

# Seminario 2: Sockets en Python

## Sistemas Distribuidos

Pablo García Sánchez

Departamento de Ingeniería Informática  
Universidad de Cádiz



Curso 2019 – 2020

# Indice

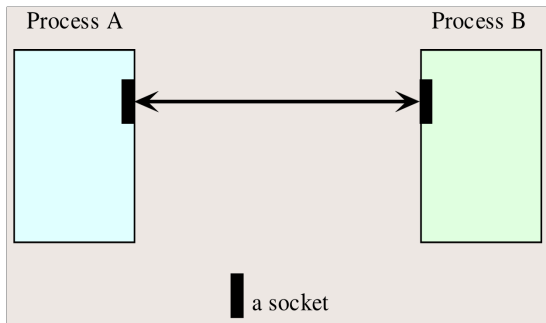
- 1 Sockets
- 2 Sockets UDP
- 3 Sockets TCP
- 4 Bibliografía

# Sección 1 | Sockets

# Sockets

- API para comunicación entre procesos creada en Berkeley
- Implementada en prácticamente todos los SO (es un estándar de facto)
- Es la base para construir cosas más complejas

# Modelo conceptual



# La API Socket

- Crear y usar un constructo denominado “socket”.
- Si se actúa como servidor: asignarle una dirección y un puerto (ojo, los puertos <1024 están reservados)
- Pueden usar el protocolo TCP o el UDP
  - UDP: Socket Datagram. No orientado a conexión. Los paquetes pueden llegar desordenados o perderse.
  - TCP: Socket stream. Orientado a conexión. Se garantiza que todos los paquetes llegan ordenados.

# El módulo Socket de Python

- import socket
- s = socket.socket (socket\_family, socket\_type)
  - socket\_family: AF\_UNIX o AF\_INET
  - socket\_type: SOCK\_STREAM (TCP) o SOCK\_DGRAM

## Sección 2 | Sockets UDP



# Métodos para los sockets UDP

- `s.sendto(mensaje, (HOST, PUERTO))`: Envía un mensaje a ese host:puerto. Puede perderse si no está a la escucha.
- `mensaje = s.recvfrom(bufferize)`: Devuelve un mensaje recibido de cualquier máquina. Es BLOQUEANTE, se queda parado hasta recibir algo.

# Servidor UDP

```
import socket
import os

HOST = 'localhost'
PORT = 1025

s_udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s_udp.bind((HOST,PORT))

print("Me quedo a la espera")
mensaje = s_udp.recvfrom(1024)

print("Recibido el mensaje" + str(mensaje))
s_udp.close()
```

# Cliente UDP

---

```
import socket
import os

HOST = 'localhost'
PORT = 1025 

s_udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

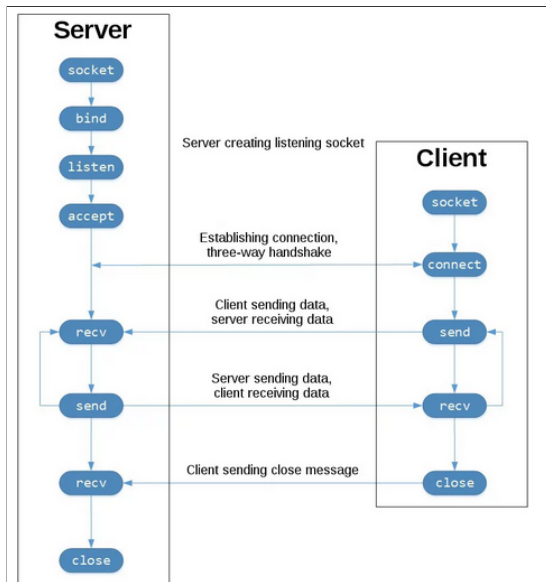
s_udp.sendto("Soy el cliente", (HOST, PORT)) 

s_udp.close() 
```

---

## Sección 3 | Sockets TCP

# Flujo socket orientado a conexión



# Métodos para los Servidores TCP

- `s.bind((HOST,PORT))` Se le asigna IP y puerto al socket que estará a la espera.
- `s.listen()` Prepara el *listener* para aceptar 1 cliente.
- `s_cliente,addr = s.accept()` Se queda bloqueado hasta que un cliente se conecta. Devuelve un nuevo socket para comunicarse con el cliente.

# Métodos para los clientes TCP

- `s.connect((HOST,PORT))` Se conecta a un servidor que esté aceptando conexiones. Si no existe, da un error.

# Métodos para los sockets TCP (servidor o cliente)

- `mensaje = s.recv(1024)` Recibe un mensaje del socket al que está conectado (en trozos de 1024 bytes). Es BLOQUEANTE, se queda parado hasta que recibe algo.
- `s.send(mensaje)` Envía un mensaje por el socket.



# Servidor TCP

---

```
import socket

HOST = 'localhost'
PORT = 1025

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

socketServidor.bind((HOST, PORT))
socketServidor.listen(1)
print("Nos quedamos a la espera...")
s_cliente, addr = socketServidor.accept()

mensaje = s_cliente.recv(1024)
print("Recibo:[" + mensaje + "] del cliente con la direccion " + str(addr))

s_cliente.send("Hola, cliente, soy el servidor")

s_cliente.close()
socketServidor.close()
```

---

# Cliente TCP

---

```
import socket
import os

HOST = 'localhost'
PORT = 1025

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

s.send("Hola servidor")

mensaje = s.recv(1024)
print("RECIBIDO: [" + mensaje + "] del servidor")

s.close()
```

---

## Sección 4 | Bibliografía

# Bibliografía

- <https://realpython.com/python-sockets/>
- <https://docs.python.org/2/howto/sockets.html>
- [https://www.tutorialspoint.com/python/python\\_networking.htm](https://www.tutorialspoint.com/python/python_networking.htm)