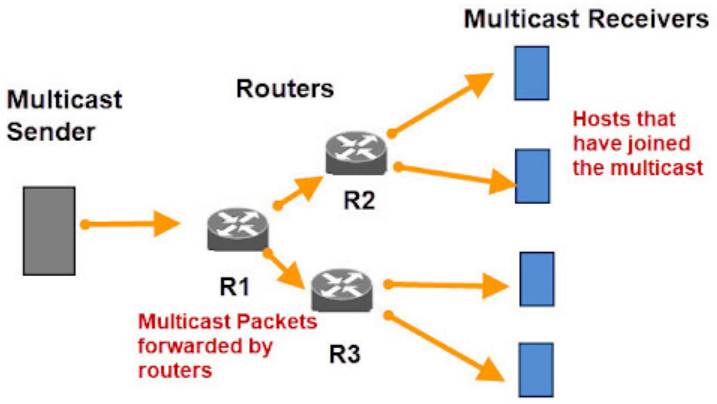
UDP Multicast

Multicast es una técnica de comunicación en redes informáticas donde un único flujo de datos se envía a múltiples receptores interesados de manera eficiente. En lugar de enviar copias separadas de los datos a cada receptor (como en unicast), multicast permite que los datos se envíen a un grupo de receptores simultáneamente mediante el uso de una dirección IP especial.



Multicasting on the Internet

Las direcciones IP de multicast pertenecen a un rango específico en el espacio de direcciones IPv4 e IPv6:

- **IPv4:** Las direcciones de multicast en IPv4 van desde 224.0.0.0 hasta 239.255.255.255.
- **IPv6:** Las direcciones de multicast en IPv6 comienzan con ff00::/8.

Sus características son:

- **Eficiencia en el Uso del Ancho de Banda:** Multicast permite el envío de un único flujo de datos que es replicado por los routers de la red solo cuando es necesario, ahorrando ancho de banda en comparación con el unicast.
- **Escalabilidad:** Multicast es altamente escalable, permitiendo que un gran número de receptores reciban datos simultáneamente sin aumentar la carga en el emisor.
- **Direcciones IP Especiales:** Utiliza direcciones IP en rangos específicos reservados para multicast. Los hosts que desean recibir datos de multicast deben unirse a estos grupos mediante la configuración de sus interfaces de red.
- **Protocolos de Administración**: Protocolos como IGMP (Internet Group Management Protocol) para IPv4 y MLD (Multicast Listener Discovery) para IPv6 se utilizan para gestionar la membresía en grupos de multicast.
- **Enrutamiento Multicast:** Los routers de la red deben soportar protocolos de enrutamiento multicast como PIM (Protocol Independent Multicast) para dirigir los flujos de datos de multicast de manera eficiente.

Desarrollo:

Inicialmente creamos un proyecto en Visual Studio Code, basta con crear una carpeta y tres scripts con extensión .py para esta práctica se nombraron de la siguiente manera:

- cliente.py
- servidor_1.py
- servidor_2.py

Ahora se buscará en la presente práctica enviar un mensaje al servidor 1 y al servidor 2 que se encuentran en ejecución en un grupo multicast, desde el cliente al mandar un mensaje su funcionamiento sería recibir ese mensaje en el grupo y puerto asociado a ese servidor.

Se realizará en la dirección multicast siguiente:

- 224.1.1.1

En el puerto:

- 5004

Primero comenzaremos definiendo la lógica que se va a utilizar para el cliente, principalmente se utiliza la biblioteca socket para enviar mensajes a un grupo de multidifusión, se configura la dirección de multidifusión 224.1.1.1 y el puerto 5004, y establece un TTL (Time to Live) de 2 saltos en la red. Luego, crea un socket UDP (socket.AF_INET para IPv4 y socket.SOCK_DGRAM para UDP) y configura la opción de socket para definir el TTL usando setsockopt. El código entra en un bucle que pide al usuario ingresar mensajes. Cada mensaje ingresado se convierte a bytes (encode('utf-8')) y se envía al grupo de multidifusión utilizando sendto. Si el usuario ingresa 'salir', el bucle se rompe y el programa termina, permitiendo así al usuario enviar múltiples mensajes hasta que decida detener el programa.

