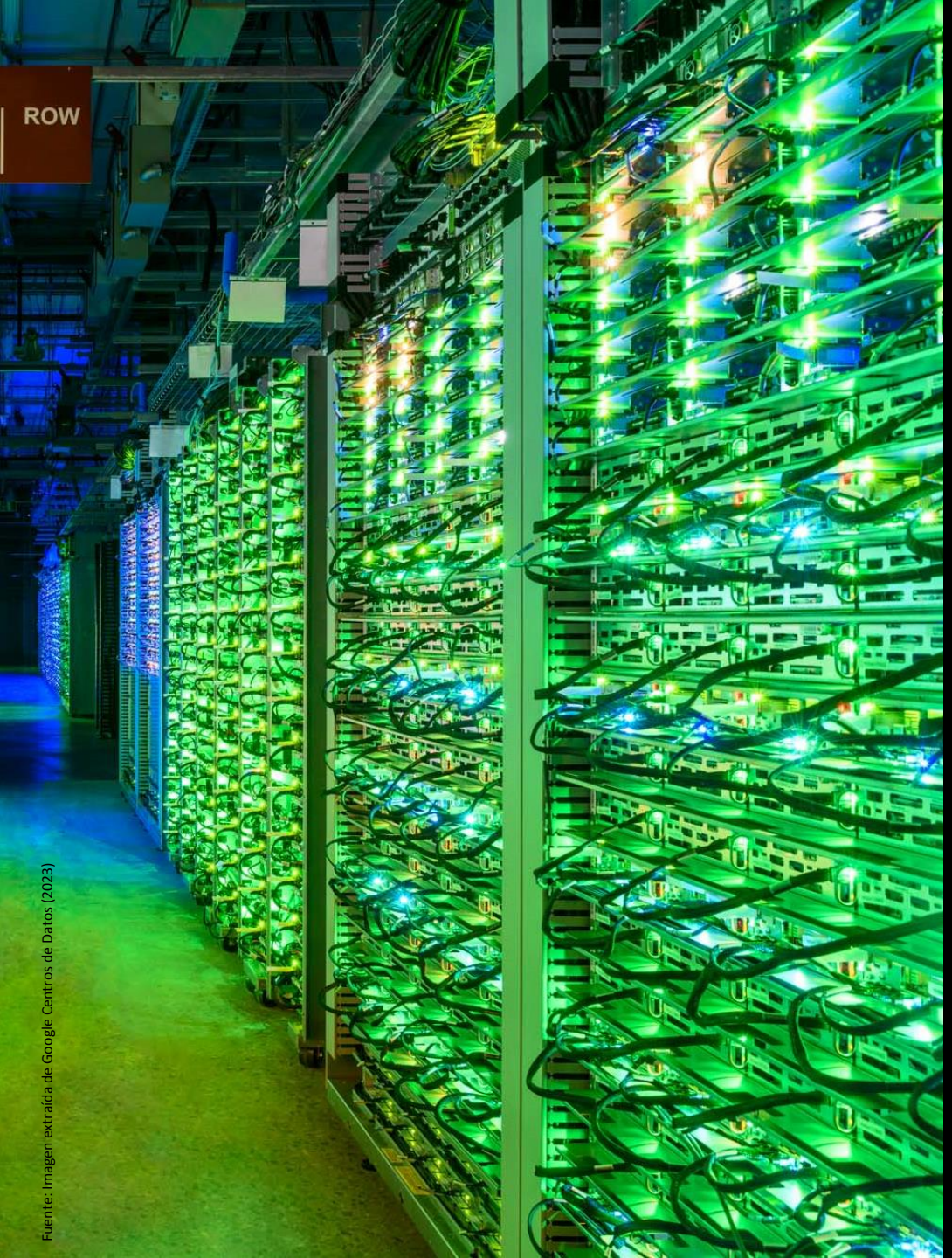


Fuente: Imagen extraída de Google Centros de Datos (2023)

