

Nombre: Luis Alberto Vargas González.

Fecha: 24/01/2025.

U1 A1: Socket UDP.

Clase: Sistemas Distribuidos.

Maestría en Ciberseguridad.

Descripción de programas.

Se hace envío de 2 programas hechos en lenguaje de programación Python en Sistema operativo Linux, el primero es el Servidor y el segundo del cliente, ambos programas comienzan importando la librería socket, la cual nos permitirá poder establecer la conexión que necesitamos en este caso UDP, por lo que es indispensable contar con ella, después ambos programas establecen los parámetros de conexión; IP de puerto, puerto y tamaño de buffer para poder establecer conexión entre cliente y servidor.

```
import socket

#Configuracion del servidor
SERVER_IP = "127.0.0.1"
SERVER_PORT = 12345
BUFFER_SIZE = 1024
```

Después ambos programas establecen una función que serán las encargadas de establecer las conexiones entre cliente-servidor, por lo que dentro de ambas funciones se crea el socket específicamente usando la librería antes mencionada.

```
#Crear el socket UDP
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((SERVER_IP, SERVER_PORT))
print(f'Servidor escuchando en en {SERVER_IP}: {SERVER_PORT}')
```

En este caso que es la creación del socket del server, se crea un objeto de tipo socket para poder después usar un bind() y vincular este objeto a los parámetros antes definidos de Ip, puerto y buffer, posteriormente se imprime en consola el puerto e IP donde está escuchando el server.

```
#Creación del socket
```

```
cliente_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

En este caso que es el del cliente solo se crea el objeto de tipo socket

```
while True:
```

```
#Este while recibe datos de cliente y responde
```

```
message, client_address = server_socket.recvfrom(BUFFER_SIZE)
```

```
print(f'Mensaje recibido de {client_address}: {message.decode()}')
```

```
response = f'Mensaje recibido {message.decode()}'
```

```
server_socket.sendto(response.encode(), client_address)
```

```
print(f'Respuesta enviada a {client_address}')
```

Posteriormente procedemos con el while a recibir y responder datos del cliente por parte del servidor, se imprime el mensaje que cliente envió y finalmente el servidor le responde bidireccionalmente al cliente para la confirmar de recibido el mensaje.

```

while True:
message = input ("Introduce mensaje para enviar a server (escribe exit para
salir): ")
if message.lower() == "exit":
print("Cerrando cliente UDP")
break

cliente_socket.sendto(message.encode(), (SERVER_IP, SERVER_PORT))
#Aqui podemos verificar que le llegó al server la info correcta
response, server_address = cliente_socket.recvfrom(BUFFER_SIZE)
print(f'Respuesta del server: {response.decode()}')

cliente_socket.close()

```

En el caso del cliente podemos notar que en el ciclo while se nos pide introducir el mensaje a mano y nos seguirá pidiendo un mensaje hasta no escribir “exit”, posteriormente se hace el envío del mensaje con .sendto(), el método recvfrom nos sirve para poder verificar que el mensaje coincida con la capacidad que establecimos del buffer.

Por último, en ambos programas podemos ver que se utiliza una invocación de las funciones en el método main que creamos.

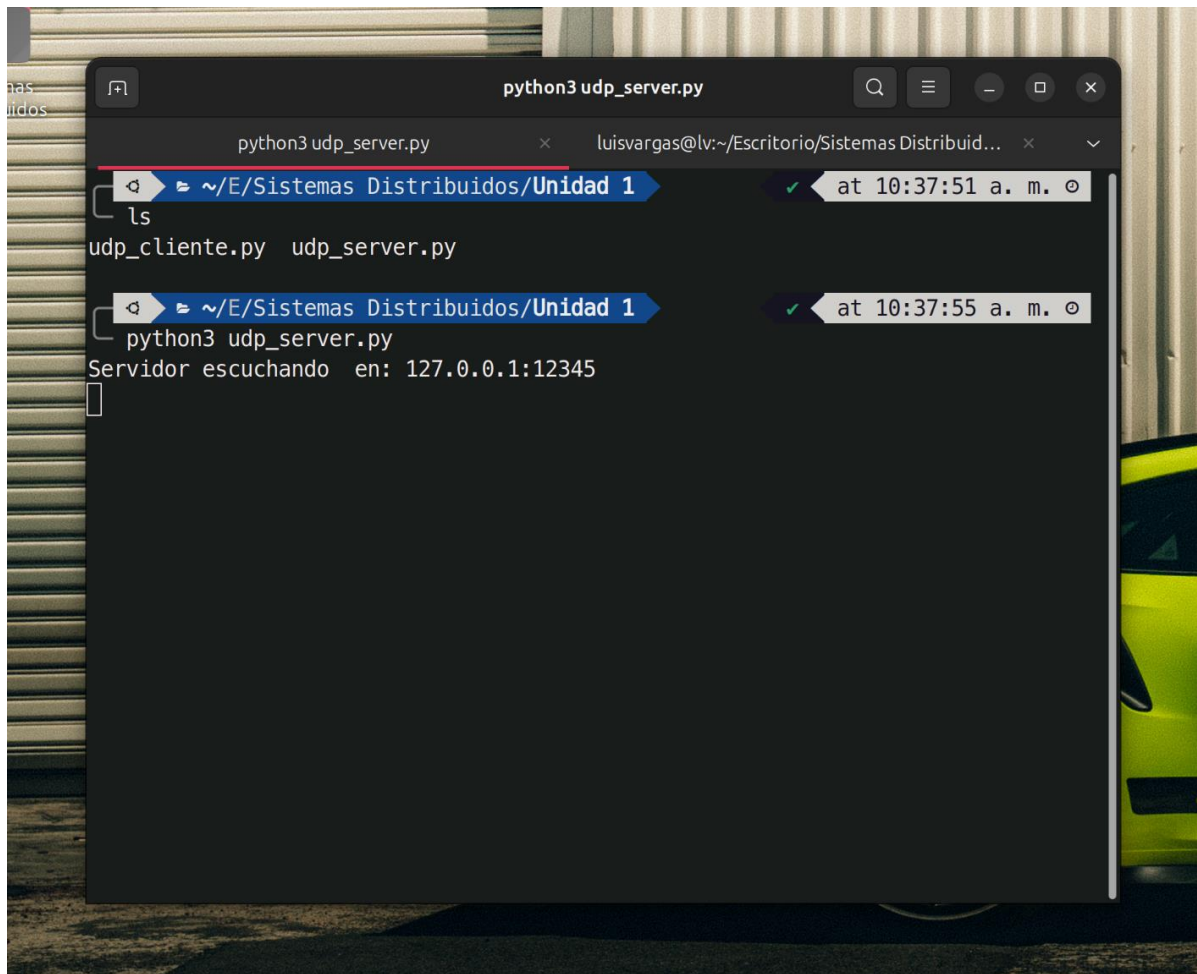
```

