PROJECTES EN ARQUITECTURA DISTRIBUIDA

Paula Rodriguez y Javier Gaig

Ejercicio 0:

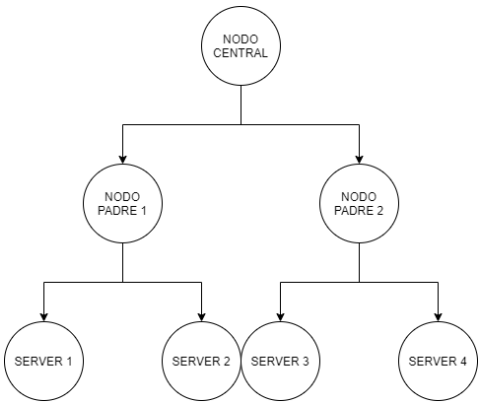
1. **Explica el funcionamiento del algoritmo**

La lógica del algoritmo esta basada en la única información que tenemos:

- Se trata de una aplicación para un banco
- Se han de implementar las funcionalidades de obtener y actualizar el valor.

A partir de aquí deducimos que lo importante es la velocidad de la aplicación, ya que, al tratarse de un banco, creemos que es prioritario la respuesta rápida a las funcionalidades en contra al coste computacional que puede llegar a costar. Una vez aclarado esto decidimos que, entre las dos funcionalidades, es preferible priorizar que cuando pida el valor me lo devuelva cuanto antes mejor, ya que una actualización de este valor, que tarde un poco más o menos, nos parece un precio asequible.

Una vez teniendo esta base clara y sabiendo que teníamos que implementarlo mediante sockets, llegamos a las que es nuestra arquitectura del proyecto.



En este caso se trata de un ejemplo con 4 servidores. En cualquier ejemplo nuestra arquitectura siempre estará formada por estos 3 niveles:

- **Nivel 3 - Servidores:** los servidores son los encargados de ir realizando el conjunto de peticiones de obtención y/o actualización del valor. Para realizar esta petición se realiza un bucle que va conectándose y desconectándose continuamente de su nodo padre (N2), para así poder enviar todas las funcionalidades a tiempo real.

En caso de tener que actualizar el valor, al conectarse el servidor a su padre le pasa este valor actualizado. En caso de que quiera obtener el valor total, se conecta a su padre, le manda una trama informativa de que espera recibir este valor, y se queda esperando a recibirlo.

