PROTOCOLOS DE INTERNET

1 CUATRIMESTRE DE 2024

PRÁCTICA

Trabajo Práctico N.º 1 Parte B

Comisión: FN

Profesor/a: Javier Adolfo Ouret

N°	Nombre y apellido	Carrera	DNI	Email
1	Christian Balderrama	Ing. Informática	42118900	christianbalderrama@uca.edu.ar
2	Fiorella Insfran Sanabria	Ing. Informática	96142187	fiorellainsfran@uca.edu.ar
3	Pablo Joaquin Cardozo	Ing. Informática	45178142	cardozopabloj@uca.edu.ar
4	Pablo Joaquin Margewka	Ing. Informática	37754332	pablomargewka@uca.edu.ar
5	Fabián Leonardo De Simone	Ing. Informática	39433563	fdesimone96@uca.edu.ar

Para la parte C de este trabajo práctico, implementamos un modelo cliente-servidor utilizando Python. El servidor debe mostrar las direcciones IP y puerto de cada uno de los clientes conectados y debe enviar al cliente el día y la hora de la conexión y el tiempo que se encuentra conectado con el cliente. Además implementamos que la conexión se pueda finalizar tanto del lado del cliente, con el comando “salir”, como del lado del servidor, mediante una interrupción dada por “CTRL+C”.

Este servidor lo hicimos con concurrencia real y con concurrencia aparente y comparamos los resultados obtenidos en ambos casos.

Adjuntamos el código del cliente, servidor con concurrencia real y aparente, en el siguiente GIthub para una mayor comodidad al momento de probar los códigos: https://github.com/pablo29m/protocolos/tree/main

A continuación presentamos el código del cliente.

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_dir = ('localhost', 6667)
print('Conectando a %s puerto %s' % server_dir)
sock.connect(server_dir)

try:
  # Recibir mensaje de bienvenida del servidor
  print(sock.recv(1024).decode('utf-8'))

  while True:
    # Enviar mensaje al servidor
    mensaje = input("Mensaje para el servidor: ")
    sock.sendall(mensaje.encode('utf-8'))
```

```
        # Si el mensaje es "salir", cerrar la conexión y salir del bucle
        if mensaje.lower() == "salir":
            print("Cerrando conexión con el servidor...")
            break

        # Recibir respuesta del servidor
        try:
            data = sock.recv(1024)
            if not data:
                print("La conexión con el servidor se cerró.")
                break
            print(data.decode('utf-8'))
        except ConnectionResetError:
            print("La conexión con el servidor se cerró inesperadamente.")
            break

finally:
    print('Cerrando socket')
    sock.close()
```

Una vez que se conecte con el servidor, el usuario puede enviar un mensaje hacia el servidor y el servidor va a responder indicando que recibió el mensaje y devolviendo el día, la hora y el tiempo de conexión

A continuación presentamos el código del servidor aparente

home > pablo > tpc > nuevoaparente.py

```python
import socket
import select
import signal
import sys
from datetime import datetime

# Diccionario con los nombres de los meses en español
meses = {
    1: "enero", 2: "febrero", 3: "marzo", 4: "abril", 5: "mayo", 6: "junio",
    7: "julio", 8: "agosto", 9: "septiembre", 10: "octubre", 11: "noviembre", 12: "diciembre"
}

# Lista para mantener las conexiones activas
conexiones_activas = []
sock = None
manejando_interrupcion = False

def manejar_interrupcion(signal, frame):
    global manejando_interrupcion
    if manejando_interrupcion:
        return
    manejando_interrupcion = True

    print("\nSe ha detectado una interrupción. ¿Qué acción deseas tomar?")
    print("1. Cerrar solo el socket del servidor.")
    print("2. Cerrar todas las conexiones activas y luego cerrar el socket del servidor.")
    print("3. Salir sin cerrar nada.")

