Computação Distribuída

Distributed Photo Organizer

Diogo Fernandes, 78323

João Matos, 103341

Docentes:

Diogo Gomes (dgomes@ua.pt)

Nuno Lau (nunolau@ua.pt)

João Rodrigues (jmr@ua.pt)

25 de junho de 2022

1 – Introdução

No âmbito da unidade curricular de Computação Distribuída, foi-nos proposto um projeto consistindo no desenvolvimento de um sistema distribuído de organização de fotos.

Este deveria ser capaz de, dada uma pasta com fotos, de vários formatos e tamanhos, partilhar essas imagens com todos os outros amigos no mesmo sistema *peer-to-peer* (P2P) e garantir que, no caso de perdas de ficheiros, todos os amigos tivessem acesso a cópias das fotos perdidas.

2 – Funcionamento do Sistema

O programa contém 2 scripts executáveis. O primeiro é o *daemon.py*, requerendo um argumento obrigatório – a pasta com as imagens a partilhar no sistema – e um opcional, que permite inicializar o node do *daemon* como sendo o *main node*. O segundo é o *client.py*, que requer dois argumentos, sendo estes o *host* e a *port* em que se conectará ao seu *daemon node* respetivo.

O *client* pretende simular um utilizador do sistema, e permite duas operações. A primeira lista todas as imagens existentes no sistema no momento, e a segunda pede uma imagem ao sistema para ser visualizada pelo cliente.

Assim, todas as funcionalidades do sistema estão dentro do *daemon*. O *cliente* apenas permite especificar o que quer que o sistema faça.

Daemon

O Daemon é a nossa rede P2P. Esta rede é composta por nós independentes que se podem conectar quando quiserem. Existe um nó (normalmente o primeiro) que fica designado como o *main node*. Este fica responsável por processar a logística associada à localização de cada

imagem na rede, e disseminar essa informação por todos os outros nodes da rede. No entanto, caso o *main node* se desconecte da rede, haverá outro nó a assumir esta responsabilidade.

Identificação unívoca de uma dada imagem

O sistema utiliza um algoritmo de *hashing* para identificar uma dada imagem. Ficheiros contendo fotos iguais terão o mesmo *hashCode*, o que permite ao sistema reconhecer quando existe vários ficheiros com a mesma imagem no sistema.

Transferência de imagens entre nós

As imagens são enviadas entre nós do sistema através de sockets TCP. Através das mensagens *send_img* e *recv_img* definidas no protocolo desenvolvido, um ficheiro contendo uma imagem é enviado por partes.

Pesquisa por uma dada imagem pelo seu identificador

O *client* pode pedir a lista das imagens existentes na rede. A rede retorna então uma lista das imagens, ignorando as cópias de segurança existentes na rede. Assim o *client* pode escolher, a partir da lista, a imagem que quer visualizar.

Eficiência de armazenamento

O sistema garante que, a qualquer momento, existam apenas duas imagens com o mesmo *hashCode* dentro do próprio sistema.

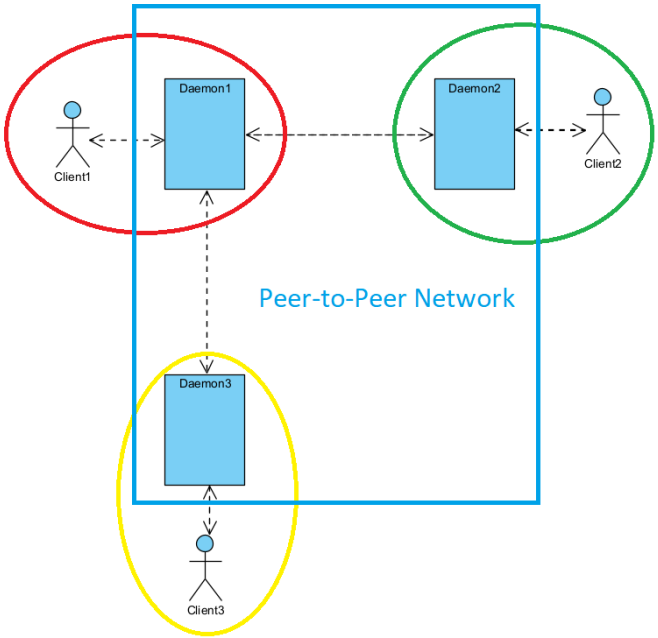
Além disso, quando um novo nó, com os seus respetivos ficheiros entra no sistema, é chamada uma rotina que distribui todas as novas imagens pelo sistema, visando que nenhum nó centralize os ficheiros e que todos os nós contenham aproximadamente a mesma carga de armazenamento, e eventualmente removendo cópias de imagens que já continham duas cópias no sistema.

