

UNIVERSIDAD DIEGO PORTALES

INGENIERÍA CIVIL INFORMÁTICA Y
TELECOMUNICACIONES

Laboratorio N°6

Redes de Datos

Author:

Sebastian DIAZ

DANIEL UTRERAS

Ayudante:

Alexis INZUNZA

Profesor:

José PÉREZ



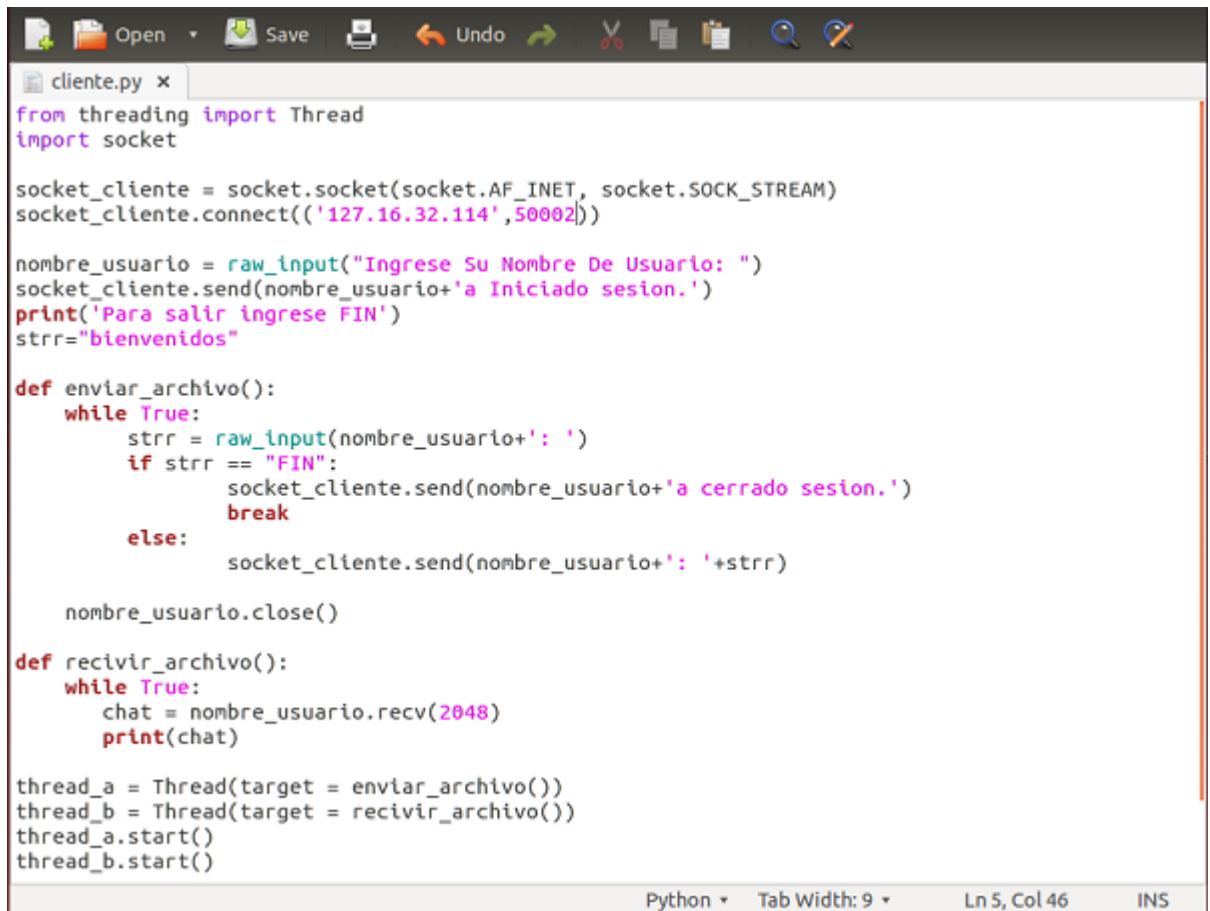
June 9, 2017

1 ÍNDICE

2 ACTIVIDAD 1	PÁG 2-6
3 ACTIVIDAD 2	PÁG 7
6 PREGUNTAS PROPUESTAS	PÁG 10-15
6.1	PÁG 10
6.2	PÁG 10
6.3	- PÁG 11
6.4	- PÁG 11
7 CONCLUSIÓN	PÁG 15
8 BIBLIOGRAFÍA	PÁG 15