    opcion = input("Selecciona una opción: ")

    if opcion == "1":
        cerrar_socket()
    elif opcion == "2":
        cerrar_conexiones()
        cerrar_socket()
        print("Saliendo del programa.")
        sys.exit(0)
    else:
        print("Saliendo sin cerrar nada.")
        sys.exit(0)
    manejando_interrupcion = False

def cerrar_socket():
    global sock
    if sock:
        try:
            sock.close()
            print("Socket del servidor cerrado.")
            sys.exit(0)  # Salir del programa después de cerrar el socket del servidor
        except Exception as e:
            print("Error al cerrar el socket del servidor:", e)

def cerrar_conexiones():
    global conexiones_activas
    for conn in conexiones_activas:
        try:
            conn.close()
        except Exception as e:
            print("Error al cerrar conexión:", e)
    print("Todas las conexiones activas han sido cerradas.")

def proceso_hijo(conn, addr):
    global conexiones_activas, cerrando
    # Añadir la conexión activa a la lista
    conexiones_activas.append(conn)
    # Momento en que se establece la conexión
    inicio_conexion = datetime.now()

    print('Conexión establecida con IP: %s y Puerto: %s' % (addr[0], addr[1]))

    # Obtener la fecha y hora actual de conexión
    month_name = meses[inicio_conexion.month]  # Nombre del mes en español
    formatted_time = inicio_conexion.strftime("%H:%M")  # Formato de la hora
    friendly_date_time = f"{inicio_conexion.day} de {month_name} a las {formatted_time} hs"

    # Send the welcome message along with the date and time of connection
    mensaje_bienvenida = f"Servidor: Conexión establecida el {friendly_date_time}. Puedes enviar mensajes al servidor.\n"
    conn.send(mensaje_bienvenida.encode('UTF-8'))

    cliente_desconectado = False
```

```
23
24      print("\nSe ha detectado una interrupción. ¿Qué acción deseas tomar?")
25      print("1. Cerrar solo el socket del servidor.")
26      print("2. Cerrar todas las conexiones activas y luego cerrar el socket del servidor.")
27      print("3. Salir sin cerrar nada.")
28
29      opcion = input("Selecciona una opción: ")
30
31      if opcion == "1":
32          cerrar_socket()
33      elif opcion == "2":
34          cerrar_conexiones()
35          cerrar_socket()
36          print("Saliendo del programa.")
37          sys.exit(0)
38      else:
39          print("Saliendo sin cerrar nada.")
40          sys.exit(0)
41      manejando_interrupcion = False
42
43  def cerrar_socket():
44      global sock
45      if sock:
46          try:
47              sock.close()
48              print("Socket del servidor cerrado.")
49              sys.exit(0)  # Salir del programa después de cerrar el socket del servidor
50          except Exception as e:
51              print("Error al cerrar el socket del servidor:", e)
52
53  def cerrar_conexiones():
54      global conexiones_activas
55      for conn in conexiones_activas:
56          try:
57              conn.close()
58          except Exception as e:
59              print("Error al cerrar conexión:", e)
60      print("Todas las conexiones activas han sido cerradas.")
61
```

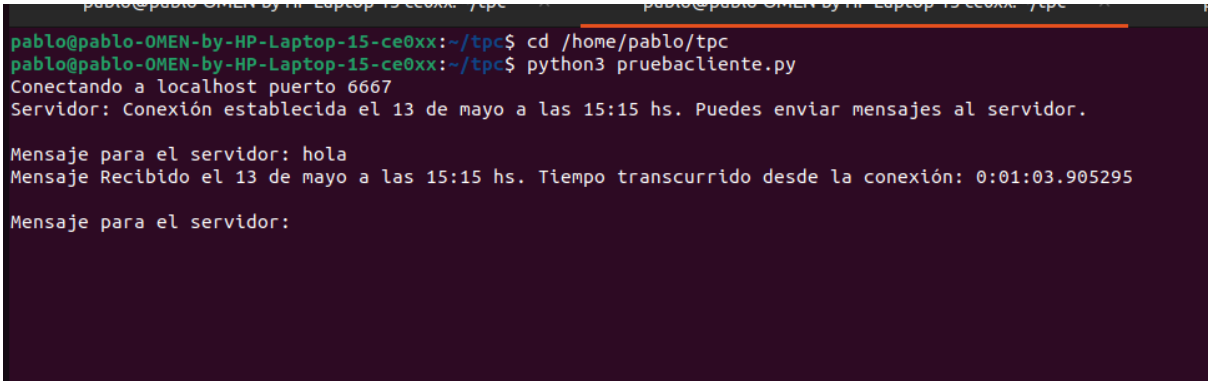
```python
def proceso_hijo(conn, addr):
    global conexiones_activas, cerrando
    # Añadir la conexión activa a la lista
    conexiones_activas.append(conn)
    # Momento en que se establece la conexión
    inicio_conexion = datetime.now()

