75.74

Sistema Distribuidos 1

Paulo César Cuneo

84840

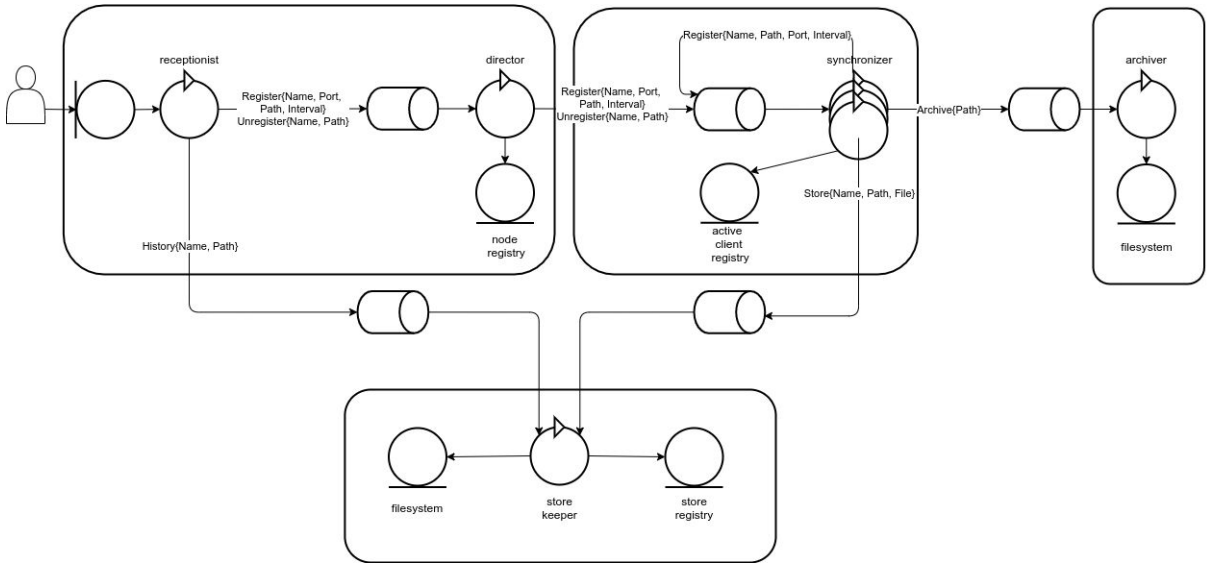
TP1

Backup Server

Correcciones

Índice

Diagrama de robustez



El Usuario del sistema emite comando con netcat contra un servidor principal encargado de coordinar las operaciones.

Todo los comandos son recibidos por un thread Receptionist, que es el encargado de hacer el dispatch del comando hacia a otros threads si correspondiera.

Para las peticiones de mostrar historial backups, es el mismo Receptionist quien busca en el historial y devuelve el resultado.

Para registrar y desregistrar nodos, el Receptionist envía el mensaje al thread Director, este está encargado de administrar los procesos Synchronizer.

Los thread Synchronizers son los encargados de la programación de los backups para cada nodo y path dados, es decir, para un intervalo de tiempo dado ejecutan una petición contra los procesos Archiver, que están corriendo en los nodos a backupear.

El proceso Archiver recibe por comando el path a archivar, y genera un tar con el contenido del path, antes de enviar el tar al Synchronizer, calcula el hashmd5 y si el hash cambio responde transfiriendo el archivo, en otro caso simplemente responde un ACK.

En caso que el Synchronizer reciba una transferencia de archivo, además de persistirla, avisa al thread StoreKeeper que se ha guardado una nueva versión del path.

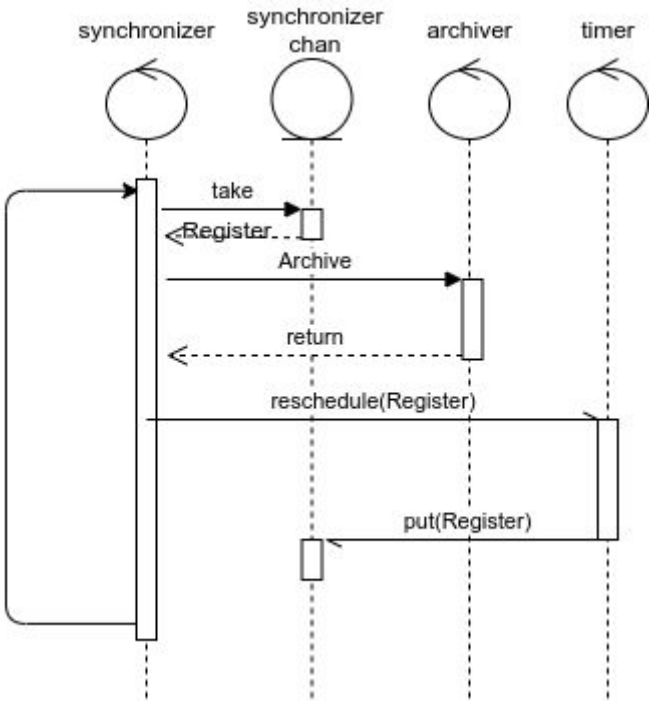
El proceso StoreKeeper al ser notificado verifica que solamente se estén guardando las ultimas 10 versiones del path en cuestión.

