Informe TP4

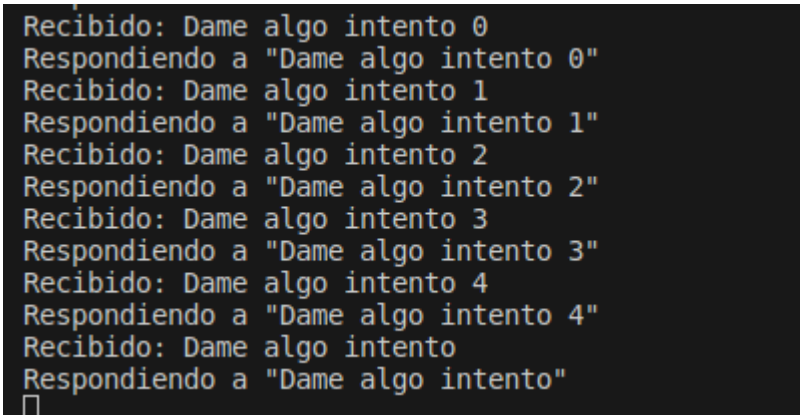
Alumno: Facundo Gaviola

Legajo: 13508

Request-Reply

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

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

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

  for (let i = 0; i < 5; i++) {
    const request = `Dame algo intento ${i}`

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

    console.log(result.toString())
  }
  const request = "Dame algo intento"

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

  console.log(result.toString())
}

run()
```



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?

Cuando ejecutamos el cliente antes que el servidor este se queda esperando la respuesta del servidor. Una vez ejecutamos el servidor, este recibe las respuestas y por consiguiente devuelve los mensajes al cliente. Esto ocurre porque utilizamos el protocolo TCP el cual está orientado a conexiones y garantiza la correcta entrega de la información. Si utilizaramos otros protocolos, como por ejemplo UDP o HTTP, podríamos evitar el bloqueo que se produce en el cliente que se queda esperando la respuesta del servidor.

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?

Es posible listar los puertos abiertos de una máquina mediante el comando **ss -tulpn**. La principal diferencia entre las máquinas cliente y servidor es que en la máquina del servidor nos vamos a encontrar con el puerto 3000 escuchando peticiones TCP.

Publisher-Subscriber

1. ¿Que pasa si solo apago uno de los dos programas?

En el caso de que el programa apagado sea el subscriber nos encontraremos con que el publisher sigue enviando mensajes aunque no haya nadie recibiendolos. En caso de que apaguemos el publisher veremos que el subscriber se queda esperando a recibir algún tipo de mensaje desde el subscriber.

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

Los mensajes siguen siendo enviados por parte del publisher, incluso aunque no haya ningún subscriber escuchando.

b. ¿Hay algo que te llame la atención?

El primer mensaje enviado por el publisher no es recibido por el subscriber.

2. Probá ejecutando dos o más subscriber en dos consolas distintas.

Como es de esperarse, el mensaje enviado por el publisher es recibido por las 2 consolas.

3. Ahora intente probar ejecutar dos publisher en el servidor. ¿Funciona? ¿Por qué no?

Al intentar ejecutar 2 publisher se lanza un error. Esto ocurre ya que estamos intentando utilizar la misma ip y el mismo puerto para enviar los mensajes y por lo tanto genera un error.

4. ¿Qué pasa si cambio el tópico en el cliente o en servidor?

Si cambiamos el tópico en cualquiera de los 2 códigos, entonces ya no se va a poder realizar la comunicación entre ellos.

5. ¿Puede un pub tener más de un tópico en ejecución? Y un subscriber atender múltiples tópicos?

Si es posible que un publisher pueda ejecutar más de un tópico a la vez, siempre y cuando utilicemos un puerto distinto para cada tópico. Lo mismo ocurre para el caso del subscriber, permitiendo que se suscriba a más de un tópico.

Push-Pull

1. ¿Qué pasa si solo apago uno de los dos programas? ¿Qué diferencia esto del patrón pub-sub?

Si apagamos el programa producer entonces el programa worker se queda esperando a recibir mensajes. Si apagamos el programa worker entonces este se queda esperando a realizar la conexión con el producer.
A diferencia del patrón pub-sub, en el patrón push-pull nos encontramos con que se espera a realizar una conexión para enviar los mensajes.

2. Probá ejecutando dos o más worker en dos consolas distintas. De nuevo, ¿que diferencia esto del patrón pub-sub?

En este caso, los mensajes son recibidos por los workers y se van distribuyendo de forma equitativa entre los distintos workers. Esto se diferencia del patrón pub-sub en que dicho patrón no permite tener 2 servidores escuchando mutuamente.