



Nombre: Luis Alberto Vargas González.

Fecha: 1/02/2025.

U4 A1: Servidor Estrella.

Clase: Sistemas Distribuidos.

Maestría en Ciberseguridad.

Descripción de programas.

En esta práctica se implementaron 2 códigos los cuales son cliente y servidor respectivamente, en el código de cliente se presentan dos funciones las cuales se encargan de registrar los nodos en el servidor y de recibir mensajes provenientes del servidor, usando los métodos listen y bind así como los métodos connect y sendall para enviar los datos del nodo al servidor.

Código de cliente.

```
import socket
import argparse
import threading

# Función para escuchar mensajes del servidor
def escuchar_mensajes(puerto_escucha):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as listener_socket:
        listener_socket.bind(("127.0.0.1", puerto_escucha))
        listener_socket.listen()
        print(f"Escuchando en el puerto {puerto_escucha} para mensajes entrantes...")
        while True:
            conn, addr = listener_socket.accept()
            with conn:
                mensaje = conn.recv(1024).decode('utf-8')
                if mensaje:
                    print(f"Mensaje recibido: {mensaje}")
                else:
```

```

break

# Función de cliente TCP para enviar mensajes (solo para registro)
def registrar_nodo(mi_ip, mi_puerto):
    print("Registrando el nodo...")
    # Conectarse al servidor para registrarse
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
        client_socket.connect(("127.0.0.1", 12345))
    # Enviar la información del cliente (IP y puerto)
    client_socket.sendall(f"{mi_ip}:{mi_puerto}".encode('utf-8'))
    # Recibir la respuesta del servidor
    respuesta = client_socket.recv(1024).decode('utf-8')
    print(respuesta)

# Configuración del cliente
if __name__ == "__main__":
    # Leer los parámetros del puerto del cliente desde la línea de comandos
    parser = argparse.ArgumentParser(description="Nodo Cliente P2P.")
    parser.add_argument("--puerto", type=int, required=True, help="Puerto en el que este nodo escuchará conexiones.")
    args = parser.parse_args()

    MI_IP = "127.0.0.1" # Dirección IP fija para este ejemplo
    MI_PUERTO = args.puerto

    # Iniciar hilo para escuchar mensajes del servidor
    hilo_escucha = threading.Thread(target=escuchar_mensajes,
    args=(MI_PUERTO,))
    hilo_escucha.start()

```

```
# Iniciar el cliente para registrarse en el servidor  
registrar_nodo(MI_IP, MI_PUERTO)
```

Código de servidor.

```
import socket  
import threading  
  
# Lista que almacenará los nodos registrados como tuplas (IP, Puerto)  
nodos_registrados = []  
  
# Función para manejar la conexión de cada cliente  
def manejar_cliente(conn, addr):  
    print(f"Conexión establecida con {addr}")  
    # Recibir la información del cliente (IP y Puerto)  
    datos_cliente = conn.recv(1024).decode('utf-8')  
    if datos_cliente:  
        ip, puerto = datos_cliente.split(':')
```

```

puerto = int(puerto)
nodo = (ip, puerto)
if nodo not in nodos_registrados:
    nodos_registrados.append(nodo) # Registrar el nodo
    print(f"Nodo registrado: {nodo}")
    conn.sendall(f"Nodo registrado: {nodo}".encode('utf-8'))
else:
    conn.sendall(f"Este nodo ya está registrado.".encode('utf-8'))
conn.close()

# Función para enviar mensajes a todos los nodos registrados
def enviar_mensaje_a_nodos():
    while True:
        # Leer mensaje desde la consola
        mensaje = input("Escribe un mensaje para enviar a todos los nodos: ")
        # Enviar mensaje a todos los nodos registrados
        if mensaje.lower() == "exit":
            print("Saliendo del servidor...")
            break
        for nodo in nodos_registrados:
            try:
                with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
                    client_socket.connect(nodo)
                    client_socket.sendall(mensaje.encode('utf-8'))
                    print(f"Mensaje enviado a {nodo}")
            except ConnectionRefusedError:
                print(f"No se pudo conectar a {nodo}")

# Función que maneja las conexiones entrantes

