Luciano Masuelli

Sistemas Distribuidos

Trabajo Práctico 4

Patrón Request-Reply

5. Modificando el ejercicio

a. **Cambiar el ejemplo para que el mensaje sea diferente / la respuesta del servidor sea diferente.**

Cambio el mensaje del cliente:

```
async function run() {
  const sock = new zmq.Request

  sock.connect(`tcp://${config.server_host}:3000`)
  console.log(`Cliente conectado (?) al server ${config.server_host}`)

  const request = "Hola soy el cliente!"

  await sock.send(request)
  const [result] = await sock.receive()

  console.log(result.toString())
}
```

Cambio la respuesta del servidor:

```
async function run() {
  const sock = new zmq.Reply

  await sock.bind("tcp://*:3000")
  console.log('Reply server iniciado')

  for await (const [msg] of sock) {
    console.log(`Recibido: ${msg.toString()}`)
    const reply = `Soy el servidor, recibí tu mensaje: ${msg.toString()}`
    await sock.send(reply)
    console.log(reply)
  }
}
```

b. **Detener el servidor y ejecutar el cliente, ¿que pasó? Luego ejecutar el servidor. ¿Qué pasó con el cliente? ¿Esto es lo mismo en otros protocolos conocidos? ¿Por qué cree que pasa esto?**

Al listar los puertos se nota que el servidor se mantiene escuchando en el puerto 3000, mientras que el cliente no.

c. **Investigue cómo listar los puertos abiertos tanto en la máquina que hace de cliente como la que hace de servidor con solo un programa andando. ¿Cuál es la diferencia entre ambos?**

El cliente se queda esperando una respuesta del servidor. Al ejecutar el servidor, el cliente recibe inmediatamente la respuesta. Este comportamiento en otros protocolos no es exactamente igual, puede variar según la implementación. Esto sucede porque permite mantener una correspondencia uno a uno entre las solicitudes y las respuestas.

Patrón Publisher-Subscriber

1. ¿Que pasa si solo apago uno de los dos programas?
Si apago el publisher, el subscriber se mantiene esperando mensajes de este. Si apago el subscriber, la máquina publisher continúa enviando mensajes de igual manera.

a. **¿Si no hay cliente, qué pasa con los mensajes publicados?**

Se siguen publicando mensajes y estos se pierden.

b. **¿Hay algo que te llame la atención?**

2. Probá ejecutando dos o más subscriber en dos consolas distintas.
Ambos subscribers reciben los mismos mensajes.

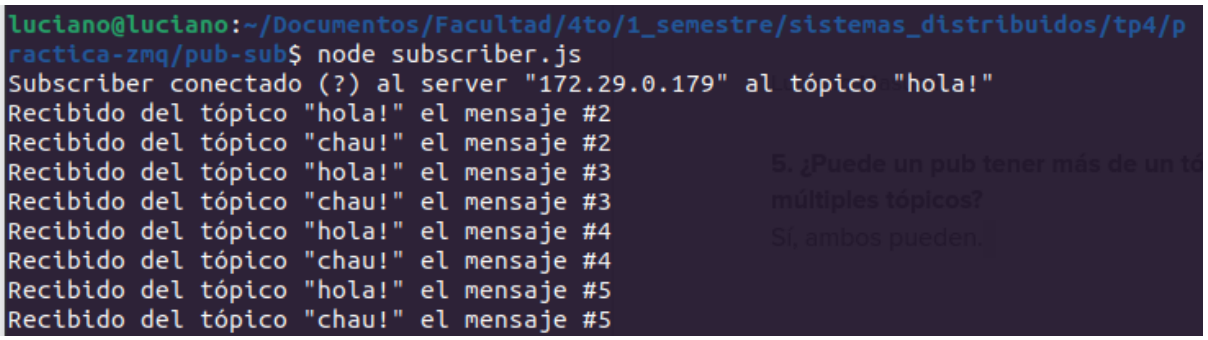
3. Ahora intente probar ejecutar dos publisher en el servidor. ¿Funciona? ¿Por qué no?
No funciona porque el puerto ya está en uso.

4. ¿Qué pasa si cambio el tópico en el cliente o en servidor?
Si los tópicos son distintos, el subscriber no recibe los mensajes.
5. ¿Puede un pub tener más de un tópico en ejecución? Y un subscriber atender múltiples tópicos?

Sí, ambos pueden.

Publisher enviando ejecutando dos tópicos:

Subscriber recibiendo ambos tópicos:



Patrón Push-Pull

1. ¿Qué pasa si solo apago uno de los dos programas? ¿Qué diferencia esto del patrón pub-sub?
Tanto el worker como el producer esperan a que el otro programa reciba o envíe respectivamente.
2.Probá ejecutando dos o más worker en dos consolas distintas. De nuevo, ¿que diferencia esto del patrón pub-sub?
Cada worker recibe un mensaje diferente, a diferencia de pub-sub donde todos los subscribers reciben los mismos mensajes.

Probando portabilidad

Implementé un publisher en python:

```python
import time
import zmq

context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind("tcp://*:5555")
print("Server started")
count = 0

while True:
    count += 1
    message = "Mensaje %s" % count

    socket.send_string(message)
    print("Sent: %s" % message)
    time.sleep(1)
```