IdeaBroker – Idea Management and Trading

Projeto de Sistemas Distribuídos

Projeto v1

Grupo Constituído Por:

Bruno Miguel Oliveira Rolo(2010131200)

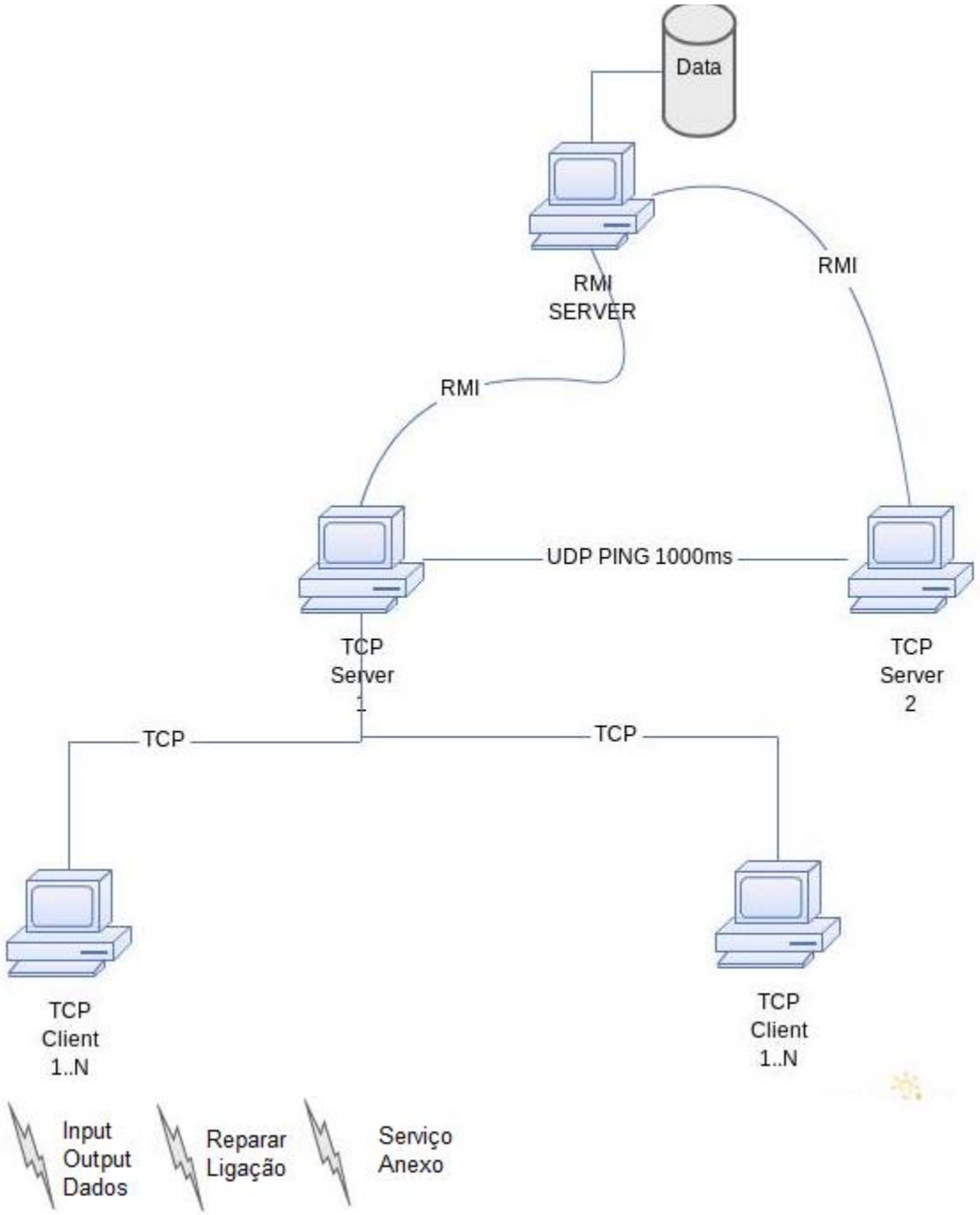
João Artur Ventura Valério Nobre(2010131516)

Introdução

Este trabalho surge no âmbito da cadeira Sistemas de Distribuídos e tendo como objetivo implementar uma rede de Ideias como um sistema distribuído.