Los threads dentro del servidor/coordinador o el nodo/cliente se comunican entre sí mediante channels.

La comunicación entre Synchronizer y Archiver es mediante una conexión TCP que se mantiene abierta durante toda la vida de ambos.

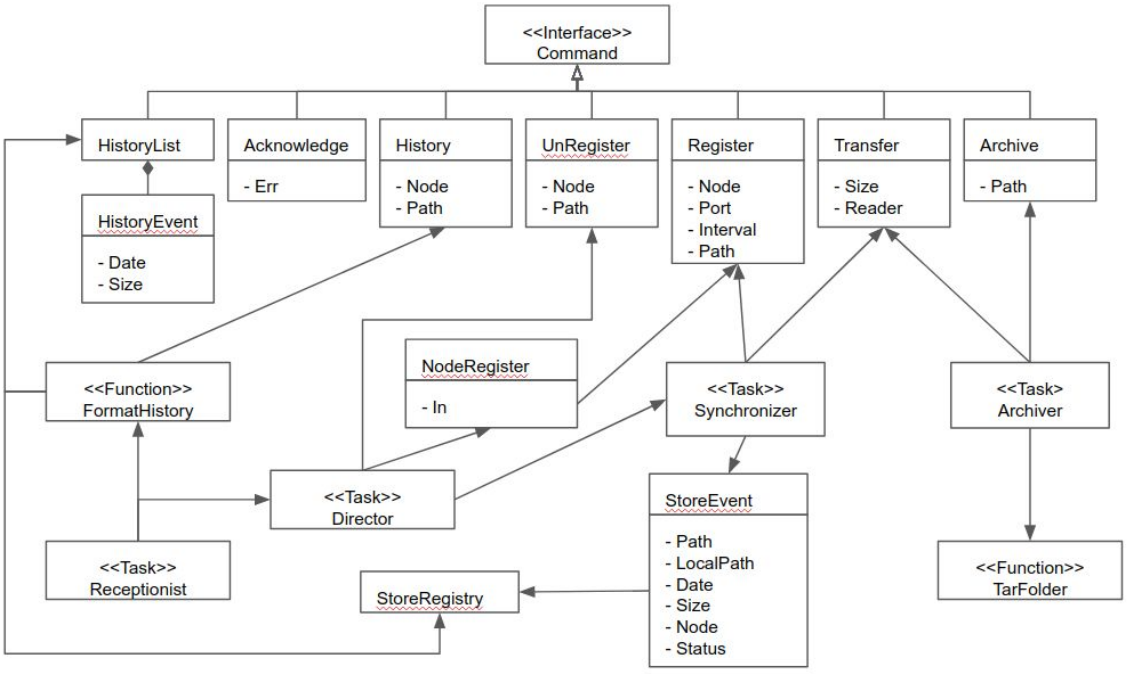
Existe un pool N Synchronizer, todos leen de la misma cola.

Cuando un synchronizer toma un registro de la queue lo procesa(yendo a buscar el archiver y pasándolo al storekeeper) y luego usando time.AfterFunc lo vuelve a encolar cuando se cumple el intervalo.



Hubiera sido mejor tener una gorutina dedica a scheduling, y alimentando la cola de synchronizer, pero por una cuestión de tiempo no se llegó cambiar el diseño.