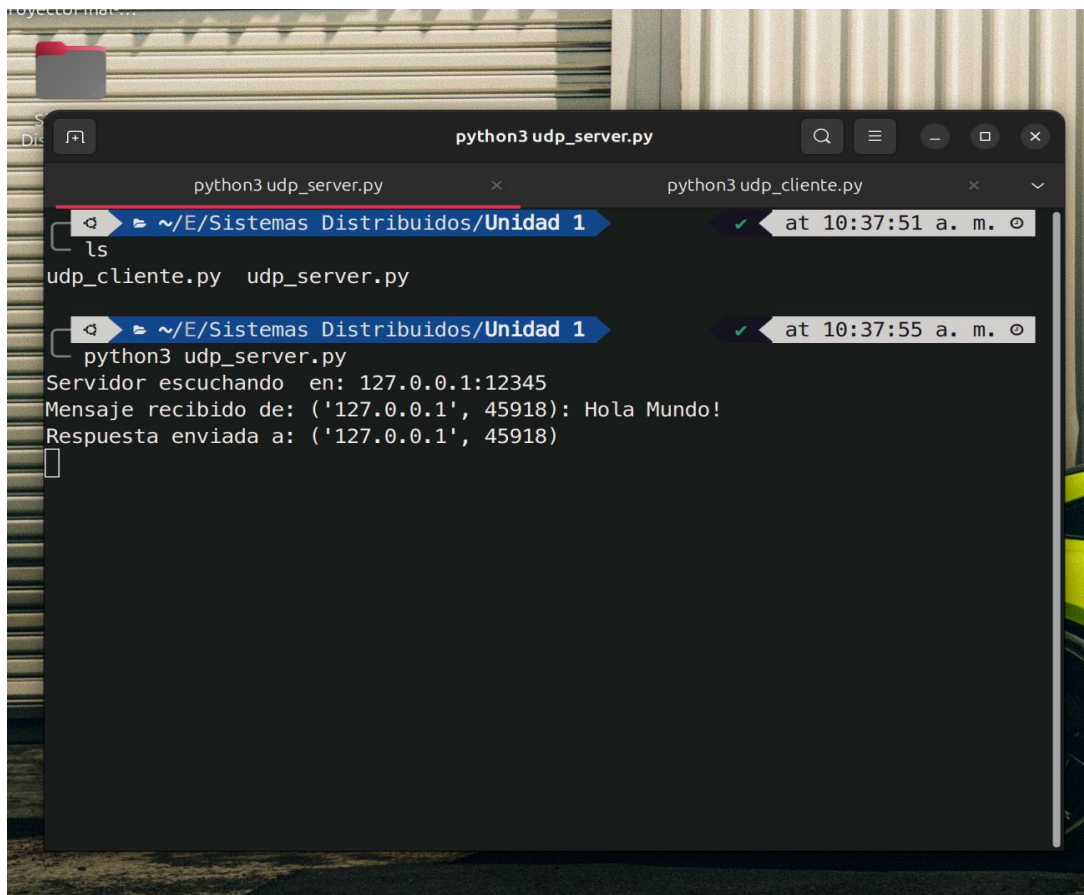
if __name__ == "__main__":