```

```

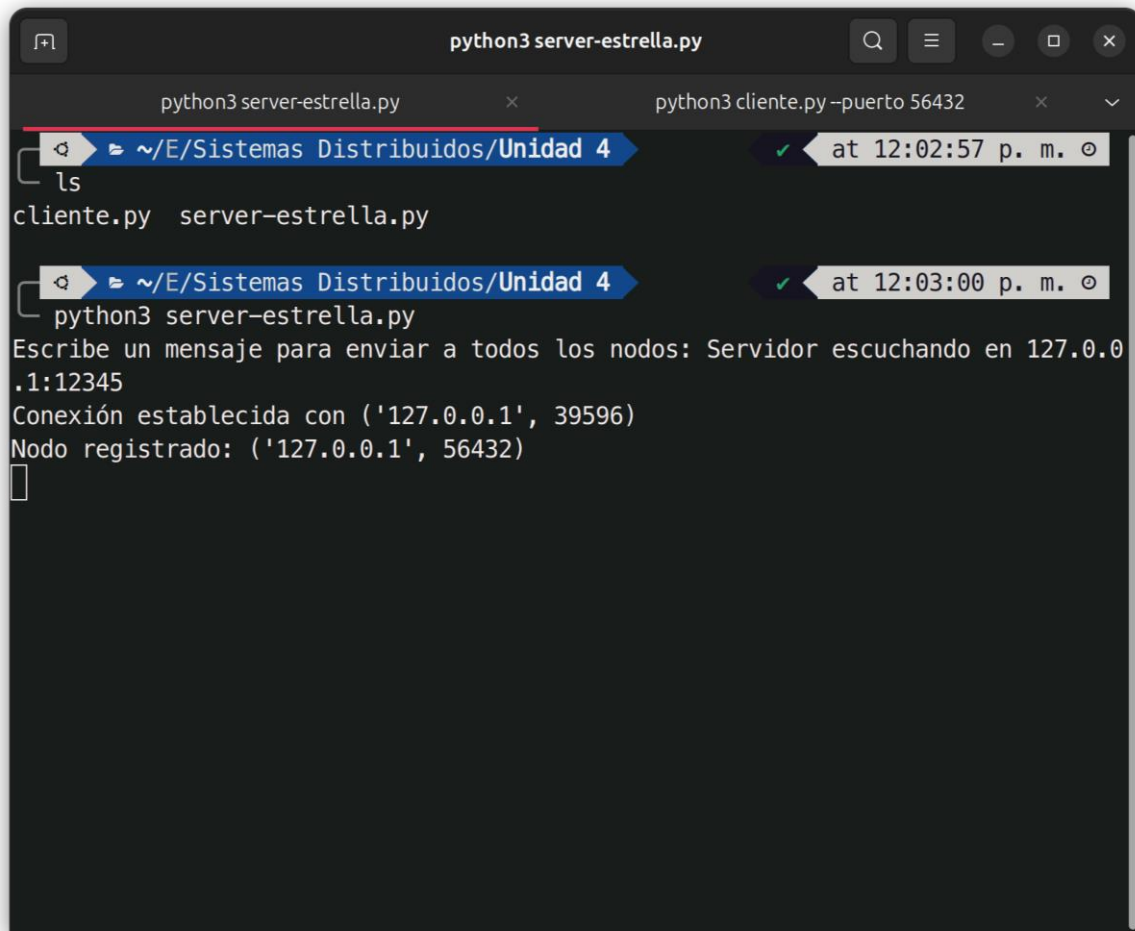
def iniciar_servidor_tcp(host, puerto):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
        server_socket.bind((host, puerto))
        server_socket.listen()
        print(f"Servidor escuchando en {host}:{puerto}")
        while True:
            conn, addr = server_socket.accept()
            hilo_cliente = threading.Thread(target=manejar_cliente, args=(conn, addr))
            hilo_cliente.start()

# Iniciar el servidor en el puerto 12345
if __name__ == "__main__":
    # Iniciar hilo para manejar la recepción de mensajes desde consola
    hilo_mensaje = threading.Thread(target=enviar_mensaje_a_nodos,
    daemon=True)
    hilo_mensaje.start()
    # Iniciar el servidor para escuchar conexiones entrantes
    iniciar_servidor_tcp("127.0.0.1", 12345)