1 IMAGEN 1:Codigo cliente	PÁG 2
2 IMAGEN 2:Codigo servidor	PÁG 3
3 IMAGEN 3: Cliente 1	PÁG 4
4 IMAGEN 4: Cliente 2	PÁG 5
5 IMAGEN 5: Servidor	PÁG 6
6 IMAGEN 6: Wireshark	PÁG 7

2 ACTIVIDAD 1



```
cliente.py x
from threading import Thread
import socket

socket_cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket_cliente.connect(('127.16.32.114', 50002))

nombre_usuario = raw_input("Ingrese Su Nombre De Usuario: ")
socket_cliente.send(nombre_usuario+'a Iniciado sesion.')
print('Para salir ingrese FIN')
strr="bienvenidos"

def enviar_archivo():
    while True:
        strr = raw_input(nombre_usuario+' : ')
        if strr == "FIN":
            socket_cliente.send(nombre_usuario+'a cerrado sesion.')
            break
        else:
            socket_cliente.send(nombre_usuario+' : '+strr)

    nombre_usuario.close()

def recibir_archivo():
    while True:
        chat = nombre_usuario.recv(2048)
        print(chat)

thread_a = Thread(target = enviar_archivo())
thread_b = Thread(target = recibir_archivo())
thread_a.start()
thread_b.start()
```

Python ▾ Tab Width: 9 ▾ Ln 5, Col 46 INS

(IMAGEN 1: Codigo cliente)

```
servidor.py (~/Desktop) - gedit
Open Save Undo
cliente.py x servidor.py x
from threading import Thread
import socket

socket_servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket_servidor.bind(('127.16.32.114', 50002))
socket_servidor.listen(2)

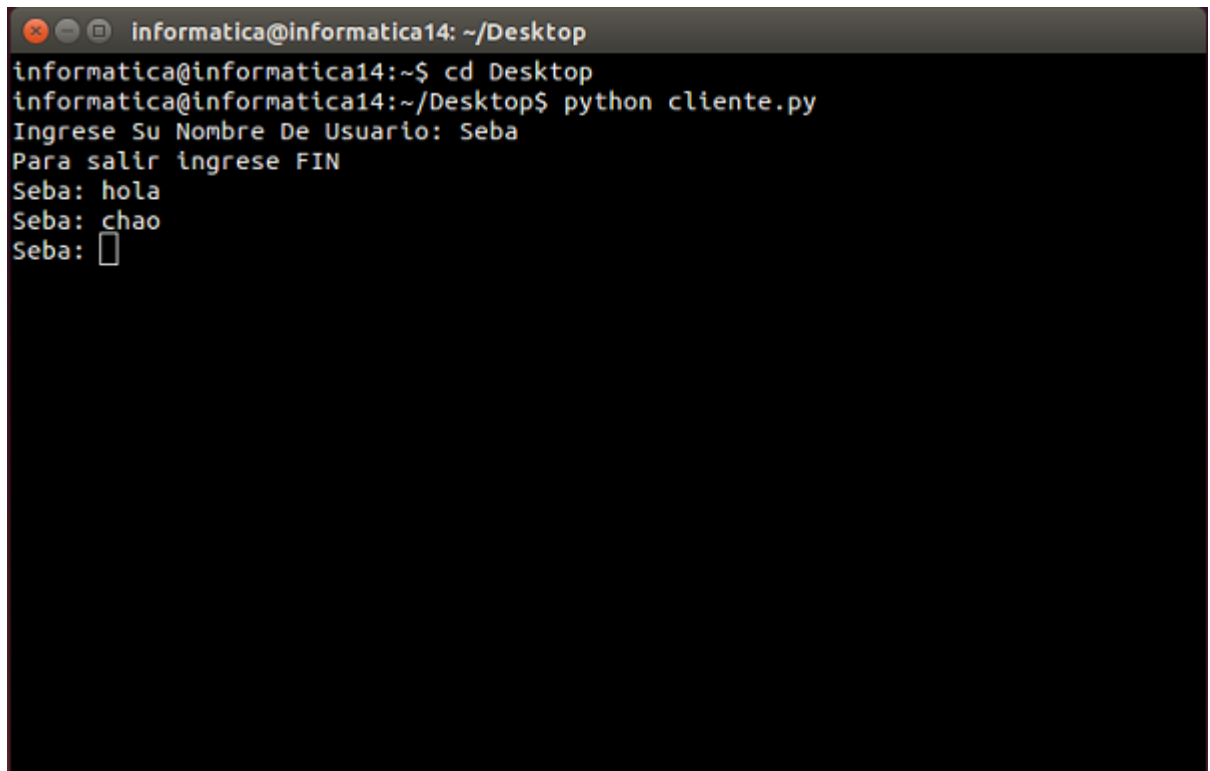
def funcion():
    sock, addr = socket_servidor.accept()
    print(sock.recv(2048))

def funcion_2(sock, addr):
    while True:
        chat = sock.recv(2048)
        print(chat)
        sock.send(chat)

while True:
    sock, addr = socket_servidor.accept()
    thread_a = Thread(target = funcion_2, args = (sock, addr))
    thread_a.start()
```

