ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

[DEIM]

Pràctica 1

Sistemes Distribuits

Online chat application

Assmaa Ouladali
David Domènech Pereira Da Silva
17/5/2024

1 GitHub:

Enllaç repositori:

https://github.com/David-Domenech-Pereira/PracticaSD

2 Abstract

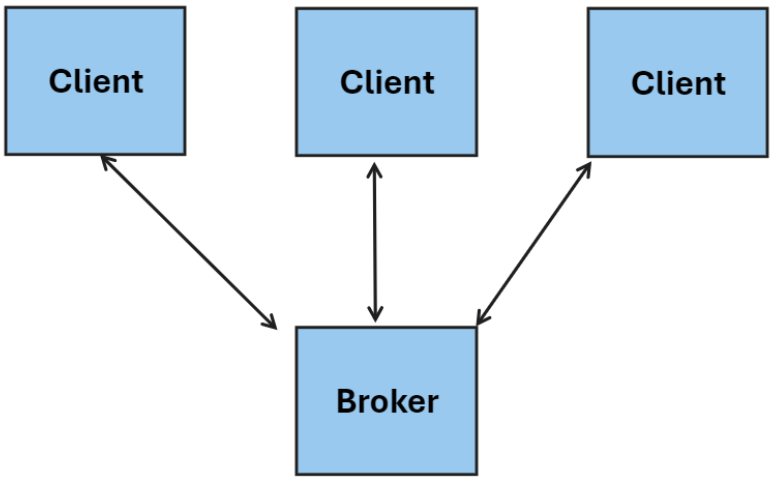
En aquest projecte s'ha desenvolupat un sistema de comunicació en línia fent servir Python i algunes de les seves llibreries més conegudes. En primer lloc, s'ha creat un xat privat entre diferents usuaris amb gRPC, on només amb el seu nom d'usuari el sistema és capaç de localitzar-lo fent servir Redis, es requereix que ambdós estiguin connectats en un mateix moment. Seguidament, s'ha creat un xat grupal amb RabbitMQ que permet enviar missatges entre diferents usuaris, a més depèn de com es configuri es pot permetre una comunicació asíncrona si algun usuari marxa. Per continuar s'ha implementat un servei discovery, també amb RabbitMQ, que permet veure els usuaris que estan connectats i per acabar s'ha aprofitat els coneixements adquirits a RabbitMQ per a fer un servidor d'insults amb el que qualsevol usuari pot enviar un insult a la xarxa.

3 Disseny del sistema i discussió:

Per a desenvolupar aquesta pràctica s'han diferenciat 2 tipus d'elements dins de l'ecosistema de la aplicació:

- Client: Aplicació que executa cada usuari.
- Broker: Aplicació que executa RabbitMQ i Redis.

Dins del funcionament normal, es poden tenir n clients i només un broker, no s'ha contemplat cap manera d'escalar-lo, encara que pot acabar sent un punt crític.



Il·lustració 1: Diagrama bàsic de funcionament.

3.1 Client:

El client indica a l'usuari un menú d'opcions, on aquest escull entre:

1- Enviar un missatge.
2- Subscriure's a un grup.
3- Veure grups i usuaris connectats.
4- Servidor d'insults.

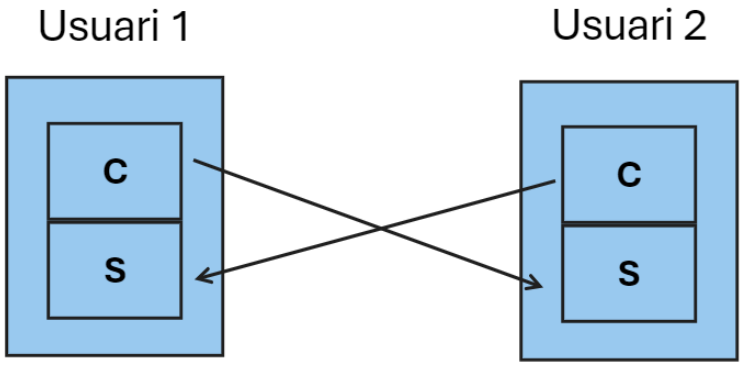
3.1.1 Enviar un missatge.

A l'hora d'enviar un missatge, es pregunta un destinatari a l'usuari, aquest pot introduir tant usuari real com el nom d'un grup.

En primer lloc, el sistema busca si existeix el nom d'usuari, si existeix estableix una connexió 1-1 (Privats) amb aquest client. Si no existeix, busca un grup amb aquest nom, si no existeix tampoc el crea.

3.1.2 Missatges Privats

Per al funcionament dels missatges privats, cadascun dels usuaris dins del seu client conté un client gRPC i un servidor gRPC.



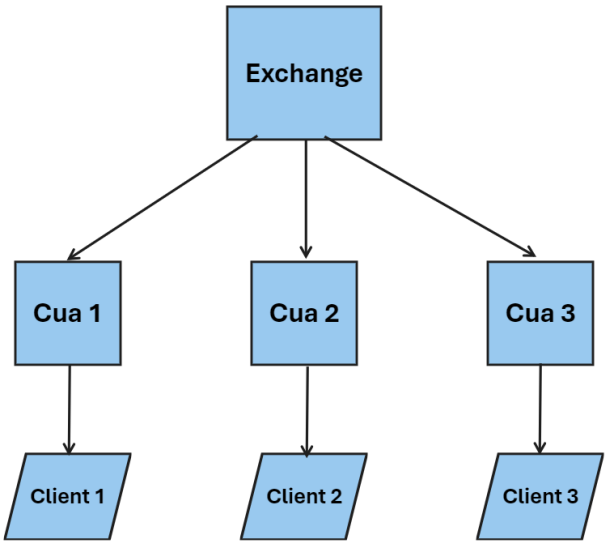
Il·lustració 2: Diagrama de la arquitectura pel funcionament del gRPC.