Na primeira meta deste projeto é necessário apresentar um sistema com uma interface básica, que possua todas funcionalidades pedidas e que permita ser acedido tanto através do protocolo TCP que por sua vez liga ao Data Server pelo protocolo RMI, ao mesmo tempo que suporte uma solução de fail-over transparente para o utilizador. Nas várias secções deste relatório, descrevemos de que forma abordamos estes requisitos no nosso trabalho.

Arquitetura Interna da Aplicação



Tal como se pode ver pelo esquema, temos três servidores, dois deles aceitam pedidos do protocolo TCP e o que contêm a Base de dados aceita pedidos por RMI dos servidores TCP. Entre os servidores TCP existe uma constante verificação da conexão de um com o outro, algo que será explicado na secção de Fail-Over.

Ambos os servidores possuem as mesmas funcionalidades. O servidor primário, o que não está em modo de backup, possui uma main thread que está constantemente a aceitar pedidos por parte de cliente TCP, enviando-os para outra thread, onde o pedido é processado, bem como processa pedidos a realizar ao RMI.

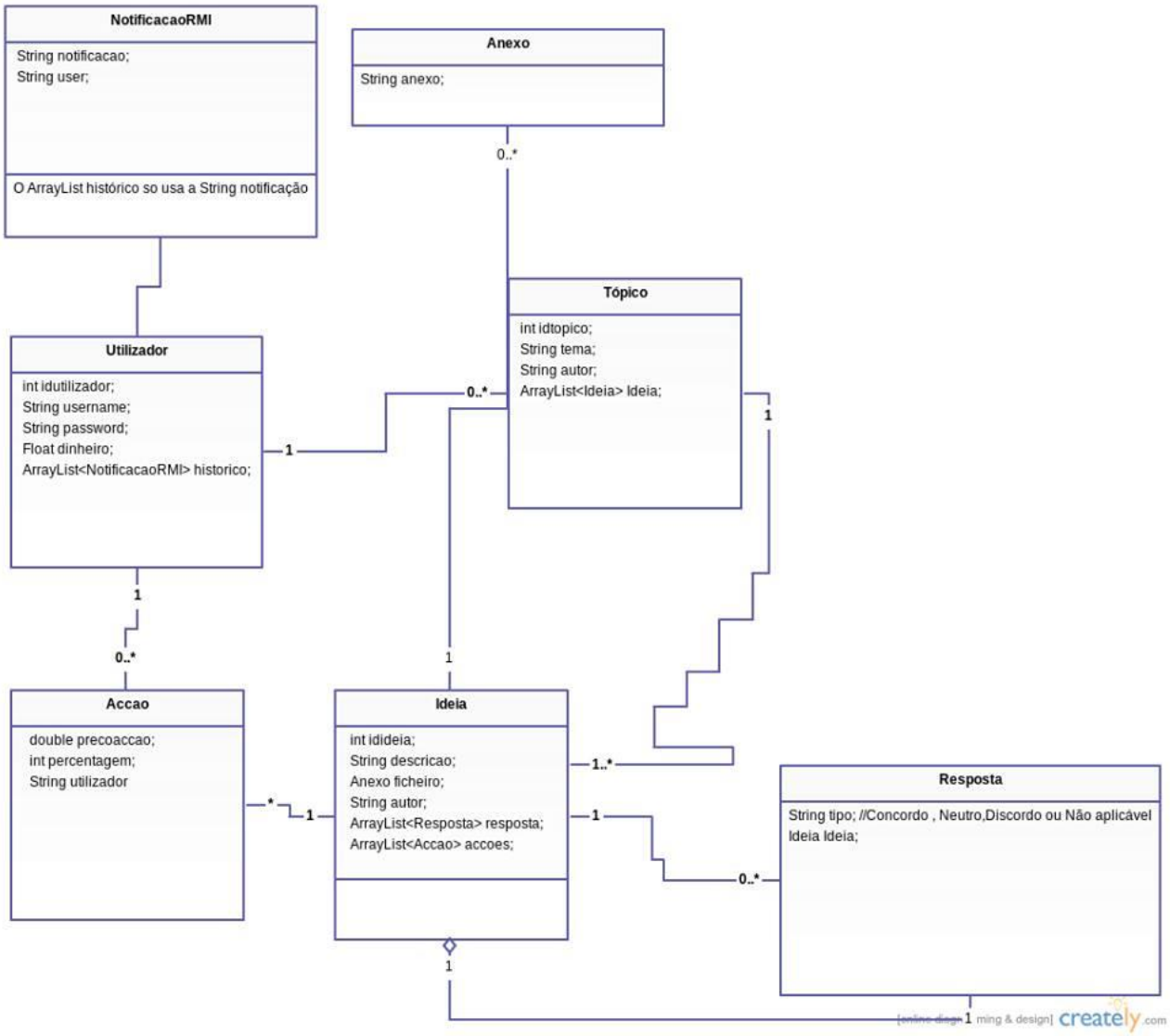
Sempre que o cliente TCP envia um ficheiro de anexo, uma nova thread é lançada para processar o recebimento e a escrita do mesmo para o disco.