REDES DE COMPUTADORAS

Ciclo 2023-1

Semana 06

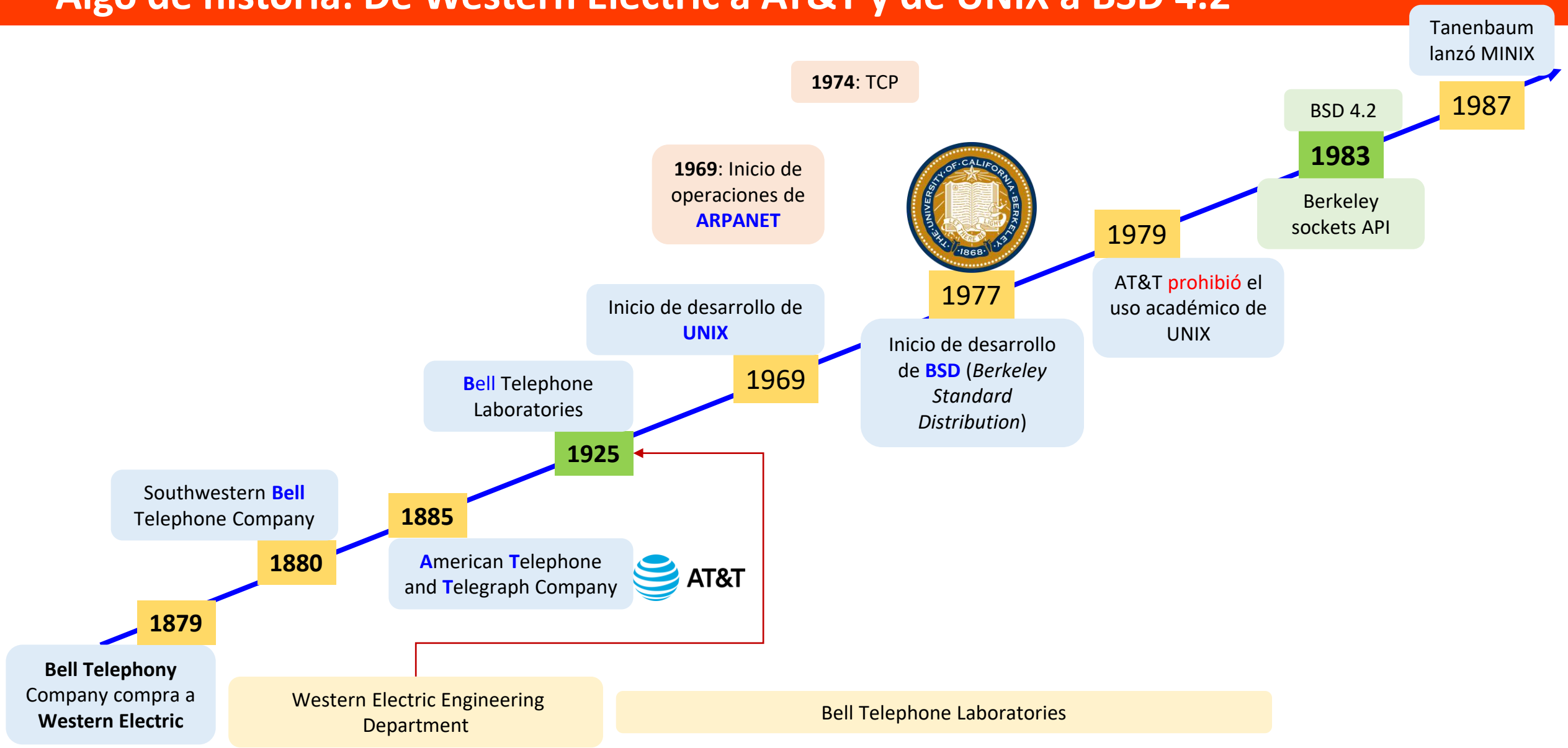
Sockets

UNIVERSIDAD DE LIMA

Logro de la semana

Al final de la semana el estudiante comprende el **funcionamiento de los sockets** y su programación usando lenguaje Python