```

Como se puede notar, se hace uso de una estructura de datos en Python, en este caso una lista vacía la cual será la encargada de almacenar los datos de ip y puerto de cada nodo registrado. Además, es importante evidenciar que se usa un ciclo for para poder recorrer esa lista y asegurarnos que no se vuelvan a registrar puertos anteriormente registrados

Evidencia de funcionamiento.



```
python3 server-estrella.py
python3 cliente.py --puerto 56432

~ / E / Sistemas Distribuidos / Unidad 4 at 12:02:57 p. m.
ls
cliente.py server-estrella.py

~ / E / Sistemas Distribuidos / Unidad 4 at 12:03:00 p. m.
python3 server-estrella.py
Escribe un mensaje para enviar a todos los nodos: Servidor escuchando en 127.0.0.1:12345
Conexión establecida con ('127.0.0.1', 39596)
Nodo registrado: ('127.0.0.1', 56432)
```

Aquí se puede ver cómo se registró en el server un nodo

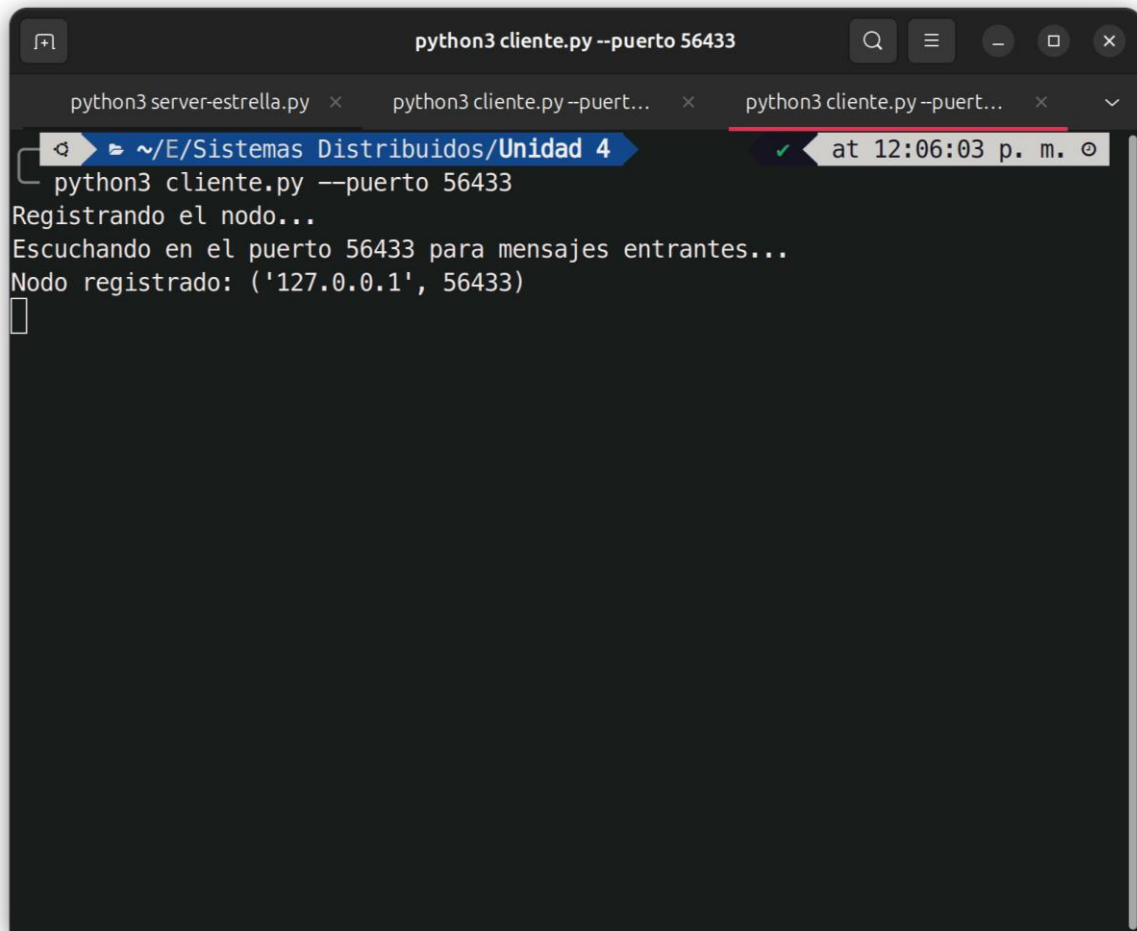
The image shows a terminal window with two tabs. The active tab is titled 'python3 cliente.py --puerto 56432'. The terminal output shows the following sequence of events:

```
python3 cliente.py --puerto 56432
ls
cliente.py  server-estrella.py

python3 cliente.py --puerto 56432
Registrando el nodo...
Escuchando en el puerto 56432 para mensajes entrantes...
Nodo registrado: ('127.0.0.1', 56432)
```