La idea principal es que cada Usuari contingui un servidor que està sempre encés, aquest servidor permet rebre missatges de qualsevol usuari. Quan un usuari vol enviar un missatge li envia als servidor d'aquest.

El servidor s'executa en un thread independent, que comunica al client quan rep un missatge per a mostrar-lo per pantalla. Quan el servidor s'inicia anuncia la seva ip i port al broker que conté el servidor Redis i el servidor RabbitMQ, per a que s'inclogui als 2.

3.1.3 Missatges de grup

Els grups es troben en cues al broker RabbitMQ, per a fer la implementació s'ha definit un Exchange on cada usuari té la seva pròpia cua:



Il·lustració 3: Arquitectura del sistema de cues RabbitMQ

Quan es vol enviar un missatge pel grup s'envia al Exchange i aquest li fa arribar a totes les cues una copia.

També es pot indicar si es vol fer el grup amb persistència o no, a l'enunciat es demanava **demostrar que els grups persistents permeten la comunicació asíncrona**. La implementació de RabbitMQ permet que quan un usuari connectat es desconnecti, al tornar-se a connectar vegi tots els missatges que s'han enviat mentre estava absent, això permet la comunicació asíncrona si un usuari es desconnecta, però no ho permet amb els nous usuaris que es puguin connectar.

3.1.4 Missatges "Discovery"

Per a poder saber quins usuaris estan connectats sense fer servir el servidor Redis, s'ha programat de tal manera que es tingui un Exchange comú amb tots els usuaris, on dins d'ell cada usuari tindrà una cua exclusiva, d'una manera molt similar a com s'ha fet als missatges en grup.

Quan s'envia un paquet al Exchange, el paquet s'envia a totes les cues, així aconseguim una distribució broadcast.

Un cop el client rep el missatge, respon amb el seu alies i ip al emissor.

En comparació amb Redis, aquesta implementació és molt més lenta, ja que requereix que el missatge arribi a cadascun dels clients i aquests facin una

resposta, establir una connexió entre N dispositius sempre és més lent que connectar-se amb un i que aquest doni la resposta el més ràpid possible.

3.1.5 Servidor d'insults

Per a aquesta implementació, diferents clients es subscriuen a una mateix cua, quan s'envia un missatge a aquesta cua, només un dels clients la reb, el primer que es trobi a la cua.

Pot ser que qui rebi el insult sigui el mateix que l'ha rebut, ja que es troba també subscrit a la cua.

3.2 Broker:

El servidor s'inicialitza a través del seu propi Main, aquest element engega un servidor RabbitMQ i un servidor Redis, cadascun en un thread diferent per a permetre una execució simultània.

4 Preguntes

Q1 Els xats privats són persistents? Sinó, com podríem donar-los persistència?

Per defecte els chats privats no són persistents, ja que es tracten de comunicacions stateless, és a dir, no es té en compte el estat anterior del sistema quan s'envia i es rep un missatge.

Per afegir persistència, la única manera que creiem és afegir una base de dades a cadascun dels clients que guardin els missatges que es reben i/o envien, ja que no tenim cap intermediari dins de la connexió.

Q2 Hi ha patrons de comunicació stateful en el vostre sistema?

En el nostre sistema, a la part de gestió de cues si que hi ha patrons stateful, ja que el broker emmagatzema quants usuaris hi ha subscrits a cadascuna de les cues, així es guarda l'estat del sistema i cada cop que s'envia un missatge, per exemple un discover, s'envia a tots els clients, per tant la resposta no serà sempre igual.

Q3 En quin tipus de patró es basen els xats de grup? En termes de funcionalitat, compareu la comunicació transitiva i persistent en xats de grup mitjançant RabbitMQ.

Els chats grupals fan servir un patró publiser-subscriber, on tenim un conjunt de clients subscrits a un canal i on cada cop que s'envia un missatge es fa una publicació que la reben tots els subscriptors.

La comunicació transitiva i persistent funcionen d'una manera molt similar, la única diferencia és que la persistent guarda al disc del broker els missatges de cada cua que no arriben als clients, així quan un usuari està desconnectat i es torna a connectar pot recuperar els seus missatges que encara té pendents a la cua. Aquesta diferència és una petita configuració que es fa alhora de configurar el RabbitMQ i no altera el seu funcionament normal.´

Q4 Redis també pot implementar cues i patrons pubsub. Prefereixes utilitzar Redis que RabbitMQ per a esdeveniments? Podríeu tenir xats de grup transitoris i persistents? Compara els dos enfocaments

Hem analitzat com realitzar el patró pubsub a Redis, d'una banda, hem vist que Redis generalment treballa sobre la memòria RAM, per tant no pot garantir una persistència quan s'aturi la màquina, malgrat tenir alguna funcionalitat de persistència en memòria no és tan robusta com RabbitMQ (es poden garantir els xats de grup transitoris i persistents). D'altra banda, Redis és molt més eficient i té una manera nativa d'implementar cues, per tant ens hagués facilitat el desenvolupament.

Nosaltres creiem que cadascun té la seva funció, Redis és ideal per garantir una resposta ràpida, però potser no ens pot garantir tota la persistència que ens garanteix RabbitMQ, a més a més, RabbitMQ ens garanteix la completa rebuda dels missatges amb molta més fiabilitat que Redis, a banda d'estar pensat explícitament per la gestió de cues.