Algo de historia: De Western Electric a AT&T y de UNIX a BSD 4.2



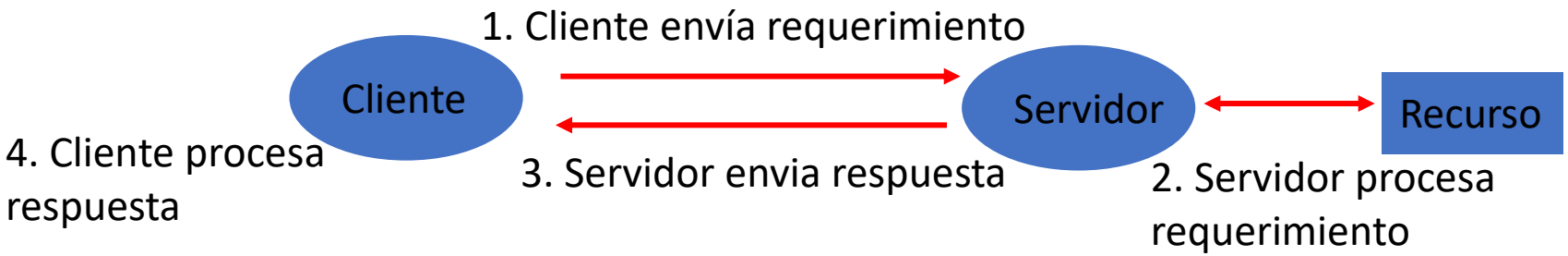
Fuente: Imagen extraída de Google Centros de Datos (2023)

1.- Capa de transporte

UNIVERSIDAD DE LIMA

Sockets

- Un socket es una conexión virtual que permite comunicar dos procesos.
- Sockets proveen una transmisión bi-direccional (**full duplex**)
- Como conexiones virtuales, los sockets pueden ser creados en forma dinámica
- La interface de sockets fue desarrollada a principios de los 80´ en la universidad de Berkeley
- Sockets son la herramienta de comunicación más utilizada en aplicaciones de internet.
- TCP/IP usa sockets.



Métodos

```
import socket   #Permite trabajar con el módulo de sockets de Python
```

Métodos:

```
socket ()
bind ()
listen ()
accept ()
connect ()
send ()
recv ()
close ()
```

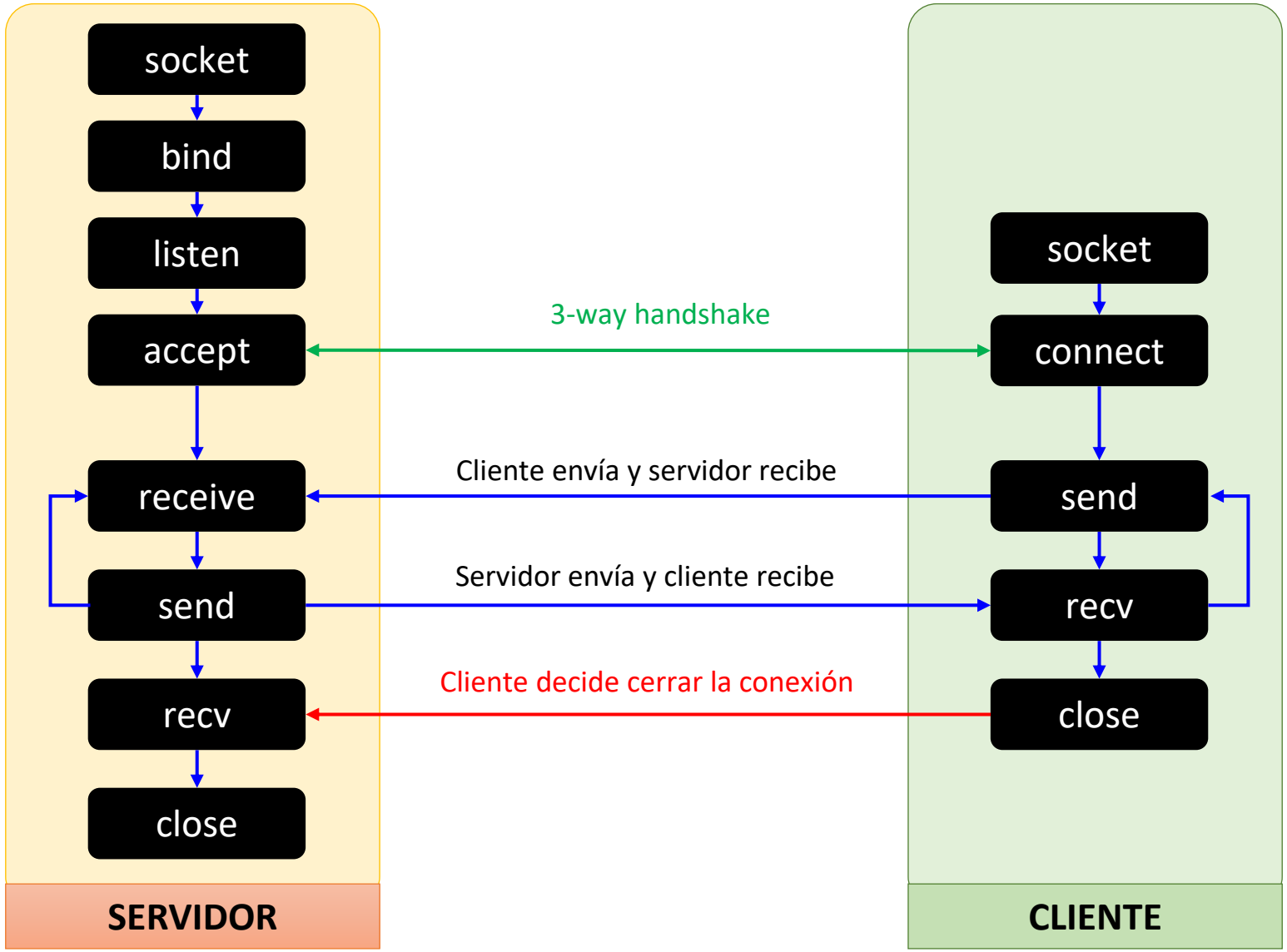
Para crear un socket se usa el objeto socket.socket()

```
# Para crear un socket IPv4 (INET) con TCP (STREAM):
#Opción 01:
socket_server = socket.socket (socket.AF_INET, socket.SOCK_STREAM)

#Opción 02:
socket_server = socket.socket ()
```

```
# Para crear un socket IPv4 (INET) con UDP (STREAM):
socket_server = socket.socket (socket.AF_INET, socket.SOCK_DGRAM)
```