The terminal window has a dark theme and includes standard window controls (minimize, maximize, close) and a search bar. The path '~ / E / Sistemas Distribuidos / Unidad 4' is visible in the terminal's title bar.

Aquí se puede ver que el cliente con el puerto 56432 se registra en el servidor.

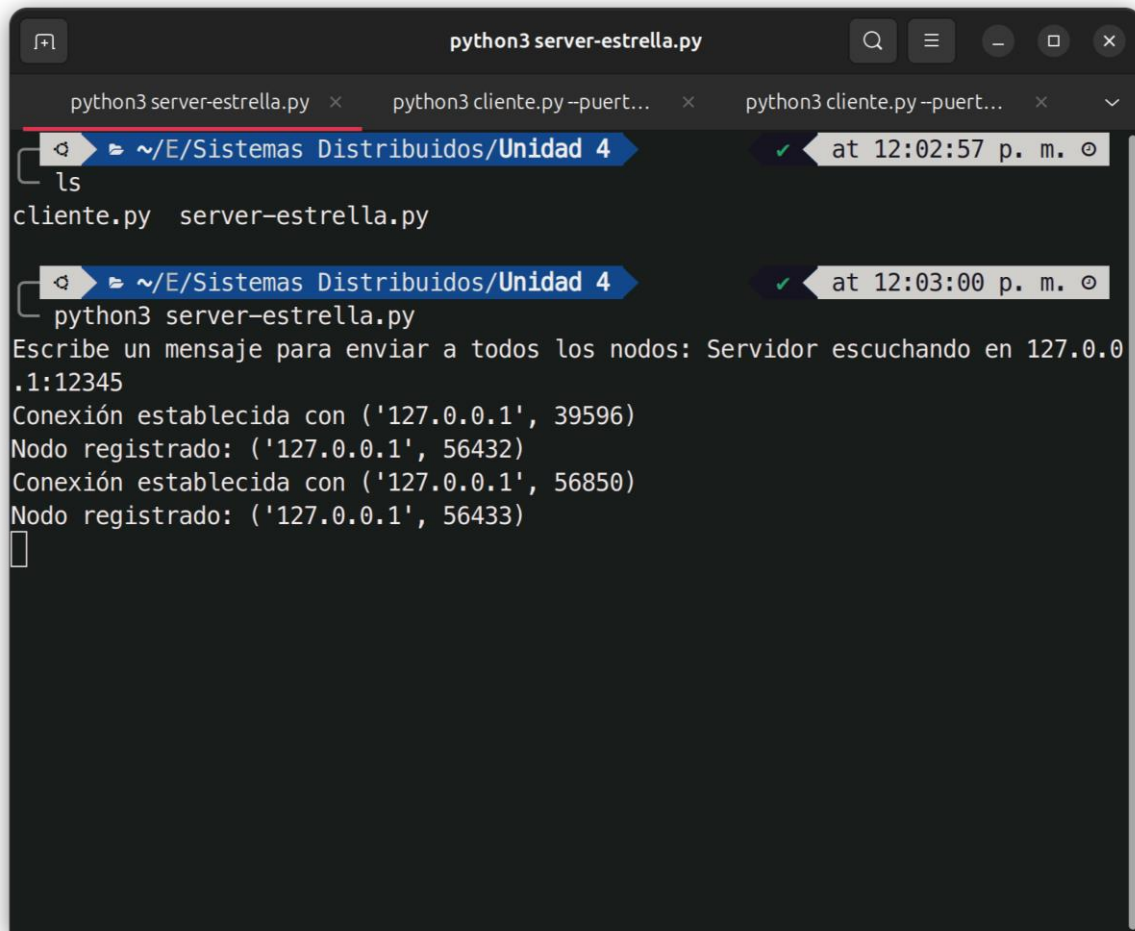


A terminal window with a dark background. The title bar at the top reads "python3 cliente.py --puerto 56433". There are three tabs open: "python3 server-estrella.py", "python3 cliente.py --puert...", and "python3 cliente.py --puert...". The active tab is the third one. The terminal shows the following output:

```
python3 cliente.py --puerto 56433
Registrando el nodo...
Escuchando en el puerto 56433 para mensajes entrantes...
Nodo registrado: ('127.0.0.1', 56433)
```

The cursor is at the end of the last line. A status bar at the bottom right shows a green checkmark and the text "at 12:06:03 p. m.".