    print('Conexión establecida con IP: %s y Puerto: %s' % (addr[0], addr[1]))

    # Obtener la fecha y hora actual de conexión
    month_name = meses[inicio_conexion.month]  # Nombre del mes en español
    formatted_time = inicio_conexion.strftime("%H:%M")  # Formato de la hora
    friendly_date_time = f"{inicio_conexion.day} de {month_name} a las {formatted_time} hs"

    # Send the welcome message along with the date and time of connection
    mensaje_bienvenida = f"Servidor: Conexión establecida el {friendly_date_time}. Puedes enviar mensajes al servidor.\n"
    conn.send(mensaje_bienvenida.encode('UTF-8'))

    cliente_desconectado = False

    while not cliente_desconectado:
        lista_para_leer, _, _ = select.select([conn], [], [], 0.1)
        for s in lista_para_leer:
            data = s.recv(1024).decode('utf-8')
            if not data:
                print('Cliente {}:{} se desconectó.'.format(addr[0], addr[1]))
                cliente_desconectado = True
                break
            if data.lower() == 'salir':
                print('Cliente {}:{} ha cerrado la conexión.'.format(addr[0], addr[1]))
                cliente_desconectado = True
                break  # Exit the loop and terminate the child process

            if manejando_interrupcion==False:
                print('Mensaje recibido de {}:{}: {}'.format(addr[0], addr[1], data))

            # Calculate the amount of time elapsed since the start of the connection
            tiempo_transcurrido = datetime.now() - inicio_conexion
            tiempo_transcurrido_formateado = str(tiempo_transcurrido)
```

```python
            # Send the received message along with the amount of time elapsed
            mensaje_respuesta = f"Mensaje Recibido el {friendly_date_time}. Tiempo transcurrido desde la conexión: {tiempo_transcurrido_formateado}\n"
            try:
                conn.send(mensaje_respuesta.encode('UTF-8'))
            except OSError:
                cliente_desconectado = True
                break

    # Remove the active connection from the list
    try:
        conexiones_activas.remove(conn)
    except ValueError:
        pass
    conn.close()

# servidor concurrente
host = "127.0.0.1"
port = 6667

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
print("Socket creado")
sock.bind((host, port))
print("Enlace del socket completado")
sock.listen(5)
print("Socket en modo escucha")