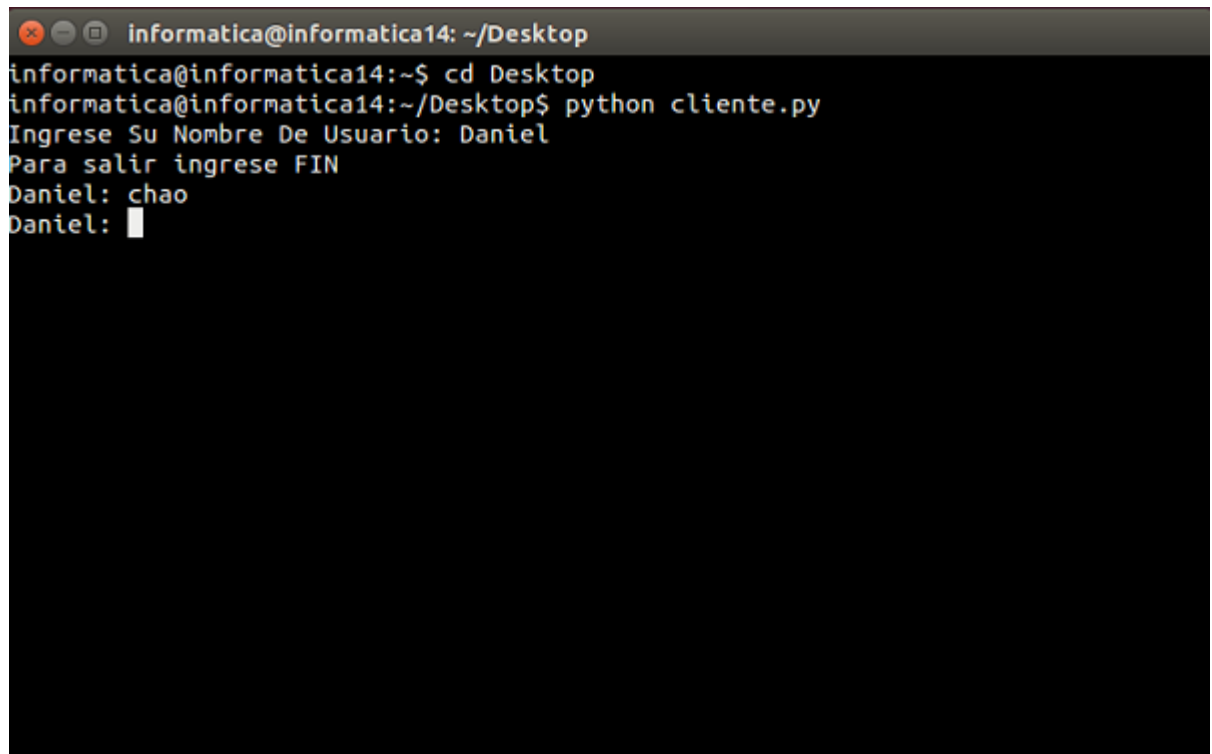
Python ▾ Tab Width: 9 ▾ Ln 22, Col 1 INS