Secuencia de llamada al API de socket



Elaborado con información de: https://realpython.com/python-sockets

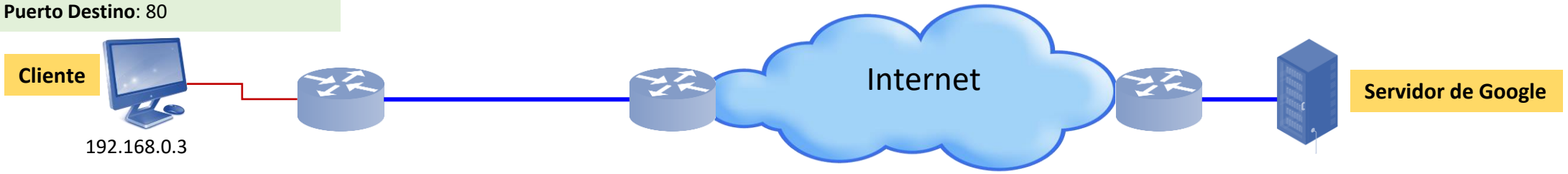
Fuente: Imagen extraída de Google Centros de Datos (2023)

2.- Ejercicios

UNIVERSIDAD DE LIMA

Ejercicio 01: Conexión desde el cliente (mi PC) al Servidor de Google

IP Destino: Ip del Servidor de Google
Puerto Destino: 80




```
In [1]: # importar libreria de sockets
        import socket

In [2]: # crear el socket TCP con IPv4:
        s =socket.socket(socket.AF_INET,socket.SOCK_STREAM)

        # Imprime que el socket se creó correctamente:
        print("Socket creado exitosamente")

        Socket creado exitosamente

In [3]: # Obtiene la IP de google.com
        host_ip = socket.gethostbyname("www.google.com")

In [4]: #La conexión se realizará por el puerto 80:
        port = 80

        #Se establece la conexión a la IP y puerto indicado
        s.connect((host_ip, port))

In [5]: # imprime que los resultados fueron exitosos:

        print("Conexión Ok")
        print("IP Destino = ", host_ip)
        print("puerto_destino = ", port)

        Conexión Ok
        IP Destino =  172.217.3.68
        puerto_destino =  80
```

```
# Para conectarme a un servidor (desde el cliente)

s.connect (IP_Servidor, Puerto_Destino)

#donde «s» representa al nombre que le pusiste al socket (al momento de crearlo)
```

3-way Handshake:

También puedes filtrar con **tcp.port == 80**

tcp.stream eq 61

No.	Time	Source	Destination	Protocol	Length	Info
711	10.246665	192.168.0.3	172.217.3.68	TCP	66	54757 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
712	10.349838	172.217.3.68	192.168.0.3	TCP	66	80 → 54757 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
713	10.349908	192.168.0.3	172.217.3.68	TCP	54	54757 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0

