Escuela de Ingeniería Informática
Sistemas Distribuidos
Prof. Luis Expósito

Proyecto II

Venezuela a nivel nacional está sufriendo de muchos problemas de suministro eléctrico, y los informáticos de la UCAB se están viendo afectados por estos "apagones" dado que el sistema de manejo de versiones (SMV) con el que cuentan está centralizado en un solo servidor, lo que resulta ineficiente y peligroso, es por esto que se requiere que se diseñe e implemente un SMV distribuido, tolerante a fallas, que permita realizar *commits*, *updates* y *checkouts* (de acuerdo a los términos usados en SVN).

Un SMV es una herramienta que facilita la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico).

El control de versiones se realiza para controlar las distintas versiones del código fuente dando lugar a la sistemas de control de código fuente o SCM (siglas del inglés Source Code Management). Sin embargo, los mismos conceptos son aplicables a otros ámbitos como documentos, imágenes, sitios web y otros.

Existirá un servidor principal encargado de recibir las solicitudes de los clientes y ejecutar las acciones requeridas de acuerdo al servicio solicitado por el cliente servidores de almacenamiento encargados de mantener copias de los archivos controlados por el SMV.

Los servidores de almacenamiento podrán ser pasivos (sólo almacenan los archivos) o activos (pueden responder a alguna o todas las funcionalidades). Esta decisión de diseño es responsabilidad de cada grupo.

El SMV debe ser k-tolerante a fallas silenciosas, por lo tanto todos los archivos que se guarden en el SMV deberán replicarse en $k+1$ servidores. K es un parámetro del sistema.

Especificaciones Técnicas:

***update, chechout*:** Cuando un cliente solicite un determinado archivo, se le entregará la última versión de ese archivo. Para esto es necesario asignarle a cada archivo un *timestamp* y ante la solicitud, retornar la versión con mayor *timestamp*.

***commit*:** Cuando un cliente desee guardar una versión del archivo, se crearán *k+1* copias entre los servidores.

El servidor principal debe asegurarse de asignar balanceadamente las copias.

Los servidores deberán mantener en todo momento una lista con la información actualizada de los demás servidores: nombre de la máquina, nombres de los archivos que almacena con su versión y cualquier otra información que consideren útil.

Si el servidor principal falla, los servidores de almacenamiento deben iniciar un proceso de elección para sustituirlo.

La comunicación de control entre los servidores se realizará a través de un *socket* conectado a una dirección multicast, a la cual se suscribirán los servidores al momento de levantarse. Los mensajes que se envían a esta dirección pueden ser multicast, de importancia para todos, o unicast, dirigidos a un servidor específicamente (el equipo puede proponer otro modelo de comunicación).

La comunicación correspondiente a los servicios que se proveen al cliente, se realizará con el servidor principal a través de Java RMI seguro u otro modelo de comunicación que el equipo proponga.

El cliente, puede ser una interfaz gráfica muy sencilla o líneas de comandos.

Es claro que debe haber un proceso de sincronización de relojes entre los servidores para manejar las versiones de los archivos que mantiene el SMV.

Deben definir la estructura de datos y el protocolo necesario para cumplir con las funcionalidades requeridas. Esto da cierta libertad a la hora de definir el tipo de interacción que tendrán los servidores entre ellos.

Entregar: Informe en el que se indiquen los detalles relevantes en cuanto a la implementación, así como también el código del proyecto.