Viernes 25 de Marzo 2022

PROBLEMA 1/4

Sistemas Inteligentes Distribuidos
Universitat Politècnica de Catalunya
2021-2022 Q2

AUTORES:
CARLA CAMPÀS GENÉ
GIL RALLÓ GENERÓ
JOAQUIN FIGUEIRA CHACÓN

1 INTRODUCCIÓN

El objetivo de esta práctica era crear dos tipos de agentes (Termómetro y Termostato). Mediante el uso de la librería JADE y las diferentes behaviours que esta tiene deberemos tomar decisiones para buscar el mejor método de implementación para que estos dos puedan existir en un sistema multiagente con correctitud.

En cuanto al termómetro deberá ser capaz de tomar la temperatura de usando las siguientes una probabilidad p:

- Con probabilidad 1-(p/100) la temperatura generada deberá estar en el rango [m-r, m+r]
- Con probabilidad (p/100) la temperatura generada deberá estar fuera del rango [m-3*r, m+3*r]

Por otra parte el termostato deberá ser capaz de obtener las diferentes temperaturas de cada termómetro y hacer algún cálculo, en nuestro caso, la media ponderada de cada temperatura que le llega de los termómetros, para poder gestionar la diferencia entre estas. Explicaremos cómo ponderamos estos valores posteriormente.

Además vamos a tener que tomar una decisión de cómo queremos que los termostatos consigan las temperaturas de los diferentes termómetros del sistema multiagente y de esta forma poder tomar decisiones en base de la temperatura media que reciben.

A continuación explicaremos más en detalle las diferentes decisiones que hemos tomado en cada agente para poder lograr los objetivos de esta práctica.

2 TERMÓMETRO

La implementación del termómetro es la más sencilla de las dos, pues dado que utilizamos un sistema de pulling el grueso de las tareas del sistema recaen en el termostato. Por esta razón, lo único que hace el termostato es generar aleatoriamente la temperatura utilizando la fórmula enunciada en la práctica y simplemente esperar a recibir mensajes del termostato para responder a cada uno de ellos con su temperatura. Para hacer lo anterior el agente implementa dos behaviours, un ***tickerBehaviour*** que cada "tick" genera una nueva temperatura (la cual es almacenada en el estado del agente para su posterior utilización), este ticker va a hacerse cada s segundos, pasados como parámetro por el usuario. Los parámetros para calcular la temperatura también serán esos que pasa el usuario (*m, r, p)*. Además tendremos un ***cyclicBehaviour*** que espera y responde a los mensajes del termostato con la última temperatura generada.

Con tal de poder descubrir correctamente los termómetros desde el termostato también hemos reprogramado las funciones ***setup()*** y ***takeDown()***. Estas se encargaran de inscribir el termómetro en el DFService (***setup***) y quitarlo cuando el termómetro pare de funcionar (***takeDown***). De esta forma nos será muy fácil encontrar todos los termómetros disponibles desde el termostato.

3 TERMOSTATO

Para implementar el termostato nuestra primera idea fue un sistema de push para consultar los termómetros. Sin embargo, debido a dificultades de sincronización en el recibimiento de los mensajes y a que el termostato debe monitorizar constantemente el estado de los termómetros para detectar anomalías decidimos que sería más simple y adecuado usar un sistema de pulling.

Este sistema de pulling lo implementamos con un ***tickerBehaviour*** que cada cierto tiempo hace una búsqueda en la plataforma para encontrar todos los termómetros disponibles. Este tiempo no nos vendrá por parte del usuario, por lo tanto deberemos decidir nosotros un número estándar para este termómetro. Como parte de nuestras pruebas, y como queríamos ver resultados rápidos hemos puesto un valor muy bajo (1 segundo). Lo cual creemos adecuado para las pruebas que tenemos que hacer para la asignatura. En el caso que queramos usar esto en una situación real sería correcto incrementar la cantidad de segundos para reflejar un periodo de tiempo que haya algún cambio significativo en temperatura.

