UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA DEP. ENGENHARIA INFORMÁTICA	**Distributed Systems** **1st Semester 2011-2012** **PROJECT #1** **Deadline: 21-Oct-2011**

"MyDropBox": Online Storage for Sharing Files and Tasks

Alunos:

Igor Nelson Garrido da Cruz #2009111924

Saul Miguel Pereira da Costa Santos #2009118782

Introdução / Objectivos:

Neste projecto foi-nos proposta a concepção de um software de comunicação entre computadores idêntico ao famoso "DropBox". Nesta aplicação deve haver uma relação servidor – cliente, em que o cliente pode ver/adicionar/editar/remover tarefas, ver os utilizadores online, ver/fazer upload/download de ficheiros do servidor e ainda apagar ficheiros do servidor.

Para o seu desenvolvimento recorremos aos conhecimentos adquiridos nas aulas, tais como a implementação de serviços distribuídos usando Sockets TCP/IP, Threads e JavaRMI.

Como correr a aplicação:

-Para executar o programa é necessário ter o **jnotify** na path do sistema;

-Tanto o cliente como o serivor devem ser executados a partir da sua directoria de forma a automaticamente criar a pasta partilhada;

-O cliente é executado passando como argumentos os ips do servidores 1 e 2:

<ex: java -jar DropBitsFinal.jar localhost 192.168.1.65>

-O servidor recebe como parâmetros o ip do servidor de destino e o ip dele próprio ou localhost:

<ex: java -jar MyDropBoxServer.jar 192.168.1.65 localhost>

-A recepção de ficheiros não está implementada de forma transparente, é assim necessário executar os comandos:

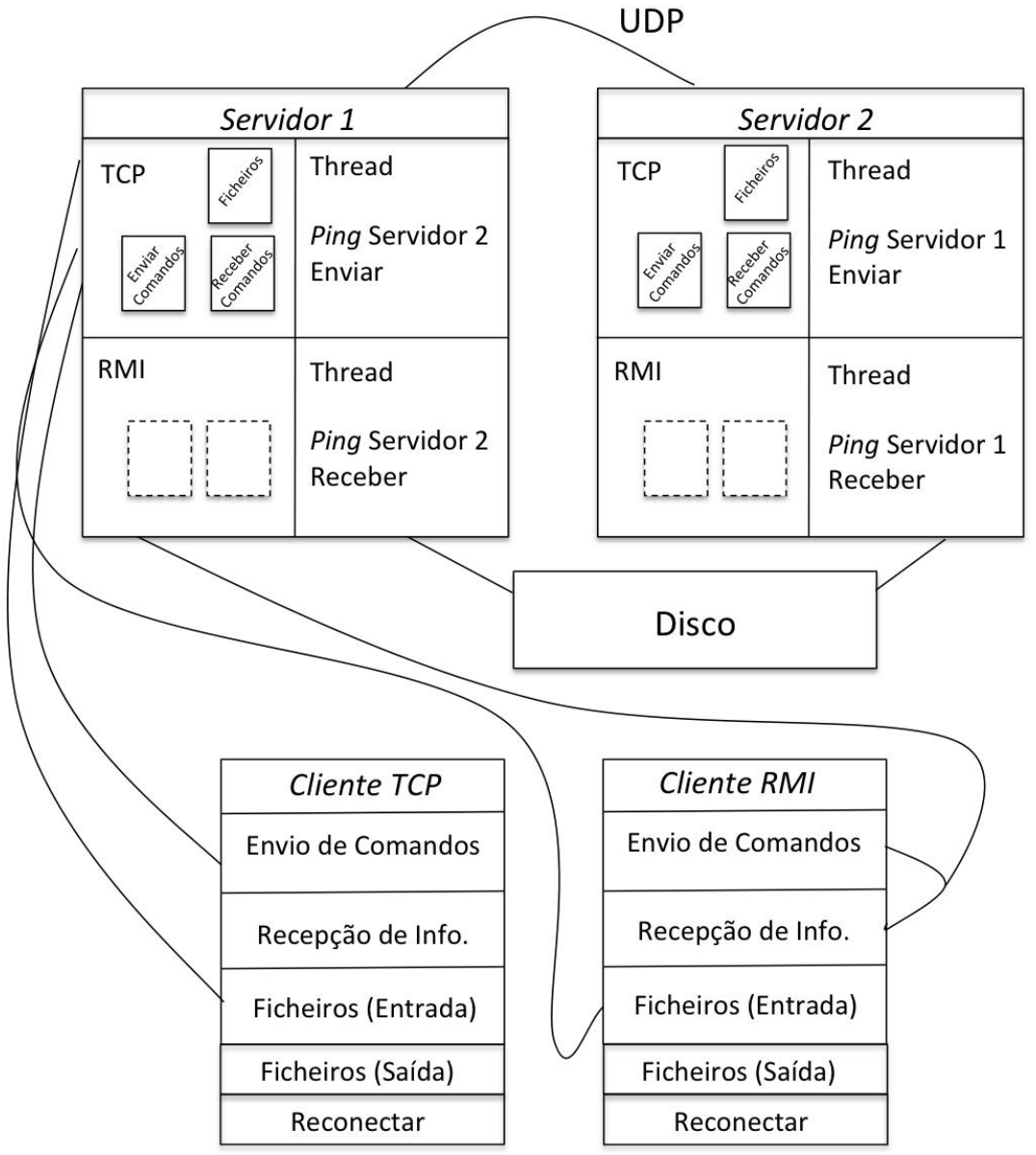
download file

Insert Filename:

nome_do_ficheiro

-Para Ficheiros grandes, deve aumentar a heap do java com -Xmx2g (tanto no server como no client).

Arquitectura Interna:



Descrição da Arquitectura:

Na nossa arquitectura há dois servidores a funcionar em paralelo. Estes servidores estão preparados para receber ligações de clientes TCP para os quais criam diferentes threads: cada nova ligação é processada numa nova thread, e por sua vez cada nova ligação terá acesso a três threads diferentes: uma para tratamento de ficheiros, outra para envio de comandos e ainda outra para recepção de comandos. No caso das ligações recebidas por clientes RMI, também cada nova ligação é tratada numa nova thread. No entanto, no que diz respeito à troca de informação, a tecnologia RMI trata de criar as threads automaticamente. Para além disto, os servidores têm ainda duas outras threads para o envio e recepção de mensagens entre os dois, mensagens essas que são utilizadas para controlar o mecanismo de "Fail-Over".

Relativamente ao cliente TCP, são criadas seis threads: uma para envio de comandos, outra para recepção de informação, duas para entrada e saída de ficheiros uma responsável pela reconecção em caso de queda da ligação e por fim uma thread que corre o jnotify para sabermos quando ocorrem eventos nos ficheiros.

Quanto ao RMI são criadas cinco threads: uma para envio de comandos, duas para entrada e saída de ficheiros, uma para reconeções ao servidor e por fim uma thread para correr o jnotify.

Comunicação:

No nosso programa há dois tipos de Clientes: TCP e RMI. No cliente TCP são criados dois sockets de comunicação com o servidor, um para comandos (porta 9500) e outro para ficheiros (porta 9501). Desta forma, caso o utilizador queira enviar um ficheiro muito grande, não há perigo da aplicação ficar muito tempo em espera, uma vez que são utilizados sockets diferentes para os ficheiros e para as restantes funcionalidades. No caso do Cliente RMI, também separamos a ligação para transmissão de objectos de comunicação da ligação de transferência de ficheiros. Para tratar da comunicação utilizamos o RMI *Naming.lookup*, e para os ficheiros voltamos a recorrer a um socket na porta 9501, tal como no Cliente TCP. O servidor está programado para receber ligações por TCP e por RMI, podendo lidar em simultaneo com clientes de ambos os tipos.

Sincronização de ficheiros (TCP):

O nosso programa permite aos utilizadores o upload, download e eliminação de ficheiros e também modificar o nome dos mesmos. Para desenvolver esta caracterísitca recorremos ao Jnotify – que é uma biblioteca que nos permite receber informação de quando um ficheiro é adicionado, removido ou modificado numa determinada directoria. Para além disso permite tratar essas notificações de forma a que sejam chamados os métodos correspondentes para adição/remoção/modificação dos ficheiros no servidor.

Tratamento de excepções (Sockets, RMI):

Na nossa implementação as excepções foram devidamente tratadas tanto no cliente TCP como no cliente RMI. Caso o servidor falhe, o cliente vai receber a excepção e vai tentar reconectar-se. Para além disso, enquanto o servidor estiver offline, o cliente vai indicar essa falha na ligação mas permite ao utilizador continuar com algumas das suas operações: permite-lhe adicionar tarefas – que ficam armazenadas numa ArrayList em memória – e permite-lhe adicionar ficheiros para upload – que também ficam numa ArrayList até que a ligação seja restabelecida. Quanto às outras operações o cliente envia ao utilizador uma mensagem a informar de que não está ligado ao servidor, e fica a aguardar que o utilizador escolha outra opção.

Mecanismo de "Fail-Over":

Na nossa implementação temos dois servidores a correr em paralelo, e que enviam mensagens UDP entre eles, para controlo de actividade. Quando o primeiro servidor falhar, vai deixar de enviar essas mensagens ao segundo, e este ao deixar de os receber deve passar para primeiro plano nas ligações aos clientes e continuar as tarefas que estavam a ser executadas pelo primeiro. Entretanto, quando o servidor um estiver novamente online, deverá ficar também à escuta de mensagens do servidor dois (que está neste momento activo), e quando (ou se..) os deixar de receber, toma o seu lugar, e assim sucessivamente. Desta forma o nosso sistema terá um uptime superior, em principio capaz até de funcionar 24x7!

Manual do Utilizador:

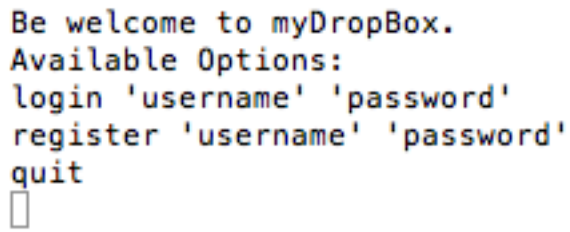


Fig1. Menu Login no Cliente

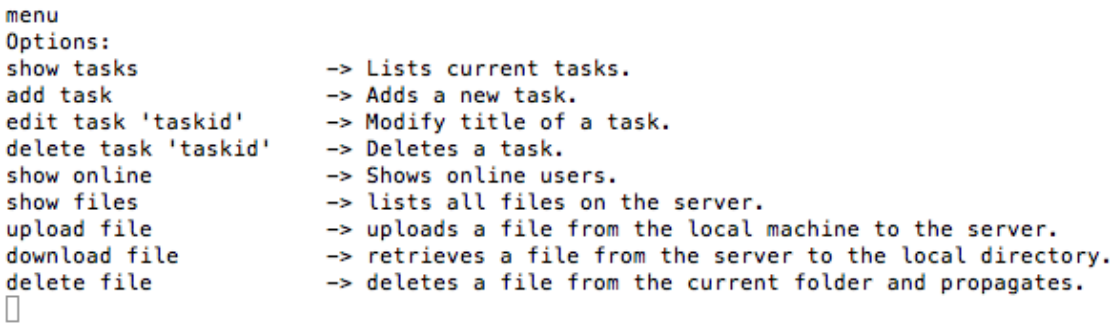


Fig2. Menu Opções no Cliente

Ao correr a aplicação o cliente encontrará um menu de registo / login. Caso não tenha conta na ***MyDropBox*** deverá criar uma, caso já tenha basta que introduza o username e password para efectuar o login.

Após a autenticação bem sucedida, o cliente terá acesso a um menu de opções:

(1) **Mostrar Tarefas** – onde poderá visualizar a lista de tarefas ;

(2) **Adicionar Tarefa** – que, tal como o nome indica, lhe permite adicionar uma tarefa à lista. Esta nova tarefa terá um ID gerado automaticamente e os outros clientes serão notificados da sua criação;

(3) **Editar Tarefa** – permite ao cliente modificar o título de alguma(s) tarefa(s);

(4) **Remover Tarefa** – permite marcada a tarefa como concluída e eliminá-la da lista. Mais uma vez, os outros clientes serão notificados;

(5) **Utilizadores Online** – nesta opção o cliente poderá visualizar a lista de clientes que estão online nesse momento;

(6) **Mostrar Ficheiros** – mostra ao cliente uma lista com os ficheiros existentes no servidor;

(7) **Enviar Ficheiro** – permite ao cliente enviar (upload) um ficheiro para o servidor. Este ficheiros será partilhados com os outros clientes;

(8) **Transferir Ficheiro** – tal como o nome indica, permite descarregar um ficheiro do servidor para o computador local do cliente;
(9) **Apagar Ficheiro** – permite ao cliente apagar um ficheiro da sua pasta de partilha, que será também apagado do servidor e das pastas dos outros clientes;

T-SPEC: Descrição dos testes feitos à aplicação:

Em baixo estão apresentados alguns testes feitos à nossa aplicação.

Teste 1 – Utilização normal do programa – Interacção Cliente/Servidor

Foi testada a ligação entre o cliente e o servidor, assim como as diferentes operações implementadas. Todas elas funcionarm sem problemas.

Teste 2 – O servidor falha com um cliente a correr (Fail-Over)

Dois servidores a funcionar, um activo outro em espera, um cliente com login efectuado. O servidor activo falhou (Ctrl-C). O cliente recebeu a mensagem de que o Servidor estava teporariamente offline, entretanto o servidor espera tornou-se activo, o cliente recebe a mensagem de que a ligação está de novo activa. O mecanismo de Fail-Over está a funcionar correctamente.

Teste 3 – O cliente adiciona uma tarefa após uma quebra de ligação

Dois servidores estão a correr em paralelo, um cliente está ligado. O primeiro servidor falha (Ctrl-C). O cliente recebe a informação de que a ligação caiu, mas ainda assim insere uma nova tarefa. Recebe informação de que a tarefa foi adicionada à lista temporária, enquanto a ligação está a ser corrigida. Entretanto o servidor 2 activou-se, o cliente recebe essa informação. Tal como esperado, a tarefa adicionada offline foi automaticamente enviada ao servidor e foi adicionada à lista.

Teste 4 – Notificação de remoção de tarefas enviada para todos os clientes

Estão dois clientes ligados (para efeitos de teste utilizou-se um TCP e outro RMI). Uma determinada tarefa é removida por um utilizador. Tal como esperado, o outro cliente recebe informação de que o utilizador "Y" apagou a tarefa.