Universidade do Minho

Trabalho Prático nº2

Licenciatura em Engenharia

Comunicações por Comutador

2ºSemestre
Ano lectivo 2014/2015

Grupo: 4

Número de aluno	Nome
A57917	Miguel Rodrigues
A67736	Marcelo Gonçalves
A67728	Ricardo Silva

Data: 9 de Junho

1 Introdução

No âmbito da unidade curricular de Comunicações por Computador foi proposto ao grupo de trabalho que se desenvolvesse um jogo. Esse jogo teria de estar distribuído por vários servidores, e deveria permitir a vários jogadores competirem entre si. A linguagem escolhida para a realização deste trabalho foi JAVA.

O objetivo principal deste projeto é a utilização de diferentes protocolos de comunicação, nomeadamente o TCP, para a comunicação entre servidores, e o UDP para a comunicação entre servidor e cliente.

Ao longo deste relatório será explica a forma como foi desenvolvido o projeto, bem como as dificuldades que foram surgindo á medida que este era desenvolvido.

2 Diferenças/Adições no Protocolo sugerido

Durante a implementação foram efetuadas algumas alterações em relação ao protocolo inicialmente sugerido.

As alterações efetuadas não são sobre o protocolo em si, são simplesmente a inclusão de pequenas mensagens extra em relação ao sugerido. Nomeadamente:

- Foi acrescentado ao campo ‘0’, do Reply, uma pequena mensagem, esta mensagem serve para dizer se correu tudo bem ou não.
- Na mensagem de “Logout” foi inserido o nome do utilizador;
- Foi inserido o campo ‘30’ para designar a porta onde o cliente será atendido.

3 Implementação

3.1 Protocolo

O primeiro detalhe da implementação foi a implementação do protocolo de implementação. Como definido no enunciado as mensagens trocadas entre cliente e servidor tinham por base PDU's, por isso decidimos criar uma classe PDU com os campos base, e uma lista com os campos que designam mensagens. Esta lista de campos é baseada numa outra classe Fields que designa o tipo de pedido.

Segue em seguida a arquitetura implementada.

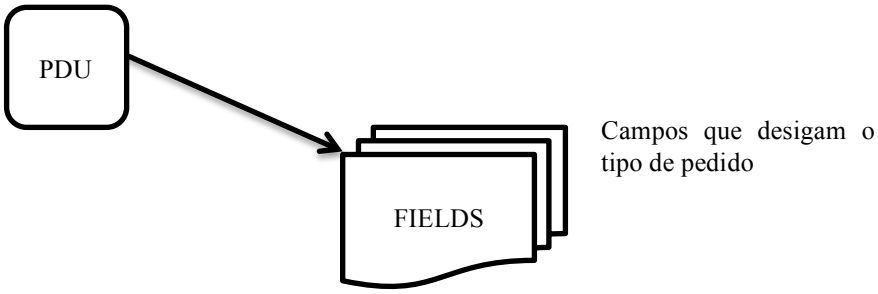

Figure 1 Arquitetura de um PDU

3.2 Cliente Servidor

Sendo que a aplicação a desenvolver tem de interagir com vários clientes em simoltáneo, decidimos que teríamos uma Thread a aceitar clientes, que por sua vez iria criar uma outra Thread, sendo esta segunda thread que interage com o cliente a partir do momento que este se conecta.

Em suma, o cliente envia o pacote “Hello”, o servidor cria uma nova Thread e esta envia a resposta com um pacote “Reply” que contém a nova porta para a qual o cliente deve enviar os seus pedidos, ou seja, a nossa aplicação cria um Thread por cliente, e cada thread tem um socket UDP para comunicar com o seu cliente.

Em seguida é apresentada a arquitetura da nossa aplicação.

