

# Práctica 2: Programación en Sockets

Rafael Rodríguez Calvente  
Javier David García-Pardo Montero

3 de Mayo de 2020

## 1 Programa 1: Probar Sockets

El primer programa será implementar un programa cliente servidor que comunique dos procesos. Probar las diferencias que ocurren si el socket es TCP o UDP. ¿Qué diferencias hay? ¿Qué pasa si el cliente se inicia antes que el servidor en el caso TCP y UDP?

Para responder a las cuestiones anteriormente planteadas, es conveniente comentar las diferencias entre los protocolos TCP (*Transmission Control Protocol*) Y UDP (*User Datagram Protocol*):

- TCP está orientado a la **conexión** mientras que UDP es un protocolo **sin conexión**.
- TCP es altamente confiable para transferir datos ya que garantiza un acuse de recibo de la información enviada y vuelve a enviar los paquetes perdidos si los hay. En el caso de UDP, si el paquete se pierde, no solicitará su retransmisión y el ordenador de destino recibirá un dato corrupto. Por lo tanto, UDP no es un protocolo fiable
- TCP es mas lento en comparación con UDP, ya que TCP establece la conexión antes de transmitir los datos y garantiza la entrega adecuada de los paquetes. Por otro lado, UDP no reconoce si los datos transmitidos son recibidos o no.

Si el cliente se inicia antes que el servidor, el cliente estaría enviando a una dirección puerto que no existe aún ya que no hay nadie esperando un mensaje.

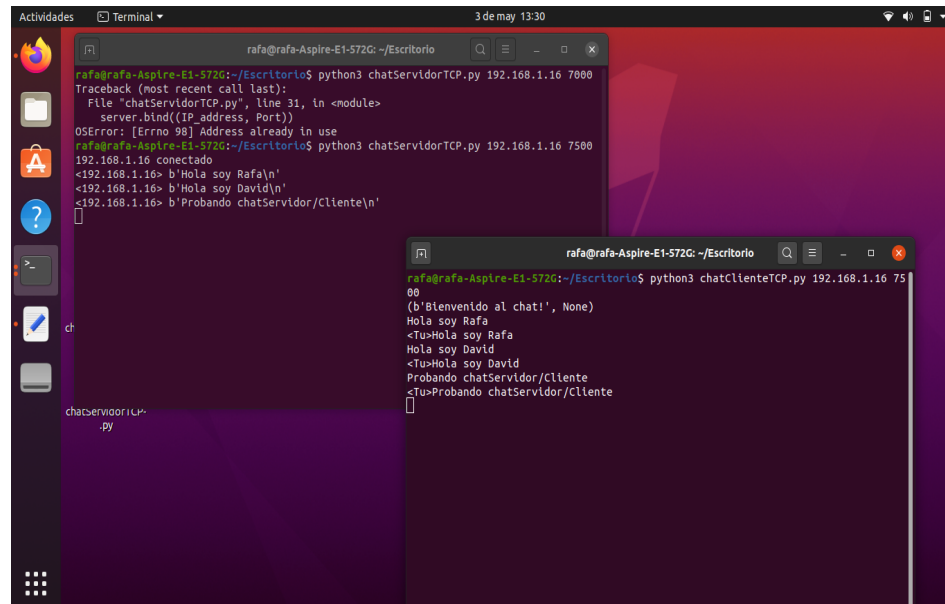
## 2 Programa 2: Chat Simple

Realizar un chat simple, en principio síncrono: un usuario habla, y se espera a que el otro usuario responda. Investigar qué función de Python hay para leer

por teclado. Implementar una versión con TCP y otra con UDP. ¿Qué diferencias hay entre ellas? Describir qué flujo ocurre entre los participantes. A continuación se detalla el funcionamiento de la aplicación y posteriormente se puede encontrar una captura de pantalla del funcionamiento del chat TCP.

1. Se ejecuta el programa servidor indicando la dirección IP de la máquina que lo ejecuta y el puerto por el cual se desea que se realice la posterior conexión con el/los programas cliente. **python3 chatServidorTCP.py 192.168.1.16 9090**. NOTA: el puerto indicado en este caso es el 7500, pero cualquiera podrá ser utilizado siempre y cuando no sea uno inferior a 1024 puesto que están reservados. Hay disponible 60.000 puertos aproximadamente.
2. Se ejecuta el/los programas cliente indicando la misma dirección (ya que se está ejecutando desde la misma máquina pero en otra terminal) y el mismo puerto. **python3 chatClienteTCP.py 192.168.1.16 7500**.
3. La conexión se establece automáticamente si todo fue correcto.
4. El/los clientes pueden comunicarse entre si. NOTA: todos los mensajes escritor por cada cliente aparecerán en todos los clientes que estén dentro de dicho chat, y además también aparecerá dentro de la terminal el programa servidor.

