TP 1 - PARTE C

Escribir una aplicación cliente servidor que muestre las direcciones y puertos de todos los clientes conectados del lado del servidor y devuelva a cada cliente el día y hora de conexión, y el tiempo que estuvo (o está conectado).

Realizar la misma aplicación tanto para C-S con Concurrencia Aparente (Select) como C-S Concurrente.

De ser necesario agregar tiempo de espera, loop, sleep, con contadores para demorar los procesos.

Implementar una limpieza de recuersos al salir del los programas (agregar opción de pregunta al usuario para cerrar los clientes)

Código cliente:

```
import socket



# Creando un socket TCP/IP

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)



# Conectar al servidor

server_dir = ('localhost', 6667)

sock.connect(server_dir)



# Recibir la hora de conexión del servidor

hora_conexion_servidor = sock.recv(1024).decode('utf-8')

print("Hora de conexión del servidor:", hora_conexion_servidor)



# Intercambio de mensajes

for i in range(5):
```

```
    # Enviar mensaje al servidor
    mensaje = f"Mensaje {i+1} del cliente"
    sock.sendall(mensaje.encode())

    # Recibir respuesta del servidor
    respuesta = ""
    while not respuesta.endswith('\n'):
        chunk = sock.recv(1024).decode('utf-8')
        respuesta += chunk

    print("Respuesta del servidor:", respuesta.strip())

# Recibir la duración de la conexión del servidor
duracion_conexion = sock.recv(1024).decode('utf-8')
print("Duración de la conexión:", duracion_conexion.strip())

# Cerrar el socket
sock.close()
```

Código servidor

```
import socket
import threading
import datetime
import time
```

```
host = "127.0.0.1"
port = 6667

def manejar_cliente(cliente_socket, cliente_direccion):
   print(f"Nueva conexión desde {cliente_direccion}")

   # Obtener la fecha y hora de conexión del servidor
   fecha_conexion_servidor = datetime.datetime.now()
   hora_conexion_servidor = fecha_conexion_servidor.strftime("%Y-%m-%d %H:%M:%S")
   cliente_socket.send(f"{hora_conexion_servidor}\n".encode())

   contador = 0
   while contador < 5:
       # Recibir datos del cliente
       datos = cliente_socket.recv(1024).decode('utf-8')
       if not datos:
           break

       print(f"Recibido de cliente {cliente_direccion}: {datos}")

       # Enviar datos de vuelta al cliente
       respuesta = f"Respuesta del servidor: {datos}"
       cliente_socket.sendall(f"{respuesta}\n".encode())

       contador += 1
```

```
    time.sleep(2)

    # Calcular la duración de la conexión y enviar al cliente

    duracion_conexion = datetime.datetime.now() - fecha_conexion_servidor

    cliente_socket.send(f"{duracion_conexion}\n".encode())



    # Cerrar la conexión con el cliente

    cliente_socket.close()

    print(f"Cliente {cliente_direccion} desconectado")



# Configurar el socket del servidor

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as servidor_socket:

    servidor_socket.bind((host, port))

    servidor_socket.listen()



    print(f"Servidor escuchando en {host}:{port}")



    while True:

        cliente_socket, cliente_direccion = servidor_socket.accept()

        threading.Thread(target=manejar_cliente, args=(cliente_socket,
cliente_direccion)).start()
```

Output Cliente:

Hora de conexión del servidor: 2024-05-13 17:53:40

Respuesta del servidor: Respuesta del servidor: Mensaje 1 del cliente

Respuesta del servidor: Respuesta del servidor: Mensaje 2 del cliente

Respuesta del servidor: Respuesta del servidor: Mensaje 3 del cliente

Respuesta del servidor: Respuesta del servidor: Mensaje 4 del cliente

Respuesta del servidor: Respuesta del servidor: Mensaje 5 del cliente

Duración de la conexión: 0:00:02.005418

Output Servidor:

Servidor escuchando en 127.0.0.1:6667

Nueva conexión desde ('127.0.0.1', 56738)

Recibido de cliente ('127.0.0.1', 56738): Mensaje 1 del cliente

Recibido de cliente ('127.0.0.1', 56738): Mensaje 2 del cliente

Recibido de cliente ('127.0.0.1', 56738): Mensaje 3 del cliente

Recibido de cliente ('127.0.0.1', 56738): Mensaje 4 del cliente

Recibido de cliente ('127.0.0.1', 56738): Mensaje 5 del cliente

Cliente ('127.0.0.1', 56738) desconectado

1) Concurrencia aparente

Para crear la aplicación cliente servidor con concurrencia aparente, se modificó los archivos de python subidos al github. Dentro de las modificaciones se incluyó la librería time para poder registrar el tiempo. Además se agregó que, al finalizar la lectura del último mensaje enviado por el cliente, el servidor envíe el tiempo total y el tiempo inicial de la conexión.