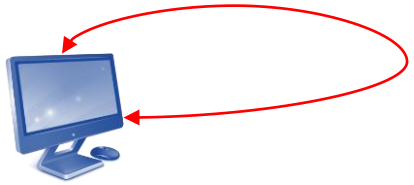
```
C:\Users\Un>netstat -n -o -a -p TCP | findstr 80
  TCP    0.0.0.0:7680           0.0.0.0:0              LISTENING       4492
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       980
  TCP    127.0.0.1:8071         0.0.0.0:0              LISTENING       13740
  TCP    127.0.0.1:8071         127.0.0.1:59428        ESTABLISHED     13740
  TCP    127.0.0.1:59380        127.0.0.1:59381        ESTABLISHED     13992
  TCP    127.0.0.1:59381        127.0.0.1:59380        ESTABLISHED     13992
  TCP    127.0.0.1:59428        127.0.0.1:8071         ESTABLISHED     2916
  TCP    192.168.1.7:59659      142.251.0.104:80       TIME_WAIT       0
  TCP    192.168.1.7:59669      38.90.226.37:80        TIME_WAIT       0
  TCP    192.168.1.7:59670      64.233.186.99:80       ESTABLISHED     19112
```

Visualiza el PID del proceso

Ejercicio 02.1: Programar el servidor en el puerto 12345 y forzamos un cliente TELNET

Servidor

Cliente



Loopback

127.0.0.1:12345

1°: En Spyder escribir el siguiente código

server.py*

```python
# importa la libreria socket
import socket
# luego crea el objeto socket
s = socket.socket()
print ("Socket creado exitosamente")
# reserva un puerto
port = 12345
# luego enlaza el socket al puerto
# al no poner IP, el servidor escucha los requerimientos
# que lleguen de otras computadoras
s.bind(('', port))
print ("El socket está enlazado a %s" %(port))
# pone el socket en modo escucha
s.listen(5)
print ("El socket está en modo escucha")
# bucle infinito hasta interrupción o error
while True:
    # Establece conexión con el Cliente y crea un nuevo Socket (ej: "con")
    # addr representa al puerto del Cliente
    con, addr = s.accept()
    print ('Conexión recibida de', addr)
    # envía un mensaje de agradecimiento al Cliente
    con.send(b'Gracias por conectarse')
    # Cierra la conexión con el Cliente
    con.close()
```

2°: En Wireshark, habilitar la captura de la interfaz loopback

VirtualBox Host-Only Network

Adapter for loopback traffic capture

3°: Ejecuta el servidor desde el terminal (puedes usar Spyder)

```
PS C:\Users\Unknown> python server.py
Socket creado exitosamente
El socket está enlazado a 12345
El socket está en modo escucha
Conexión recibida de ('127.0.0.1', 63159)
```

4°: Crea una conexión TELNET al Localhost:12345

```
C:\Users\Unknown>telnet localhost 12345
```

5°: Filtra solo las conexiones en el puerto 12345

tcp.port==12345

6°: Se indicará que se recibió una conexión

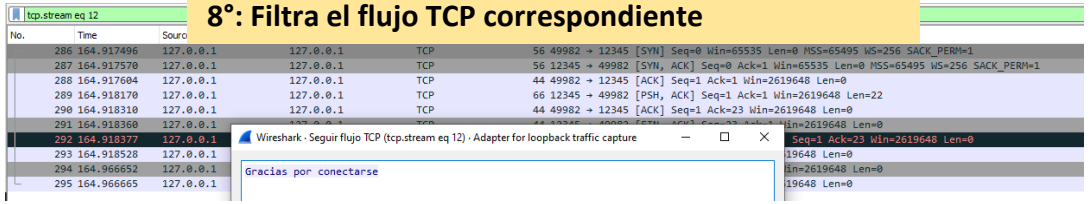
```
Conexión recibida de ('127.0.0.1', 49982)
```

7°: La conexión TELNET terminará

```
Gracias por conectarse

Se ha perdido la conexión con el host.
```

8°: Filtra el flujo TCP correspondiente



Ejercicio 02.2: Programar el cliente

1°: En Spyder escribir el siguiente código

server.py × cliente.py ×

```
1   # Importa el módulo socket
2   import socket
3   # Crea un objeto socket
4   sock1 = socket.socket()
5   # Define el puerto al cual se conectará
6   port = 12345
7   # Se conecta al Servidor en su computadora local
8   sock1.connect(('127.0.0.1', port))
9   # recibe datos del Servidor
10  print (sock1.recv(1024))
11  # Cierra la conexión
12  sock1.close()
```