# Manejar la señal de interrupción (Control+C)
signal.signal(signal.SIGINT, manejar_interrupcion)

while True:
    conn, addr = sock.accept()
    proceso_hijo(conn, addr)
```

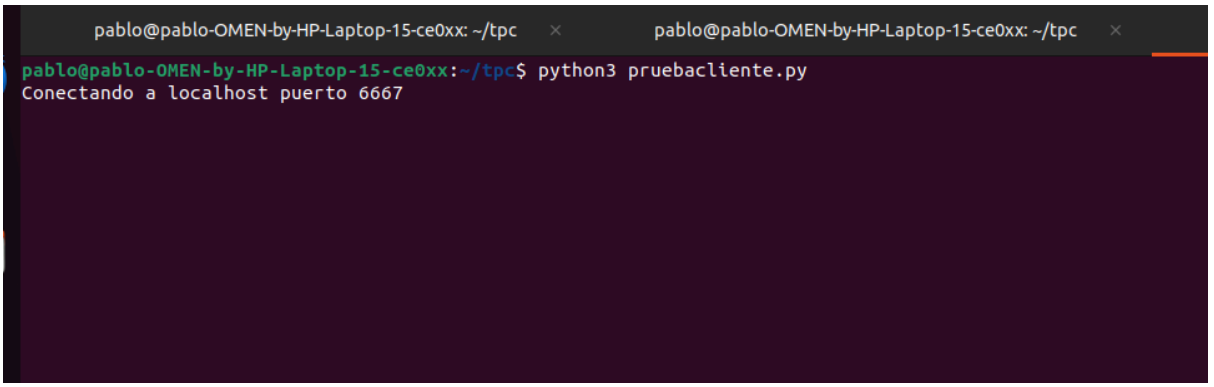
Para este caso lo que hicimos fue abrir 4 terminales. En 1 ejecutamos el servidor y en el resto de las terminales ejecutamos los clientes. Lo que sucede es que el servidor se va a conectar con el primer cliente, mientras que los demás están en cola de espera.

El servidor se conecta con el primer cliente

Desde el primer cliente enviamos un mensaje al servidor y el servidor nos contesta

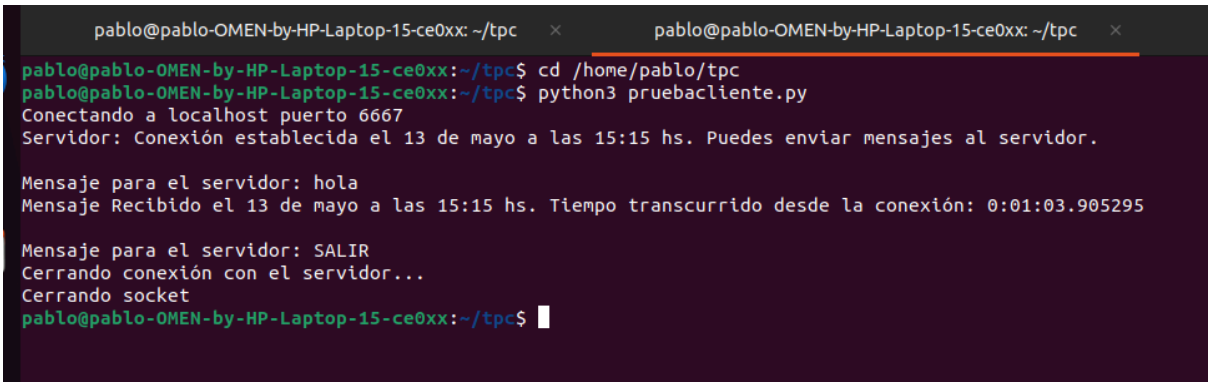


El resto de los clientes permanece en este estado, esperando para conectarse con el servidor

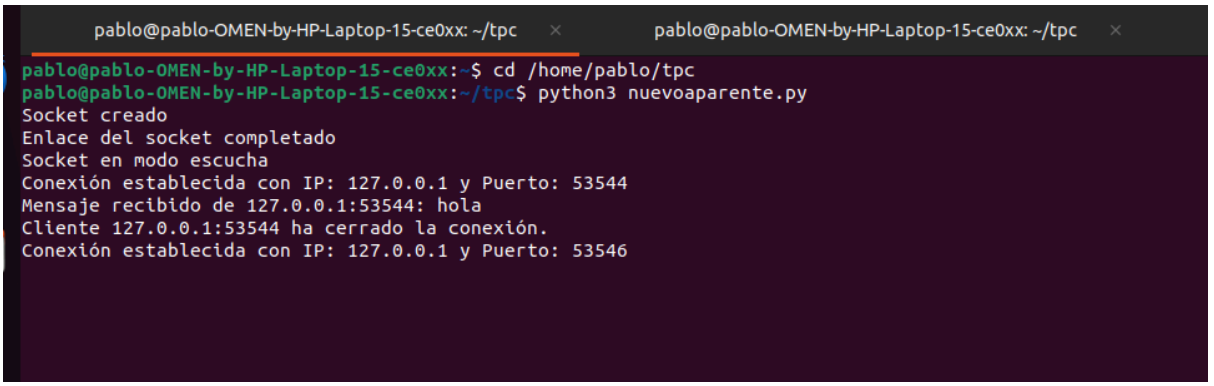