Código cliente:

```
import socket
import sys

mensajes = [
    'Este mensaje ',
    'es enviado ',
    'en partes.',
]
```

```
dir_servidor = ('localhost', 10000)

# Creo socket
socks = [
    socket.socket(socket.AF_INET, socket.SOCK_STREAM),
    socket.socket(socket.AF_INET, socket.SOCK_STREAM),
]

# onectar el socket al puerto en el cual el servidor está escuchando
print('conectando a {} puerto {}'.format(*dir_servidor),
      file=sys.stderr)
for s in socks:
    s.connect(dir_servidor)

for mensaje in mensajes:
    datos_salientes = mensaje.encode()

    # envío mensajes en ambos sockets
    for s in socks:
        print('{}: enviando {!r}'.format(s.getsockname(),
                                         datos_salientes),
              file=sys.stderr)
        s.send(datos_salientes)

    # leo respuestas en ambos sockets
    for s in socks:
        data = s.recv(1024)
        print('{}: recibido {!r}'.format(s.getsockname(),
                                         data),
              file=sys.stderr)
for s in socks:
        data = s.recv(1024)
        print('{}: recibido {!r}'.format(s.getsockname(),
                                         data),
              file=sys.stderr)
```

Código del servidor:

```
import select
import socket
import sys
import queue
```

```
import time

# Creando un socket TCP/IP
servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
servidor.setblocking(0)

# Hago Bind del socket al puerto
dir_servidor = ('localhost', 10000)
print('iniciando en {} port {}'.format(*dir_servidor),
      file=sys.stderr)
servidor.bind(dir_servidor)

# Escucho conexiones entrantes
servidor.listen(5)

# Sockect que espero leer
entradas = [servidor]

# Sockets que espero enviar
salidas = []

# Cola de mensajes salientes
cola_mensajes = {}

# Diccionario de tiempos
tiempos= {}

while entradas:

    # Espero a que al menos uno de los sockets este listo para ser
procesado

    print('esperando el próximo evento', file=sys.stderr)
    readable, writable, exceptional = select.select(entradas,
                                                    salidas,
                                                    entradas)

    if not (readable or writable or exceptional):
        print('  tiempo excedido....',
              file=sys.stderr)
        continue
    # Manejo entradas
    for s in readable:
```

```
        if s is servidor:
            # Un socket "leíble" está listo para aceptar conexiones
            con, dir_cliente = s.accept()
            print('  conexión desde: ', dir_cliente,
                  file=sys.stderr)
            con.setblocking(0)
            entradas.append(con)

            tiempo_actual = time.time()
            puerto_cliente = con.getpeername()[1]
            tiempos[puerto_cliente]=tiempo_actual

            # Le asigno a la conexión una cola en la cuál quiero enviar
            cola_mensajes[con] = queue.Queue()
            #cola_mensajes[con].put(cadena_tiempo.encode())

        else:
            data = s.recv(1024)
            if data:
                # Un socket leíble tiene datos
                print('  recibido {!r} desde {}'.format(
                    data, s.getpeername()), file=sys.stderr,
                )
                cola_mensajes[s].put(data)

                # Enviar hora de conexion si es el ultimo mensaje
                puerto_cliente = s.getpeername()[1]
                if(data.decode()=="en partes."):

#cola_mensajes[s].put(tiempos[puerto_cliente].encode())
                    tiempo_actual=time.time()
                    tiempo_anterior=tiempos[puerto_cliente]
                    tiempo_conexion= tiempo_actual - tiempo_anterior
                    mensaje="Hora de
conexion:"+time.ctime(tiempo_anterior)+". Tiempo total de conexion:
"+str(tiempo_conexion)
                    cola_mensajes[s].put(mensaje.encode())

                # Agrego un canal de salida para la respuesta
                if s not in salidas:
                    salidas.append(s)
            else:
```

```
                # Si está vacío lo interpreto como una conexión a
cerrar
                print('  cerrando...', dir_cliente,
                      file=sys.stderr)

                # dejo de escuchar en la conexión
                if s in salidas:
                    salidas.remove(s)
                entradas.remove(s)
                s.close()

                # Rremueve mensaje de la cola
                del cola_mensajes[s]
    # Administro salidas
    for s in writable:
        try:
            next_msg = cola_mensajes[s].get_nowait()
        except queue.Empty:
            # No hay mensaje en espera. Dejo de controlar para posibles
escrituras

            print('  ', s.getpeername(), 'cola vacía',
                  file=sys.stderr)
            salidas.remove(s)
        else:
            print(' enviando {!r} a {}'.format(next_msg,
s.getpeername()), file=sys.stderr)
            s.send(next_msg)

  # Administro condiciones excepcionales"
    for s in exceptional:
        print('excepción en', s.getpeername(),
              file=sys.stderr)
        # Dejo de escuchar en las conexiones
        entradas.remove(s)
        if s in salidas:
            salidas.remove(s)
        s.close()

        # Remuevo cola de mensajes
        del cola_mensajes[s]
```

