

PROTOCOLOS DE INTERNET
1 CUATRIMESTRE DE 2024
PRÁCTICA

Trabajo Práctico N.º 1 Parte B

Comisión: FN

Profesor/a: Javier Adolfo Ouret

Nº	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

```
1  import socket
2  import select
3  import signal
4  import sys
5  from datetime import datetime
6
7  # Diccionario con los nombres de los meses en español
8  meses = {
9      1: "enero", 2: "febrero", 3: "marzo", 4: "abril", 5: "mayo", 6: "junio",
10     7: "julio", 8: "agosto", 9: "septiembre", 10: "octubre", 11: "noviembre", 12: "diciembre"
11 }
12
13 # Lista para mantener las conexiones activas
14 conexiones_activas = []
15 sock = None
16 manejando_interrupcion = False
17
18 def manejar_interrupcion(signal, frame):
19     global manejando_interrupcion
20     if manejando_interrupcion:
21         return
22     manejando_interrupcion = True
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
62 def proceso_hijo(conn, addr):
63     global conexiones_activas, cerrando
64     # Añadir la conexión activa a la lista
65     conexiones_activas.append(conn)
66     # Momento en que se establece la conexión
67     inicio_conexion = datetime.now()
68
69     print('Conexión establecida con IP: %s y Puerto: %s' % (addr[0], addr[1]))
70
71     # Obtener la fecha y hora actual de conexión
72     month_name = meses[inicio_conexion.month] # Nombre del mes en español
73     formatted_time = inicio_conexion.strftime("%H:%M") # Formato de la hora
74     friendly_date_time = f"{inicio_conexion.day} de {month_name} a las {formatted_time} hs"
75
76     # Send the welcome message along with the date and time of connection
77     mensaje_bienvenida = f"Servidor: Conexión establecida el {friendly_date_time}. Puedes enviar mensajes al servidor.\n"
78     conn.send(mensaje_bienvenida.encode('UTF-8'))
79
80     cliente_desconectado = False
81
```

```

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

```

```

12 def proceso_hijo(conn, addr):
13     global conexiones_activas, cerrando
14     # Añadir la conexión activa a la lista
15     conexiones_activas.append(conn)
16     # Momento en que se establece la conexión
17     inicio_conexion = datetime.now()
18
19     print('Conexión establecida con IP: %s y Puerto: %s' % (addr[0], addr[1]))
20
21     # Obtener la fecha y hora actual de conexión
22     month_name = meses[inicio_conexion.month] # Nombre del mes en español
23     formatted_time = inicio_conexion.strftime("%H:%M") # Formato de la hora
24     friendly_date_time = f"{inicio_conexion.day} de {month_name} a las {formatted_time} hs"
25
26     # Send the welcome message along with the date and time of connection
27     mensaje_bienvenida = f"Servidor: Conexión establecida el {friendly_date_time}. Puedes enviar mensajes al servidor.\n"
28     conn.send(mensaje_bienvenida.encode('UTF-8'))
29
30     cliente_desconectado = False
31
32     while not cliente_desconectado:
33         lista_para_leer, _, _ = select.select([conn], [], [], 0.1)
34         for s in lista_para_leer:
35             data = s.recv(1024).decode('utf-8')
36             if not data:
37                 print('Cliente {}:{} se desconectó.'.format(addr[0], addr[1]))
38                 cliente_desconectado = True
39                 break
40             if data.lower() == 'salir':
41                 print('Cliente {}:{} ha cerrado la conexión.'.format(addr[0], addr[1]))
42                 cliente_desconectado = True
43                 break # Exit the loop and terminate the child process
44
45             if manejando_interrupcion==False:
46                 print('Mensaje recibido de {}:{}: {}'.format(addr[0], addr[1], data))
47
48             # Calculate the amount of time elapsed since the start of the connection
49             tiempo_transcurrido = datetime.now() - inicio_conexion
50             tiempo_transcurrido_formateado = str(tiempo_transcurrido)
51
52             # Send the received message along with the amount of time elapsed
53             mensaje_respuesta = f"Mensaje recibido: {data} {friendly_date_time}. Tiempo transcurrido desde la conexión: {tiempo_transcurrido_formateado}\n"
54             try:
55                 conn.send(mensaje_respuesta.encode('UTF-8'))
56             except OSError:
57                 cliente_desconectado = True
58                 break
59
60     # Remove the active connection from the list
61     try:
62         conexiones_activas.remove(conn)
63     except ValueError:
64         pass
65     conn.close()
66
67 # servidor concurrente
68 host = "127.0.0.1"
69 port = 6667
70
71 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
72 sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
73 print("Socket creado")
74 sock.bind((host, port))
75 print("Enlace del socket completado")
76 sock.listen(5)
77 print("Socket en modo escucha")
78
79 # Manejar la señal de interrupción (Control+C)
80 signal.signal(signal.SIGINT, manejar_interrupcion)
81
82 while True:
83     conn, addr = sock.accept()
84     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

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc × pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 nuevoaparente.py
Socket creado
Enlace del socket completado
Socket en modo escucha
Conexión establecida con IP: 127.0.0.1 y Puerto: 53544
█
```

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

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc × pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 pruebacliente.py
Conectando a localhost puerto 6667
Servidor: Conexión establecida el 13 de mayo a las 15:15 hs. Puedes enviar mensajes al servidor.

