SD

Meta 1

Elaborado por:

Oleg Tryembich - 2011159465

Miguel Veloso - 20111527540

UNIVERSIDADE DE COIMBRA

Índice

Introdução **- 3**

Arquitetura **- 4**

Modelo de Dados **- 5**

Protocolos de Comunicação **- 6**

Exception Handling e Fail-Overs **- 8**

Instalação e Configuração **- 9**

Descrição de testes **- 10**

Conclusão **- 11**

Introdução

Este projeto tem como objetivo o desenvolvimento de uma aplicação de marcação de reuniões, havendo interação entre utilizadores.

A arquitetura baseia-se nas ligações Cliente – Servidor TCP – Servidor RMI. No caso de o Servidor TCP não se encontrar disponível, o cliente liga-se a um Servidor Secundário que serve como "backup server". Os servidores intermédios (TCP) estabelecem uma comunicação via UDP com a finalidade de saberem o estado um do outro. Finalmente, o servidor RMI contém toda a informação sobre a aplicação, guardando-a sempre que o utilizador faz uma ação, prevenindo assim casos de falha. Inicialmente o RMI faz "load" de toda a informação, guardando-a em "runtime" e atualiza a base de dados sempre que se produz uma alteração.

A aplicação disponibiliza várias opções como: criar reunião normal, criar reunião personalizável, aceitar/recusar convites, convidar utilizadores para reunião, criar "action items", criar "key decisions", criar tarefas destinadas a um utilizador, "chat" por "action item", etc.

Seguidamente, iremos analisar a arquitetura da aplicação, ligações, testes e "fail-overs".

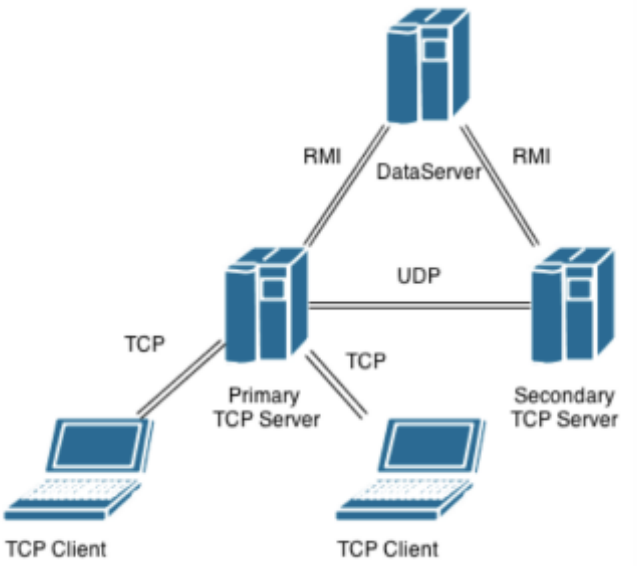
Arquitetura

Na arquitetura deste trabalho dispomos de 4 entidades distintas: o Cliente, o Primeiro Servidor, o Segundo Servidor e o Servidor RMI. A ligação entre Cliente e Primeiro/Segundo Servidor é feita através de TCP. Simultaneamente o Primeiro Servidor testa se o Segundo Servidor está em funcionamento, através de uma ligação UDP, recebendo uma mensagem que descreve o estado da ligação do mesmo.

A ligação do cliente aos Servidores TCP depende da disponibilidade dos mesmos, isto é, se o Servidor 1 não se encontrar disponível o cliente tenta mais uma vez a ligação ao mesmo e em caso de falha liga-se ao Servidor TCP "backup" (Servidor 2).

Existe uma ligação prévia do Servidor Primário ou Secundário ao Servidor RMI. É este que gere toda a informação que é apresentada e guardada/enviada pelo Cliente.

Toda a informação percorre o seguinte caminho:



Modelo de Dados

Nesta meta, optámos por guardar toda a informação em ficheiros. Quando o Servidor RMI é executado faz "load" de toda a informação disponível nos ficheiros .txt e guarda-a numa estrutura de dados em "runtime". Essa estrutura é baseada numa arquitetura orientada a objetos (Java), utilizando "arrayLists" para guardar Reuniões, Utilizadores, Action Items, etc.

Sempre que existe alguma atualização aos dados disponíveis, o Servidor RMI guarda toda a informação nos ficheiros, prevenindo assim que falhas técnicas levem à perda de informação.

O Servidor RMI dispõe das classes Utilizadores, Reunião e ActionItem, essenciais para guardar toda a informação. Por exemplo, na classe Utilizadores é guardada toda a informação inerente aos mesmos, como o "username", "password", reuniões em que está inscrito e convites disponíveis; na classe Reuniao encontramos o nome, data, local, objetivo, utilizadores inscritos na mesma, administrador, "action items", "key decisions" e tarefas; por último, na classe ActionItem encontram-se o nome e o "chat" relativo a esse "action item".

Em suma, toda a informação é guardada em ficheiros .txt e em "runtime" nas classes apropriadas.

Protocolos de Comunicação

TCP

Usando este protocolo no projeto (criando Sockets TCP) conseguimos garantir uma caraterística essencial para uma aplicação deste género: garantia de entrega de uma mensagem.