Convém ainda notar que sempre que o servidor de backup entra passa a servidor primário este faz uma leitura do disco, para que os dados estejam consistentes em ambos os servidores.

O servidor TCP , para cada cliente tem uma thread e cada thread tem a sua própria ligação para o servidor RMI, ou seja N clientes N ligações RMI , N threads no Servidor TCP. Por este motivo tivemos que sincronizar os acessos ao disco do RMI não permitindo dados inconsistentes , por exemplo , um cliente está a adicionar uma ideia no servidor RMI enquanto no Servidor TCP está a ser feita uma verificação de se uma X ideia já existe quando a ideia está a ser adicionada, logo é necessário esta sincronização adicional. Esta sincronização não era necessária se a verificação fosse feita no servidor RMI visto que ele tem o seu metódo de sincronização interno.

Por último o cliente TCP possui também uma thread que gere o input e output com o servidor e com o cliente, ou seja a interatividade com o utilizador, possuindo também uma main thread que verifica problemas de conexão e repara-os se necessário. Existindo mais uma thread que gera o envio de pacotes de informação para o servidor, informação essa que consiste em quais os comandos introduzidos em modo offline, e que deverão ser processados pelo servidor uma vez reestabelecida a ligação. O cliente TCP possui ainda uma thread que é lançada sempre que o cliente deseja fazer o upload de um anexo para o servidor, gerindo então o envio do ficheiro.

Modelo de dados da aplicação



Analisando o esquema, facilmente se percebe que existe um conjunto de ficheiros que armazenam informação permanente, um conjunto de estruturas de dados, bem como o servidor RMI mantem essas coleções dessas estruturas em memória

O servidor RMI contém uma lista de utilizadores que contém todos os utilizadores registados no sistema, uma lista de Tópicos que corresponde a todos os tópicos do sistema e uma lista de notificações que contém uma lista de todas as notificações do sistema. Sendo que o servidor RMI escreve em disco cada vez que recebe novos dados para adição do servidor TCP para manter a sincronização, o processo inverso também ocorre que é sempre que for requisitado dados pelo utilizador ,é feito um pedido ao servidor rmi para enviar os dados actuais no sistema. Por exemplo, um utilizador vai a funcionalidade ver tópicos , a primeira coisa que é feita é chamar o método remoto getTopicos() que retorna todos os topicos existentes no momento no servidor RMI. Fazendo este processo asseguramos que temos sempre a informação mais recente disponivel ao cliente no momento do pedido feito por este.

De seguida apresenta-se uma descrição do conteúdo dos ficheiros:

“utilizadores.ser” – contém a lista de todos os utilizadores registados
“topicos.ser” – contém todos os tópicos do sistema
“notificacao.ser” – contém todas as notificações

Especificação dos Protocolos

Nas ligações por TCP, utiliza-se o protocolo TCP para efectuar a comunicação entre o servidor e os vários clientes, o protocolo UDP para a comunicação entre o servidor primário e o backup.

O primeiro servidor a ser iniciado, o primário, cria dois serverSockets, o de "informacao" no porto 1234 e o "estado" no 1233. Ficando a aguardar ligações por parte dos clientes. O servidor backup, o secundário, cria igualmente dois sockets nas portas 2234 e 2233, mas apenas aceita ligações quando este alternar para o servidor primário.

No socket "informacao" é utilizado um Stream para envio de notificações e controlo de fail-over, verificando se existe algum problema de ligação. Por sua vez, no socket de "estado" é usado para um stream para troca de mensagens entre o cliente e o servidor, para controlo do funcionamento da aplicação (escolha da funcionalidade pretendida por parte do cliente) e troca de dados (mensagens de sucesso etc).

Para o envio do ficheiro em anexo, é criado um socket TCP no porto 3333 que através da criação de um ObjectStream transfere o ficheiro entre o servidor e o cliente.

No RMI, o servidor regista um objecto remoto chamado "Ideia" no RMIRegistry no porto default 1099 e no porto 6000. Para caso falhe um porto tenha outra para alternar. O cliente quando é iniciado tenta aceder ao objecto no porto default 1099 do endereço do servidor RMI em caso de insucesso, tanta aceder ao objecto no porto 6000.

No protocolo UDP é utilizado um Datagram Socket no porto 3000 que envia de 2 em 2 segundos uma mensagem de controlo (ping) do servidor secundário para o primário de modo a controlar se este está activo e quando não receber resposta, o de backup assume o seu papel.

Tratamento de excepções em sockets e RMI

Tanto na versão TCP como na RMI, quando o cliente fica sem conexão com o servidor, tenta reconectar-se novamente 3 vezes com intervalos de tempo ao servidor primário. Caso não consiga fazer a ligação, tenta o servidor de backup. Se a ligação voltar a não ser bem sucedida, volta a tentar o servidor primário.

No caso de a ligação entre o cliente e o servidor falhar, o utilizador pode continuar a utilizar a aplicação, tanto esteja a utilizar a versão TCP como a RMI. Se optar por criar uma Ideia ou um Topico ou Responder a um Topico poderá fazê-lo sem se aperceber que não se encontra conectado. Estes pedidos serão guardados num buffer e enviados para o servidor logo que seja possível, o qual irá recebê-los e processá-los.

Por opção, o buffer encontra-se limitado a 5 pedidos por utilizador. Caso o utilizador pretender efetuar um pedido diferente destes, será mostrada uma mensagem a informá-lo que não possui uma ligação estabelecida como servidor e apresentado o menu principal.

Solução Fail-Over

Quando existe um “crash” do servidor principal este vai ser detetado pelo servidor de backup, através do ping contínuo feito por UDP que existe entre os dois. Após a deteção de um crash o servidor de backup vai esperar durante 3 s que o servidor principal retome funcionamento. Caso tal não aconteça o servidor backup vai assumir controlo, lendo a versão mais recente dos dados existente no disco.

Enquanto toda esta operação acontece, o cliente vai ter uma thread a iterar pelos endereços conhecidos, tentando conectar-se a um deles, e esperando 1s entre cada tentativa. O utilizador pode continuar a utilizar o cliente sem se aperceber de nada, desde que não utilize nenhuma das opções que necessitam de comunicação com o servidor, caso isso aconteça vai-lhe ser apresentada uma mensagem avisando que existe um problema, e pedindo para que aguarde. Se o utilizador quiser criar um post ou enviar uma mensagem privada (apenas na versão TCP) pode fazê-lo dado que existe um buffer que armazena os posts e mensagens privadas criadas enquanto os servidores estão em baixo.

Após a retoma de serviço pelo servidor de backup, caso o servidor que “crashou” consiga retomar execução, vai se aperceber que já existe um servidor a executar, e vai então tomar o papel de backup.

Quando existe um “crash” do servidor principal este vai ser detetado pelo servidor de backup, através do ping contínuo feito por UDP que existe entre os dois.

Descrição dos Testes Efectuados

Para testar este projecto foram efectuados vário testes, entre os quais:

- Num cenário de apenas 1 servidor (TCP e RMI) e 1 cliente (TCP), testamos todas as funcionalidades pedidas.

- Num cenário de 2 servidores (TCP e RMI), testar a interação entre eles (comunicação UDP e interação a quando de falhas).

- Num cenário de 2 servidores (TCP) e 1 cliente (TCP) e o serivdor RMI, efetuamos os mesmos testes descritos em cima, ou seja, tiramos partido das funcionalidades existentes ao mesmo tempo que eram simulados erros e retomas dos servidores, tentando verificar a permanência dos dados e não existência de erros.

No entanto , no failover no caso do cliente estar num submenu e acontecer a troca do servidor primário com o servidor secundário o servidor que vai passar a servidor primário não consegue localizar-se na zona do cliente.Isto poderia ser resolvido se passasemos a informação atómicamente por String em vez de estar constantemente a trocar dados entre servidor-cliente.