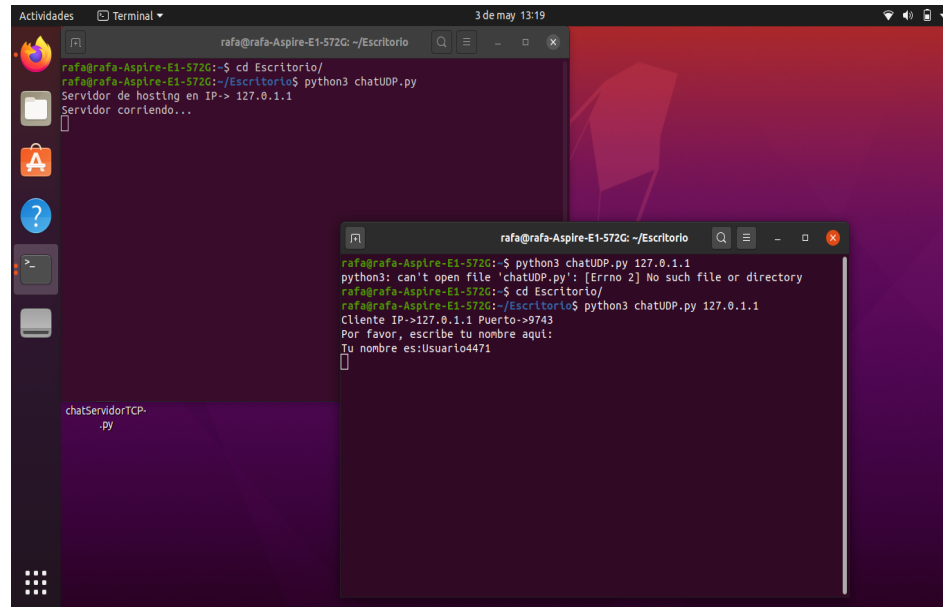
Veamos una captura del chat TCP en funcionamiento.



```
rafa@rafa-Aspire-E1-572G: ~/Escritorio
rafa@rafa-Aspire-E1-572G:~/Escritorio$ python3 chatServidorTCP.py 192.168.1.16 7000
Traceback (most recent call last):
  File "chatServidorTCP.py", line 31, in <module>
    server.bind((IP_address, Port))
OSError: [Errno 98] Address already in use
rafa@rafa-Aspire-E1-572G:~/Escritorio$ python3 chatServidorTCP.py 192.168.1.16 7500
192.168.1.16 conectado
<192.168.1.16> b'Hola soy Rafa\n'
<192.168.1.16> b'Hola soy David\n'
<192.168.1.16> b'Probando chatServidor/Cliente\n'

rafa@rafa-Aspire-E1-572G:~/Escritorio$ python3 chatClienteTCP.py 192.168.1.16 7500
(b'Bienvenido al chat!', None)
Hola soy Rafa
<Tu>Hola soy Rafa
Hola soy David
<Tu>Hola soy David
Probando chatServidor/Cliente
<Tu>Probando chatServidor/Cliente
```

La versión UDP sería la siguiente:



```
rafa@rafa-Aspire-E1-572G: ~/Escritorio
rafa@rafa-Aspire-E1-572G:~$ cd Escritorio/
rafa@rafa-Aspire-E1-572G:~/Escritorio$ python3 chatUDP.py
Servidor de hosting en IP-> 127.0.1.1
Servidor corriendo...

rafa@rafa-Aspire-E1-572G:~$ python3 chatUDP.py 127.0.1.1
python3: can't open file 'chatUDP.py': [Errno 2] No such file or directory
rafa@rafa-Aspire-E1-572G:~$ cd Escritorio/
rafa@rafa-Aspire-E1-572G:~/Escritorio$ python3 chatUDP.py 127.0.1.1
Cliente IP->127.0.1.1 Puerto->9743
Por favor, escribe tu nombre aquí:
Tu nombre es:Usuario4471
```

### 3 Programa 3: FTP Simple

Escribir un programa cliente/servidor en el que el servidor recibe por socket el nombre de un fichero y se lo envía al cliente.

Su funcionamiento y posterior captura de pantalla es la siguiente.

1. El programa servidor es ejecutado en una terminal, sin indicar dirección IP ni puerto, puesto que éste es establecido por el propio programa al iniciarse. La dirección IP se consigue utilizando el método **gethostname()** del módulo **socket**, el puerto asignado es el 60.000.
2. El programa cliente se ejecuta en una terminal si indicar ni IP ni puerto, porque cuando se inicie se le asignarán los puertos e IP del mismo modo que en el programa del servidor.
3. La conexión es establecida, el cliente le envía el mensaje "hola servidor" al programa servidor para confirmar que la conexión fue establecida con éxito. Tras ello, el programa servidor abre el archivo de texto, lee su contenido (el cual ha de ser menor a 1024 bytes) y lo enviará al cliente, el cual muestra por pantalla dicho mensaje en su terminal para verificar que se realizó correctamente