Luego, cuando desde el primer cliente escribimos la palabra “SALIR”, se cierra la conexión con el servidor, lo que habilita a otro cliente a poder conectarse con el servidor

Desde el lado del cliente



Desde el lado del servidor



Se puede ver como el servidor informa que se cerró la conexión con el cliente desde el puerto 53544 y como se establece una nueva conexión con el puerto 53546.

Podemos confirmar que el servidor se comporta de forma aparente, ya que utiliza un único hilo para atender todas las solicitudes de los clientes.

A continuación, mostraremos el código de un servidor con concurrencia real y observaremos las diferencias con respecto al aparente.

e > pablo > tpc > real.py

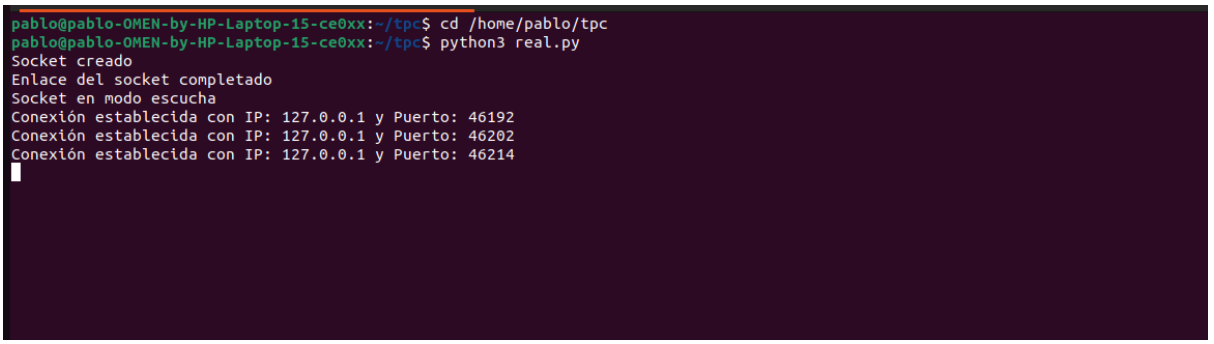
```python
import socket
import threading
import signal
import sys
from datetime import datetime