Resiliência a falhas

No sistema, uma dada imagem tem sempre uma cópia armazenada num node distinto. Assim, quando um nó se desconecta da rede (e, consequentemente, as imagens que este continha são perdidas), o sistema garante que as imagens contidas no nó perdido possam ser recuperadas por via da cópia existente.

Exemplo de network

Na imagem seguinte está um exemplo de uma possível rede. Dentro do retângulo azul está a rede P2P, e nas cores restantes estão as conexões entre os clientes e o seu nó respetivo do Daemon. Neste exemplo, a rede vermelha é considerada a main.



3 – Protocolo

O nosso protocolo contém 7 tipos de mensagens e 4 métodos de comunicação. Nesses quatro métodos, dois são para enviar e receber mensagens definidas (*send_msg* e *recv_msg*) pelo nosso protocolo e os outros dois enviam e recebem imagens (*send_img* e *recv_img*).

Utilizando *sockets* TCP, tanto mensagens e imagens são enviadas em duas partes:

- o *header* vai primeiro, e tem um tamanho fixo de 4 bytes; este contém o tamanho da mensagem/imagem
- o conteúdo da mensagem, que é enviado sempre depois do *header*.

ImageRequest

Esta mensagem é enviada sempre que o cliente ou um nó pretende pedir uma imagem a outro nó. Esta contém como argumentos o identificador da imagem pretendida, e um argumento booleano que indica se foi um cliente que pediu a imagem. Caso verdadeiro, o cliente vai mostrar a imagem no browser após recebê-la.

ImageReply

Esta mensagem é enviada ao cliente aquando este está prestes a receber uma imagem. Assim este sabe que, após este mensagem, virá uma imagem, que posteriormente deverá mostrar no browser. Recebe como argumento o identificador da imagem que será recebida.

ImageRemove

Esta mensagem é enviada entre nós para indicar uma imagem a ser removida, devido ao facto de já se encontrarem duas cópias da imagem na rede. Recebe como argumento o identificador da imagem a remover.

ListRequest

Esta mensagem é utilizada para o cliente pedir a lista das imagens na rede a um nó da rede.

ListReply

Esta mensagem é utilizada para que um nó, após pedido do cliente, responder com a lista das imagens existentes na rede. Recebe como argumento a lista das imagens existentes na rede.

NotifyMain

Esta mensagem é enviada de um nó que acabou de entrar na rede. Esta é utilizada para informar o nó principal das imagens que o novo nó traz para a rede, como do tamanho total do seu diretório de imagens, para que posteriormente, os nós com menor tamanho guardem as cópias de segurança das imagens. Tanto a lista das imagens no nó como o tamanho do diretório são passados como argumento desta mensagem, além do endereço de quem está a enviar a mensagem.

ImageNodes

Esta mensagem é enviada do nó principal para os nós que integram a rede. Esta contém um dicionário como argumento; este associa a cada imagem as suas localizações na rede. Além disso, informa os nodes do sucessor do nó principal, caso este vá abaixo, num segundo argumento.