Desarrollo de un sistema distribuido con Zookeeper

Carlos García Guzmán
Universidad de Málaga
carlosgarciag552@uma.es

1. Funcionamiento básico del sistema

Se ha desarrollado una solución en Python src/app.py que orquesta un conjunto de nodos sensores coordinados mediante Apache ZooKeeper. El sistema cumple con los requisitos de tolerancia a fallos y elección automática de líder.

1.1. Lógica de la Aplicación

Cada instancia de la aplicación realiza dos funciones concurrentes:

1. **Sensor (Thread secundario)**: Genera mediciones aleatorias cada 5 segundos y las publica en ZooKeeper. Se utilizan **znodes efímeros** en la ruta /mediciones/app{id}. Esto asegura que las mediciones de un nodo solo existan mientras este permanezca conectado.
2. **Coordinador (Thread principal/Elección):** Utilizando la receta Election de la librería kazoo. El nodo escogido como líder es el encargado de recolectar los datos que escriben todos los nodos (incluido él) y enviarlos a la API desarrollada en la práctica anterior.

1.2. Funcionalidad del Líder

El líder ejecuta un bucle infinito donde:

- Recupera la lista de nodos hijos en /mediciones.
- Lee el valor de cada nodo, gestionando posibles condiciones de carrera (nodos que se desconectan durante la lectura).
- Calcula la media aritmética de los valores disponibles.
- Envía el resultado a una API externa (http://127.0.0.1:80/nuevo) mediante una petición HTTP GET.

1.3. Robustez

El código incluye manejo de señales (SIGINT) para un cierre ordenado de la conexión con ZooKeeper y bloques try-except para gestionar errores de red al comunicar con la API o inconsistencias temporales en los datos de ZooKeeper.

1.4. Ejemplo de Código

A continuación se muestra el fragmento de código encargado de la actualización de las mediciones en ZooKeeper, el cual utiliza la fuente Cascadia Code configurada:

```
try:
    client.create(path, str(value).encode('utf-8'), ephemeral=True)
except NodeExistsError:
    client.set(path, str(value).encode('utf-8'))
```