Propuesta de solución para un sistema de archivos basado en tags.

1. Arquitectura del sistema

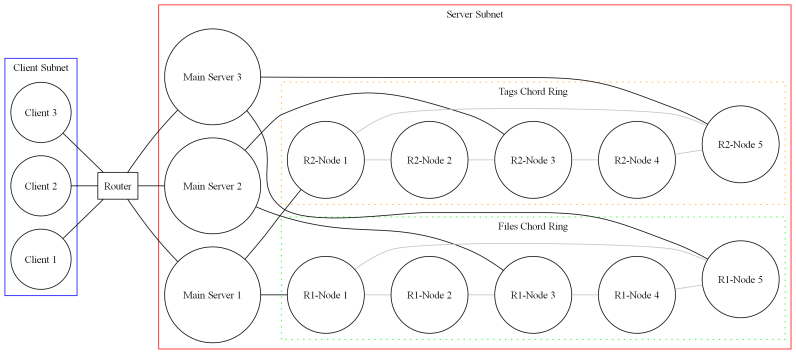

Figura 1: La arquitectura

Habrán dos redes, una para clientes y otra para las componentes del servidor, conectadas por un router. La arquitectura del lado del servidor se puede describir como orientada a servicios. Habrán S servidores principales, cuyo rol será procesar los comandos mandados por los clientes. En estos servidores no se guardarán los datos del sistema de archivos, estos serán almacenados en dos DHT distintas, que serán anillos de Chord. Una DHT para almacenar por cada tag, el nombre de los archivos que lo tienen, y otra DHT para almacenar por cada archivo, su contenido.

2. Procesos

En cada cliente estará corriendo un programa cuyo objetivo principal es mandar los comandos al servidor, junto con el contenido de los archivos de ser necesario, y recibir los archivos del servidor. Todos los procesos que corren del lado del servidor (servidores principales y nodos de DHT) tendrán una estructura similar: un hilo que espera conexiones, el cual puede crear otros hilos para atender dichas conexiones, lo que consiste principalmente en meter comandos en una cola, y un hilo que va sacando comandos de esa cola y ejecutándolos.

3. Comunicación

La comunicación se basará en Sockets. Los clientes mandarán los comandos que introduzca el usuario como un string a los servidores principales. Los

servidores se mandarán también comandos entre ellos, lo que de más bajo nivel que los de cliente-servidor; por ejemplo, entre dos nodos de la DHT pueden enviarse algo como "Encuentra el nodo antecesor de la llave X" (codificado de manera más compacta claro).

4. Coordinación

La coordinación dentro de un mismo proceso la garantiza principalmente el hecho de que haya un solo hilo ejecutando acciones, y que la cola de acciones sea thead-safe (lo que se garantiza con mutex lock/unlock). La toma de decisiones distribuidas es necesaria dentro de los anillos de Chord, por su propia definición.

5. Nombrado y localización

La correcta localización de los archivos y sus tags lo garantizan las DHTs. Para que cada servidor pueda tener localizado a las DHT, este debe conocer la dirección de al menos uno de sus nodos, y si se quiere garantizar un nivel k de tolerancia a fallas, debe conocer la dirección de $k+1$ nodos. Igual sucede con los clientes para tener correctamente localizados a los servidores.

6. Consistencia y Replicación

En nuestro problema, la consistencia y replicación son principalmente necesarias tenerlas en cuenta en las DHTs. La replicación se garantiza, en esencia, guardando un par (llave,valor) en lugar de solamente en el sucesor de la llave, en los $k+1$ sucesores (para nivel de tolerancia k). Es necesario tener en cuenta el tema de la consistencia por ejemplo cuando un nuevo nodo se une a la red de Chord, ya que no solo hay posicionarlo correctamente en la red (encontrar su antecesor y sucesor), sino que hay que copiarle las llaves que le corresponderían.

7. Tolerancia a fallas

La tolerancia a fallas se garantiza principalmente con que si un nodo necesita tener siempre disponible otro nodo con tal rol, que conozca la dirección de $k+1$ nodos con dicho rol (para un nivel k de tolerancia a fallas). Así por ejemplo, cada cliente conocería la dirección de $k+1$ servidores principales, cada servidor principal conocería la dirección de $k+1$ nodos de cada anillo de chord, y cada nodo de cada anillo de chord conocería la dirección de sus $k+1$ sucesores.

8. Seguridad

El problema no requiere autenticación. En teoría, para cada comunicación se podría hacer un intercambio de llaves y usar criptografía asimétrica.