# Diccionario con los nombres de los meses en español
meses = {
    1: "enero", 2: "febrero", 3: "marzo", 4: "abril", 5: "mayo", 6: "junio",
    7: "julio", 8: "agosto", 9: "septiembre", 10: "octubre", 11: "noviembre", 12: "diciembre"
}

# Lista para mantener las conexiones activas
conexiones_activas = []
sock = None
manejando_interrupcion = False

def manejar_interrupcion(signal, frame):
    global manejando_interrupcion
    if manejando_interrupcion:
        return
    manejando_interrupcion = True

    print("\nSe ha detectado una interrupción. ¿Qué acción deseas tomar?")
    print("1. Cerrar solo el socket del servidor.")
    print("2. Cerrar todas las conexiones activas y luego cerrar el socket del servidor.")
    print("3. Salir sin cerrar nada.")

    opcion = input("Selecciona una opción: ")

    if opcion == "1":
        cerrar_socket()
    elif opcion == "2":
        cerrar_conexiones()
        cerrar_socket()
        print("Saliendo del programa.")
        sys.exit(0)
    else:
        print("Saliendo sin cerrar nada.")
        sys.exit(0)
    manejando_interrupcion = False
```

```python
def cerrar_socket():
    global sock
    if sock:
        try:
            sock.close()
            print("Socket del servidor cerrado.")
            sys.exit(0)  # Salir del programa después de cerrar el socket del servidor
        except Exception as e:
            print("Error al cerrar el socket del servidor:", e)

def cerrar_conexiones():
    global conexiones_activas
    for conn in conexiones_activas:
        try:
            conn.close()
        except Exception as e:
            print("Error al cerrar conexión:", e)
    print("Todas las conexiones activas han sido cerradas.")

def proceso_hijo(conn, addr):
    global conexiones_activas, cerrando
    # Añadir la conexión activa a la lista
    conexiones_activas.append(conn)
    # Momento en que se establece la conexión
    inicio_conexion = datetime.now()

    print('Conexión establecida con IP: %s y Puerto: %s' % (addr[0], addr[1]))

    # Obtener la fecha y hora actual de conexión
    month_name = meses[inicio_conexion.month]  # Nombre del mes en español
    formatted_time = inicio_conexion.strftime("%H:%M")  # Formato de la hora
    friendly_date_time = f"{inicio_conexion.day} de {month_name} a las {formatted_time} hs"

    # Enviar el mensaje de bienvenida junto con la fecha y hora de conexión
    mensaje_bienvenida = f"Servidor: Conexión establecida el {friendly_date_time}. Puedes enviar mensajes al servidor.\n"
    conn.send(mensaje_bienvenida.encode('UTF-8'))
```

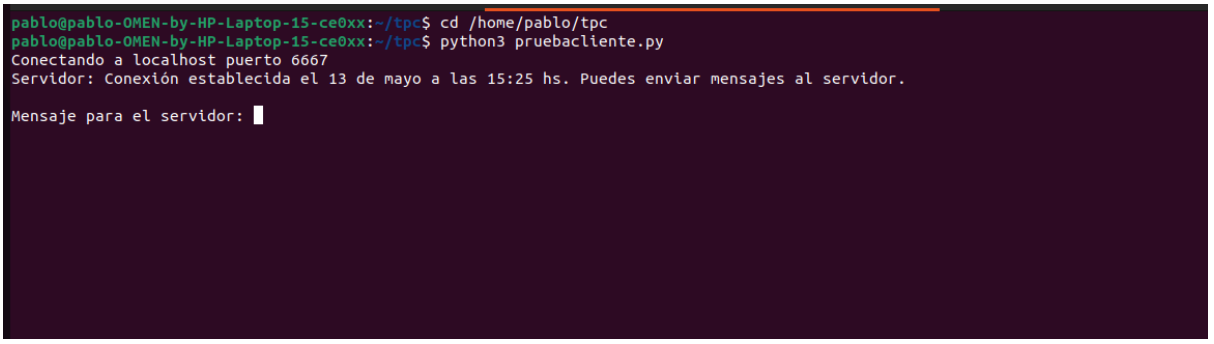
```python
    while True:
        data = conn.recv(1024).decode('utf-8')
        if not data:
            print('Cliente {}:{} se desconectó.'.format(addr[0], addr[1]))
            break
        if manejando_interrupcion==False:
            print('Mensaje recibido de {}:{}: {}'.format(addr[0], addr[1], data))


        # Calcular la cantidad de tiempo transcurrido desde el inicio de la conexión
        tiempo_transcurrido = datetime.now() - inicio_conexion
        tiempo_transcurrido_formateado = str(tiempo_transcurrido)

        # Enviar el mensaje recibido junto con la cantidad de tiempo transcurrido
        mensaje_respuesta = f"Mensaje Recibido el {friendly_date_time}. Tiempo transcurrido desde la conexión: {tiempo_transcurrido_formateado}\n"
        try:
            conn.send(mensaje_respuesta.encode('UTF-8'))
        except OSError:
            break