Aqui podemos ver que se registra otro nodo, con el puerto 56433

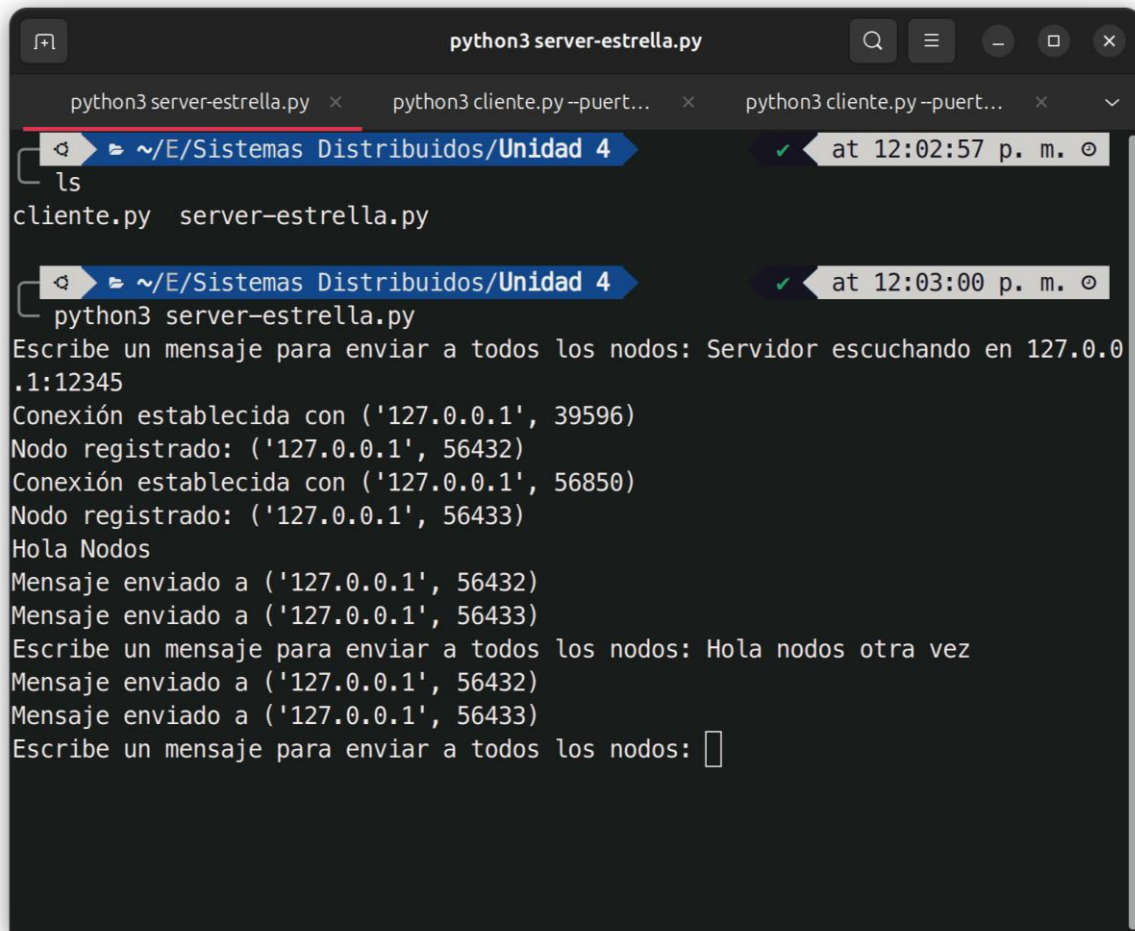


The image shows a terminal window with a dark background. The title bar at the top reads "python3 server-estrella.py". There are three tabs open: "python3 server-estrella.py", "python3 cliente.py --puert...", and "python3 cliente.py --puert...". The first tab is active. The terminal shows the following commands and output:

```
~ / E / Sistemas Distribuidos / Unidad 4
ls
cliente.py  server-estrella.py

~ / E / Sistemas Distribuidos / Unidad 4
python3 server-estrella.py
Escribe un mensaje para enviar a todos los nodos: Servidor escuchando en 127.0.0.1:12345
Conexión establecida con ('127.0.0.1', 39596)
Nodo registrado: ('127.0.0.1', 56432)
Conexión establecida con ('127.0.0.1', 56850)
Nodo registrado: ('127.0.0.1', 56433)
█
```

Nuevamente nos trasladamos a la ventana del servidor y se hace la impresión de los clientes conectados.

