarlo como un socket de broadcast.

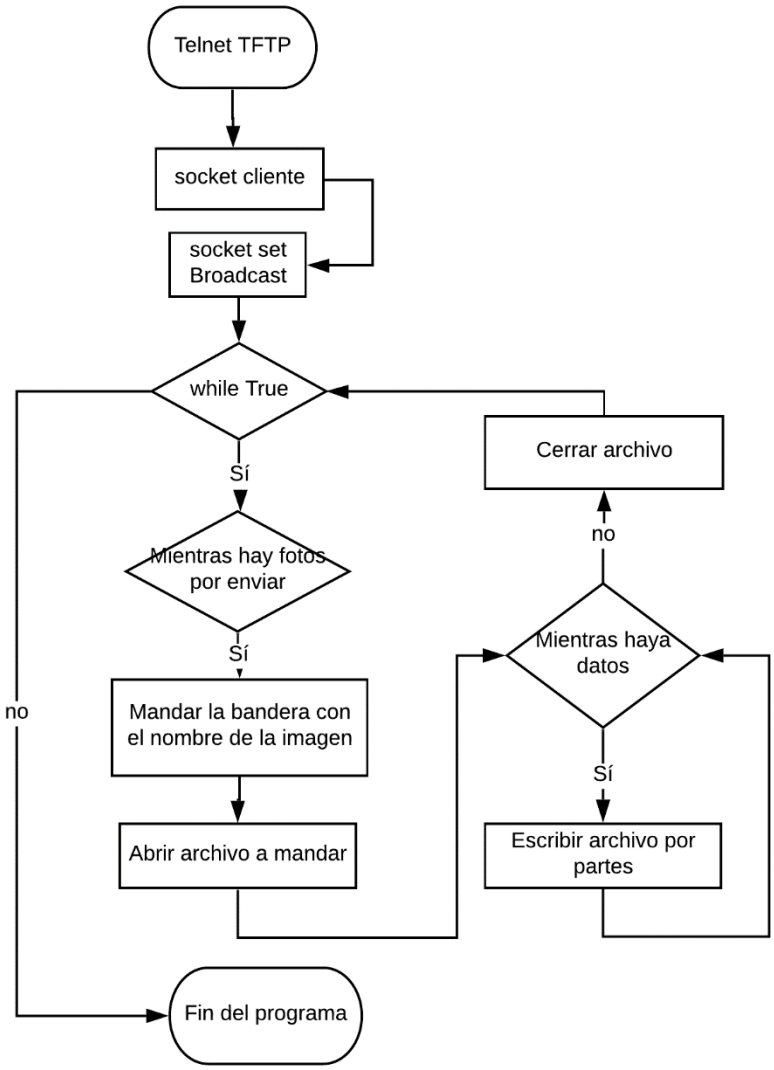
Entonces comenzamos a recibir datos, para verfificar que sea declare el inicio de un archivo con un identificador. Si es así, guardamos el nombre del archivo con el mensaje recibido que era el nombre.

Posteriormente de aceptar el nombre, abrimos un archivo en modo escritura, y empezamos a aceptar los paquetes de mientras existan datos que aceptar y no esta la bandera de final.

Código del cliente

```
import socket
import zipfile
import sys
import threading
import os

port = 37020

#configuracion de conexion por broadcast
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP
)   #Se crea el socket UDP
client.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
    #Se configura como broadcast
client.bind(("", port))
#Se vincula con la dirección broadcast y el puerto

print('[+] esperando imagenes...')

filename = "/home/"+ socket.gethostname() +"/default.png"

while True:
    data, addr = client.recvfrom(513)
    print("Conexion de: {}".format(addr[0]))
    #verifica si es el comienzo de una nueva imagen
    if data[0] == 51:
        name = data[1:len(data)]
        filename = "/home/"+ socket.gethostname() +"/"+str(name)+".png"
    elif data[0] == 49:
        f = open(filename, "wb")
        f.write(data[1:len(data)])
        while True:
            data, addr = client.recvfrom(513)
            if data[0] == 50:
                f.write(data[1:len(data)])
            elif data[0] == 48:
                f.write(data[1:len(data)])
                f.close()
                break
        print("guardado")


client.close()
```

Diagrama del Servidor



En este algoritmo primero creamos los sockets y los habilitamos como sockets de broadcast.

Entonces entramos a nuestro ciclo infinito por ser el servidor, y empezamos a enviar las imágenes, primeramente, enviando una bandera con el nombre, posteriormente abrimos el archivo que queremos enviar.

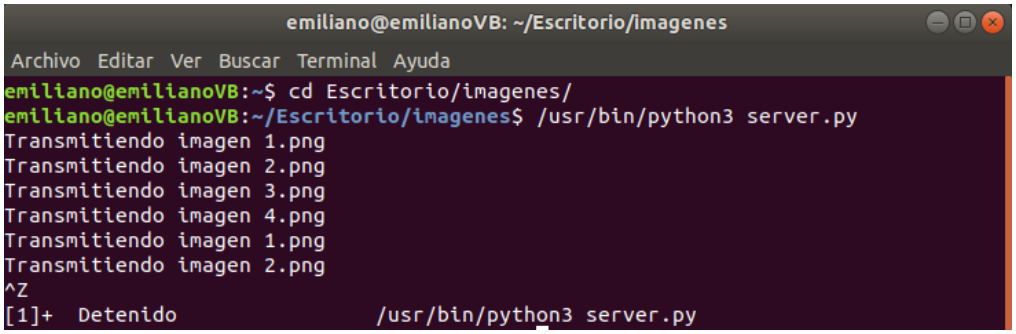
Comenzamos a enviar todo nuestro archivo por parte, finalmente cerramos el archivo y proseguimos con los siguientes archivos.

