Proyecto final de Sistemas Distribuidos
Plataforma de Agentes

Javier Rodríguez Sanchez C-411
Luis Alejandro Rodríguez Otero C-411

September 3, 2024

Contents

1 Introducción

El proyecto propone una plataforma de agentes, la cual permite que se creen soluciones distribuidas utilizando recursos y conocimientos distribuidos. Estos recursos y conocimientos se implementan en forma de agentes, cuya principal característica es que poseen autonomía para decidir realizar cualquier acción. Por tanto la plataforma provee los mecanismos para que estos agentes puedan ser contactados, se realice la interacción con ellos y que estos puedan realizar las acciones que se les piden. Para ello cada agente puede registrarse (registrar sus puntos de acceso y sus funcionalidades) y la plataforma puede otorgarle una identidad única dentro de ella. Además, la plataforma garantiza que se pueda buscar a los agentes a partir de sus funcionalidades. El software cuenta con amplia flexibilidad de usos, permitiendo montar aplicaciones enteras sobre la plataforma. Se garantiza un nivel de tolerancia a fallas 2 en los servicios de la plataforma.

2 Diseño de la Plataforma de Agentes

2.1 Aquitectura

El sistema cuenta con una arquitectura principalmente orientada a servicios. Además cuenta con un middleware que permite mayor nivel de seguridad y abstracción. Toda la información relativa se guarda en una base de datos con arquitectura de anillo de Chord.

Existen 3 componentes principales:

- **Los host de agentes o clientes**: Estos contienen a los agentes de la plataforma, asi como una interfaz gráfica que permite a un usuario interactuar con la plataforma. Cada cliente cuando inicia por primera vez en la plataforma debe proporcionar un id único (nombre de usuario) con el cual pueda ser identificado en la plataforma.

- **Los intermediarios o servers**: Estos permiten la comunicación entre dos clientes o entre un cliente y la base de datos, además de llevar un control sobre los nodos de la base de datos existentes. Debe existir al menos un server para que la plataforma funcione y se sugiere que exista al menos 3 de estos.

- **Los nodos de la base de datos**: Almacenan toda la información de la plataforma, siguen la estructura de un anillo de chord para mayor sostenibilidad. Debe existir al menos 1 para que la plataforma funcione y se sugiere que exista al menos 3 de estos.

2.2 Agentes

Para que un cliente pueda crear un agente, su código debe ser puesto en una carpeta con el nombre con el que se quiera llamar a dicho agente dentro de la carpeta 'Agents'. Dentro de esa carpeta debe existir un archivo con el nombre 'main.py' el cual debe contener:

- **Una función llamada get_info**, la cual no recibe parámetros, y debe devolver un diccionario con las llaves:
  - **name**: Nombre del agente.
  - **description**: Descripción general del agente.
  - **actions**: La lista de acciones que puede realizar el agente.
  - **action description**: Una descripción para cada una de las acciones
- **Una clase con el nombre del agente**, donde contenga una funcion por cada acción que el agente pueda realizar. Una acción debe solamente recibir 3 parametros.

Todo agente que cumpla este contrato será reconocido por la plataforma, y podrá ser registrado en esta.

La implementación de las acciones de un agente quedan a la libertad e imaginacion del usuario, pero debe tener en cuenta que el valor retornado sera interpretado como string, sea cual sea este.

Los 3 argumentos que reciben las acciones son:

- **La función search_agents**, la cual recibe un string y devuelve una lista con todos los agentes activos en la plataforma, con su respectiva información, que ejecutan la funcionalidad especificada en el string.
- **La función call_agent**, la cual recibe un agente, una acción y un string, y devuelve la respuesta de la acción de dicho agente recibiendo el string como entrada.
- **Un string args** con los argumentos de la función.

Mediante este diseño se garantiza una manera simple de implementación de los agentes por parte del cliente, además se brindan todos los medios para una interacción entre varios agentes de ser necesario.

2.3 Protocolo

El protocolo de comunicación está inspirado en el modelo REST, aunque con acciones más específicas de la plataforma. Aqui todos los mensajes son enviados como string encriptados en cadenas de bytes mediante el protocolo TCP_IP, y tienen la forma:

ACCION**1**ARGUMENTO_A**1**ARGUMENTO_B**1**....

