Sistemas Distribuídos

Meta 1

2018/2019

Francisco Alves 2012136614

Madalena Santos 2016226726

João Silva 2016252535

Arquitetura do software detalhadamente descrita

1. MulticastServer

Este programa simula um servidor que comunica apenas através de UDP Multicast, excepto na transferência de ficheiros musicais. É o servidor central, que é replicado N vezes, e que armazena todos os dados da aplicação, suportando por essa razão todas as operações necessárias através de pedidos recebidos em datagramas multicast. Estes pedidos, por sua vez, estão descritos detalhadamente na secção Protocolo de Mensagens Multicast.

Para lidar com cada solicitação recebida, o programa MulticastServer gera uma nova *thread*, que trata de satisfazer o pedido e enviar uma resposta para o Servidor RMI.

Apenas no caso de pedido para transferência de um ficheiro musical (download ou upload), o servidor UDP se liga diretamente ao Cliente, através de uma conexão TCP. Para executar este pedido, é criada uma *thread* apenas para esse efeito, que termina quando acaba a transferência.

2. RMIServer

Existem dois servidores RMI que não armazenam os dados localmente e que disponibilizam, através da sua interface remota, um conjunto de métodos acessíveis às aplicações Cliente. Quando um método é invocado, o servidor RMI traduz esse pedido num datagrama UDP enviado por Multicast para os servidores Multicast, aguardando depois pela resposta enviada igualmente por Multicast.

Em caso de não existência de resposta por parte dos servidores Multicast (N servidores Multicast em baixo), a mensagem é enviada novamente, na esperança de que um dos servidores Multicast volte a estar funcional. Desta maneira, avarias de até 30 segundos nos N servidores Multicast são invisíveis para os Clientes.

A Interface *Services* implementa todos os métodos remotos do servidor RMI:

- ➔ ***int hello ()***
- ➔ ***void newClient (int port)***
- ➔ ***void ping ()***
- ➔ ***int register (String username, String password)*** Realizar registo de novo utilizador. Retorna o nível de privilégio do novo user ou 4, em caso de erro.
- ➔ ***int login (String username, String password)*** Realizar login de utilizador. Retorna o nível de privilégio do user, ou 4, em caso de erro.
- ➔ ***boolean logout (String username)*** Realizar logout de utilizador. Retorna *true.*
- ➔ ***String search (String username, String keyword, String object)*** Realizar pesquisa por um certo objeto. Retorna a lista de objetos, ou uma mensagem de erro.
- ➔ ***String details (String username, String object, String artist, String title)***
- ➔ ***String details (String username, String object, String title)***
- ➔ ***String showGroups (String username)***
- ➔ ***String joinGroup (String username, String group)***
- ➔ ***String newGroup (String username)***

- ➔ ***String changeInfo(String username, String groups, String type, String s1, String s2, String s3, String s4)***
- ➔ ***String changeInfo (String username, String groupIDs, String title, String artist, String musics, String year, String publisher, String genre, String description)***
- ➔ ***String addInfo (String username, String groups, String type, String title, String artist, String genre, String duration)***
- ➔ ***boolean review(String title, String artist, String user, String review, int rating)***
- ➔ ***boolean givePermissions (String perk, String username, String newUser, String groupID)***
- ➔ ***int uploadFile (String username, String musicTitle, String artistName)***
- ➔ ***String getMusics (String username)***
- ➔ ***boolean shareMusic (String username)*** Realiza a partilha de ficheiros musicais para os users de certos grupos. Retorna *true.*
- ➔ ***String showRequests (String username)***
- ➔ ***boolean manageRequests (String username, String newUser, String groupID, String toDo)***
- ➔ ***int downloadFile (String username, String musicTitle, String artistName)*** Realizar download de ficheiro a partir de um servidor Multicast. Retorna o porto em que o servidor Multicast está à escuta.

Todos estes métodos *throw* java.rmi.RemoteException. Para lidar com esta excepção, o cliente tenta ligar-se de novo ao RMI.

3. RMIClient

É o cliente RMI usado pelos utilizadores para aceder às funcionalidades do DropMusic. A interface é bastante simples e o Cliente limita-se a invocar os métodos remotos no servidor RMI.

Ao fazer transferências de ficheiros musicais, o Cliente liga-se diretamente a um servidor Multicast.

