Relatório Projeto
Sistemas Distribuídos
Meta 1

UNIVERSIDADE D
COIMBRA

Trabalho realizado por:
Guilherme Gaspar - 2018278327
José Diogo Gaspar - 2018306763

Arquitetura de software

O servidor RMI corre três threads (a main, a VerificaServer e a VerificaBackupServer). Ao iniciar, o server carrega os dados que estão nos ficheiros de objetos para as respetivas estruturas (ArrayLists). De seguida, dependendo de ser o server principal ou o server de backup, irá chamar uma das threads VerificaBackupServer e VerificaServer. Caso o servidor seja o prinicpal, significa que o porto ao qual o "LocateRegistry.createRegistry(porto)" se ligou foi o 7001. Portanto tal sucedido indica que a thread chamada é a VerificaServer. Esta thread verifica via udp se packets estão a ser recebidos para o porto 7070. Caso estejam a ser recebidos packets para esse porto significa que o server passará a ter função de secundário/backup pois o server ligado com o porto 7002 ao "LocateRegistry.createRegistry(porto)" já está definido como principal e a enviar pakctes para este. Na eventualidade de nenhum packet ser recebido, significa que este será o primário e irá então ele enviar packets, neste caso para o porto 7069. Se a thread chamada for a thread VerificaBackupServer, irá tentar primeiramente receber packets do porto 7069. Se durante 5 segundos não for recebido nenhum packet, significa que se deu uma avaria no servidor principal, e terá que passar a ser este servidor a tomar esse papel, ficando portanto a enviar packets para o porto 7070.

A adminConsole corre duas threads, a main e a ContaTempo. Ao iniciar a consola, ela vai procurar por qual o servidor que está ativo dentro do grupo de dois portos possíveis (7001, 7002). Ao conseguir conectar-se com um dos servidores inicia uma verificação das eleições que estão guardadas nos ficheiros de objetos a partir da thread ContaTempo. Nesta thread, as eleições irão ser vítimas de uma verificação às duas datas (inicial e final) de forma a que quando uma tenha de ser iniciada ou terminada isso aconteça. Na consola é também disponibilizado um menu de interação para gerir várias features do programa.

A MesaDeVoto começa por correr 3 threads (a run(), a AtualizaMesa e a verificaMesa). Ao longo do programa, a thread HandleSession é iniciada caso entre um eleitor e é encerrada quando o eleitor sai. A nível de sockets tem 3 sockets ativos e todos no mesmo porto, apenas em grupos diferentes. Um para procurar terminais, outro para lidar com a sessão dos votantes e outro para receber pings dos terminais.

O Voting Terminal começa por correr 2 threads (a run() e a Atualiza). Ao longo do programa, a thread Session é iniciada caso entre um eleitor, juntamente com a ControlaTempoSessão, que limita o tempo ativo da sessão. Ambas as threads são encerradas quando o eleitor sai ou quando é dado um timeout. A nível de sockets tem 3 sockets ativos e todos no mesmo porto, apenas em grupos

diferentes. Um para ficar à escuta de mesas de voto, outro para lidar com a sessão dos votantes e outro para enviar pings à mesa de voto.

As únicas threads implementadas fora do ficheiro das classes em que são chamadas são a ContaTempo, a VerificaServer e a VerificaBackupServer).

Detalhes sobre o funcionamento do servidor Multicast

Protocolo

Todas as mensagens enviadas pelo servidor são precedidas por um "$" e todas as mensagens enviadas pelos terminais são precedidas por um "@".

- *1 - type | search; available | no* - Mesa de voto pergunta se há terminais disponíveis.
- *2 - type | search; available | yes; terminal | 123* - Terminal responde se tiver disponível.
- *3 - type | ack; terminal | 123* - Mesa de voto informa terminais qual o terminal que capturou da pool.
- *4 - type | welcome; user | 30120* - Mesa de voto envia ao terminal o cartão de cidadão do user que entrou no terminal
- *5 - type | login; cc | 30120; username | gui; password | 123* - Terminal envia ao server os dados de login introduzidos pelo user no terminal
- *6 - type | status; logged | on/off; msg | Bem-vindo ao eVoting/Username ou Password incorretos!* - Estado de login após mesa de voto receber dados de login.
- *7 - type | item_list; cc | 30120; item_count | 2; item_0_name | A Candidate List ; item_1_name | Another Candidate List* - Envia ao voting terminal com o user 30120 as listas de candidatos daquela eleição
- *8 - type | vote; cc | 30120; list | A Candidate List* - Voting terminal envia ao server a lista em que votou.
- *9 - type | timeout; user | 30120;* - Voting terminal avisa mesa de voto que deu timeout.
- *10 - type | update; terminal | 123;* - Voting terminal envia ping a mesa de voto a informar que ainda está a correr.