    # Eliminar la conexión activa de la lista
    try:
        conexiones_activas.remove(conn)
    except ValueError:
        pass
    conn.close()

# servidor concurrente
host = "127.0.0.1"
port = 6667

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
print("Socket creado")
sock.bind((host, port))
print("Enlace del socket completado")
sock.listen(5)
print("Socket en modo escucha")

# Manejar la señal de interrupción (Control+C)
signal.signal(signal.SIGINT, manejar_interrupcion)

while True:
    conn, addr = sock.accept()
    threading.Thread(target=proceso_hijo, args=(conn, addr)).start()
```

Para este caso abrimos 4 terminales. En una de ellas ejecutamos al cliente y en el resto de las terminales a los clientes. Veremos como el servidor se va a conectar con cada uno de los clientes, creando un proceso hijo por cada cliente, a través del uso del fork

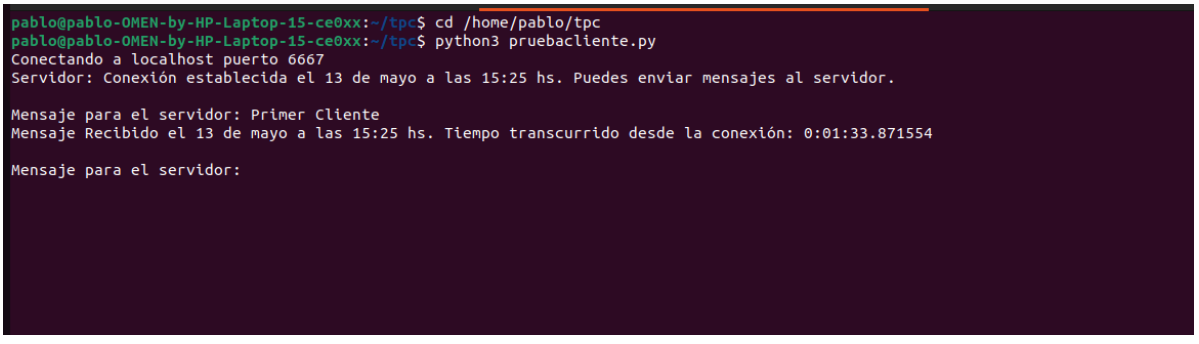
En este caso, podemos observar que el servidor se pudo conectar con los 3 clientes en simultáneo.



Cada cliente mostrará la siguiente salida

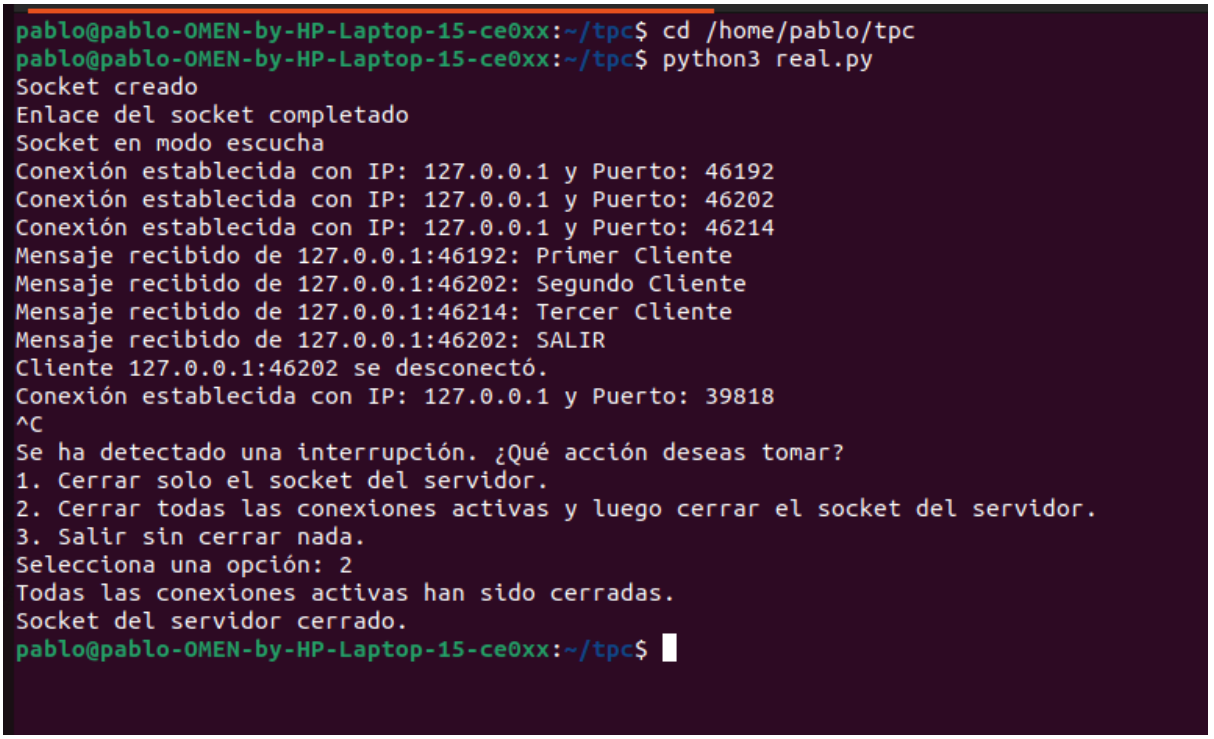