Para que seja possível o envio de notificações em tempo real para cada Cliente, foi necessária a implementação de uma Interface, *Clients*, que contém os seguintes métodos:

- ➔ ***void notification (String message)***
- ➔ ***String getUsername ()***

Distribuição de tarefas pelos elementos do grupo

A distribuição de tarefas foi a proposta no enunciado:

> *"Elemento 1 será responsável pelo Multicast Server e o elemento 2 pelo RMI Server e pelo RMI Client. Esta divisão assume que o protocolo multicast é especificado inicialmente e que as alterações serão daí em diante mínimas. Caso exista um terceiro elemento, este ficaria responsável pelo upload/download de ficheiros e pelas notificações instantâneas (isto é, reduz o trabalho aos outros dois elementos)"*

No entanto, acabámos por nos ajudar uns aos outros e portanto os três elementos conhecem bem a arquitetura do programa, as soluções para os problemas que foram surgindo nas várias fases do projeto e o código que foi feito.

Testes de software

Requisito	Descrição detalhada do teste	Pass/Fail ?
Consulta de detalhes sobre álbum	É possível ter acesso à pontuação média do álbum resultante de todas as críticas dadas	
	É possível consultar a descrição, a lista de músicas, as críticas escritas por outros users e a pontuação média	
	A lista de músicas de um álbum inclui informações como título, artista, duração	
Consulta de detalhes sobre artista	É possível consultar o nome, a descrição, a lista de concertos, a lista de álbuns e o género de um artista	
Download	Após receber acesso a um ficheiro musical (através da partilha de outro user), é possível ao user fazer download desse mesmo ficheiro para a sua máquina	
Escrita de uma crítica a um álbum	Qualquer user pode escrever uma crítica a um álbum	
Grupos	Todos os users registados pertencem ao grupo público (ID=1)	
	Qualquer user pode criar um novo grupo	
Introdução de novos álbuns, artistas ou músicas	Apenas o editor pode inserir novas informações	
	As informações que são inseridas estão apenas disponíveis para grupos escolhidos pelo editor que as inseriu	
Login	O login só é permitido se o user estiver registado e inserir a password com que se registou	
Notificações	Quando um elemento recebe privilégios, é notificado imediatamente se estiver online	
	Todas as notificações que não são entregues imediatamente porque o user está offline, são entregues quando este se volta a registar	
	Um user que editou a descrição de um álbum é notificado sempre que essa descrição é editada	
	Após receber acesso a um ficheiro musical (através da partilha de outro user), o user é notificado imediatamente	
Partilha de música para permitir o download	Qualquer user que tenha acesso a um ficheiro no servidor (seja porque foi ele a fazer o upload, seja porque essa música já foi partilhada com ele), pode escolher partilhar o ficheiro com certos grupos, de forma a permitir que os users nesse grupo tenham acesso ao download	
Pesquisa	Um user pode escolher pesquisar por músicas, álbuns ou artistas, inserindo uma keyword que verifica o match com qualquer atributo dos objetos que escolheu pesquisar	
Privilégios	Um owner de um grupo pode dar privilégios de owner/editor a membros desse grupo	
	O editor de um grupo pode dar privilégios de editor a outro user nesse grupo	
	Apenas o owner de um grupo pode recusar ou confirmar a pertença de users ao grupo após pedido	
	Tal como para o grupo público (ID=1), o owner é o utilizador que criou o grupo (foi o primeiro a entrar)	
Registo	O primeiro user a registar-se tem privilégios de owner no grupo público	
	Todos os users, excepto o primeiro a registar-se, entram no sistema sem privilégios de editor ou owner	
Tratamento de Excepções	A avaria de um servidor RMI não tem qualquer efeito nos clientes (FAILOVER)	
	Não se perdem ou duplicam músicas no caso de avaria do RMI	
	O programa funciona na totalidade havendo apenas um servidor Multicast	
	Avarias em todos os servidores Multicast ao mesmo tempo apenas são visíveis ao fim de 30 segundos	
	Todos os requests são processados por pelo menos um servidor Multicast	
	Pedidos de leitura são respondidos apenas por um servidor Multicast	
Upload de ficheiro para o servidor	O utilizador não consegue dar upload de um ficheiro associado a uma música que não tenha acesso/que não exista	
	Após acontecer o upload de um ficheiro para o servidor, conseguimos perceber que este teve sucesso pois é possível abrir o ficheiro na nova diretoria, e ouvir a música	
	É possível a qualquer user a transferência de um ficheiro musical, que fica associado a uma música já existente e que fica apenas acessível no perfil do user	
Tratamento de Excepções	Se, enquanto está em curso a transferência de um ficheiro por TCP, a ligação falhar, a operação não fica a meio	
Failover	O servidor secundário testa periodicamente o servidor primário através de uma chamada RMI. Em caso de x chamadas falhadas seguidas, o servidor secundário assume que o primário avariou e liga-se para o substituir	
Failover	Se o servidor RMI original recuperar, deverá tomar o papel de secundário e não de primário	