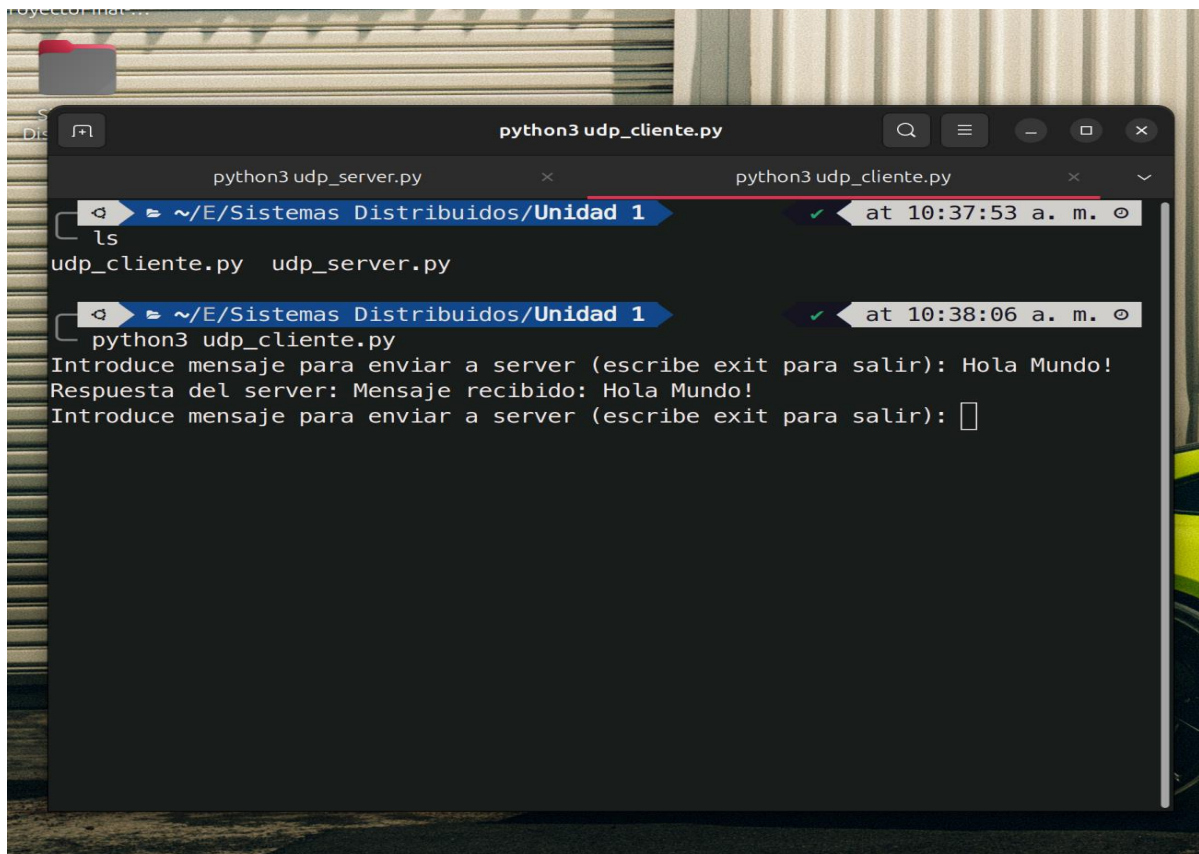
```

```
udp_cliente()
```

Evidencia de funcionamiento.







```
python3 udp_cliente.py
python3 udp_server.py
python3 udp_cliente.py
ls
udp_cliente.py  udp_server.py
python3 udp_cliente.py
Introduce mensaje para enviar a server (escribe exit para salir): Hola Mundo!
Respuesta del server: Mensaje recibido: Hola Mundo!
Introduce mensaje para enviar a server (escribe exit para salir):
```

Como se puede notar primero se levanta el servidor después el cliente, cuando el cliente envía el mensaje el servidor lo recibe, lo imprime en pantalla y devuelve el mensaje recibido para su confirmación de recepción.

Conclusiones:

Al finalizar esta actividad nos pudimos percatar que la realización de la misma conlleva un esfuerzo pequeño pues el algoritmo en si es muy sencillo de implementar por lo tanto es fácil de entender y de modificar en caso de ser requerido. Es importante entender de que va el protocolo UDP y TCP para poder entender a mayor profundidad los sistemas distribuidos, pues tanto TCP como UDP son usados para diferentes tipos de sistemas y en distintas topologías de red.