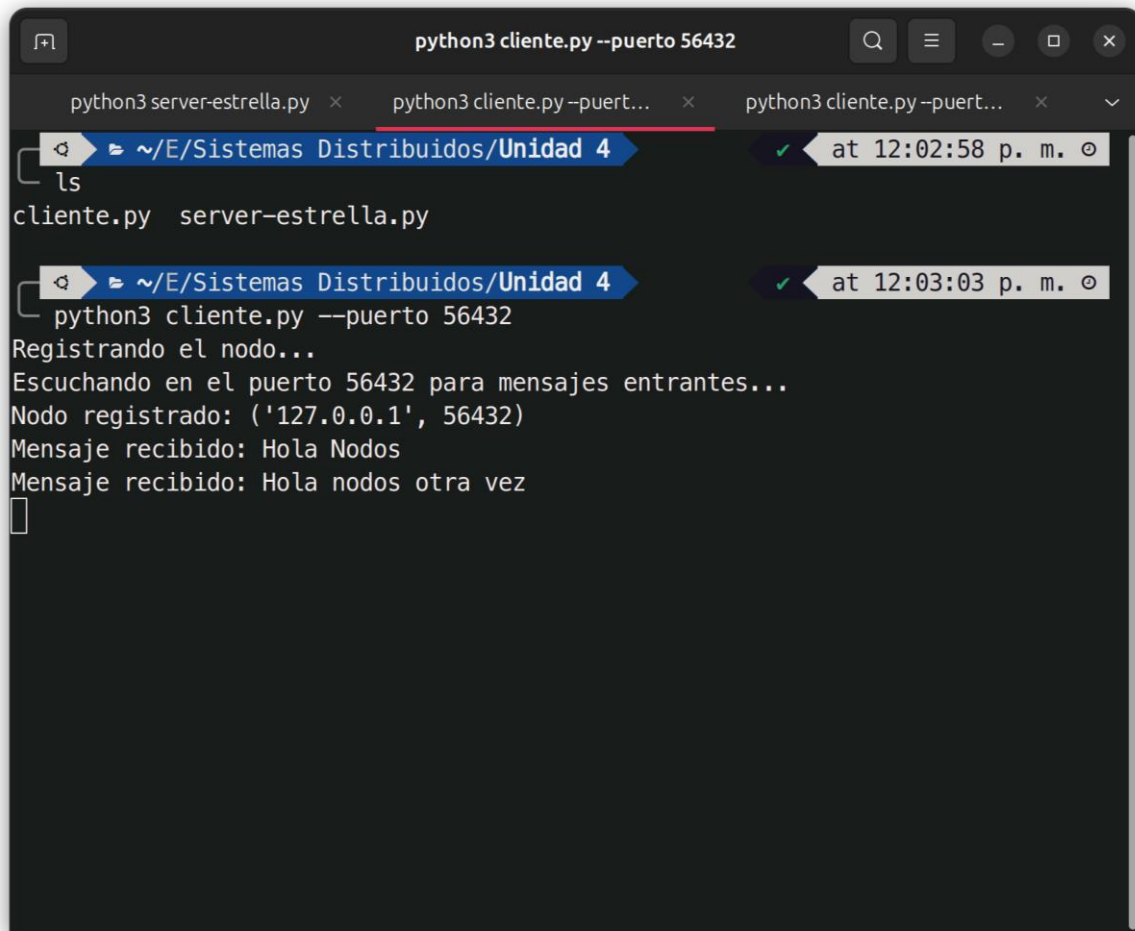


The screenshot shows a terminal window with three tabs: 'python3 server-estrella.py', 'python3 cliente.py --puert...', and 'python3 cliente.py --puert...'. The active tab is 'python3 server-estrella.py'. The terminal output shows the following sequence of events:

```
python3 server-estrella.py
ls
cliente.py  server-estrella.py

python3 server-estrella.py
Escribe un mensaje para enviar a todos los nodos: Servidor escuchando en 127.0.0.1:12345
Conexión establecida con ('127.0.0.1', 39596)
Nodo registrado: ('127.0.0.1', 56432)
Conexión establecida con ('127.0.0.1', 56850)
Nodo registrado: ('127.0.0.1', 56433)
Hola Nodos
Mensaje enviado a ('127.0.0.1', 56432)
Mensaje enviado a ('127.0.0.1', 56433)
Escribe un mensaje para enviar a todos los nodos: Hola nodos otra vez
Mensaje enviado a ('127.0.0.1', 56432)
Mensaje enviado a ('127.0.0.1', 56433)
Escribe un mensaje para enviar a todos los nodos: 
```