O servidor MultiCast recebe como argumentos o seu endereço de multicast e o seu departamento. Verifica a que server se vai ligar e se existe uma mesa de voto nesse mesmo departamento (através de uma chamada de um método remoto).

Ao correr, cria um socket no port 4321 e esse socket é adicionado ao grupo que está a correr no endereço introduzido como argumento. Inicia também a thread

que vai estar constantemente à espera de receber pings dos terminais para enviar a informação para o servidor RMI e a thread que vai avisar o servidor que esta mesa de voto está ligada.

De seguida, fica à espera de um user que introduza o seu cartão de cidadão. Ao introduzir um cartão de cidadão válido (verificado através da chamada de um método remoto), é mostrado as eleições que o user pode votar (recebidas através do RMI), este seleciona a eleição e a Mesa de Voto num sistema de Envia (protocolo 1) - Recebe (protocolo 2) - Envia (protocolo 3), obtém as informações do terminal a que se pode ligar e inicia a thread HandleSession que vai lidar com toda a sessão do user. A partir daqui, a mesa de voto fica pronta para receber outro user.

No HandleSession, a mesa de voto recebe os dados de autenticação do voting terminal e se estiverem corretos (verificados através de um método remoto) envia as listas candidatas da eleição selecionada previamente ao User. Este escolhe em qual pretende votar e a mesa de voto chama um método do servidor RMI para registar o voto e termina a thread. Se a sessão do user der timeout, ou por ter falhado a autenticação mais de 3 vezes ou pelo tempo de sessão de 120s ter sido excedido, a thread é também encerrada.

Em caso de Failover, o Multicast server lida da mesma forma que a admin Console.

Detalhes sobre o funcionamento do servidor RMI

O servidor RMI tem um total de 35 métodos remotos. Temos vários métodos que fazem operações semelhantes mas com dados diferentes. Os nomes são bastante explicativos das tarefas que fazem e os métodos estão descritos em JavaDoc.

- Métodos que adicionam dados: registar, addLista, criarEleição, addGrupo, addMesa, adicionaPessoaLista, adicionaVoto, writeBD.
- Métodos que removem dados: rmvLista, rmvGrupo, rmvMesa, removePessoaLista.
- Métodos que atualizam dados: mudaNomeLista, atualizaDescricao, atualizaDataInicio, atualizaDataFim, atualizaTitulo, writeBD.
- Métodos usados para obter dados: obterValor, getEleição, getPessoas, readBD, getEleições, getPessoa, filterEleições.
- Métodos usados para verificar dados: verificaCC, verificaEleicao, verificaLista, verificaEleitor, verificaOnServer.
- Métodos de callBack com mesa de voto e admin: olaAdmin, adeusAdmin e olaMesaVoto.

Failover

Quando se encontram dois servidores ligados, um principal e um secundário/backup, o servidor principal está constantemente a enviar packets ao secundário para em caso de avaria do principal, o secundário ao deixar de receber

os packets durante 5s se tornar no principal, ficando este a enviar os packets, para quando o server principal se iniciar novamente, ficar a receber os packets e fazer portanto a sua atribuição de secundário/backup.

Neste caso de failover do servidor principal e apenas ficar o servidor secundário/backup como principal, tanto a adminConsole como as MesaVoto se comportam de forma semelhante. A partir de um while true em cada, estas estão sempre a tentar ligar-se a um dos server, caso o outro não dê, a partir dos try e catch. Desta forma, aquando uma falha no servidor, tanto a adminConsole, como a MesaVoto ficam a aguardar por uma interação, e aquando esse acontecimento, tentam entrar no outro servidor. No entanto apenas se ligam a esse servidor após ele ser reconhecido como principal (não receber durante 5s os packets) assim como depois de terem alguma interação nelas mesmas, de forma a que tentem contactar o servidor, vejam que existiu uma falha, podendo assim ligar-se ao outro.

Distribuição de tarefas pelos elementos do grupo

Numa fase inicial começámos ambos a desenvolver os métodos básicos da AdminConsole e do RMIServer. Após essa fase, o Diogo Gaspar continuou encarregue destas duas classes e o Guilherme Gaspar encarregue da MesaDeVoto e do VotingTeminal.

O trabalho foi sempre muito harmonioso e flexível, sempre que o Guilherme precisava de desenvolver um método no RMIServer ou na AdminConsole para testar a fase do Multicast em que estava fazia-o sem problema.

O Diogo ficou também encarregue do Failover e implementou o que precisou no Multicast também sem problemas.