Donde la cantidad de argumentos depende de la accion a realizar y ACCION es una palabra clave que identifica una acción o método a ejecutar.

2.4 Coordinación

Una gran ventaja de la plataforma es la no necesidad de procesos centralizados, ya que la inserción, edición, eliminación y consulta solo se realizan sobre una única copia de la información. Cuando un nuevo nodo aparece en el anillo de chord, este busca automáticamente su sucesor y ocupa el lugar que le corresponde. El propio server es el que se encarga de realizar la búsqueda sobre la base de datos por lo que no existió necesidad de la creación de lider.

Cada dato en la base de datos tiene tres copias en nodos diferentes consecutivos, y la sincronización de estos ocurre mediante reloj de lamport. Aqui cada nodo tiene la última hora de sincronización con su sucesor, y cada cierto tiempo envia toda la información nueva posterior a dicha hora.

2.5 Formas de descubrimiento

La plataforma tiene dos vías principales de autodescubrimiento:

- **Ip cacheado**: Cada elemento es creado con una lista de posibles servidores(en el caso de los cliente) o nodos(para servidores y otros nodos de la base de datos)

- **Multicast**: Cada servidor y cada nodo estan suscritos a sus grupos multicast respectivamente. En el caso de los nodos y clientes, cuando se acaban los ip de cache, lanzan un mensaje a los multicast de nodos y servidores respectivamente, y obtienen el primero que responda. En el caso de servidores y nodos, estos estan pendientes a escuchar los latidos de corazón(mecanismo donde cada nodo de la base de datos está constantemente emitiendo mensajes indicando que está vivo) de los nodos de la base de datos, y de esta manera descubren que nodos estan vivos.

2.6 Tolerancia a Fallos

El sistema a partir de 3 servidores y 3 nodos de la base de datos, tiene un nivel de tolerancia a fallos 2 y es consistente y disponible.

Por la parte de los clientes, el sistema no se ocupa de la tolerancia a fallos de estos, ya que sus agentes pueden ser en sí sistemas distribuidos enteros, donde el nodo suscrito sea el lider de dicho sistema. Luego, el sistema solo provee cobertura a las fallas del servidor y la base de datos.

Todos los servidores son independientes entre si, y no guardan ningun tipo de informacion relevante del sistema, por lo que si este cae, cualquier otro ocupara su lugar en la solicitud de sus clientes. Si un servidor cae en medio de una solicitud de un cliente, el cliente detectara que este cayó y buscará otro servidor.

Los nodos de la base de datos por otra parte, contienen información de la plataforma. Cada nodo tiene los ip de sus tres nodos siguientes, por lo que la desaparición de dos nodos nunca desconectará el anillo. Si un nodo desaparece, el que lo detecta es su antecesor inmediato, y este automáticamente estabiliza la red avisando a su nuevo sucesor y recuperando la información del nodo perdido entre los nodos que estaban alrededor de él. En el caso de la finger table de los nodos, como estas se revisan periodicamente, cuando un nodo se detecta como desaparecido es eliminado de toda la finger table al mismo tiempo.

El tiempo de recuperación máxima de un error, en el peor de los casos, es max(2t,2q) donde t es el tiempo que toma detectar que un nodo desapareció durante la actualizacion de la finger table, y q es la duración máxima que puede durar una acción en ejecutarse (En caso de que caigan 2 servidores justo en el momento de dar una respuesta).

3 Conclusiones

En este proyecto hemos explorado el diseño e implementación de una plataforma de agentes distribuida utilizando muchos de los conocimientos adquiridos en la asignatura. La plataforma demuestra un alto nivel de escalabilidad y flexibilidad dando al cliente total libertad en la implementación de los agentes asi como permitiendo la adición o eliminación de agentes sin afectar el funcionamiento general del sistema. Meneja los fallos de forma eficiente teniendo un nivel de tolerancia a fallos 2, lo que significa que en todo momento el sistema es capaz de recuperarse de la falla de hasta 2 nodos de la base de datos de forma simultánea, sin pérdida de información. Se utilizó un protocolo de información práctico y simple, entendible para cualquier persona que deba trabajar con la plataforma.

Se alcanzaron muchas de las metas propuestas al inicio del proyecto, aún asi, todavía existen mejoras que pueden realizarse a la plataforma en trabajos futuros, como nuevas formas de autodescubrimiento o una interfaz gráfica más avanzada.