Informe sobre la arquitectura del proyecto de distribuidos.

Primero veamos las particularidades de nuestro proyecto, siendo los requisitos de la app de sporify distribuidos solo los de escuchar, subir y buscar canciones hay algunas que se nos facilitan, al no tener que editar o eliminar canciones nos ahorramos problemas de desincronización de datos por ejemplo. Sobre las arquitecturas hay dos propuestas que hemos pensado, como esta es una evaluación mas oral que escrita pensamos discutir el grueso del contenido de manera precensial. Plasmaremos en este documento las ideas generales en las que nos basamos.

Kademlia como DHT :

Nuestra idea principal idea para distribuir los datos, su búsqueda y replicación es usar una red de kademlia. Ya que le da solución a los problemas antes mencionados.Usando los k-buckets de kademlia para la replicación y en la red de kademlia en principio no necesitariamos roles o algoritmos de concenso, porque no hay problemas de sincronización. En caso de necesitarse se podria implementar el algoritmo de concenso raft en la red de kademlia. Así se resolvería el problema de la sincronización. En nuestro proyecto se consideran dos canciones iguales si tienen el mismo título, artista y álbum. asi que el identificador único de cada canción estaría compueto por un hash de esos tres atributos.

Procesamiento dentro de un nodo de kademlia :

Cada petición a un nodo de kademlia se ejecutará en un hilo diferente en principio aunque sabemos que python solo simula el multithreading. En un futuro podriamos analizar que se puede ejecutar como procesos independientes sin mucho coste para mayor eficiencia. En este caso no hay demasiados recursos que puedan compartir los hilos como para que ocurran condiciones de carrera pero en todo caso habria un lock para cada uno de esos recursos.

Comunicación entre nodos de la red de kademlia :

Tenemos pensado usar RPC por su simplicidad y la abstracción que proporciona, haciendo posible interactuar con nodos en la red como si fuera un objeto en memoria. Cada nodo puede enviar peticiones a otros nodos y recibir peticiones. Ahora, a la hora de implementar el protocolo hay dos variantes que tenemos: Como cada petición se ejecuta en un hilo diferente a partir de peticiones que llegan al nodo, algunas de esas peticiones necesitan hacer a su vez peticiones a otros nodos. La primera opción es que cada hilo habra un puerto diferente para hacer peticiones a otros nodos y recibir el resultado por ese mismo nodo, o que cada nodo tenga un puerto para hacer peticiones y otro para recibir, eso significa que el socket para hacer peticiones a otros nodos es un recurso compartido, pero su uso no es extendido, ya que solo se usa para mandar la peticion, la respueta se recibiria por el hilo que escucha, osea el puerto de escuchar peticiones tambien escucharia respuestas a peticiones que haga ese mismo nodo.

Comunicación entre cliente y servidor

Primero entendamos que el proyecto esta hecho de manera que todo el codigo representan dos capas independientes de la applicación y que son facilmente removibles o agregables, Las capas de la aplicacion en orden son: Interfaz, Middleware, Vistas, Capa Distribuida, Servicios. Siendo las capaz Interfaz, Vistas y Servicios indiferentes a si el sistema es distribuido o no. La interfaz hace las peticiones al middleware como si se las estuviera haciendo directamente al backend. El middleware es el que esta consiente de que el backend es un sistema distribuido y tiene varias direcciones ip de servidores y trata de conectar con alguno, se comunica con la capa de vistas a traves de http. A su vez las vistas le hacen la peticion a la capa distribuida como si se la estuviera haciendo a la capa de servicios, la capa distribuida es la que decide si acceder a servicios o debe pedir la inforamacion a otro nodo en la red.

Otros temas :

Sobre la tolerancia a fallas, la forma de localizar datos o servicios, la caída de nodos y la distribución equitativa de la información para evitar la sobregarca de nodos, una de las razones por la cual elegimos kademlia es que ya trae por defecto soluciones a estos problemas. Importante también seria como manejar el streaming eficiente de la música de manera distribuida, para lo cual tenemos otra alternativa que esperamos consultar en persona. En caso de que nos diera tiempo a implementar seguridad en el proyecto las comunicaciones que habria que cifrar serían tanto la http de middleware con vistas y los bytes transmitidos por RCP entre nodos de kademlia, la informacion requerida en el RCP se estan codificando en json, esto nos permite convetir en bytes hasta objetos simples del lenguaje.