Se hace el envío de mensajes a todos los nodos registrados simultáneamente

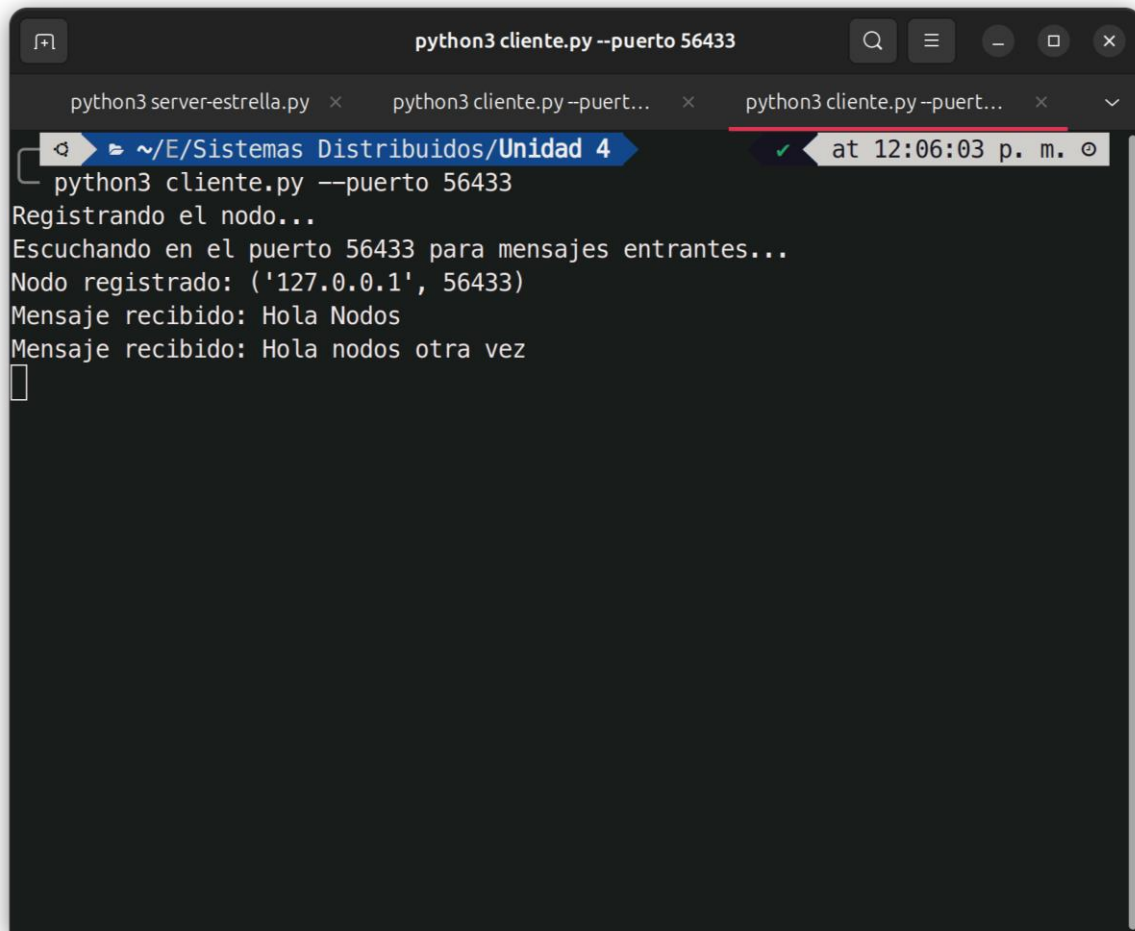


A terminal window with a dark background and light text. The title bar at the top reads "python3 cliente.py --puerto 56432". There are three tabs: "python3 server-estrella.py", "python3 cliente.py --puert...", and "python3 cliente.py --puert...". The terminal shows the following commands and output:

```
~ / E / Sistemas Distribuidos / Unidad 4
ls
cliente.py  server-estrella.py

~ / E / Sistemas Distribuidos / Unidad 4
python3 cliente.py --puerto 56432
Registrando el nodo...
Escuchando en el puerto 56432 para mensajes entrantes...
Nodo registrado: ('127.0.0.1', 56432)
Mensaje recibido: Hola Nodos
Mensaje recibido: Hola nodos otra vez
█
```

Aquí podemos ver que el cliente con puerto 56432 recibió los mensajes del servidor.



A terminal window titled "python3 cliente.py --puerto 56433" is shown. The window has three tabs: "python3 server-estrella.py", "python3 cliente.py --puert...", and "python3 cliente.py --puert...". The active tab is the third one. The terminal output shows the following sequence of events:

```
python3 cliente.py --puerto 56433
Registrando el nodo...
Escuchando en el puerto 56433 para mensajes entrantes...
Nodo registrado: ('127.0.0.1', 56433)
Mensaje recibido: Hola Nodos
Mensaje recibido: Hola nodos otra vez
█
```

The terminal window also shows a status bar at the bottom with a green checkmark and the text "at 12:06:03 p. m.".

Aquí podemos ver que el cliente con puerto 56433 recibió los mensajes del servidor.

Conclusiones:

Como se pudo notar, la práctica cumplió con los criterios necesarios para su evaluación, un gran aprendizaje que se obtiene es que siempre que debamos usar puertos hay que cerrarlos con precaución pues personalmente me sucedió que los puertos antes de poder finalizar esta práctica quedaron bloqueados los puertos 56432, y 56433 por lo que fue complicado, pero se logró desbloquearlos activando y desactivando el firewall de la máquina Linux.