(IMAGEN 2: Codigo servidor)

A terminal window with a dark background and light-colored text. The window title bar shows standard Linux window controls (close, maximize, and a third icon) followed by the text 'informatica@informatica14: ~/Desktop'. The terminal content shows a user navigating to the Desktop directory and running a Python script named 'cliente.py'. The script prompts for a username, which is 'Seba', and then displays two messages: 'hola' and 'chao'. The prompt 'Seba:' is followed by a cursor, indicating the user is ready to enter more input.

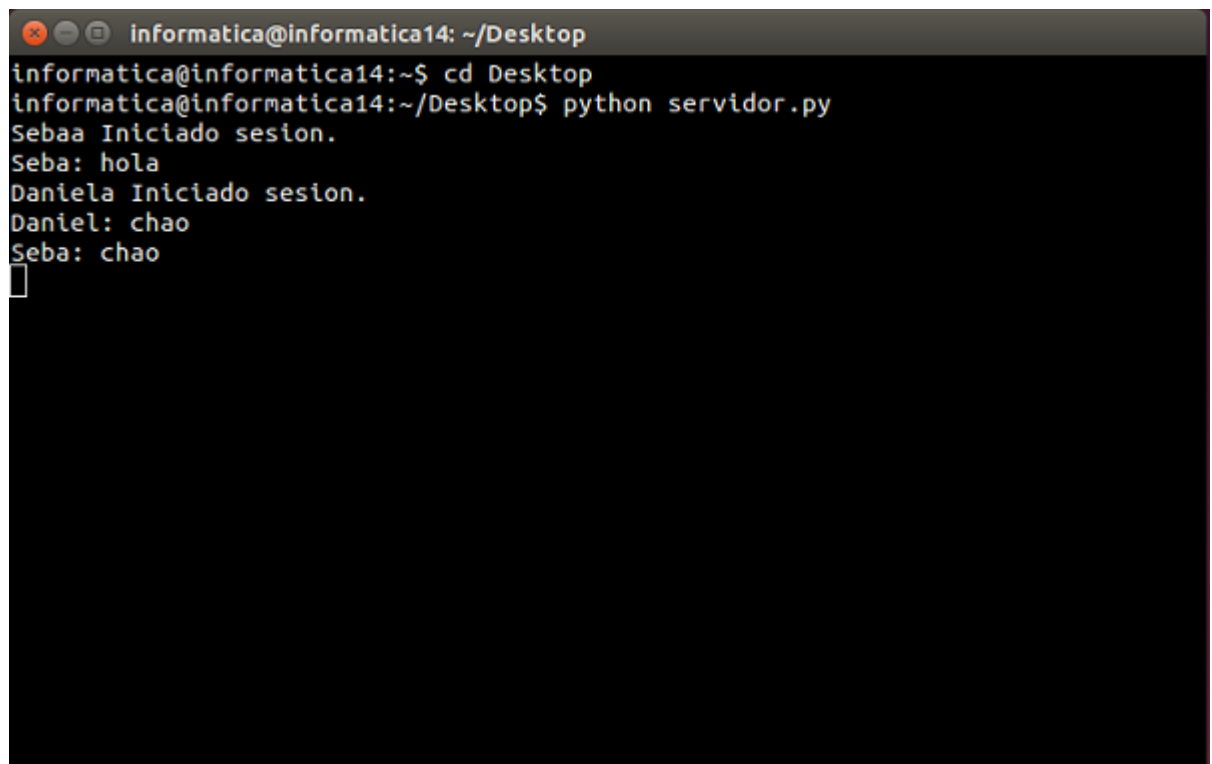
```
informatica@informatica14: ~/Desktop
informatica@informatica14:~$ cd Desktop
informatica@informatica14:~/Desktop$ python cliente.py
Ingrese Su Nombre De Usuario: Seba
Para salir ingrese FIN
Seba: hola
Seba: chao
Seba: 
```

