Projecto Semestral Computação Distribuida - Distributed Photo Organizer

Rafael Remígio 102435

Junho 2022

Contents

1 Introdução

Neste documento explicarei a resolução do Projecto semestral no ambito da disciplina de Computação Distribuida que tem como objetivo o desenvolvimento de um protocolo P2P que permita a transferência de imagens entre Clientes. É necessário permitir ao utilizador do sistema:

- Listar todas as imagens em sistema (através do seu identificador)
- Obter uma dada imagem, com base no seu identificador

Ambas estas funcionalidades foram implementadas com sucesso.

2 Rede Overlay

Na minha rede Overlay usei uma topologia Mesh onde qualquer Nó tem uma ligação através de sockets TCP. O ancoramento da rede é feito por um daemon iniciado com um Port 5000 que vai esperar por entradas de outros nós na rede. Um *daemon* é iniciado e tem como argumentos a sua pasta de imagens, o seu Port, e o Port do nó onde se vai ligar. Deste modo um nó pode se ligar a qualquer nó e será incluido na rede a partir desse nó. Isto é feito através da troca de mensagens entre os nós.
Ações de um Daemon a entrar na rede.

1. Enviar mensagem de registo ao Daemon a que se está a ligar
2. Recebe mensagem com uma lista de todos os nós existentes na rede
3. Liga-se a todos os Nós existentes através de um mensagem de registo
4. Envia a informação referente às suas imagens a todos os nós da rede
5. Envia uma mensagem de Pedido para todos os nós a pedir informação sobre as imagens da rede.
6. A partir desta ação seriam enviadas mensagens referentes à tolerancia de falhas, isto é, gerar cópias de imagens na rede. Isto seria feito através do envio de Imagens para outros nós.

O problema de usar uma topologia mesh é que muito rapidamente temos um grande número de conexões a gerir tal temos bastante tráfego a circular na rede. Tornando-a deste modo muito pouco escalável. As vantagens desta topologia são a facilidade e rapidez no envio de mensagens, podendo fazer tudo directamente sem necessidade de enviar imagens por multiplos nós.

2.1 TCP

Neste projeto usei sockets TCP pela necessidade de segurança no envio de dados devido ao transporte de grandes imagens. O uso de TCP garante a receção de dados pelo nó recetor e previne erros na transferência de dados.

Python Selectors

Usei Python Selectors neste projecto por já ter usado previamente no Primeiro Guião Pratico da disciplina o que permitiu a reutilização de código e pelo facto de permitirem de forma muito facil tratar multiplexação.

2.2 Client

O Client interage com o **Daemon** através de uma socket onde enviará 2 pedidos: Lista de Imagens e uma imagem Especifica que depois é descodificada e mostrada através da biblioteca de Python PILLOW e imagemagick. Um Cliente conecta-se a um Daemon enviando uma mensagem de registo. O **Daemon** mantém um dicionário onde mapea os diferentes clientes que pode ter à sua respectiva socket.

3 Imagens

A informação sobre as imagens, como já foi referido, é enviada pelos **Daemons** após o envio e receção das mensagens de Registo. As imagens são depois mapeadas com o seu respectivo **Nome**, **Hash**, **Tamanho em Bytes** e a **socket respectiva** ao Daemon a quem pertence esta imagem.

3.1 Identificação unívoca de Imagens

Na rede imagens são identificadas pelo seu hash. Para o Cliente são só conhecidos os nomes das imagens que são usadas como identificador para o Cliente. O hashing das imagens é feito a partir da utilização da biblioteca Python PILLOW e imagehash e as imagens são mapeadas localmente com o seu nome indetificador o seu hash e o seu tamanha em Bytes que é depois usado na eficiencia de armazenamento. Na eventualidade de haver duas imagens com o mesmo hash localmente é removida a imagem que ocupa o menor espaço (Isto foi feito de modo a manter a imagem com mais informação que

pode ser interpretada como a melhor imagem da duas. Isto só pode ser feito se assumirmos claro que as imagens estão a usar o mesmo tipo de compressão o que em retrospectiva foi uma falha mas facilmente alterável).

3.2 Transferência de Imagens

A tranferência de Imagens é feita através da codificação em **base64** da imagem e envio entre sockets TCP. A quando do pedido de um Client uma imagem através do envio de uma mensagem **GetImageRequest**, se está iamgem residir no Daemon respectivo ao cliente este envia diretamente a mensagem ao Cliente. Se a imagem residir noutro Daemon este enviará uma mensagem **AskforImage** ao Daemon que contém a imagem, este enviará a imagem ao Daemon que fez o pedido que a reenviará ao Client. Deste modo garantimos que outros Daemons não precisam saber da existencia de Clientes nem que Clientes necessitem de interagir com outros Daemons.

3.3 Eficiência de armazenamento

Na eventualidade de um **Daemon** receber uma imagem com um hash igual a um hash das suas imagens locais ele fará a comparação entre os tamanhos das 2 imagens. Na eventualidade da sua imagem for a *"pior"* ou igual à da já na rede o daemon eliminará a imagem da sua lista de imagens e do seu repositório. Se a sua imagem for *"melhor"* este nó enviará uma mensagem ao outro nó a pedir para este eliminar esta específica imagem e depois reenvia a todos os nós a nova informação das imagens (em retrospectiva isto é algo altamente inseguro permitindo a um **rogue daemon** simplesmente apagar todas as imagens da rede).

4 Resiliência a falhas

Resistencia a falhas não foi completada neste Projecto, a quando de um Daemon falhar são perdidas da rede todas as suas imagens. Esta seria feita através do envio da mensagem *keepThis* que enviaria a um outro daemon na rede a imagem codificada. Esta seria guardada pelo Daemon e mapeada a uma socket respectiva. O uso de TCP permitiria saber quando um Nó se desconecta o que permitiria a reentrodução da Imagem na rede por este Nó e que depois reenviaria a todos os outros Nós a actualização da informação

sobre as imagens que seria actualizada. A seleção de para que Nó enviar estas mensagens pode ser feita através de um algoritmo Round Robin, permitindo Load Balacing na entrega das réplicas.

5 Escalabilidade e Trabalho futuro

Como já foi referido o uso de uma topologia Mesh é bastante conviniente para redes pequenas e para este trabalho mas tornou este trabalho muito pouco escalável. Na minha prespectiva o meu Sistema Distribuido poderia ser futuramente melhorado pela adição de **SuperPeers**, estruturando deste modo o sistema peer-to-peer num Sistema Hibrido, permitindo a criação de outros sub-sistemas que poderiam ser referentes a outros grupos de amigos de outras cidade. Poderia também ser melhorado a eficiencia do Sistema através de caching de imagens que sejam pedidas frequentemente. Algo que também poderia ser melhorado no meu Sistema seria um Daemon não manter nele mesmo todas as imagens que este introduzio à rede, sendo possível as imagens serem guardadas em diferentes nós aumentando fazendo deste modo **Load Balacing**

6 Conclusão

Com a realização do trabalho foi possível compreender a criação de um Sistema Distribuída, foi um trabalho muito desafiador. além disso, é interessante ressaltar que os conhecimentos adquiridos no decorrer do semestre foram fundamentais para a execução do projeto atual. Este trabalho não foi completado não por falta de interesse ou por falta de compreensão do que era necessário fazer mas sim por falta de coordenação e tempo