2°: Ejecutar el servidor desde Spyder (Run file: F5)

```
In [1]: runfile('C:/Users/Unknown/server.py', wdir='C:/Users/Unknown')
Socket creado exitosamente
El socket está enlazado a 12345
El socket está en modo escucha
```

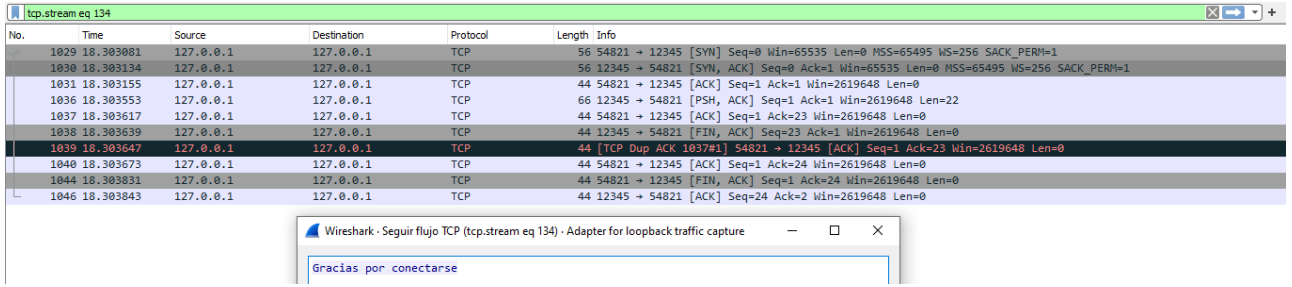
3°: En una nueva consola, ejecutar el cliente desde Spyder (Run file: F5)

```
In [1]: runfile('C:/Users/Unknown/cliente.py', wdir='C:/Users/Unknown')
b'Gracias por conectarse'
```

4°: En la consola del servidor se visualiza la conexión exitosa

```
In [1]: runfile('C:/Users/Unknown/server.py', wdir='C:/Users/Unknown')
Socket creado exitosamente
El socket está enlazado a 12345
El socket está en modo escucha
Conexión recibida de ('127.0.0.1', 63254)
```

5°: Realizar la captura En Wireshark



6°: Verificar que el Socket está en modo TIME_WAIT

```
C:\Users\Unknown>netstat -a -o -n -p TCP

Conexiones activas

  Proto  Dirección local        Dirección remota       Estado           PID
  TCP    127.0.0.1:12345        127.0.0.1:55795        TIME_WAIT        0
  TCP    127.0.0.1:12345        127.0.0.1:55802        TIME_WAIT        0
  TCP    127.0.0.1:12345        127.0.0.1:55808        TIME_WAIT        0
  TCP    127.0.0.1:12345        127.0.0.1:55815        TIME_WAIT        0
  TCP    127.0.0.1:12345        127.0.0.1:56481        TIME_WAIT        0
  TCP    127.0.0.1:12345        127.0.0.1:56483        TIME_WAIT        0
```

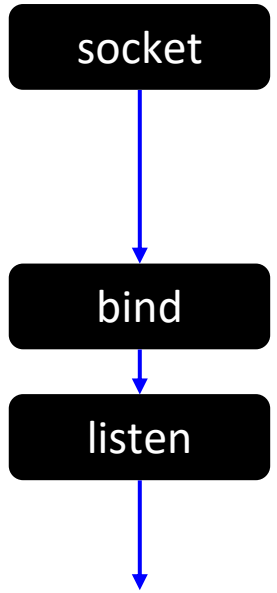
Recuerda que también puedes hacer las pruebas desde el terminal

```
PS C:\Users\Unknown> python server.py
Socket creado exitosamente
El socket está enlazado a 12345
El socket está en modo escucha
Conexión recibida de ('127.0.0.1', 55500)
Conexión recibida de ('127.0.0.1', 57699)
```

```
PS C:\Users\Unknown> python cliente.py
b'Gracias por conectarse'
PS C:\Users\Unknown> python cliente.py
b'Gracias por conectarse'
PS C:\Users\Unknown>
```