(IMAGEN 3: Cliente 1)

A terminal window with a dark background and light-colored text. The window title bar shows standard Linux window controls (close, maximize, minimize) and the text 'informatica@informatica14: ~/Desktop'. The terminal content shows a user navigating to the Desktop directory and running a Python script named 'cliente.py'. The script prompts for a username, and the user enters 'Daniel'. It then prompts for a message to exit, and the user enters 'chao'. The prompt 'Daniel: ' is followed by a cursor, indicating the user is still in the process of entering a response.

```
informatica@informatica14: ~/Desktop
informatica@informatica14:~$ cd Desktop
informatica@informatica14:~/Desktop$ python cliente.py
Ingrese Su Nombre De Usuario: Daniel
Para salir ingrese FIN
Daniel: chao
Daniel: 
```

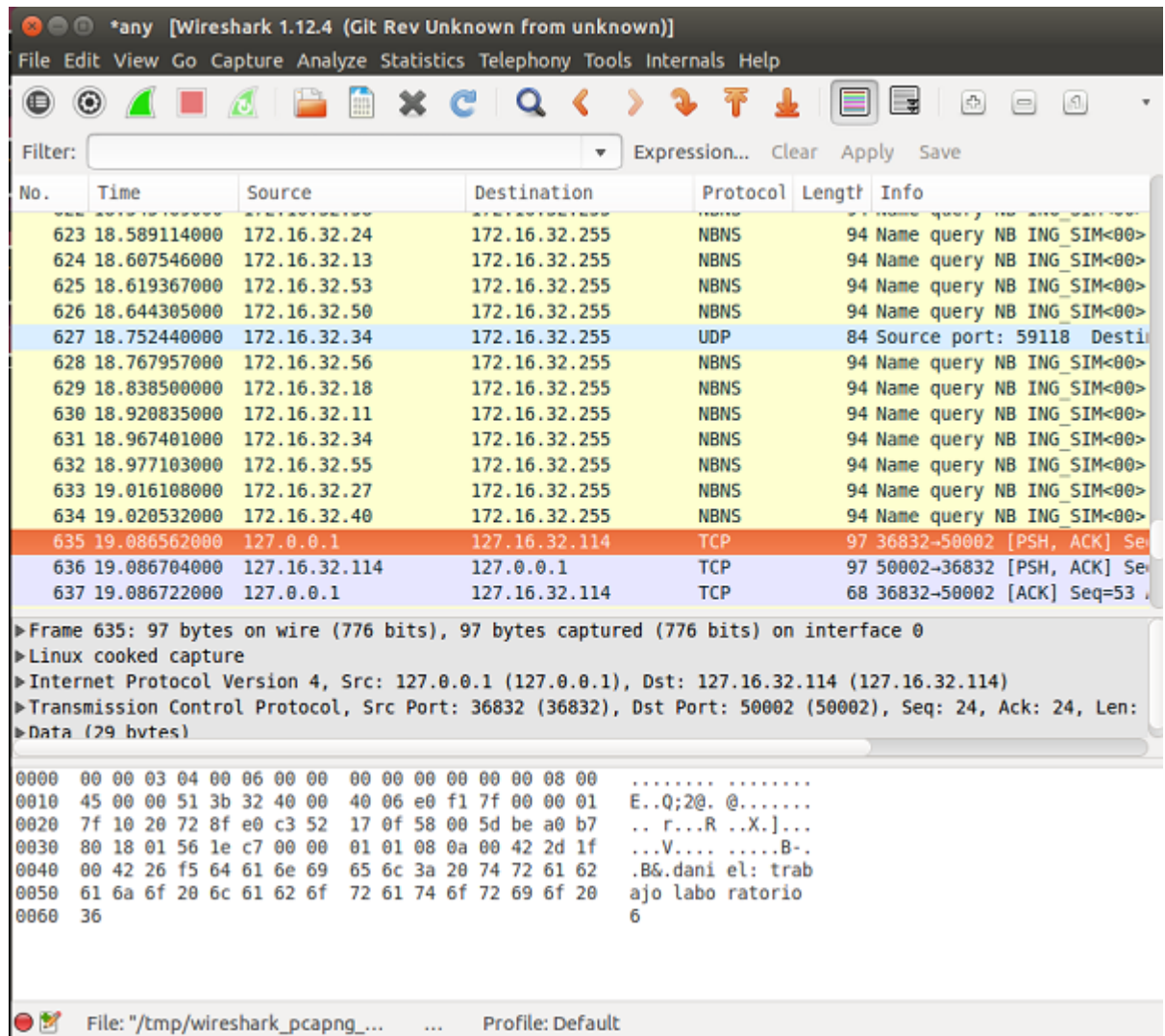
(IMAGEN 4: Cliente 2)



```
informatica@informatica14: ~/Desktop
informatica@informatica14:~$ cd Desktop
informatica@informatica14:~/Desktop$ python servidor.py
Sebaa Iniciado sesion.
Seba: hola
Daniela Iniciado sesion.
Daniel: chao
Seba: chao
█
```