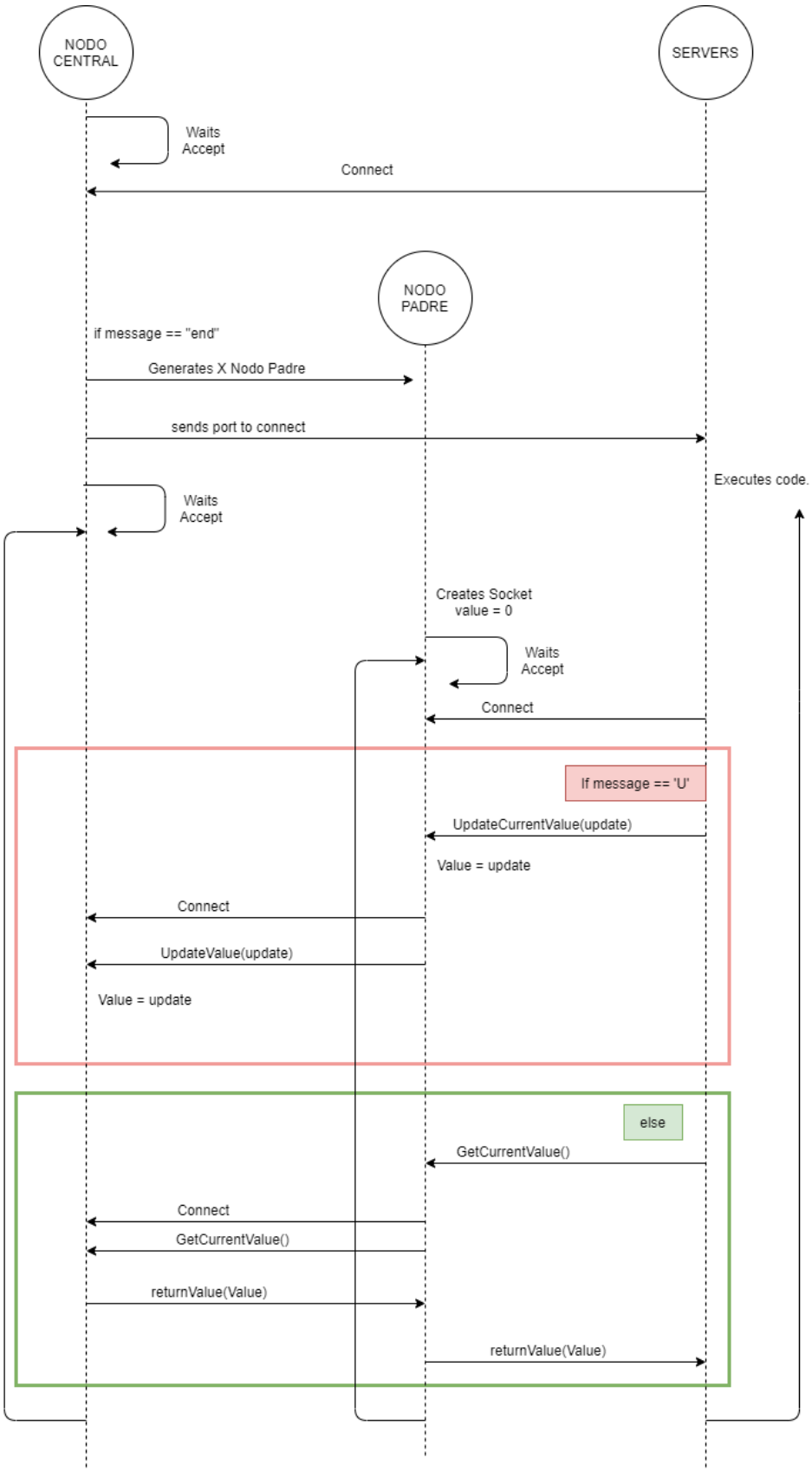
- **Nivel 2 – Nodos Padre:** los nodos padres tienen 2 funciones, ya que están conectados en modo “cliente” con el nodo central (N1), y en modo “servidor” con los servidores (N1). Su primera función es, cada vez que recibe una actualización del valor, se conecta al nodo central y le pasa este valor actualizado. Al recibir una trama de que un hijo quiere consultar el valor, mantiene su conexión abierta, realiza otra con el nodo central (N1) y, al igual que en el N3, manda una trama informativa de que espera recibir este valor mientras se queda esperando a recibirlo.

- **Nivel 1 – Nodo Central:** único e irremplazable. Este nodo es el encargado de recibir todas las peticiones de la misma forma que en el N3 con sus padres, es decir, se mantiene en un bucle esperando una conexión que van haciendo los nodos padre de forma cíclica.

PROJECTES EN ARQUITECTURA DISTRIBUIDA

Paula Rodriguez y Javier Gaig

Diagrama de secuencia del algoritmo:



2. **¿Qué limitaciones tiene el algoritmo?**

    - Si cae un nodo, se pierde la conexión de todos aquellos clientes que estén conectados a dicho nodo
    - Si cae el nodo central, el problema es aun mayor, ya que, no se podrá actualizar ningún valor ni consultar alguno.
    - Cuello de botella, el central se puede saturar si hay muchos conectados y recibe muchas peticiones.

3. **Como afectaría al algoritmo que los servidores que hacen el update puedan ejecutar otras operaciones (resta, multiplicación...) y/con otros números (-1, 2, 100, 0, ...)?**

Inicialmente se tendría que *parsear* la operación que se indica hacer, es decir, el nodo central, al recibir esta trama, tendría que traducirla a multiplicación, división, resta..., y, modificar el valor.

En función de la compilación, ha habido algunos casos en los que el programa hace algún update antes de un get y viceversa. En estos casos si que afectaría al resultado del algoritmo.

