Proyecto Final de Sistemas Distribuidos: WhatsApp

Bryan Machín García, José Alejandro Solís Fernández y Adrianna Alvarez Lorenzo

Universidad de La Habana,
San Lázaro y L. Plaza de la Revolución, La Habana, Cuba
{bryan.machin,jose.solis,adrianna.alvarez}@estudiantes.matcom.uh.cu
http://www.uh.cu

Resumen La mensajería tiene un papel importante en nuestros días. Empezando con los correos electrónicos hasta las aplicaciones de mensajes más completas. Se quisiera además que el servicio de mensajería no dependiera de un ente centralizado que regula los perfiles y potencialmente "lee" nuestras conversaciones. En lugar de esto se desea proveer de una arquitectura descentralizada y segura, en la que, cualquiera pueda recibir y mandar mensajes con la garantía de que solo lo podrán ver los involucrados en la conversación.

Palabras Claves: cliente, conexión, servidor, red

Abstract. Messaging has an important role in our days. Getting started with emails even the most complete messaging applications. It would also be desired that the courier service does not depend on a centralized entity that regulates the profiles and potentially "reads" our conversations. Instead of this, it is desired to provide a decentralized and secure architecture, in which anyone can receive and send messages with the guarantee that only the involved in the conversation.

Keywords: client, connection, network, server

1. Requerimientos

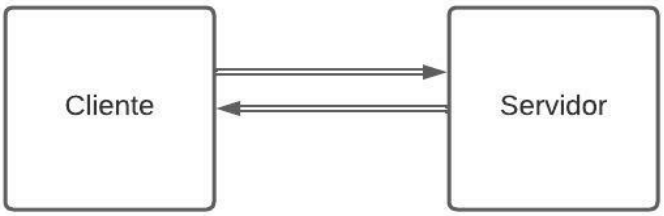
- **FastAPI**: es un marco web moderno y rápido (de alto rendimiento) para crear API con Python 3.6+ basado en sugerencias de tipo Python estándar. Constituye un conjunto de herramientas que permite a los desarrolladores utilizar una interfaz REST para llamar a funciones de uso común para implementar aplicaciones. Se accede a través de una API REST para llamar a bloques de creación comunes para una aplicación

- **Uvicorn**: es una implementación de servidor web ASGI para Python. Además, es el elemento de enlace que maneja las conexiones web desde el navegador o el cliente de API y luego permite que FastAPI sirva la solicitud real. Uvicorn escucha en un socket, recibe la conexión, procesa un bit y entrega la solicitud a FastAPI, de acuerdo con la interfaz ASGI.

- **Requests**: es el estándar de facto para realizar solicitudes HTTP en Python. Abstrae las complejidades de hacer solicitudes detrás de una API simple para que pueda concentrarse en interactuar con los servicios y consumir datos en su aplicación.

2. Aquitectura

El sistema está diseñado con dos estructuras fundamentales: un servicio en función a un cliente y otro capacitado como un servidor.

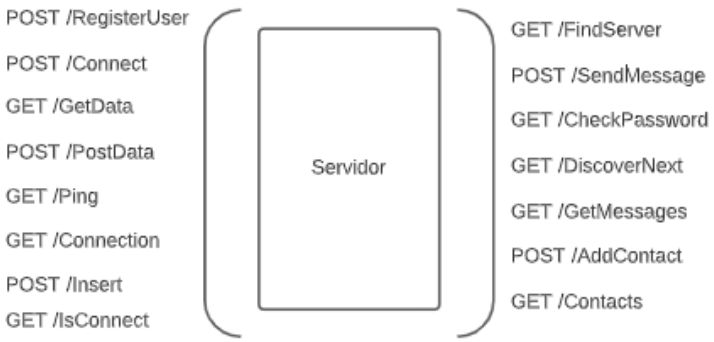


Este modelo de diseño de software permite que las tareas se distribuyan entre los proveedores de servicios (*servidor*) y los demandantes (*cliente*). El cliente solicita varios servicios al servidor, y este se encarga de dar la respuesta demandada por el cliente.

