SISTEMAS DISTRIBUIDOS

INFORME DE AEC2: OBJETOS REMOTOS

Antonio Luis Ojeda Soto

1.Explicación del proceso de la práctica.

Después de haber estudiado la unidad 6 y probado los códigos que vienen en la misma, me dispongo a iniciar la práctica habiendo también leído y visto el vídeo previamente.

Como en la anterior práctica empiezo primero intentando hacer el **Twitter casero** de forma básica. Para ello comienzo con la interfaz del servidor y luego implementando los métodos de ésta. Los compilo y aquí debería de crearse el resguardo (**stub**) y el esqueleto (**skeleton**) en la parte servidora, pero hoy en día no hace falta usar la herramienta rmic para ejecutar un mandato, ya que todo esto se hace de forma abstracta por el sistema estando esta herramienta obsoleta. Una vez que tengo esto me dispongo a definir el cliente objeto y el servidor objeto modificando un poco el código ofrecido en la unidad 6 para su forma básica. Lo pruebo y funciona correctamente, mandando todos los mensajes que quiera el cliente.

Una vez que concluyo la parte básica me dispongo a realizar la parte de callbacks.

Lo intento hacer de forma básica sin los métodos **listaApodos()** y **follow()**, solamente que se puedan enviar los tweets a los clientes que estén registrados. Comienzo por analizar el código ofrecido en el libro sobre **callbacks**. Una vez estudiado y analizado, comienzo a desarrollar en papel cómo será el código y qué tendrá cada parte de este. Finalmente, con cambiar unas cuantas líneas y modificar los métodos ofrecidos que acabo de implementar, lo compruebo y funciona correctamente. Lo dejo aquí y no sigo más, ya que veo que con los dos métodos mencionados anteriormente se complica más y por lo tanto me llevaría más tiempo que no podría dedicar a otras asignaturas. Quería hacer por lo menos hasta aquí para ver el funcionamiento de RMI, tanto en su forma básica como con callbacks. Me ha parecido, igual que la anterior, una práctica interesante. Pero esta vez me ha costado más que la anterior por el nivel de abstracción el cual es mayor aquí.

2.Documentación del código implementado.

Para AEC2_RMI_Basico: en el fichero interfaz **ServInt** pongo la plantilla de los dos métodos a implementar. En la clase **ServImpl** implemento los dos métodos de la interfaz. En el método **registro()** se registrará el apodo pasado por argumento y se pasará un mensaje, tanto a servidor como al cliente, de verificación de que el apodo ha sido registrado. En el método **tweet()** se le devolverá al cliente la hora a la que se mandó su mensaje confirmando que fue recibido. En servidor objeto **ServidorBasico** inicializaremos el registro de objetos mediante el método **inicioRegistro()** y registraremos los objetos con **Naming.rebind** pasándole como argumentos la URL y el objeto que queremos registrar. En el cliente objeto **ClienteBasico** se buscará el objeto remoto con **Naming.lookup** y una vez encontrado se invocarán los métodos del objeto remoto: **registro()** y **tweet()**.

Para AEC2_CallbackBasico: Comienzo por eliminar la clase **Sensor** del ejemplo del libro. El servidor lo dejo tal cual como estaba en el libro, esto es, que inicie un registro y registrar los métodos de la interfaz del servidor. Con la salvedad que aquí incluyo una pequeña modificación de unidades anteriores: donde se pregunta al usuario que

especifique un puertoRMI por consola, si no se introduce nada, que dé por defecto el puerto 1099, especificado en el libro.

La interfaz del cliente la llamo **CallbackInter** y su implementación **CallbackImpl**. Aquí, al igual que en el ejemplo del libro, modifico el método **notificaMe()**, pasándole como argumentos el apodo y el mensaje.

La interfaz del servidor la llamo **ServInt** y su implementación **ServImpl**. Aquí pongo tres métodos: el método **registro()**, muy similar al método registro del libro, pero en este caso, además del callback, también registro el apodo. El método **eliminarReceptor(),** en donde pasa lo mismo que en el método del libro, pero también eliminando, además del callback, el apodo. Y por último el método **tweet()**, el cual es una modificación del método **darAlarma()** del libro. Aquí le pasaremos como argumento el apodo y el mensaje en lugar de sensor. Y en la llamada al método **notificaMe()**, le pasaremos también estos dos argumentos. La última modificación de este método es pasarle un mensaje al cliente para indicarle que llegó correctamente su tweet.

La parte del Cliente la llamo **UsuarioReceptor** y consiste en varias modificaciones que la parte del cliente del libro, **Receptor**. La primera modificación consiste en que si no se introduce una dirección y un puerto te dé por defecto “localhost” y “1099” respectivamente. La segunda modificación consiste en pedir al usuario que escriba por consola su apodo. Este quedará registrado en el servidor junto con el callback. Y como última modificación, se le pedirá al usuario que introduzca un tweet para mandárselo al resto de clientes registrados. Esta petición la introduzco en un bucle while para que el usuario pueda introducir tantos tweets como quiera, hasta que se salga del bucle con el tweet “fin”.