UTE EC

EDUCACIÓN EN LÍNEA

ZeroMQ

Grupo 3

Integrantes:

Carlos Cadena

Jonathan Sánchez

Esteban Perugachi

¿Qué es?

- Es una biblioteca de comunicaciones de alto rendimiento que esta orientada a mensajes, que esta para la construcción de aplicaciones distribuidas

- Está basada en colas de mensajes

- No necesita un bróker intermedio

Qué nos ofrece

- Enviar mensajes intraproceso, ICP,TCP,UDP y multicast
- Patrones de conexión: Publish-Suscribe, Pepeline
- Funcionamiento asíncrono
- Muy rápido
- Soporta mas de 20 lenguajes incluyendo java
- Soporta la mayoría de sistemas operativos

Principales Características

- Usa transportes eficientes como Multicast o IPC
- Usa Batching
- Simplicidad(API muy simple)
- Escalabilidad a través de su diseño de brokerless

Pasos para implementar una capa de mensajería

- Elegir un transporte
- Configurar la infraestructura
- Elegir un patrón de mensajería

Tecnología

- ZeroMQ es un middleware de comunicaciones orientado a mensajes (en inglés, Message-oriented middleware). Este tipo de sistemas utiliza una arquitectura basada en un servidor de mensajes, también llamado broker, que funciona como una aplicación intermedia situada entre los emisores y los receptores de los mensajes.

Tecnología

- Los diseñadores de ZeroMQ se propusieron construir un sistema que consiguiera un mayor rendimiento y una latencia más baja, precisamente eliminando ese broker intermedio y pasando así de una arquitectura de clientes simples (que sólo saben comunicarse con el servidor) y servidores inteligentes (con la lógica de almacenamiento y envío de los mensajes), a otra en la que todos son pares inteligentes que gestionan tanto la gestión de los mensajes como su enrutamiento.

Patrones de Mensajería

- La arquitectura de un sistema distribuido puede describirse utilizando patrones de mensajería. Estos patrones se encargan de especificar y detallar la topología del conjunto del sistema y el flujo de mensajes entre sus distintas partes. ZeroMQ implementa varios de estos patrones. Para ello proporciona pares de sockets preoptimizados para cada tipo de comunicación que se desee realizar. En concreto, los patrones soportados son:

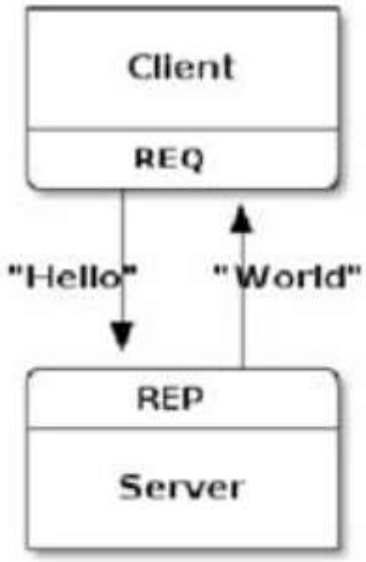
Patrones de Mensajería

- **Pair:** el más parecido a un socket clásico, comunicación bidireccional entre un servidor y un cliente único.
- **Request-Replay:** conecta un conjunto de clientes con un conjunto de servidores. Se utiliza para implementar distribución de tareas y llamadas a procedimientos remotos (RPC).
- **Publish-Subscribe:** conecta un conjunto de publicadores con un conjunto de subscriptores. Se utiliza para la distribución de datos mediante mensajes categorizados (Publish-subscribe pattern).
- **Pipeline:** conecta nodos organizados en pipelines de varias etapas o pasos secuenciales. Se utiliza para organizar flujos de procesamiento de mensajes en paralelo y por etapas.

Ejemplo REQUEST/REPLY

```
import org.zeromq.ZMQ;
public class hwclient {
    public static void main (String[] args){
        ZMQ.Context context = ZMQ.context (1);
        ZMQ.Socket socket = context.socket (ZMQ.REQ);
        socket.connect ("tcp://localhost:5555");
        socket.send ("Hello", 0);
        System.out.println (socket.recv(0));
    }
}
```

```
import org.zeromq.ZMQ;
public class hwserver {
    public static void main (String[] args) {
        ZMQ.Context context = ZMQ.context(1);
        ZMQ.Socket socket =
  context.socket(ZMQ.REP);
        socket.bind ("tcp://*:5555");
        while (true) {
            byte [] request = socket.recv (0);
            socket.send("World", 0);
        }
    }
}
```



Bibliografía:

- Gracia, L. (2013, 25 febrero). Un poco de ZeroMQ. Un poco de Java y +. https://unpocodejava.com/2013/02/25/un-poco-de-zeromq/
- zeromq - Empezando con zeromq | zeromq Tutorial. (s. f.). Riptutorial. Recuperado 21 de enero de 2021, de https://riptutorial.com/es/zeromq

UTE EC

¡GRACIAS!

TRAS CENDE MOS