3. Servidor

Cada servidor dispone de un proceso de verificación que mantiene la conectividad enlazada en la red. La forma de realizarse es consultar si su predecesor inmediato se encuentra activo en el sistema. En caso de verificarse como falso, actualiza como su sucesor el segundo predecesor que contiene en su tabla distribuida de conexiones y asegura la consistencia para este y para su previo.

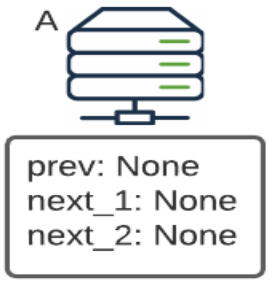
A continuación se muestran los distintos recursos o servicios que brinda un servidor propio del sistema:



Mediante el servicio `/Connect` es que se efectúa la inserción de un servidor nuevo a la red, especificando la dirección del servidor al que se desea conectar.

3.1. Estructura

Un servidor se compone de una tabla basada en una estructura de hash distribuida, conocida por las siglas DHT (del inglés, *Distributed Hash Tables*), donde se registra la dirección para acceder a los dos servidores inmediatos que le preceden(`next_1` y `next_2`) y el que le antecede(`prev`). Si solo existe un servidor activo en la red, entonces dicha tabla contiene valores nulos en estos campos, como se evidencia en la siguiente figura:



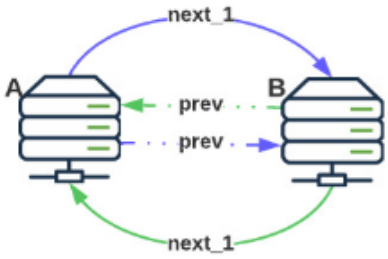
Cada servidor actúa como depósito de datos mediante la inclusión de dos bases de datos en su estructura. En la primera almacena su información propia, que hace referencia a los usuarios registrados en él, y en la segunda base de datos, la información correspondiente de los datos heredados, como parte de la concepción del modelo de sistema distribuido diseñado.

3.2. Conexión de Servidores

3.3. Inserción

La inserción de un nodo o servidor al sistema se efectúa a través de una petición a cualquier servidor que se encuentre en la red del sistema, la cual se valida mediante la verificación de una contraseña común para todos los servidores de la red. En esta figura se puede apreciar como se realiza el proceso de inserción

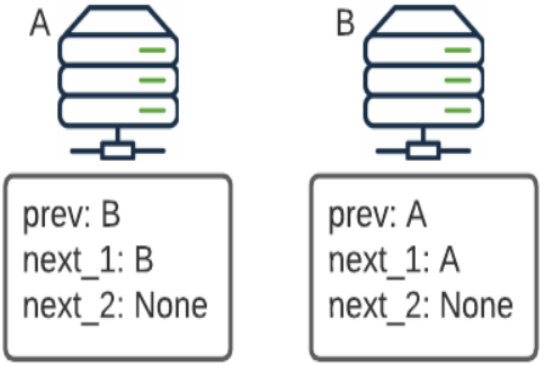
(a) Estado Inicial



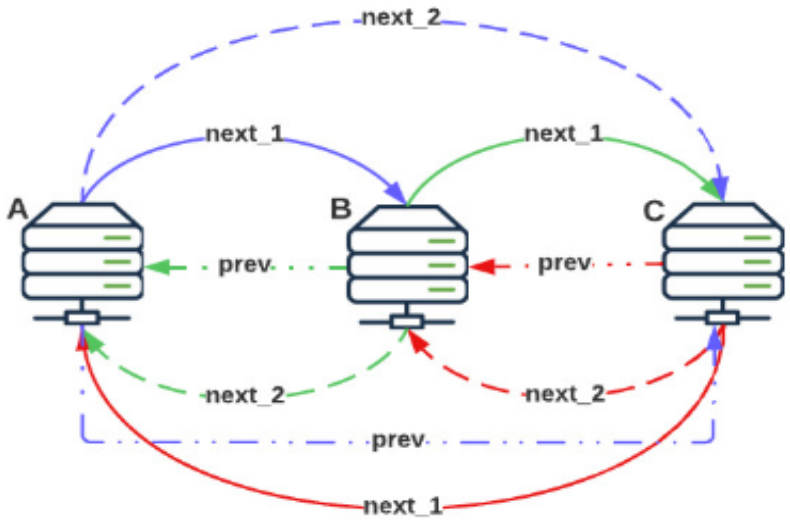
(b) Estado resultante luego de la inserción del servidor B al servidor A

de un servidor a la red si la petición de conexión se autoriza. Como resultado, el nodo B actualiza su tabla de distribución de conexión, donde asigna como su predecesor inmediato al nodo A, y a su vez designa a este como su previo.