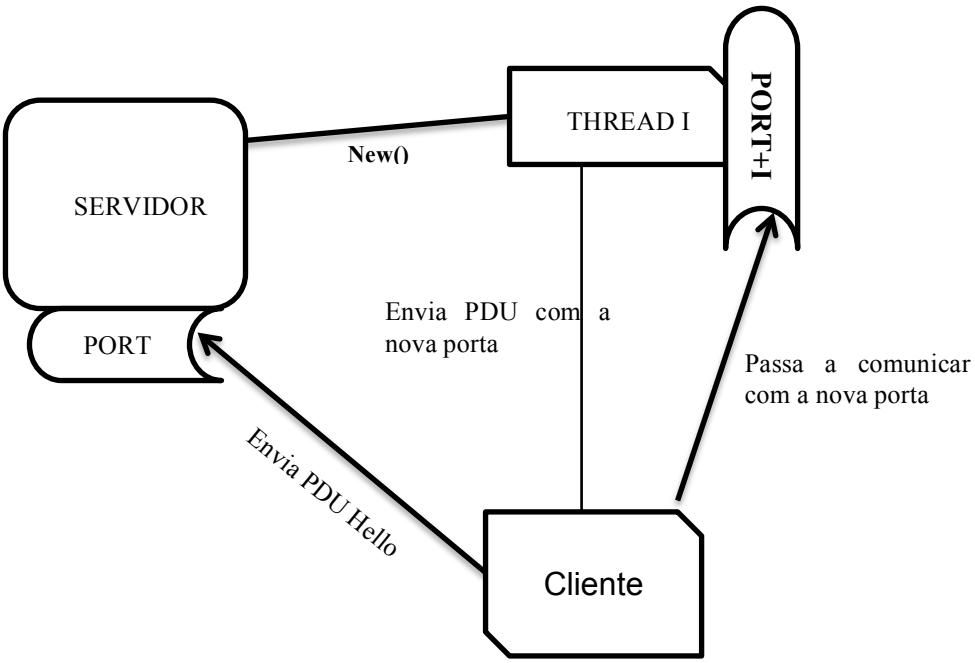

Figure 2 Arquitetura Cliente Servidor

3.3 Memória Partilhada

À medida que os clientes se vão conectando, a nossa aplicação vai criando várias Threads que possuem memória partilhada, foi portanto necessário proteger esta memória para o múltiplo acesso.

Para isso foi criada uma classe Manager que contém toda a informação presente no servidor, é a base de dados do servidor. Esta classe foi encapsulada, e através do mecanismo de "locks" e "conditions" foi protegida de maneira a que só um processo possa aceder á informação de cada vez.

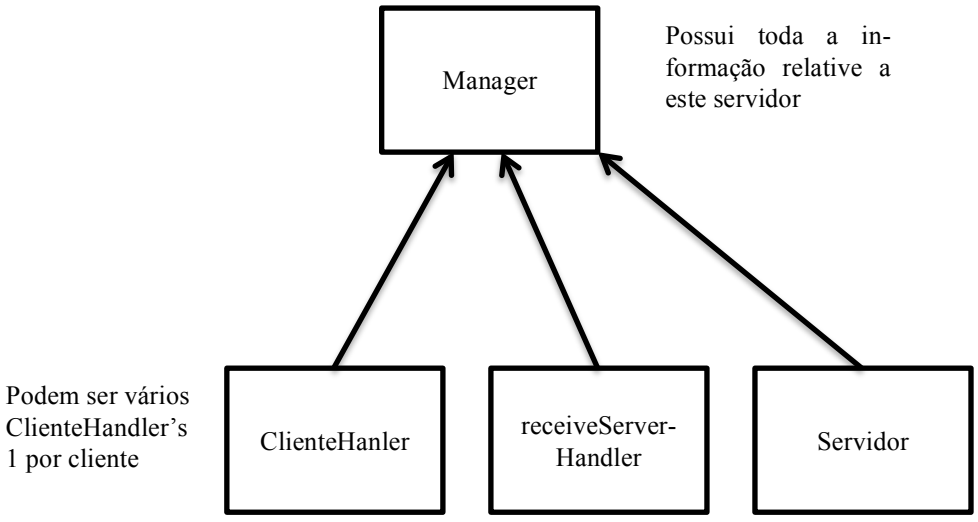

Figure 3 Classes que partilham memória

3.4 Jogo/Reenvio de pacotes

Como o principal objetivo deste projeto é desenvolver um jogo, foi necessário proceder á sua implementação. Um jogo como descrito no enunciado é um conjunto de perguntas, ás quais está associado uma música e uma imagem. Quando um cliente decide jogar um jogo, a nossa aplicação tem de á hora de início enviar as perguntas para o cliente. O principal problema aqui existente é que como o protocolo utilizado é UDP, existe perda de pacotes, logo, é necessário corrigir os erros quando eles existirem.

Para garantir que o cliente recebe a questão completa, ou seja, as perguntas, opções imagem e música, a nossa aplicação envia divide o desafio em PDU's, em seguida envia todos os PDU's para o cliente, e aguarda uma mensagem do cliente. Por sua vez, o cliente vai recebendo os PDU's que lhe vão chegando, e quando deixa de os receber, procede á sua verificação para saber se estão completos, para em seguida apresentar a questão ao cliente. Apresentamos de seguida o algoritmo seguido para comprovar que possuímos todos os pacotes.

1. Recebe os pacotes que consegue;
2. Procura o último pacote, e verifica se este diz se existem mais;
   (a) Se o último pacote disser que existem mais, vai pedindo ao servidor até chegar ao último.
3. Quando chegar ao último, vai percorrer todos os pacotes pela sua ordem para detetar falhas, e pedir os pacotes que faltarem.
4. Quando possuir todos os pacotes envia mensagem ao servidor a dizer está tudo bem;
5. Apresenta a questão;

Este algoritmo é efetuado para todas as questões até ao final do desafio. No final do desafio só são apresentados os resultados quando todos os utilizadores tiverem respondido a todas as perguntas.