Basicamente, ambos trabalhamos no RMIServer e AdminConsole, o Diogo tratou mais sozinho da parte do Failover e o Guilherme tratou mais sozinho da parte do MultiCast.

Consideramos que tivemos os dois a mesma quantidade de esforço exigido e gostamos bastante de fazer este projeto (tanto pelo projeto em si como pelo nosso grupo), apesar de não ter ficado como nós queríamos.

Testes

Foram efetuados vários e diversos testes, que se podem encontrar na tabela abaixo.

	Descrição do teste	Resultado
1	Em caso de avaria do server principal, o servidor de backup deixa de receber packets via UDP durante 5s e passa a ter o papel de servidor principal	
2	Ao voltar a ligar, o server principal passa a receber os packets via UDP, visto que o server de backup passou a estar definido como principal	
3	Ao repetir os processos 1 e 2, tudo acontece da mesma forma	
4	Nunca permite a introdução de strings onde é estritamente necessário o campo ser só de inteiros	
5	No caso de o servidor primário ter uma avaria, tornando o de backup principal e posteriormente se tornar a ligar, se o de backup tiver uma avaria tornando novamente o primário como principal, mas ligando-se de seguida novamente, como que voltando ao estado inicial, se a mesa de voto não tenha tido qualquer interação e o tentar ter agora, ligar ao principal atual	
6	Admin Console, em caso de falha do servidor principal avariar, conecta-se ao de backup quando este se torna principal e a admin console tiver alguma interação	
7	Registar pessoa no admin console	
8	Não permitir registar pessoas com o mesmo número de cc	
9	Registar um novo estudante	
10	Registar um novo funcionário	
11	Registar um novo docente	
12	Criar uma eleição	
13	Não permite adicionar a mesma mesa de voto duas vezes a uma eleição	
14	Não permite criar duas eleições com o mesmo nome	

15	Gerir lista de candidatos a uma eleição	
16	Não permite criar duas listas com o mesmo nome	
17	Gestão automática de terminais de voto por Multicast com uma mesa de voto	
18	Identificar eleitor na mesa de voto e desbloquear um terminal de voto	
19	Login de eleitor no terminal de voto	
20	Mostra apenas as eleições em que o eleitor pode votar, ou seja, verifica se a eleição está ativa, se o user pertence a um departamento que pode votar, se o user ainda não votou nessa eleição e se pertence ao grupo de pessoas que pode votar (Funcionários, Estudantes, Docentes)	
21	Após 120 segundos de sessão ativa, o terminal de voto é bloqueado automaticamente.	
22	Votar	
23	Editar propriedades de uma eleição e apenas nas que ainda não começaram	
24	Saber em que local votou cada eleitor	
25	Consola de administração mostra mesas de voto on/off e votantes (Terminais apenas bem para 1 mesa de voto, com 2 o nº é sempre igual nas duas)	
26	Consola de administração atualizada em tempo real e nas eleições	
27	Apenas é possível votar nas listas que contém o grupo de pessoas de que a pessoa faz parte(ex.funcionário votar onde funcionários podem votar)	
28	Ao fechar um voting terminal, a mesa de voto nunca mais reencaminhar um eleitor para lá	
29	Consultar resultados detalhados de todas as eleições passadas	
30	Aquando uma avaria no servidor principal, os eleitores não têm qualquer noção que tal ocorreu	
31	Se, depois do eleitor fazer login no terminal de voto e antes de votar, o servidor principal avariar, ao votar, o eleitor não tem visão de que uma avaria se deu e o voto é registado	

32	Gestão automática de terminais de voto por Multicast com duas mesa de voto (resolvia-se enviando o nome da mesa de voto nas mensagens de protocolo e adicionando a verificação no filterMessage)	
33	O failover é invisível para clientes/eleitores no terminal de voto	
34	Não se perde/duplica votos se a comunicação Multicast tiver falhas	
35	Crash de terminal de voto é recuperado	
36	Em caso de avaria longa os servidores Multicast ligam ao secundário	
37	Criar eleição com uma data de início anterior à atual, para depois ser iniciada quando for a data de início	
38	Criar eleição com uma data de início anterior à data atual e uma data de fim superior para terminar quando a data atual for igual à de fim.	
39	Criar eleição com a data de início e a data de fim anteriores à atual para verificar se é criada como terminada	
40	Corre em duas máquinas, métodos chamados remotamente pelo server que estão na admin não correm	
41	Não deixar criar duas mesas no mesmo departamento	
42	Nos detalhes da eleição apresentar a contagem de votos por mesa	
43	Nos detalhes da pessoa mostra as eleições em que votou, a mesa de voto em que votou e o momento em que votou	