Una vez comprendido el flujo del cliente se procede a agregar el siguiente fragmento de código en el archivo cliente.py como se muestra a continuación:

```
import socket

# Configuración del grupo de multidifusión y el puerto
group = '224.1.1.1'
port = 5004

# Restricción de 2 saltos en la red
ttl = 2

# Crear el socket UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, ttl)

print("Escribe tus mensajes. Escribe 'salir' para terminar.")

while True:
    # Pedir al usuario que ingrese un mensaje
    message = input("Ingrese el mensaje a enviar: ")
```

```
    # Salir del bucle si el usuario escribe 'salir'
    if message.lower() == 'salir':
        break

    # Enviar el mensaje al grupo de multidifusión
    sock.sendto(message.encode('utf-8'), (group, port))

print("Cliente cerrado.")
```

Ahora para establecer la lógica de nuestro servidor debemos de comprender cuál será la funcionalidad de lo que se va a ejecutar, para configurar el servidor para recibir mensajes de un grupo de multidifusión se deben de importar las bibliotecas socket y struct. Define la dirección de multidifusión como '224.1.1.1' y el puerto como 5004. Luego, crea un socket UDP utilizando socket.AF_INET para IPv4 y socket.SOCK_DGRAM para UDP. Configura el socket para permitir la reutilización de direcciones con setsockopt y lo vincula al puerto de multidifusión especificado. Se utiliza struct.pack para preparar una solicitud de membresía al grupo de multidifusión y la añade al socket con setsockopt. Después se imprime un mensaje indicando que el servidor está listo. En un bucle infinito, el servidor recibe mensajes de hasta 10240 bytes y los imprime en la consola, permitiendo así recibir y mostrar todos los mensajes enviados a la dirección de multidifusión y puerto configurados.

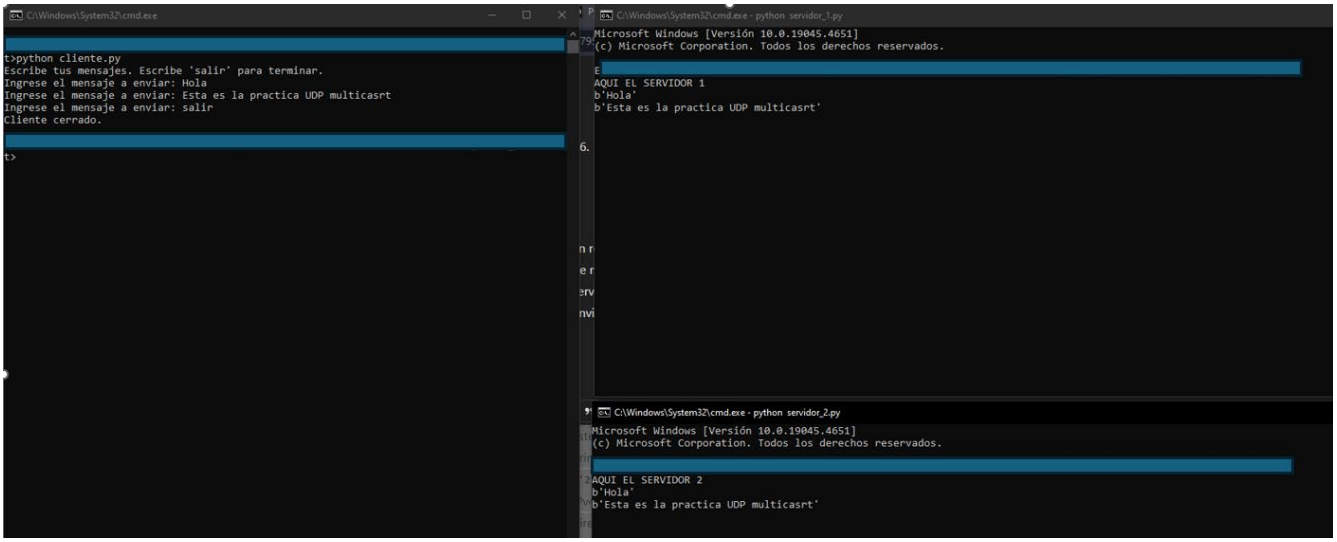
Ahora se debe agregar el siguiente fragmento de código para que el funcionamiento sea el esperado, esto es en el archivo servidor_1.py:

```
import socket
import struct
MCAST_GRP = '224.1.1.1'
MCAST_PORT = 5004
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock.bind(('', MCAST_PORT))
mreq = struct.pack("4sl", socket.inet_aton(MCAST_GRP), socket.INADDR_ANY)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
print("AQUI EL SERVIDOR 1")
while True:
    print(sock.recv(10240))
```

Lo mismo se hace con el archivo servidor_2.py:

```
import socket
import struct
MCAST_GRP = '224.1.1.1'
MCAST_PORT = 5004
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock.bind(('', MCAST_PORT))
mreq = struct.pack("4sl", socket.inet_aton(MCAST_GRP), socket.INADDR_ANY)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
print("AQUI EL SERVIDOR 2")
while True:
    print(sock.recv(10240))
```

Ahora procedemos a realizar la ejecución y ver el funcionamiento del envío de mensajes a través de una red de multidifusión:



Ahora si se observa en Wireshark se podrá ver el mensaje que es mandado a la dirección multicast: