Sistemas Distribuídos

Relatório dos Trabalhos 1 e 2

12 de Novembro de 2006

Informação dos Autores

Grupo

Diogo Emanuel de Almeida Guerra – 501041522 – deag@student.dei.uc.pt

Hugo Andre Dias Pereira Vieira Cura – 501041532 – hcura@student.dei.uc.pt

Bernardo Montezuma de Carvalho Planas Raposo – 501021206 – braposo@student.dei.uc.pt

Horas gastas no trabalho

1º Meta - 30 Horas

2º Meta - 10 Horas

Introdução

O trabalho pedido consistia na implementação do popular jogo quatro-em-linha em Java, utilizando um dos ambientes gráficos do mesmo. Como se trata de uma cadeira de Sistemas Distribuídos, o verdadeiro desafio deste trabalho era a ligação entre os vários clientes e servidores, isto através de duas abordagens diferentes.

Para tal o trabalho foi dividido em duas fases: numa primeira etapa implementámos todo o ambiente gráfico e construímos a aplicação para suportar apenas ligações por Sockets (TCP e UDP), de seguida foi-nos pedido que desenvolvessemos uma nova versão da aplicação, desta vez suportando também ligações RMI.

Este relatório visa explicar um pouco mais detalhadamente os pormenores da nossa aplicação, a que nós demos o nome de Quattro. O documento inclui descrição da arquitectura da aplicação, uma lista dos testes efectuados, um pequeno manual e outros detalhes que considerámos relevantes.

Arquitectura da Aplicação

Cliente

Sockets

É constituido pela classe gráfica Game4L que cria uma instância da classe ClientSocket.

A partir da classe ClientSocket são criadas quatro threads para recepção e envio de informação através de sockets UDP e TCP chamadas UdpListener, UdpSender, TcpListener e TcpSender.

Assim que é feito o pedido de login, é estabelecido um socket TCP, e se o login for feito com sucesso, as threads acima referidas são criadas, assim como os sockets UDP (nas respectivas threads).

No fim destas acções, são recebidos alguns dados tais como rankings e online users, ficando o client à espera de novas instruções. Neste momento podem ser feitas várias acções:

- Envio de chat global, que vai usar a thread UdpSender para enviar a mensagem.
- Início de jogo, que usa a thread TcpSender para enviar o request.
- Pedido de estatísticas pelo socket TCP.
- Logout, através do socket TCP,

A nível de receber informação, o cliente pode receber as seguintes informações:

- Na thread UdpListener:
- Rececção de chat global
- Actualização de rankings
- Actualização de users
- Servidor desconectado
- Na thread TcpListener:
- Recepção de convite de jogo

Assim que está um jogo activo, o utilizador não pode convidar ninguém e não pode ser convidado (controlado no servidor). Passa a poder enviar e receber chat privado (socket TCP) e a jogar.

A lógica de jogo é efectuada toda no servidor, apenas é também verificado no cliente a vez de quem vai jogar. No fim de cada jogo, cada jogador recebe a mensagem se ganhou, perdeu ou empatou e é-lhe perguntado automaticamente se quer começar novo jogo ou não. Durante o jogo o cliente pode também

desistir do jogo.

Todas estas acções são enviadas para o servidor por TCP e processadas no mesmo à excepção do envio de mensagens de chat globais.

RMI

No cliente de RMI, a aplicação é constituida pela mesma classe gráfica Game4L e por uma ClientRmi que comunica com o servidor.

Foi criada uma thread TempThread que simplesmente auxilia o ClientRmi em janelas gráficas para que o servidor quando executa remotamente métodos do cliente, não fique bloqueado à espera.

Todas as acções disponibilizadas no cliente baseado em sockets são executadas no cliente RMI, em que as threads Senders são substituídas por métodos do servidor chamados pelo cliente e as threads Listeners são substituídas por métodos do cliente chamados pelo servidor.

Servidor

O servidor usado é comum às duas implementações e tanto recebe ligações de clientes por sockets, como por RMI. Cada servidor assim que iniciado, tenta ligar-se à porta definida, se esta tiver livre, age como servidor principal, se não, age como watchdog.

Watchdog

É criada uma thread Server4LPinger com uma flag a indicar se o watchdog tem que actuar em modo RMI ou em modo socket, e fica em wait à espera que a thread referida lhe faça o notify.

Se for escolhido o modo socket, a thread Pinger vai ligar-se por UDP ao servidor principal (à thread Server4LResponder) e pinga-o todos os segundos para garantir que o servidor principal está sempre activo.

Se for escolhido o modo RMI, a thread Pinger vai fazer o lookup do servidor e também todos os segundos acede ao método ping.

Ao fim de 3 falhas, a thread notifica a thread principal (servidor em modo watchdog) que passa a actuar como servidor principal.

Servidor Principal

O servidor principal, assim que estabelece o socket, faz rebind à instância de si mesmo, para poder aceitar ligações RMI. Se anteriormente era watchdog, faz load a todos os ficheiros de dados no disco e no fim fica à espera de ligações TCP.

Criam-se threads para envio e recepção de sockets UDP, Server4LUdpSender e Server4LUdpListener respectivamente. Para clientes RMI, os métodos são acedidos remotamente pelo cliente, dependendo de cada acção.

Para clientes TCP, para cada socket que é estabelecido, é criada uma Server4LThread que gere toda a comunicação com esse cliente. Esta thread cria também uma Server4LThread2 que vai funcionar como thread que envia informação para o cliente quando a thread que a cria está bloqueada à espera de receber informação dos clientes. Assim que a thread recebe algum request, processa-o e executa na thread principal, podendo aceder a outro cliente (TCP ou RMI).

Existem várias estruturas para guardar os dados, todas elas baseadas em hashtables. Existem 6 hashtables cuja chave é o nome do utilizador, à excepção da hashtable games que tem como chave o id do jogo:

- **Clients** – contém objectos PlayerInfo. Toda a informação dos clientes (incluíndo a thread ou o stub de RMI) autenticados;
- **Clients_backup** – contém objectos OnlineUser2. Backup do PlayerInfo da hashtable anterior, mas sem os objectos não serializáveis.
- **Users** – contém objectos User. Registo de todos os users registados no servidor.
- **Games** – contém objectos Game. Registo de todos os jogos activos no momento.
- **Busy** – contém inteiro com o game_id. Registo de todos os utilizadores que estão ocupados a jogar.
- **Old_users** – contém objectos OnLineUser2. Esta hashtable é preenchida com uma cópia do clients_backup (este é inicializado novamente) quando um watchdog entra em modo servidor principal, e serve como um buffer para users que façam o retry, ou seja, quando um user se reconecta, o servidor vai verificar se está neste sitio, e se estiver aproveita a sua informação e remove-o de lá.

Formato das Mensagens

Para enviar mensagens entre o cliente e o servidor, na versão em que a comunicação se processa a partir de sockets, utilizámos objectos do tipo Packet. Cada Packet é constituído por um código que identifica o tipo de mensagem, e um objecto com o conteúdo, que pode ser de diversos tipos.

Na implementação baseada em RMI como os métodos são chamados remotamente, os objectos são enviados por parâmetro, não sendo necessária a utilização de objectos do tipo Packet.

Tratamento das Falhas

O tratamento de falhas transitórias no cliente é feito de duas maneiras, de acordo com o sistema de comunicação utilizado. Analisamos de seguida cada um dos casos.

Sockets

Ao se ter estabelecido a ligação entre o cliente e o servidor ficamos com um socket de ligação entre ambos. Todo o mecanismo de verificação e controlo de falhas está dependente dessa mesma ligação. Assim, cada excepção que der nessa ligação é por nós controlada de forma a que se torne perfeitamente transparente para o cliente, efectuando tentativas de estabelecer uma nova ligação (no máximo 3) até se assumir que o servidor não voltará a funcionar.

RMI

No RMI processa-se forma semelhante, usando na mesma as excepções que ocorrerem para se restabelecer a ligação ao servidor, mas em vez de sockets utiliza-se a ligação RMI activa.

Mecanismo de Replicação

O mecanismo de replicação diz respeito a como se lida com falhas do lado do servidor. De seguida abordamos o seu funcionamento.

O primeiro servidor a ser inicializado fica como primário, enquanto que outros servidores ao se tentarem ligar verificam que a porta já se encontra ocupada e ficam como watchdogs. A partir daqui os watchdogs ficam com uma thread a pingar o servidor principal para saberem se este se encontra a funcionar, sendo essa informação processada no servidor principal por intermédio de uma thread que recebe os pings.

No caso de um ping enviado por um watchdog não ser bem sucedido, esse watchdog passa a ser o servidor principal, retomando toda a actividade e dados que o servidor principal até ai vinha a utilizar. O tratamento dos dados envolvidos é explicada em detalhe na parte de arquitectura lógica.

Integração de Sockets e RMI

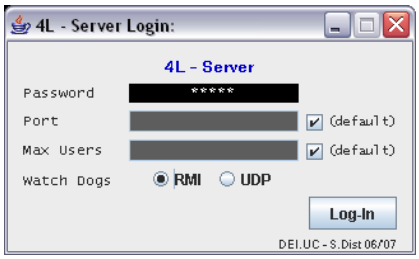
A integração entre as duas possibilidades de comunicação (Sockets e RMI) é feita de forma completamente transparente para o cliente, sendo toda a informação processada no servidor.

Assim sendo, o servidor ao receber a informação de um client verifica qual é o tipo de ligação que este utiliza e responde-lhe usando o mesmo tipo, permitindo assim que clientes com diferentes ligações comuniquem entre si através do servidor e sem qualquer alteração visível para o cliente.

Manual do Utilizador

Servidor

O primeiro passo para se inicializar o servidor é efectuar o login.



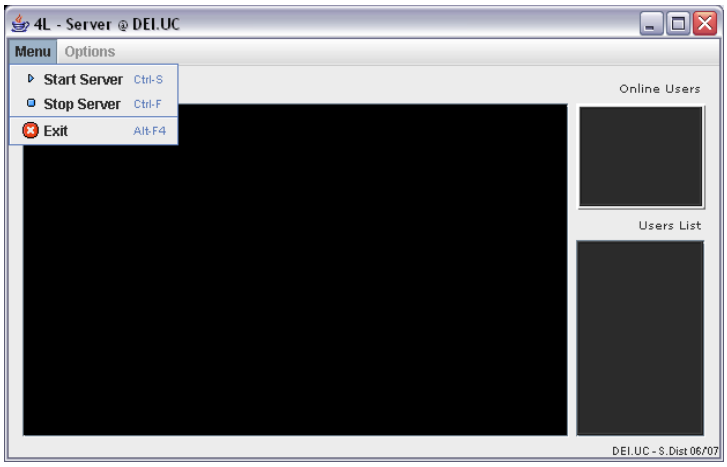
Esta imagem mostra a janela de login, onde temos de preencher os campos de modo a entrar no servidor.

Os campos são, a password, o porto(port), o número máximo de utilizadores(max users) e o modo em que queremos utilizar os watch dogs. O campo port e max users tem também valores default no caso de não serem preenchidos. Sendo que port tem um valor de 7000 e max users de 10.

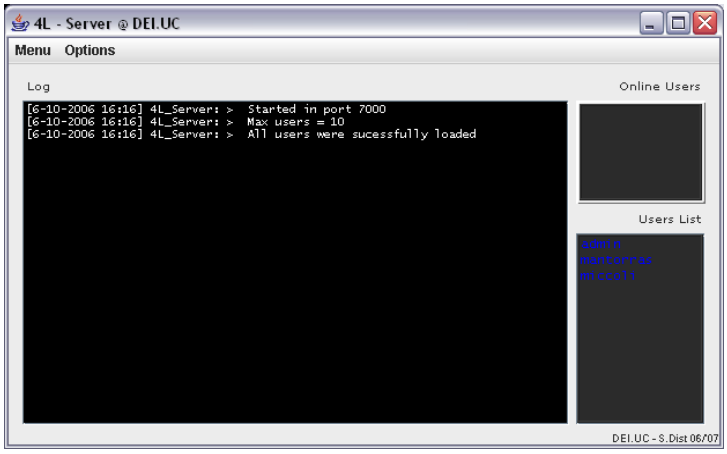
Após um login bem sucedido vamos ao encontro da seguinte janela:

Nesta janela temos uma visão geral do interface que nos permite utilizar o servidor.

Para se inicializar o servidor temos de aceder à opção start no Menu, como nos indica a figura seguinte.



Após o start, temos um exemplo do servidor já em funcionamento:



Analisando a figura podemos ver algumas zonas distintas, nomeadamente, um log dos eventos realizados no servidor, a lista de utilizadores registados, bem como a lista de utilizadores online.

Cliente

Após ter configurado e inicializado um servidor ou tendo conhecimento do endereço de um servidor, pode então começar a utilizar o nosso cliente.

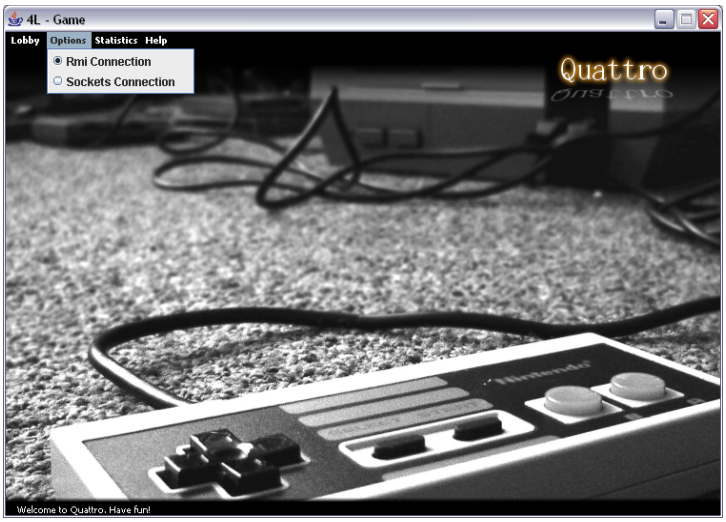
Ao executar o cliente irá ter uma vista geral do ambiente gráfico da aplicação.

Como se pode observar o ambiente gráfico é constituído por um menu com diversas opções (que serão analisadas mais à frente neste manual), a área de conteúdos e a status bar (ainda com a mensagem de boas vindas). Comecemos então por explorar o menu. Este é constituído por alguns sub-menus, nomeadamente, lobby, options, statistics e help. Segue-se em detalhe cada um dos sub-menos, com algumas imagens ilustrativas dos mesmos.

Lobby

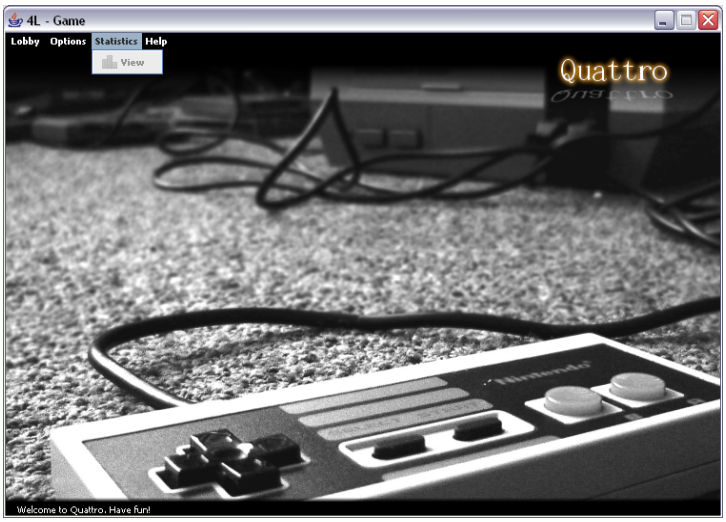
No lobby podemos efectuar o login (ou logout) e ainda terminar a aplicação.

Options



Nas options podemos optar entre a utilização de uma ligação via Sockets ou RMI

Stastistics



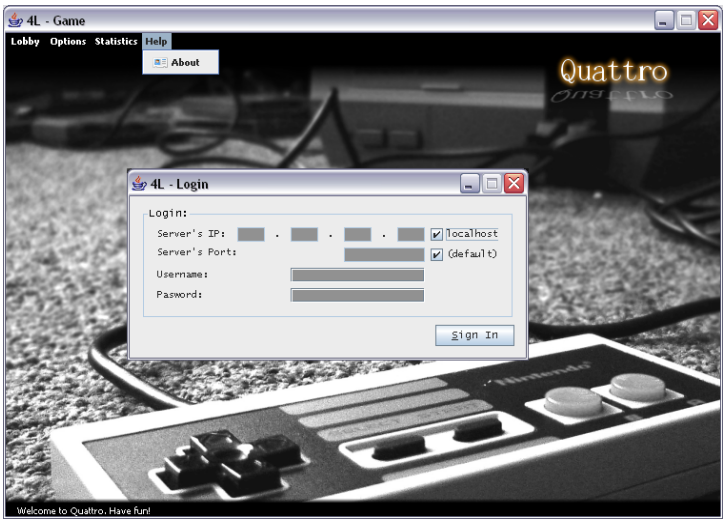
Nas statistics podemos consultar os resultados dos nossos jogos.

Help

O help apenas mostra a informação dos autores.

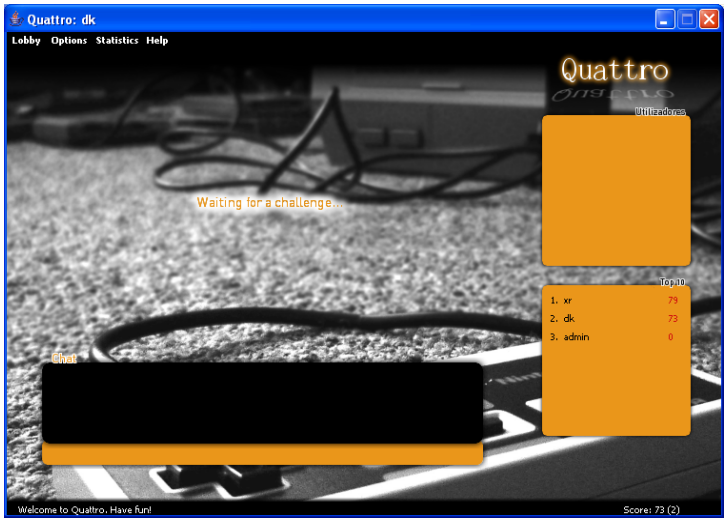
Após explorarmos o menu vamos então ver como se inicia a ligação ao servidor.

Como já vimos anteriormente para fazermos o login basta ir ao lobby no menu e carregar em login. Ao faze-lo vamos ter à seguinte janela:



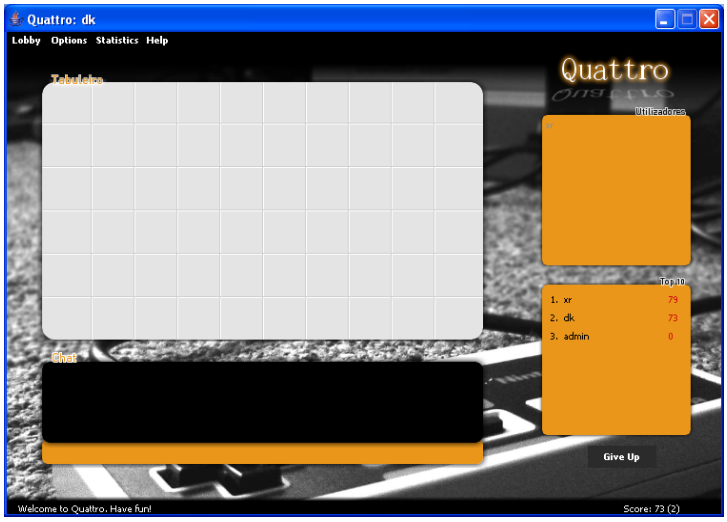
Aqui temos de preencher detalhes de ligação ao servidor, nomeadamente o endereço e o porto (valores que podem ser os default, localhost e 7000, respectivamente). Além disso temos de preencher os campos com os dados da conta de utilizador. Caso não tenha uma, ser-lhe-à perguntado se quer registar um novo utilizador, devendo dizer que sim, de modo a poder usufruir da aplicação.

Efectuado que foi o login, temos agora a àrea de conteúdos com mais elementos, como se pode ver na figura seguinte:



Agora podemos observar a janela de chat e o local para introduzir o texto, assim como a lista de utilizadores disponíveis para jogar e ainda o Top 10 de utilizadores da aplicação.

Jogo



Para se jogar basta carregar no nome do adversário pretendido da lista de utilizadores online. Após o outro jogador ter aceite o desafio, o jogo será iniciado e aparecerá o tabuleiro no ecrã principal. Caso contrário, voltará para o ecrã anterior onde poderá desafiar novos jogadores.

Manual de Instalação

1. Compilar toda a source utilizando o comando:

   javac *.java

2. Criar os stubs utilizando os comandos:

```
rmic ClientRmi
rmic Server4L
```

3. Para se poder utilizar o servidor é necessário iniciar o RMI Registry:

```
start rmiregistry
```

4. Finalmente já se pode correr os servidores e clientes. Para isso basta:

```
java ServerFrame (para o servidor)
java Game4L (para o cliente)
```

Testes Efectuados

Cliente

Ligação RMI – RMI

- Conectar-se ao servidor – OK
- Ver estatísticas pessoais – OK
- Desafiar jogador – OK
- Jogar – OK
- Desistir do jogo – OK
- Aceitar rematch – OK
- Rejeitar rematch – OK
- Utilizar o chat – OK
- Desconectar-se do servidor – OK
- Terminar cliente RMI durante o jogo – OK

Ligação RMI – Sockets

- Conectar-se ao servidor – OK
- Ver estatísticas pessoais – OK
- Desafiar jogador – OK
- Jogar – OK
- Desistir do jogo – OK

- Aceitar rematch – OK
- Rejeitar rematch – OK
- Utilizar o chat – OK
- Desconectar-se do servidor – OK
- Terminar cliente RMI durante o jogo – OK
- Terminar cliente Sockets durante o jogo – OK

Ligação Sockets – Sockets

- Conectar-se ao servidor – OK
- Ver estatísticas pessoais – OK
- Desafiar jogador – OK
- Jogar – OK
- Desistir do jogo – OK
- Aceitar rematch – OK
- Rejeitar rematch – OK
- Utilizar o chat – OK
- Desconectar-se do servidor – OK
- Terminar cliente Sockets durante o jogo – OK

Servidor

Servidor Principal

- Inicializar servidor – OK
- Definir limite de utilizadores – OK
- Desligar servidor – OK
- Adicionar utilizador – Não implementado
- Apagar utilizador – Não implementado
- Editar utilizador – Não implementado

Watchdogs RMI

- Pingar servidor principal – OK
- Entrar como servidor principal – OK

- Assumir todas as configurações do servidor principal – OK

Watchdogs Sockets

- Pingar servidor principal – OK
- Entrar como servidor principal – OK
- Assumir todas as configurações do servidor principal – OK