4. **¿Como reacciona el rendimiento del algoritmo cuando se incrementa arbitrariamente el número de procesos?**

El mayor problema que se podría tener es la sobrecarga del nodo central al este recibir muchas peticiones. La estructura está pensada para tener una carga equilibrada de trabajo en todos los nodos del N2, pero llegará un punto en el que todos los nodos padres tendrán una cantidad elevada de hijos, saturando bastante el sistema.

5. **¿Qué haría faltar cambiar para permitir que nuevos servidores se añadieran a media ejecución?**

Si el número de servers cambia una vez ya ejecutado el programa, lo primero que se debería hacer es recalcular el número de nodos padre que hay en ejecución. Si este número coincide con los ya existentes, se debería saber de ellos es el que tiene el menor número de servidores conectados a él, con un simple *array* que nos diga el número que tiene cada uno bastaría, y se podría ejecutar sin problema.

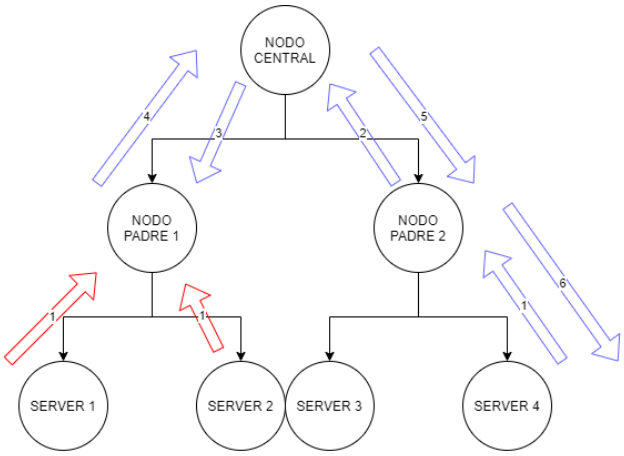
El problema viene cuando el número de nodos padre es diferente al calculado con la nueva inserción de un servidor, donde se tendría que redistribuir todo el árbol.

ANEXO:

Distribución inicial de los nodos padre según el número de Servers:

Como hemos dicho, hemos de priorizar la funcionalidad de obtención del valor, y para conseguirlo hemos tenido que encontrar una relación entre ésta y la de actualización.

Vamos a llamar 't' el tiempo que se tarda en enviar una trama por el socket entre los diferentes niveles. Teniendo en cuenta que, para la obtención del valor, se manda una trama de 2 Chars (32bits) y para la actualización del valor manda un entero (32bits), afirmamos que ambas funcionalidades tardan X*t tiempo (teniendo como X el número total de saltos hasta acabar la función).



En el ejemplo anterior, el tiempo en realizar una petición para obtener el valor (color azul), es de 6t.

1. El Server hace la petición a su padre
2. Su padre se la pasa al nodo central pasándole su acumulado
3. Nodo central le pide al otro padre su acumulado.
4. Nodo central recibe este acumulado y hace la suma
5. Nodo central le pasa al padre el resultado
6. El Server recibe el valor actualizado

Mientras que el tiempo que va a estar ocupado cada nodo padre (rojo) va a ser mínimo 2t (1 petición por cada hijo).

Para dar preferencia a la función de obtención del valor y tener un árbol balanceado a nivel de carga de trabajo de los padres, la lógica impuesta en el algoritmo es que, cada vez que la carga media de trabajo de cada padre (tiempo mínimo que va a estar ocupado/flechas rojas), sea igual al tiempo necesario para realizar una petición de obtención del valor, se ha de añadir otro padre y balancear entre los diferentes hijos. Ya que, si queremos que la obtención del valor sea rápida, no podemos permitirnos tener un padre que esté ocupado el mismo tiempo que se tarda en realizar toda la función.

PROJECTES EN ARQUITECTURA DISTRIBUIDA

Paula Rodriguez y Javier Gaig

La secuencia del grafo al ir añadiendo servers sería (Tobt: tiempo obtención valor, Tpad: tiempo mínimo que el padre estará ocupado):