Ejercicio 03.1: Aplicación para transferencia de archivos – Lado del Servidor

Importar librería de sockets:	import socket
Creación del socket “server” con TCP	server=socket.socket ()
Indicar el puerto del Servidor	port = 60000
Enlazar la IP y el Puerto al Socket	server.bind((‘’,port))
Poner al Socket en modo de escucha	server.listen(5) print (“Servidor escuchando …”)



Mientras ningún cliente se conecte al servidor, estará en el modo “listen”

```
In [1]: runfile('C:/Users/Unknown/transferserver.py', wdir='C:/Users/Unknown')
Servidor escuchando ...
```

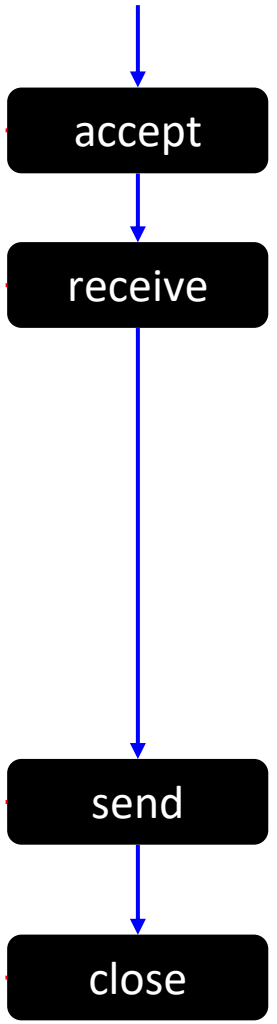
Ejercicio 03.1: Aplicación para transferencia de archivos – Lado del Servidor

Ejecutar mientras sea TRUE	`while True:`
Aceptar la conexión por medio de un nuevo socket (**conn**) con "ADDR"	`conn, addr = server.accept()` `print ('Conexión recibida de ', addr)`
En la variable "data" almacenar la información recibida (Buffer = 1024B)	`data = conn.recv(1024)` `print ('Mensaje recibido', repr(data))`
Indicar el archivo que remitirá SERVER	`filename='mi_texto.txt'`
Leer (read) el TXT en binario	`f = open(filename,'rb')`
Dividirlo en bloques de 1024 Bytes	`l = f.read(1024)`
Ejecutar mientras L aun tenga bits	`while (l):`

Enviar los bits de "L"	`conn.send(l)`
	`print ('Enviado ',repr(l))`
	`l = f.read(1024)`

Terminar la conexión	`print ('Terminó de enviar')`
	`conn.send(b'Gracias cliente por conectarse')`
	`conn.close()`

El TXT se carga en la misma carpeta del **server.py**



Ejercicio 03.2: Aplicación para transferencia de archivos – Lado del Cliente

Importar librería de sockets:

```
import socket
```

Creación del socket “s” con TCP

```
s=socket.socket ()
```

Indicar el puerto y la IP del Servidor

```
port = 60000
host = socket.gethostname()
```

Enlazar la IP y el Puerto al Socket

```
s.connect((host, port))
```

Enviar un mensaje de saludo al Servidor

```
s.send(b"Hola Servidor!")
```

Escribir los datos en un archivo llamado archivo_recibido.txt

```
with open('archivo_recibido.txt', 'wb') as f:
print('Transferencia iniciada')
```

Mientras se reciban datos, lo almacena en la variable “data”

```
while True:
    print('Recibiendo datos...')
    data = s.recv(1024)
    print('datos = %s', (data))
```

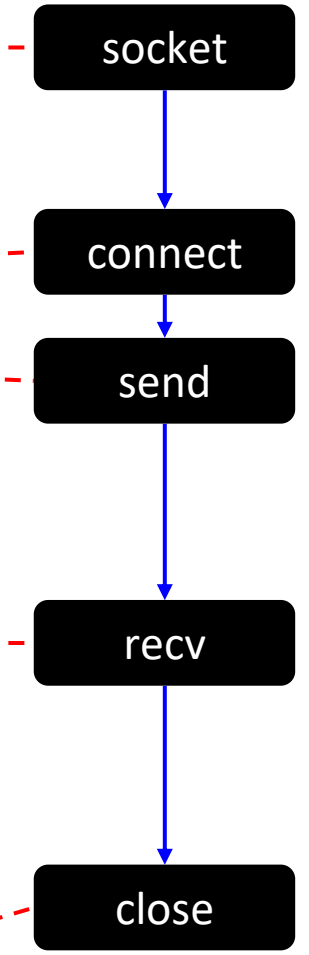
Si ya no hay más data para recibir, escribe toda la data en el archivo (TXT)

```
if not data:
    break
f.write(data)
```