La nueva tabla de hash distribuida para ambos servidores se compondría de la siguiente manera:

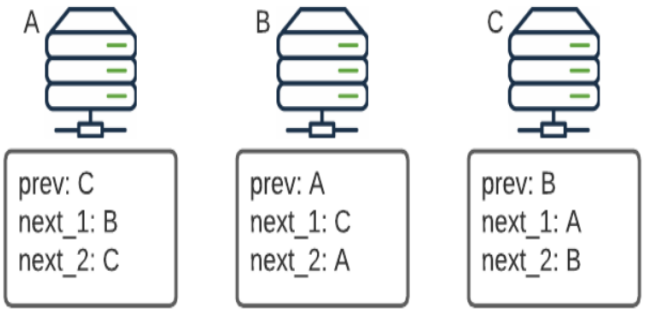


Si un servidor C realiza una petición de conexión al servidor B, y se verifica que la contraseña de acceso es válida, entonces la nueva red de conexión del sistema resulta:

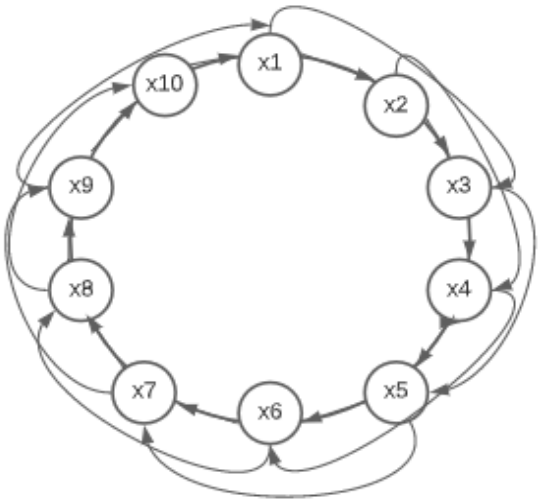


Como consecuencia, el nodo(o servidor) B designa al nodo C como su nuevo sucesor , y este a su vez actualiza su tabla de distribución de conexión, donde adopta como sus predecesores inmediatos los que contenía el nodo B, y lo designa como su previo. Para mantener consistente el sistema, como parte del proceso de actualización, el nodo A, que antes era el sucesor del nodo B, se actualiza como su segundo predecesor.

Luego de dicha inserción, las tablas de hash distribuidas se actualizan con los siguientes datos:

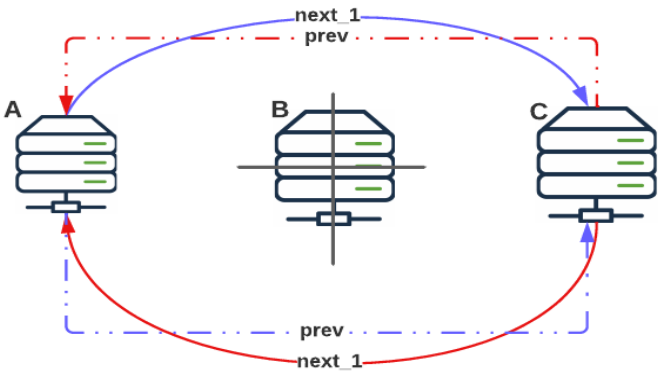


Por lo tanto, el esquema de diseño para la conexión de servidores en un sistema en el que intervengan más de 3 nodos se estructuraría de la siguiente manera:



3.4. Eliminación

Si se desconecta del sistema uno de los servidores, se debe lograr mantener la consistencia de este. Por ejemplo, si se desconecta el servidor B de la red:



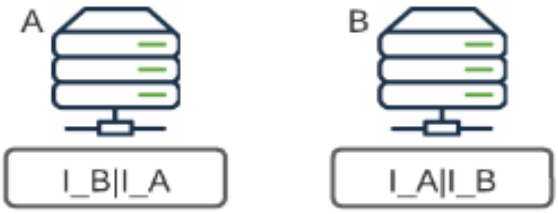
el nodo A actualiza al nodo C como su nuevo sucesor, y su segundo predecesor sería el sucesor inmediato de C, como se evidencia a continuación en sus tablas de hash distribuidas.