Cada cliente podrá enviar un mensaje al servidor y el servidor respondera

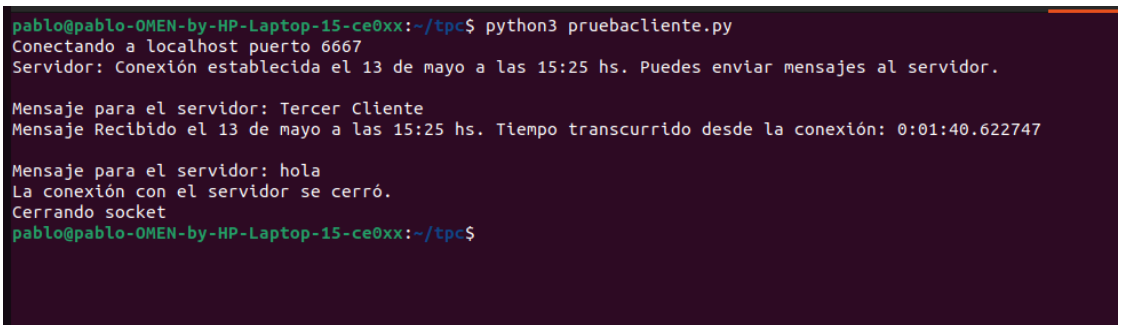


Ya que se genera un proceso hijo por cada conexión, el cliente no debe esperar a que el servidor termine de ejecutar las tareas solicitadas por el primer cliente, sino que el servidor es capaz de de responder las solicitudes en simultáneo.

Por otro lado, tanto para el servidor con concurrencia real como aparente, realizamos una función para permitir que el servidor pueda cerrar todas las conexiones activas.
SImplemente presionamos “CTRL+C” y elegimos la opción 2 para finalizar todas las conexiones



Desde el lado del cliente estaba activado un input para enviar un mensaje, pero al detectar que el cierre de la conexión, no manda ese último mensaje e informa que se cerró la conexión con el servidor y se cerró el socket.



A través de este trabajo práctico, hemos logrado obtener una mayor comprensión acerca de la diferencia entre un servidor con concurrencia real y uno con concurrencia aparente. Además hemos aprendido métodos para finalizar las conexiones tanto desde el cliente como el servidor.