Em termos de comunicação, o cliente informa o servidor de que quer aceder a uma dada funcionalidade enviando uma "string" que indica a opção escolhida. Essa opção, do lado do servidor, é interpretada para posteriormente ser ativada uma dada funcionalidade no RMI Server.

UDP

O protocolo UDP, ao contrário do TCP, não confirma que as mensagens chegam ao destinatário, preocupando-se apenas em enviá-las.

Este protocolo serve apenas para garantir o funcionamento dos Servidores no caso de haver um "crash" num deles.

RMI

Neste caso, o Primeiro ou o Segundo Servidor vão ter uma "thread" que apenas trabalha como RMI Client. Quando um cliente se conecta por RMI este possui uma interface do lado do servidor (RMI) e uma do seu lado, utilizada pelo Servidor TCP.

Através desta interface o cliente pode fazer grande parte das operações disponíveis, como “login”, registo, consultar as reuniões, criar uma reunião, etc. Todas estas operações são feitas sem a necessidade de programar o envio e recepção de mensagens por “sockets” entre o cliente e o servidor.

Nota: Cada cliente, no Servidor TCP, é tratado por uma “thread” diferente. Com isto queremos dizer que, cada vez que um cliente se conecta ao primeiro servidor, este cria uma “thread” para o tratar individualmente. Assim, não há conflito ao interpretar e executar os pedidos de cada um.

Exception Handling e Fail-Overs

Neste ponto vamos abordar as exception handling e fail-overs em conjunto devido à interacção entre os dois no nosso projecto. Com isto quer-se dizer que devido às Exceptions vamos conseguir evitar quebras no sistema (Fail-Overs).

Cada vez que há um problema na ligação do socket, caso o cliente esteja ligado ao primeiro servidor, o erro é interpretado e é tentada nova ligação ao mesmo e caso a ligação seja restabelecida o pedido do é enviado novamente. Caso o primeiro servidor esteja desligado aquando da ligação do cliente, é interpretado o erro e é logo estabelecida a ligação ao segundo servidor.

Sempre que existe um problema na ligação, é disfarçado perante o cliente, havendo um time out e pós re-conexão ou nova ligação a um dos servidores.

No caso dos dois servidores estarem desligados, após tentativa de ligação, se a falha for sucessiva o cliente desliga-se, sendo que o problema encontrado é da rede e não do sistema em si.

Qualquer erro de ligação, tanto no cliente, como no servidor TCP são interpretados e tratados sem que seja emitido um erro.

Instalação e Configuração

Após fazer download do .zip "SD Projecto", copiar as pastas para o workspace do Eclipse. Criar 4 novos projectos no eclipse com os nomes: Cliente, Servidor 1, Servidor 2 e ServidorRMI.

No eclipse, fazer "refresh" das pastas dos projectos. Irá notar que agora os projectos contêm packages relativas ao projecto, contendo as classes e o código respectivo ao mesmo.

Agora, para testar a aplicação terá de correr o Servidor RMI, Servidor 1, Servidor 2 e Cliente (a ordem por que liga os Servidores e o Cliente é importante, pois sem a ligação dos Servidores o Cliente não poderá realizar operações).

Após executar todos estes passos terá ao seu dispor a aplicação podendo utiliza-la da forma como achar melhor.

Descrição de Testes

Teste	**Funciona**
Casos de operação invalida dá erro	Sim
Cliente reconecta-se ao servidor1	Sim
Cliente conecta-se ao servidor2 em caso de falha	Sim
Login	Sim
Registo	Sim
Criar reunião	Sim
Convidar utilizadores	Sim
Aceitar/Recusar convites	Sim
Marcar reunião simples	Sim
Marcar reunião personalisada	Sim
Criar Key Decisions	Sim
Consultar Key Decisions	Sim
Criar tarefas	Sim
Consultar todas as tarefas	Sim
Consultar as minhas tarefas	Sim
Concluir tarefas	Sim
Propor uma data para a reunião	Sim
Votar numa data	Sim
Criar action item	Sim
Modificar action items	Sim
Chat (em tempo real)	Sim
Chat (guardar conversa)	Sim
Apagar action items	Sim
Consultar action items	Sim
Consulatr convites	Sim

Conclusão

Neste trabalho implementamos uma aplicação com a arquitetura Cliente – Servidor TCP – Servidor RMI.

Esta aplicação tinha como objectivo a marcação de reuniões e interacção entre utilizadores relativamente às mesmas. Tinha como opções: a marcação de reuniões simples e personalizadas, a criação de "key decisions", tarefas, datas, e "action items", sendo que o ultimo permite a comunicação, através de um chat, entre utilizadores.

Este trabalho serviu para aprofundar o conhecimento relativo aos servidores RMI, TCP e UDP, trabalhando este tipo de comunicações. Foi também útil para a pratica de tratamento de falhas de comunicação.

Conseguimos (o grupo) atingir a meta que nos propusemos alcançar, tendo preenchido grande parte, ou mesmo todos os requisitos necessários.