

# Programación Distribuida y Tiempo Real

## Práctica 4

1)

**Programar un agente para que periódicamente recorra una secuencia de computadoras y reporte al lugar de origen:**

- a) El tiempo total del recorrido para recolectar la información.**
- b) La carga de procesamiento de cada una de ellas.**
- c) La cantidad de memoria total disponible.**
- d) Los nombres de las computadoras.**

**Comente la relación entre este posible estado del sistema distribuido y el estado que se obtendría implementando el algoritmo de instantánea.**

Desarrollo:

Se crearon 2 agentes:

- **ReporterAgent:** Este agente se encargará de recorrer los distintos contenedores localizados en las distintas computadoras y enviar en cada uno un mensaje al **OriginAgent** con los datos solicitados.
- **OriginAgent:** Este agente está escuchando continuamente por mensajes del reportero y reporta en consola los datos recolectados por este.

El reportero tiene un behaviour por el que cada 10 segundos pedirá al ams (Agent Management System), quien administra los nombres de la plataforma, los nombres de los contenedores disponibles. Luego creará un behaviour por cada contenedor para migrar a este. Esto funciona dado que el movimiento entre contenedores es transparente en jade y dado a que los behaviours son no apropiativos, estos se ejecutan hasta terminar (por lo que se ejecutará un behaviour de movimiento por cada contenedor). También se utiliza `afterMove` para realizar la recolección de los datos y el envío del mensaje al origen.

Para realizar la ejecución debe ejecutarse en consola:

```
java -cp lib/jade.jar:classes jade.Boot -gui -agents  
"origin:OriginAgent;reporter:ReporterAgent"
```

Lo que inicializará los agentes y la gui en el Main-Container.

Posteriormente se inicializan los contenedores en las distintas computadoras y reportero se encargará de encontrarlos y visitarlos cada 10 segundos.

A diferencia de la implementación del algoritmo de instantáneas de Chandy y Lamport, con este algoritmo no se tienen en cuenta los mensajes salientes y entrantes entre las diferentes computadoras, por lo que se podría conseguir un corte inconsistente del estado global, ya que los mensajes del reportero pueden no llegar en orden, lo que iría en contra del orden de precedencia de sucesos que es asegurado por el algoritmo de instantáneas.

2)

**Programa un agente para que calcule la suma de todos los números almacenados en un archivo de una computadora que se le pasa como parámetro. Comente cómo se haría lo mismo con una aplicación cliente/servidor. Comente qué pasaría si hubiera otros sitios con archivos que deben ser procesados de manera similar.**

Desarrollo:

Se crea un agente llamado AdderAgent, el cual recibe por parametro mediante el metodo getArguments() el nombre del contenedor de la computadora que posee el archivo y el nombre del archivo. Una vez los recibe, el sumador migra al contenedor indicado y en el método afterMove() realiza la lectura de los números del archivo y su posterior suma.

Para la ejecucion:

- Iniciar el Main-Container

**java -cp lib/jade.jar:classes jade.Boot -gui**

- iniciar un Container con el sumador indicando el container y archivo a leer

**java -cp lib/jade.jar:classes jade.Boot -container -host localhost  
'adder:AdderAgent(Main-Container, Numbers)'**

En una aplicación cliente/servidor, se tomaría como cliente el origen del agente sumador y como servidor la computadora pasada como parámetro. De esta forma, el cliente le debería pedir al servidor la suma de los números del archivo o debería pedirle el archivo.

En caso de que hubiese más sitios con archivos para ser procesados, con jade el agente podría simplemente migrar a esos contenedores para realizar la suma, en cambio en el modelo cliente/servidor, el cliente debería conectarse a cada computadora con archivos a sumar, que cumplirían el rol de servidor.

Esto último generaría más trabajo y comunicación en el cliente, el cual debe conectarse a todos los servidores. En cambio, en la solución utilizando agentes, no se le otorga más carga de trabajo a ninguna computadora.

**3)**

**Defina e implemente con agentes un sistema de archivos distribuido similar al de las prácticas anteriores.**

**a.- Debería tener como mínimo la misma funcionalidad, es decir las operaciones (definiciones copiadas aquí de la práctica anterior):**

**leer:** dado un nombre de archivo, una posición y una cantidad de bytes a leer, retorna 1) la cantidad de bytes del archivo pedida a partir de la posición dada o en caso de haber menos bytes, se retornan los bytes que haya y 2) la cantidad de bytes que efectivamente se retornan leídos.

**Escribir:** dado un nombre de archivo, una cantidad de bytes determinada, y un buffer a partir del cual están los datos, se escriben los datos en el archivo dado. Si el archivo existe, los datos se agregan al final, si el archivo no existe, se crea y se le escriben los datos. En todos los casos se retorna la cantidad de bytes escritos.

**b.- Implemente un agente que copie un archivo de otro sitio del sistema distribuido en el sistema de archivos local y genere una copia del mismo archivo en el sitio donde está originalmente. Compare esta solución con la de los sistemas cliente/servidor de las prácticas anteriores.**

Desarrollo:

Se creo un agente MigratingFSAgent, el cual recibira como parametros en su creacion el contenedor de donde copiara datos, el nombre del archivo del cual copiar y el nombre de la copia que creará en el servidor. Utilizando estos datos, el agente migrara hacia el contenedor recibido.

Luego de cada movimiento, mediante el metodo afterMove, el agente preguntara si ya ha terminado de copiar el archivo, en caso contrario preguntara si esta en el origen, si no esta en el origen significa que esta en el servidor, por lo que comenzará a copiar el archivo pidiendo el tamaño total de este y leyendo de a 1024 bytes en un buffer. Este buffer lo utilizara para adjuntar al final de la copia del servidor 1024 bytes, y luego se movera hacia su origen.

Al moverse al origen preguntara si termino, en caso contrario, preguntara si esta en el origen, dado que ahora esta en el origen adjuntara al final de la copia local los 1024 bytes y se volvera a mover hacia el servidor.

En caso de haber terminado, no es necesario ir a buscar mas datos al servidor, por lo que adjuntara los datos leidos al final de la copia local y luego se elimina al agente utilizando doDelete.

Para ejecutar:

- Se inicia FSAgent

**java -cp lib/jade.jar:classes jade.Boot -gui fs:FSAgent**

- Se inicia TestFSAgent

**java -cp lib/jade.jar:classes jade.Boot -container -agents tester:FSTestAgent**

A diferencia de la solución cliente/servidor, en la solución de jade el agente se mueve completamente al servidor, pudiendo realizar en este las operaciones que este desee, como consecuencia se ejecutará el mismo código en el cliente y en el servidor, por lo que es necesario la utilizacion de variables para saber donde se encuentra el agente y actuar conforme a esto.

Otra diferencia es que con RMI y RPC, se le pedian servicios al servidor, el cual respondia apropiadamente. En cambio en Jade, se mueve completamente el agente, por lo que se pueden realizar todas las operaciones que se desee en el servidor. En el caso del FS, se pudo leer el tamaño del archivo, leer 1024 bytes y adjuntar 1024 bytes en la primer migracion, lo que usando los otros metodos hubiese tenido como consecuencia la utilizacion de 3 operaciones.