4. Replicación

4.1. Inserción

Cuando un nodo ingresa a la red, adquiere dos bases de datos: una que referencia a los datos heredados de la información propia de su antecesor y otra que representa su información propia.

Durante el proceso de conexión del nodo B al nodo A

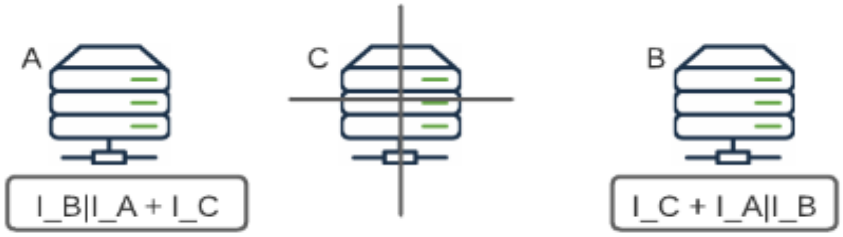


este recibe como datos heredados la información propia del nodo A(I_A), y a su vez este recibe los del nodo B(I_B), que en primera instancia de la inserción son nulos.

Luego, si el nodo C se conecta a la red a través del nodo A, entonces se procede con el mismo método de replicación de información, transfiriendo los datos propios del nodo A hacia la correspondiente base de datos heredados de su sucesor inmediato C, y en el nodo B se almacenaría como datos heredados la información propia de C.

4.2. Eliminación

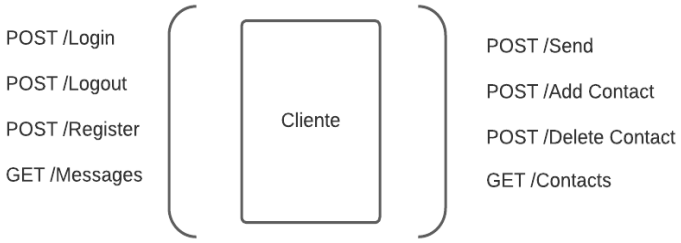
Si se efectúa el proceso de eliminación del servidor C, se debe lograr replicar la información de este hacia su antecesor y sucesor inmediato.



Como se muestra en la figura anterior, la información propia del nodo A se le adiciona a los datos heredados del servidor B, y la información heredada que este contenía previamente del nodo C se le adiciona a los datos propios del nodo A, proporcionándole así al sistema un mayor rendimiento, disponibilidad y escabilidad mediante dicho mantenimiento.

5. Cliente

La estructura del cliente dispone de los siguientes servicios:



El cliente puede registrarse a través de cualquier dirección válida en el sistema, ingresando sus datos personales: *nombre, id(nickname), contraseña* y el servidor en el que desea realizar dicho registro.

Para obtener acceso a los recursos de mensajería proporcionados en este entorno, el usuario debe iniciar sesión en el sistema mediante sus credenciales. Si dicho acceso es permitido, entonces puede hacer uso de las funcionalidades que facilita el sistema. El envío de mensajes se puede efectuar entre cualquier par de usuarios, sin requerir que el destinatario esté registrado en la lista de contactos del remitente.

Así como la estructura del servidor, la aplicación del cliente cuenta con un proceso de verificación que mantiene una lista de servidores con los cuales comunicarse con el sistema para realizar las peticiones.

6. Recomendaciones

En el caso de nuestro sistema, la comunicación se realiza a través del protocolo HTTP, con la intención de chequear el flujo de datos en un entorno de desarrollo. Para el caso de un entorno de producción, se requiere de certificados que avalen la seguridad en la transmisión y obtención de datos, para emplear un protocolo HTTPS.

Además, para mayor seguridad en la transmisión de datos, es posible realizar un algoritmo de encriptación asimétrica entre par de usuarios.