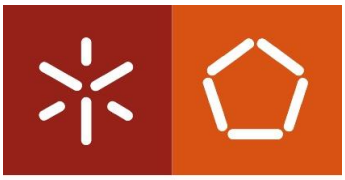

Universidade do Minho

Departamento de Informática

Troca de Ficheiros

Grupo Nº 34

José Ferreira (A83683)

João Teixeira (A85504)

Miguel Solino (A86435)

5 de Janeiro de 2020

Conteúdo

Capítulo 1

Introdução

O objetivo deste projeto é desenvolver um sistema de partilha de ficheiros de música que seja capaz de gerir os ficheiros assim como os respetivos meta-dados.
A arquitetura a ser utilizada deve seguir a lógica *Cliente, Servidor* com recurso a *Sockets TCP* e *Threads*, aplicando, assim, os conhecimentos sobre Sistemas Distribuídos leccionados durante as aulas da UC.
Como objetivo, é necessário implementar um sistema que permita a utilização concorrente de um servidor por vários clientes, assegurando a consistência de toda a informação introduzida no sistema para qualquer cliente que esteja a fazer uso da mesma.
Ao longo deste relatório vamos descrever as nossas abordagens a estes problemas.

Capítulo 2

Arquitetura e Solução do Projeto

2.1 Servidor

No modulo Servidor são tratadas todas as conexões e pedidos dos clientes, garantindo a capacidade de suportar vários clientes em simultâneo.
Para além disso, é tratada a gestão da informação relativa a logins e a músicas armazenadas de forma a que seja possível um acesso concorrente entre as diversas *Threads* lançadas.
A recessão, envio e armazenamento dos ficheiros de música e também tratado neste modulo, tendo em vista uma boa gestão de recursos, tanto a nível de memoria como a nível do *CPU*.
De forma a conseguir um servidor capaz de lidar com vários clientes em simultâneo, este é dividido em varias *Threads*, sendo que é criada uma nova para tratar cada cliente que se tente conectar, uma para receber e responder aos pedidos vindos do cliente, uma para enviar notificações sobre novas músicas adicionadas, uma para gerir os *Downloads* requisitados, e uma ultima para receber os ficheiros de música enviados pelo respetivo cliente.

2.2 Cliente

No modulo Cliente é efetuada a ponte entre o utilizador e o servidor, sendo capaz de efetuar pedidos ao servidor e apresentar ao utilizador a resposta aos pedidos efetuados. Este está preparado para complementar o servidor no envio e recessão de ficheiros de música.

2.3 Comunicação Cliente-Servidor

2.3.1 Pedidos e respostas aos pedidos

Para efetuar a comunicação entre o cliente e o servidor fixamos um formato de texto, inspirado no formato *JSON*, que no caso de pedidos ao servidor tem o formato 2.1, onde o *request_type* corresponde ao tipo de pedido que o cliente quer efetuar.
Por exemplo, para adicionar uma música iria ser *add_music*.

```
{type='request_type', content=['obj1','obj2',...]}
```

Figura 2.1: Formato dos pedidos

As respostas a pedidos segue o formato 2.2.
Neste, o *status* pode ser *ok* ou *failed* e o *res* é a resposta ao pedido efetuado ou a razão pela qual não foi possível efectuá-lo.

```
{status='status', result='res'}
```

Figura 2.2: Formato das repostas

Para comunicar com o cliente, linhas de texto com este formato são transmitidos através de uma socket criada para o efeito.

2.3.2 Uploads e Downloads

Para efetuar a transferência de ficheiros de e para o servidor, é criada uma *ServerSocket* por cada cliente, numa porta aleatória, e posteriormente e enviada ao servidor a informação para ele se conectar a essa *Socket*. Nesta *socket* apenas são enviados binários correspondentes ao ficheiro de música, em partes de um tamanho máximo definido. Embora sejam possíveis downloads e uploads concorrentes entre clientes, um cliente apenas pode efetuar um download e um upload em simultâneo.

2.3.3 Notificação de novas músicas

Quando é adicionada uma nova música ao sistema, esta é adicionada a uma *queue* de notificações que será utilizada por cada *thread* de notificações, enviando assim uma notificação a todos os clientes que estejam ligados ao servidor no momento.

Capítulo 3

Conclusão

Para concluir, conseguimos cumprir todos os requisitos propostos, criando no processo um sistema de troca de ficheiros de música capaz de utilizar uma arquitetura *Cliente, Servidor* que suporta vários clientes em simultâneo. Como trabalho futuro, gostaríamos de permitir a cada cliente ter mais do que um download e upload a correr em simultâneo.