Output del cliente:

```
conectando a localhost puerto 10000
('127.0.0.1', 34176): enviando b'Este mensaje '
('127.0.0.1', 34178): enviando b'Este mensaje '
('127.0.0.1', 34176): recibido b'Este mensaje '
('127.0.0.1', 34178): recibido b'Este mensaje '
('127.0.0.1', 34176): enviando b'es enviado '
('127.0.0.1', 34178): enviando b'es enviado '
('127.0.0.1', 34176): recibido b'es enviado '
('127.0.0.1', 34178): recibido b'es enviado '
('127.0.0.1', 34176): enviando b'en partes.'
('127.0.0.1', 34178): enviando b'en partes.'
('127.0.0.1', 34176): recibido b'en partes.'
('127.0.0.1', 34178): recibido b'en partes.'
('127.0.0.1', 34176): recibido b'Hora de conexion:Thu May  9 17:37:20 2024. Tiempo total de
conexion: 0.002445220947265625'
('127.0.0.1', 34178): recibido b'Hora de conexion:Thu May  9 17:37:20 2024. Tiempo total de
conexion: 0.0028963088989257812'
```

Output del servidor:

```
iniciando en localhost port 10000
esperando el próximo evento
  conexión desde:  ('127.0.0.1', 34176)
esperando el próximo evento
  conexión desde:  ('127.0.0.1', 34178)
  recibido b'Este mensaje ' desde ('127.0.0.1', 34176)
esperando el próximo evento
  recibido b'Este mensaje ' desde ('127.0.0.1', 34178)
 enviando b'Este mensaje ' a ('127.0.0.1', 34176)
esperando el próximo evento
   ('127.0.0.1', 34176) cola vacía
 enviando b'Este mensaje ' a ('127.0.0.1', 34178)
esperando el próximo evento
   ('127.0.0.1', 34178) cola vacía
esperando el próximo evento
  recibido b'es enviado ' desde ('127.0.0.1', 34176)
esperando el próximo evento
  recibido b'es enviado ' desde ('127.0.0.1', 34178)
 enviando b'es enviado ' a ('127.0.0.1', 34176)
esperando el próximo evento
   ('127.0.0.1', 34176) cola vacía
 enviando b'es enviado ' a ('127.0.0.1', 34178)
esperando el próximo evento
   ('127.0.0.1', 34178) cola vacía
```

```
esperando el próximo evento
  recibido b'en partes.' desde ('127.0.0.1', 34176)
esperando el próximo evento
  recibido b'en partes.' desde ('127.0.0.1', 34178)
 enviando b'en partes.' a ('127.0.0.1', 34176)
esperando el próximo evento
 enviando b'Hora de conexion:Thu May  9 17:37:20 2024. Tiempo total de conexion:
0.002445220947265625' a ('127.0.0.1', 34176)
 enviando b'en partes.' a ('127.0.0.1', 34178)
esperando el próximo evento
   ('127.0.0.1', 34176) cola vacía
 enviando b'Hora de conexion:Thu May  9 17:37:20 2024. Tiempo total de conexion:
0.0028963088989257812' a ('127.0.0.1', 34178)
esperando el próximo evento
   ('127.0.0.1', 34178) cola vacía
esperando el próximo evento
  cerrando... ('127.0.0.1', 34178)
esperando el próximo evento
  cerrando... ('127.0.0.1', 34178)
esperando el próximo evento
```

Luego de finalizar con el envío de mensajes y la hora hacia el cliente, el servidor sigue esperando nuevas conexiones.