Mensaje para el servidor: hola
Mensaje Recibido el 13 de mayo a las 15:15 hs. Tiempo transcurrido desde la conexión: 0:01:03.905295

Mensaje para el servidor:
```

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

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc × pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc ×
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 pruebacliente.py
Conectando a localhost puerto 6667
```

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

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc x pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc x
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 pruebacliente.py
Conectando a localhost puerto 6667
Servidor: Conexión establecida el 13 de mayo a las 15:15 hs. Puedes enviar mensajes al servidor.

Mensaje para el servidor: hola
Mensaje Recibido el 13 de mayo a las 15:15 hs. Tiempo transcurrido desde la conexión: 0:01:03.905295

Mensaje para el servidor: SALIR
Cerrando conexión con el servidor...
Cerrando socket
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$
```

Desde el lado del servidor

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc x pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx: ~/tpc x
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 nuevoaparente.py
Socket creado
Enlace del socket completado
Socket en modo escucha
Conexión establecida con IP: 127.0.0.1 y Puerto: 53544
Mensaje recibido de 127.0.0.1:53544: hola
Cliente 127.0.0.1:53544 ha cerrado la conexión.
Conexión establecida con IP: 127.0.0.1 y Puerto: 53546
```

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
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
maneja_interrupcion = False

def manejar_interrupcion(signal, frame):
    global maneja_interrupcion
    if maneja_interrupcion:
        return
    maneja_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)
    maneja_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"

    # 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'))

```

```

80 while True:
81     data = conn.recv(1024).decode('utf-8')
82     if not data:
83         print('Cliente {}: {} se desconectó.'.format(addr[0], addr[1]))
84         break
85     if manejando_interrupcion==False:
86         print('Mensaje recibido de {}: {}: {}'.format(addr[0], addr[1], data))
87
88     # Calcular la cantidad de tiempo transcurrido desde el inicio de la conexión
89     tiempo_transcurrido = datetime.now() - inicio_conexion
90     tiempo_transcurrido_formateado = str(tiempo_transcurrido)
91
92     # Enviar el mensaje recibido junto con la cantidad de tiempo transcurrido
93     mensaje_respuesta = f"Mensaje Recibido el {friendly_date_time}. Tiempo transcurrido desde la conexión: {tiempo_transcurrido_formateado}\n"
94     try:
95         conn.send(mensaje_respuesta.encode('UTF-8'))
96     except OSError:
97         break
98
99     # Eliminar la conexión activa de la lista
100    try:
101        conexiones_activas.remove(conn)
102    except ValueError:
103        pass
104    conn.close()
105
106 # servidor concurrente
107 host = "127.0.0.1"
108 port = 6667
109
110 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
111 sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
112 print("Socket creado")
113 sock.bind((host, port))
114 print("Enlace del socket completado")
115 sock.listen(5)
116 print("Socket en modo escucha")
117
118 # Manejar la señal de interrupción (Control+C)
119 signal.signal(signal.SIGINT, manejar_interrupcion)
120
121 while True:
122     conn, addr = sock.accept()
123     threading.Thread(target=proceso_hijo, args=(conn, addr)).start()
124

```

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.

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 real.py
Socket creado
Enlace del socket completado
Socket en modo escucha
Conexión establecida con IP: 127.0.0.1 y Puerto: 46192
Conexión establecida con IP: 127.0.0.1 y Puerto: 46202
Conexión establecida con IP: 127.0.0.1 y Puerto: 46214
█
```

Cada cliente mostrará la siguiente salida

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 pruebacliente.py
Conectando a localhost puerto 6667
Servidor: Conexión establecida el 13 de mayo a las 15:25 hs. Puedes enviar mensajes al servidor.

Mensaje para el servidor: █
```

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

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 pruebacliente.py
Conectando a localhost puerto 6667
Servidor: Conexión establecida el 13 de mayo a las 15:25 hs. Puedes enviar mensajes al servidor.

Mensaje para el servidor: Primer Cliente
Mensaje Recibido el 13 de mayo a las 15:25 hs. Tiempo transcurrido desde la conexión: 0:01:33.871554

Mensaje para el servidor:
```

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 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

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ cd /home/pablo/tpc
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 real.py
Socket creado
Enlace del socket completado
Socket en modo escucha
Conexión establecida con IP: 127.0.0.1 y Puerto: 46192
Conexión establecida con IP: 127.0.0.1 y Puerto: 46202
Conexión establecida con IP: 127.0.0.1 y Puerto: 46214
Mensaje recibido de 127.0.0.1:46192: Primer Cliente
Mensaje recibido de 127.0.0.1:46202: Segundo Cliente
Mensaje recibido de 127.0.0.1:46214: Tercer Cliente
Mensaje recibido de 127.0.0.1:46202: SALIR
Cliente 127.0.0.1:46202 se desconectó.
Conexión establecida con IP: 127.0.0.1 y Puerto: 39818
^C
Se ha detectado una interrupción. ¿Qué acción deseas tomar?
1. Cerrar solo el socket del servidor.
2. Cerrar todas las conexiones activas y luego cerrar el socket del servidor.
3. Salir sin cerrar nada.
Selecciona una opción: 2
Todas las conexiones activas han sido cerradas.
Socket del servidor cerrado.
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$
```

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.

```
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$ python3 pruebacliente.py
Conectando a localhost puerto 6667
Servidor: Conexión establecida el 13 de mayo a las 15:25 hs. Puedes enviar mensajes al servidor.

Mensaje para el servidor: Tercer Cliente
Mensaje Recibido el 13 de mayo a las 15:25 hs. Tiempo transcurrido desde la conexión: 0:01:40.622747

Mensaje para el servidor: hola
La conexión con el servidor se cerró.
Cerrando socket
pablo@pablo-OMEN-by-HP-Laptop-15-ce0xx:~/tpc$
```

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.