(IMAGEN 5: Servidor)

3 ACTIVIDAD 2



(IMAGEN 6:Wireshark con el mensaje "trabajo laboratorio 6")

4 PREGUNTAS PROPUESTAS

4.1 Explique el funcionamiento del envío y recepción de información para su algoritmo, respondiendo las interrogantes ¿Que se envía?¿Como se envía?

Se envia información, el envío y recepción de esto funciona mediante un numero de secuencias, primero creamos un sockets para instanciarlo, siguiendo con un bind para configurar las direcciones y el puerto, luego configuramos el número de conexiones usando listen y creamos 2 funciones de envío y recepción , finalizando con un ciclo el cual espera un mensaje para iniciar el chat.

4.2 ¿Porque el puerto que muestra el servidor al generar la conexión no es el mismo que el escrito en el algoritmo del cliente?

4.3 Si usted tuviese que realizar un videojuego con soporte multijugador utilizando sockets ¿Que protocolo se utilizaría?¿Porque?

El protocolo UDP sería un protocolo más adecuado para un videojuego, ya que el TCP es para funciones que exigen un nivel de transporte, detección y corrección de errores fiables, produciendo que tenga una ejecución más lenta y pesada, en cambio UDP al ser más ligero con el transporte, el videojuego correría a mayor velocidad y así evitando que en su desarrollo ocurran bucles o fallas, y en esta reaccionar rápido.

4.4 ¿Se puede utilizar cualquier numero de puerto para cualquier aplicación?

Los puertos se dividen en 3 secciones: puertos bien conocidos (0-1024), puertos registrados (1024-49151) y puertos dinámicos o privados (49152-65535aprox). Los dos primeros se le asignaran para ser utilizados en aplicaciones y el sistema operativo, dejando los puertos dinámicos para clientes.

5 CONCLUSIÓN

En este laboratorio comprendimos que función cumplen los sockets, método para el intercambio de información entre un cliente y un servidor, además de sus protocolos. Logramos establecer conexión en un servidor, mediante dos clientes utilizando python y estudiar y comprender su funcionamiento usando Wireshark.

6 BIBLIOGRAFÍA

Wikipedia