Código del Servidor

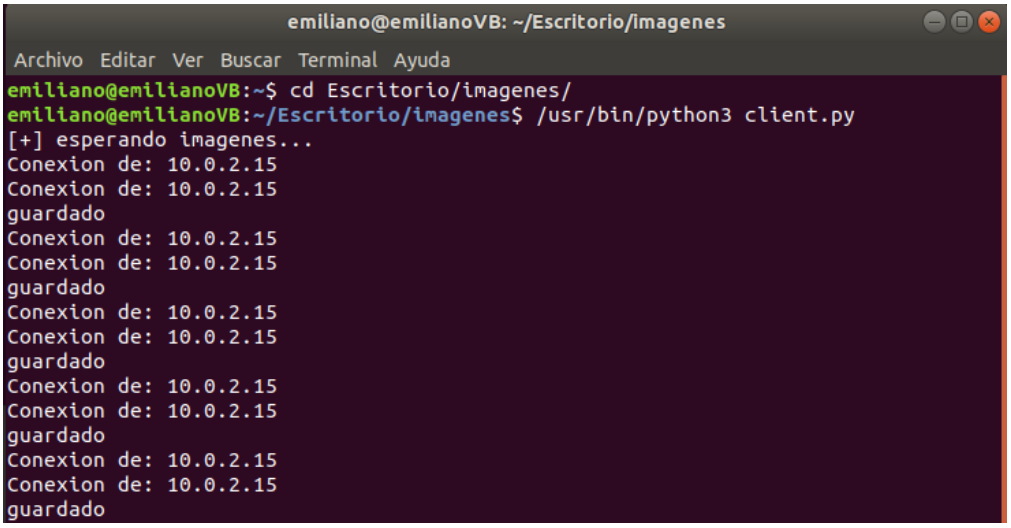
```
import os
import zipfile
import socket
import time

port = 37020
#Lista con los nombres de las 4 imagenes a enviar
imagenes = ["1.png", "2.png", "3.png","4.png"]

#definicion de modo broadcast en el socket mediante UDP
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP
)   #Se crea el socket UDP
server.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)

while True:
    for img in imagenes:
        print("Transmitiendo imagen " + img)
        file = "/home/"+ socket.gethostname() + "/" + str(img)
        #enviar titulo
        content = bytes(img, "utf-8")
        content = bytes("3", "utf-8") + content
        server.sendto(content,('<broadcast>', port))

        #abrir imagenes en bytes
        f = open(file,"rb")
        content = f.read(512)
        content = bytes("1", "utf-8") + content

        while content:
            server.sendto(content,('<broadcast>', port))
            content = f.read(512)
            if len(content) < 512:
                content = bytes("0", "utf-8") + content
                server.sendto(content, ('<broadcast>', port))
                time.sleep(2)
                break
            else:
                content = bytes("2", "utf-8") + content
        f.close()


#Cierro el socket
server.close()
```

Pruebas

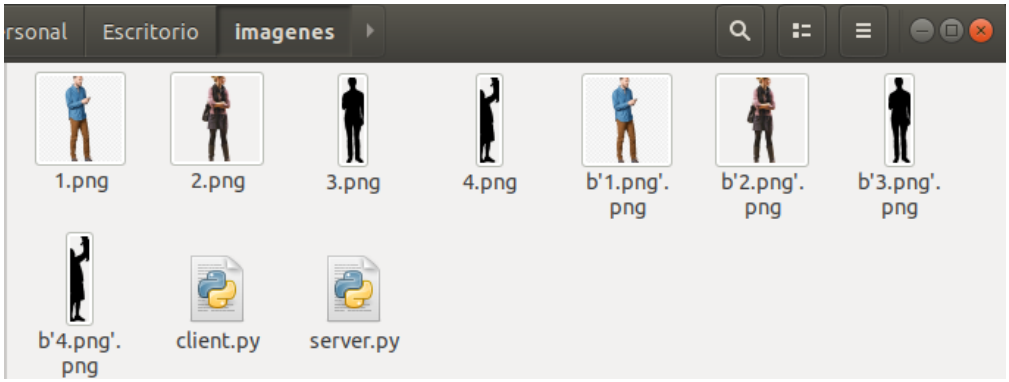
Prueba en servidor local



Servidor corriendo



Cliente corriendo, muestra la conexión y las imágenes guardadas.



Imágenes guardadas correctamente

Servidor corriendo en nuestra arquitectura

Aquí se ven ambas maquinas virtuales con la misma codificación del archivo, al correr nuestras maquinas virtuales en modo texto no tenemos como mostrar las imágenes, pero podemos mostrar con nano la imagen.

Conclusiones

Cortez Enríquez Jovanny Wilver

El desarrollo de la práctica estuvo un tanto complicado debido a que realizábamos en primera instancia, la configuración del socket de manera errónea y no podíamos mandar los datos de manera broadcast pero se logro solucionar usando la documentación de socket con Python y los usados por el lenguaje C, así mismo tuvimos problemas al cierre de los archivos que se logro solucionar sin problemas. Crear un servidor no es tan complicado como se ve en la implementación de algunos servidores que se ocupan en distribuciones Linux principalmente y es por ello que las funciones que hacen es indispensable para mantener la comunicación estable sin problemas.

González López Emiliano

Durante el desarrollo de esta práctica tuvimos complicaciones en la parte de codificación ya que teníamos problemas guardando las imágenes, porque, aunque se mandaba, se recibían incorrectamente, pero al final descubrimos que teníamos un error cerrando los archivos. Finalmente, con esto aprendimos a crear socket de tipo UDP y tambien a habilitar las transmisiones de broadcast en los routers mismas que tambien ayudan en la configuración de DHCP.