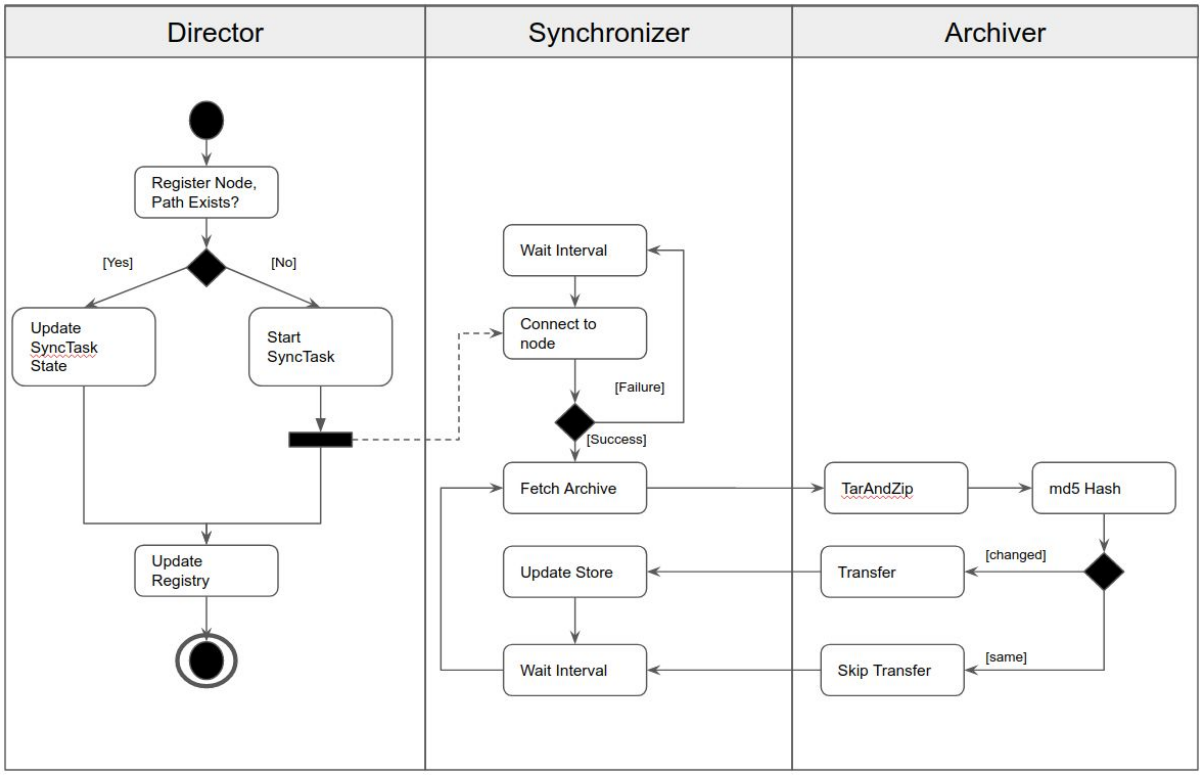
Diagrama de clases



La comunicación entre los distintos threads se hace pasando Commands a través de channels, estos representan peticiones o respuestas según sea el caso.
Algunas entidades no importantes no están implementadas como clases si no funciones/tareas.
Vale aclarar que Receptionist en la implementación actual conoce a casi todas las implementaciones de Command.
Además de los comandos tenemos, 2 conjuntos de estructuras para mantener el histórico de los backups realizados y para mantener el estado de las conexiones contra los nodos, respectivamente.
StoreRegistry y StoreEvent representan el historia de backups realizados.
NodeRegistry y NodeEntry corresponde al estado de los nodos sobre los cuales se está sincronizando.

Diagrama actividades

Backup de un nodo



Comentarios

El manejo de conexiones TCP está implementado usando un pool de goroutines, sin embargo se crea un thread sincronizador por cada nodo a sincronizar, se puede hacer algo más sofisticado utilizando priority scheduling y work stealing, pero creo que excede la complejidad sobre el beneficio.

Solo se permite un path a backupear por nodo, que se puede modificar volviendo a registrar el nodo, pero al mostrar el historial esta información no se está mostrando.

Se está usando el mismo timer para verificar si un nodo se recupera y para scheduling de la sincronización, lo hice así para sacar algo que funcione, pero dista de ser ideal.
Sería mejor, que si se detecta un nodo caído, el Synchronizer des-registrará el nodo y se finalice, una vez que el nodo vuelve a la vida, este mismo puede pedir la registración contra el coordinador.

El sistema actual no tolera una caída del coordinador porque no se está persistiendo el estado de los StoreRegistry y NodeRegistry. Idealmente al iniciar el coordinador este podría recuperar el estado anterior, relanzando todas las sincronizaciones que se estaban corriendo por última vez.