Este ***tickerBehaviour*** se encargará de enviar a todos los termómetros un request para que le envíen su temperatura (capeamos el número máximo de termómetros con los que trabajaremos para así poder hacer que la behaviour se ejecute constantemente sin demasiado lag). Se recogen todos los valores de temperaturas de los termómetros y a partir de estos calculamos su media aritmética y su desviación estándar. A partir de estas, siguiendo el mismo formato que el termómetro vamos a aceptar valores en el rango [media - 3*sd, media + 3*sd] y descartamos las que no estén en este rango. El termostato modifica en su estado interno las variables que utiliza para monitorear la temperatura global: la media y la desviación estándar.

Con tal de obtener el valor más adecuado de la temperatura actual vamos a mantener un contador interno de cuantas veces el termómetro ha dado valores fuera de nuestro rango. Esto nos ayudará a crear una media ponderada de todos los termómetros. Para actualizar la media el termostato no hace simplemente una media aritmética, sino que utiliza información sobre la fiabilidad de los termómetros (en concreto, cuántas veces ha dado una lectura correcta) para ponderarla. De esta forma obtenemos una medida de la temperatura menos sesgada por el ruido de los termostatos defectuosos (y por tanto un sistema más robusto). Si un termómetro ha dado más valores incorrectos que correctos no se usa hasta que haya dado más valores correctos que incorrectos.

Tras obtener las métricas anteriores el termostato comprueba con esta nueva información si hay anomalías en los datos. En particular, se asegura que la nueva media global de la temperatura se encuentre dentro del intervalo [a, b] que se pasa como argumento al agente. En caso negativo avisa de esta situación a todos los agentes de tipo "***alarm-management***", tal como indica el enunciado de la práctica.

Como todo, esta forma de calcular la temperatura tiene sus limitaciones. En el caso que haya únicamente un termómetro, asumirá que la temperatura que este da siempre es correcta. Por lo tanto no tendremos ninguna comprobación de que esté realmente funcione.

4 EXTRA

Cuando no haya termostatos disponibles en el sistema vamos a generar un nuevo agente termómetro desde el código y vamos a añadir este en la plataforma donde se encuentran nuestros agentes. Esto lo hacemos a partir de las clases ***AgentContainer*** y ***AgentController***. ***AgentContainer*** nos permite coger la plataforma en la que se encuentra nuestro agente y por lo tanto controlar esta desde el código. Añadimos un agente usando una llamada a la función ***createNewAgent()*** que nos empezará un nuevo agente en el mismo container y nos devolverá el correspondiente ***AgentController***. Con este podemos activar el nuevo agente Termómetro que hemos creado.

Además tenemos que contemplar dos casos distintos para esta acción. En el caso que no se haya creado ningún termómetro anteriormente no tendremos valores para el average de temperatura ni para la desviación estándar, en ese caso la temperatura que se cogerá será directamente la del Termómetro creado. Vamos a ponderar la media como la media entre a y b, los parámetros usados para enviar ***alarm-managemen***t y como desviación estándar usaremos la raíz cuadrada de la diferencia entre a y b. De esta forma nos aseguramos de que haya algún termómetro en todos momentos.

En el otro caso, que se dará cuando creemos este agente después de que se hayan eliminado todos los termómetros o que todos den temperaturas erróneas, el termómetro se añadirá teniendo en cuenta que genere temperaturas con la mediana y la desviación obtenida anteriormente.

Con tal de no manipular los valores reales, en cuanto se cree un nuevo termómetro, el termómetro creado en el código se eliminará de la plataforma. Así consiguiendo que los valores sean lo más realistas posibles.

5 REPARTO DE TRABAJO

Agente Termómetro: Carla Campàs Gené
Agente Termostato: Carla Campàs Gené y Gil Ralló Generó
Documentación:

- *Introducción:* Carla Campàs Gené
- *Termómetro:* Joaquin Figueira Chacón
- *Termostato:* Joaquin Figueira Chacón
- *Extra:* Carla Campàs Gené