3.5 Malha de Servidores

Um dos objetivos da aplicação era que esta pudesse correr com uma malha de servidores, ou seja, N clientes para N servidores. No entanto, o grupo não conseguiu atingir esse patamar. A nossa aplicação só consegue funcionar com 2 servidores em simultâneo.

No momento em que o nosso servidor começa a funcionar, a nossa aplicação cria uma thread, esta thread cria um socket TCP que fica á escuta que um novo servidor se conecte. Quando isso ocorrer, são criadas duas novas threads uma para enviar informação para o outro servidor, e outra para ficar á escuta.

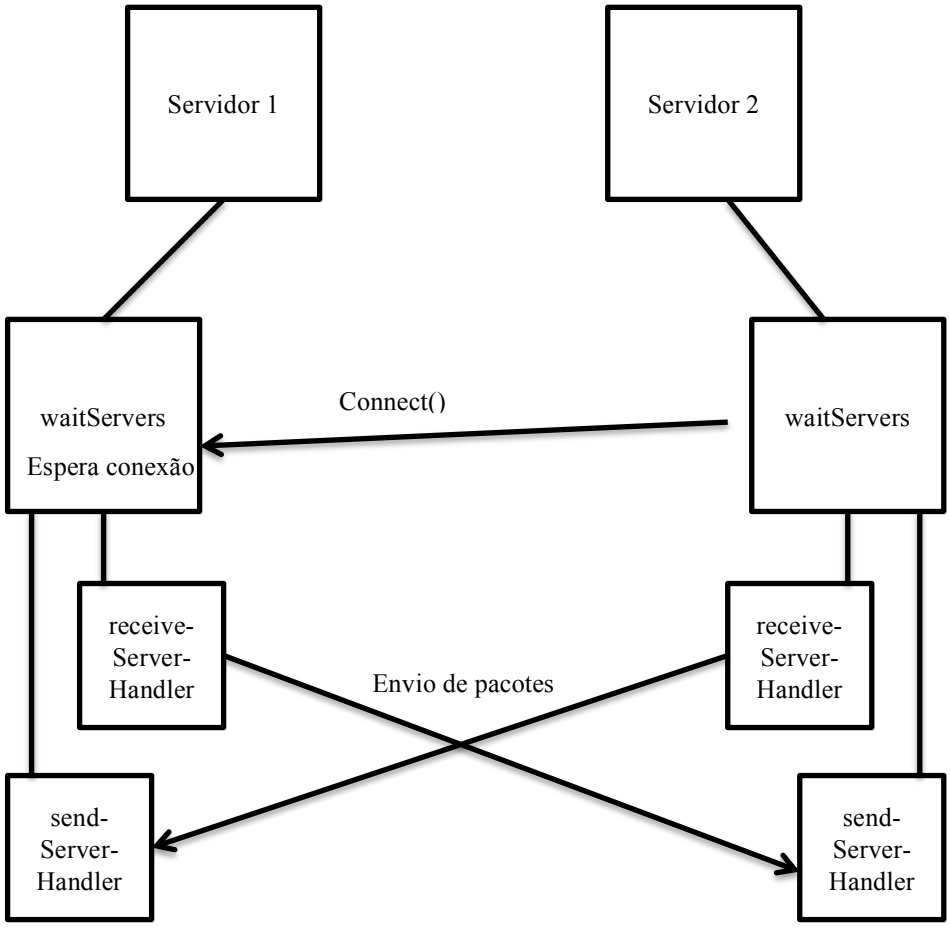

Figure 4 Arquitetura da aplicação com 2 servidores

Esse novo servidor, é adicionado à memória partilhada, e será utilizado pelo manager para obter informação que não possui, mas que o outro servidor pode possuir, ou seja, quando o cliente pedir uma listagem dos desafios o Manager vai ao novo servidor buscar os desafios, agrupa com os seus e envia para o cliente.

Concluindo quando está um novo servidor conectado, quando é solicitada ao servidor informação, está será um conjunto da informação do servidor local, com o outro servidor conectado.

4 Conclusões

Após a conclusão deste projeto, ficamos ainda mais familarizados com os conceitos do control de concorrência e concluimos que estes são parte crucial de qualquer sistema distribuído.

Com este projeto adquirimos novos conhecimentos relativos á comunicação entre computadores. Através da utilização de diferentes protocolos de transmissão de dados, permitiu-nos compreender melhor as diferenças existentes entre eles.

Concluindo pensámos, que o principal objetivo deste trabalho foi atingido, apesar de não termos conseguido implementar completamente o sistema pedido.