

**Universidade do Minho**

Escola de Engenharia, Departamento de Informática

LEI 3º ano – Comunicações por Computador

“Music Game”

**Desenho e implementação de um jogo distribuído na Internet**

**Abstract.**

Este relatório visa a documentação de todo o processo e decisões implementadas durante a construção e implementação de um serviço de distribuição de conteúdos por transmissão de informação através do uso de *sockets* TCP e UDP. Os servidores terão acesso à informação através de comunicação por *sockets* TCP e, os clientes, receberão os dados enviados pelo servidor por *sockets* UDP. O desafio é, então, implementar um procedimento fiável, com controlo de fluxo e de erros, que possibilite a comunicação entre os diferentes participantes do jogo (servidores e clientes).

Bruno Pereira, 69303 João Mano 69854 Patrícia Rocha 67636

1. Introdução

O avanço tecnológico permitiu a introdução de aplicações únicas suportadas pela internet, que permitem o fornecimento de um dado serviço. Nestas aplicações, a distribuição de informação dos servidores para os clientes (habitualmente sempre em maior número), é das maiores preocupações a ter aquando o desenvolvimento dos programas.

Para que a comunicação entre as diversas entidades envolvidas seja possível e para que a sua eficácia e desempenho seja também, tão elevada quanto possível, é necessário projetar, instalar e operar redes otimizadas de comunicação que têm em conta paradigmas como a fiabilidade, disponibilidade, atrasos e débitos dos serviços fornecidos.

Assim sendo, como objetivo deste projeto, será construído uma aplicação que fornecerá um jogo sobre música a todos os seus clientes, implementando elementos como rankings, pontuações, desafios entre jogadores, etc. Esta aplicação, sendo suportada pela internet, exige a construção de um protocolo de comunicação que será então o objetivo principal, ao longo deste documento será explicitado o processo de construção do mesmo.

1. **Arquitetura da Aplicação**

A solução desenvolvida implementa a seguinte estrutura, existe um servidor denominado MusicServer que, quando é inicializado “parte-se” em duas threads de escuta, a thread AtendimentoCliente que estará sempre à escuta de pedidos UDP, direcionada, como o próprio nome indica, para atender clientes, e a thread AtendimentoServidor que estará à escuta para pedidos TCP, de outros servidores. Esta última thread ainda se encarrega de, caso o servidor não seja primário, registar-se perante o servidor primário.

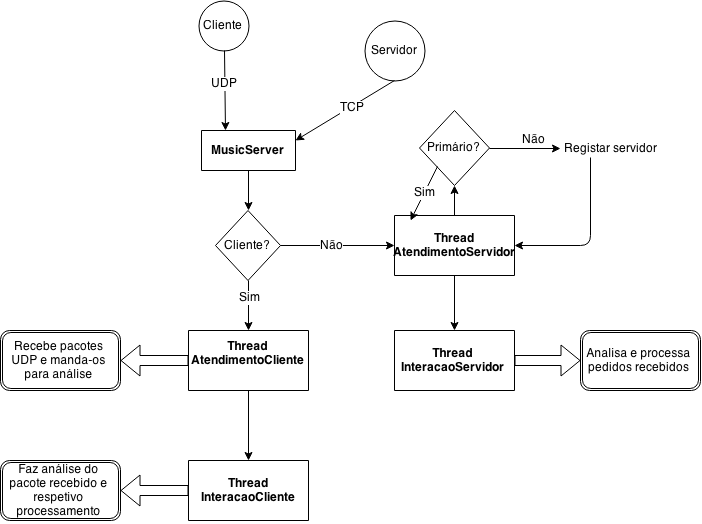


Figura 1 – Processamento de pedidos

Após a criação das threads, sempre que é recebido um pacote, visto que é preciso estar sempre “à escuta”, são criadas a thread de análise e processamento dos pedidos, denominadas InteracaoCliente para pedidos provenientes dos clientes e, InteracaoServidor para pedidos provenientes de outros servidores.

A estrutura utilizada para armazenar os dados, denominada BD é constituída por:

* Um HashMap de Utilizadores (chave é a alcunha);
* Um HashMap para o ranking local e outro para o ranking global (mapeiam a alcunha de um jogador para a sua pontuação local ou global respetivamente);
* Um HashMap de desafios locais e outro para os desafios globais (mapeiam a string com o nome do desafio para a data de início);
* Um HashMap listaDesafiosServidores para conseguir associar ao nome do desafio não local o InetAddress e a porta do servidor a quem é preciso pedir as perguntas;
* Um ArrayList de perguntas;
* Um HashMap de servidores (mapeia um InetAddress de um servidor à sua porta);
* Uma string com o nome da pasta de músicas e outra com o nome da pasta de imagens.

Esta estrutura é comum a todos os servidores e a todas as threads pertencentes ao servidor (threads de atendimento, processamento).

* 1. **PDU**

A classe PDU representa o pacote que permite a comunicação entre os diversos componentes.

Falta aqui!!!!!!!!!!!!!!!!!!!!

* 1. **Cliente**

Após a entrada de um pedido na thread AtendimentoCliente é criada então, tal como já foi referido, a thread InteracaoCliente. Esta thread analisa o tipo de pedido recebido (login, registo, aceitar jogo, etc.) e encarrega-se de tratar do seu processamento.

De todos os diversos pedidos abordaremos apenas aqueles que achamos de maior relevância, outros como Login, Registo, etc. são bastante simples e por isso não mencionados.

* + 1. **Jogar Desafio**

Tanto a aceitação como a criação de um desafio implicam a criação de uma nova thread, Jogo, para fazer os clientes aguardarem pelo jogo.

Ao criar um desafio é introduzido o desafio localmente (no HashMap de desafios locais e no de globais) e enviada a informação de um novo desafio para os restantes servidores, estes vão atualizar a estrutura de desafios globais. É ainda introduzido o jogador no desafio.

De seguida é criada a thread Jogo que, espera até à hora do desafio chegar e, posteriormente envia a primeira pergunta do desafio ao jogador respetivo se estiver tudo bem. Isto acontece também no pedido de aceitar desafio.

Assim quando o utilizador recebe a pergunta espera até receber o último pacote e realiza uma validação para verificar se todos os blocos foram recebidos, caso não tenham sido, é pedido um retransmit do bloco em falta.

Tanto os pedidos de retransmit como as respostas às questões são recebidas pela thread de escuta, AtendimentoCliente reencaminhadas à thread InteracaoCliente e, caso seja resposta à pergunta e não seja a última é enviada a nova pergunta e o processo repetido, caso seja a última é esperado que todos os utilizadores terminem e então enviadas as cotações do desafio para todos os utilizadores.

* + 1. **Controlo de perdas**

O controlo de perdas de pacote é feito da seguinte forma, quando uma resposta não chega entre um determinado período de tempo, é lançado uma TimeOutException que é apanhada e despoleta um retransmit do último pacote enviado. Desta forma é requerido que o último pacote enviado seja armazenado por parte do cliente.

Note-se que o retransmit não é infinito, ou seja, após um número estipulado de retransmit’s não se volta a pedir uma retransmissão e é lançada a exceção ServerUnreachable.

* 1. **Servidor**

Falta aqui!!!!!!!!!!!!!!!!!!!!

* 1. **Outras Estruturas**

**Diferenças/Adições na especificação do protocolo em relação à do enunciado**

**Implementação (detalhes, parâmetros, bibliotecas, funções…)**

**Testes e Resultados**

**Conclusões e Trabalho Futuro**