El TXT se recibirá en la carpeta en la que se encuentra el **cliente.py**

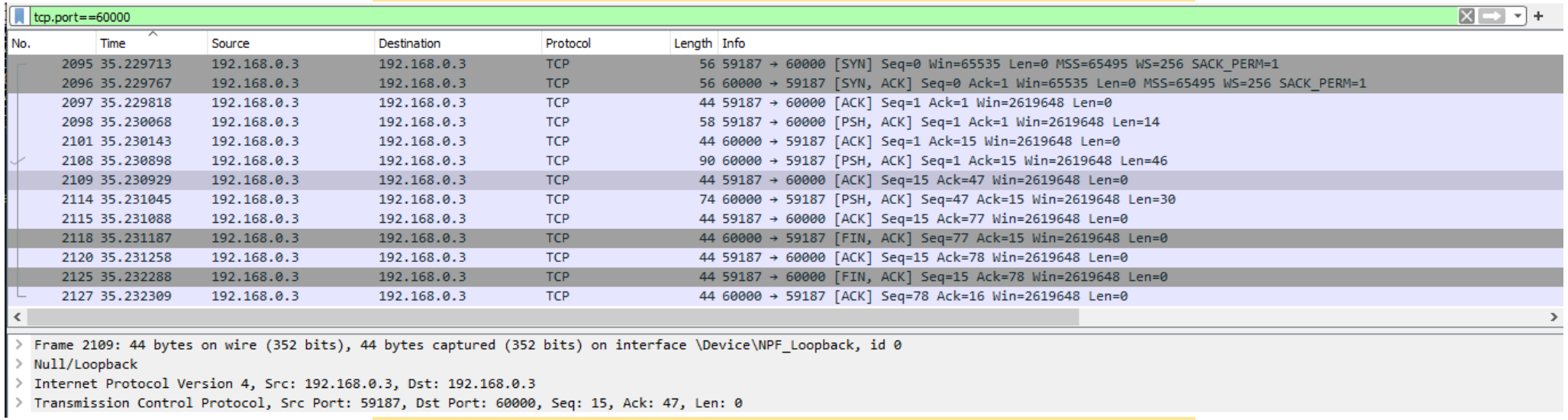
Cerrar el socket

```
print('Archivo recibido exitosamente')
s.close()
print('Conexión cerrada')
```

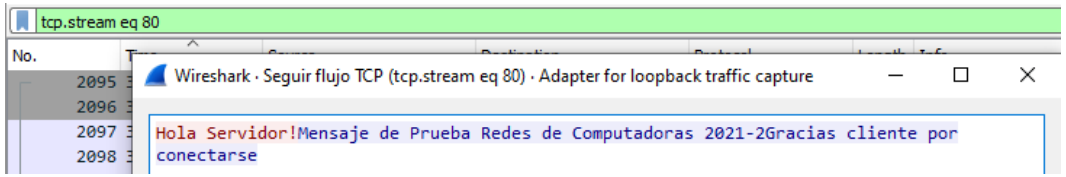


Capturas en Wireshark y en Netstat

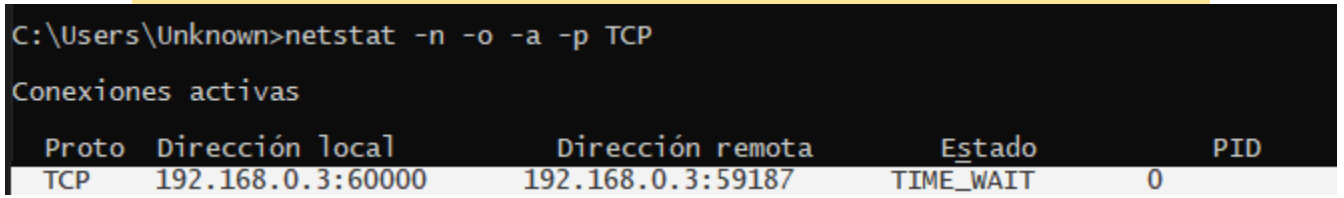
1°: Filtrar el puerto TCP = 60000



2°: Hacer un Follow TCP



3°: Verificar el estado de la conexión



Preguntas / Comentarios

UNIVERSIDAD DE LIMA