Protocolo de Comunicação Adotado:

Especificação de todos os comandos que são enviados entre o servidor RMI e os servidores Multicast.

Nos campos das mensagens enviadas não podem haver "|", ";" nem "\n" nas chaves ou valores.

; -> serve para separação dos conjuntos chave-valor

| -> serve para separação da chave do valor

\n -> fim do comando

User Registration

REQUEST: type | register ; username | new_username ; password | new password

ANSWER: type | status ; register | succeeded ou failed ; message | perks

User Login

REQUEST: type | login ; username | username ; password | password

SUCCESS ANSWER: type | login ; operation | failed

FAILURE ANSWER: type | status ; operation | succeeded ; perks | perk do user ; notifications | todas as notificações pendentes para esse user

1: user é owner de algum grupo

2: user é editor de algum grupo

3: user é um user normal

Para transmitir estas informações ao cliente, é retornada uma string com a seguinte estrutura:

"perk_number,notif1,notif2,..."

Cada notificação segue a seguinte estrutura:

"mensagem@timeStamp"

User Logout

REQUEST: type | logout ; username | username

ANSWER: type | status ; logout | succeeded ou failed

Check for groups

REQUEST: type | groups ; username | username

ANSWER: type | groups ; list | <group1,group2,...>

(isto para apresentar ao user todos os grupos aos quais ele pode juntar-se. É enviado o username para só devolver os grupos aos quais ele nao pertence)

Create Group

REQUEST: type | new_group ; username | username

ANSWER: type | new_group ; groupID | groupID ; operation | succeeded ou failed

Get group requests

REQUEST: type | get_requests ; username | username ; groupID | groupID

ANSWER: type | get_requests; operation | succeeded/failed ; (if succeeded) list | <user1, user2,...>

Manage group requests

REQUEST: type | manage_request ; username | username ; new_user | username ; groupID | groupID ; request | accept/decline

ANSWER: type | manage_request ; status | succeeded/failed ; operation | accept/decline

Pesquisar músicas

Search for musics, albums or artists

REQUEST: type | search ; username | username ; keyword | what you search for | object | type of object (musics, albums or artists)

Se pesquisou por artistas:

ANSWER: type | artist_list ; item_count | n ; item_list | artist1,artist2,...

Se pesquisou por musicas:

ANSWER: type | music_list ; item_count | n ; item_list | music1,music2,...

Se pesquisou por álbuns:

ANSWER: type | album_list ; item_count | n ; item_list | album1,album2,...

Gerir artistas, álbuns e músicas

Alterar informação de álbuns/artistas

REQUEST: type | change_info ; object | music ; username | username ; groups | groups ; title | title ; artist | artist ; genre | genre ; duration | duration

ANSWER: type | change_info ; status | success/fail

O user só pode alterar informação se tiver privilégios

Se o user tiver privilégios para alterar/adicionar a informação -> success

Se o user não tiver privilégios para alterar/adicionar a informação -> fail

Adicionar informação de álbuns/artistas

Esta função é usada quando um editor quer acrescentar um novo álbum, música ou artista às bases de dados, partilhando-as com grupos seleccionados

Parâmetros que são pedidos ao user para colocar no novo objeto:

Numa nova música: title, artist, genre, duration

Num novo álbum: title, artist, list of musics, year of publication, publisher, genre, description

Num novo artista: name, description, concerts, genre

Há duas funções diferentes no RMI para addInfo, visto que nas músicas e nos artistas precisamos de 4 parâmetros, mas no novo artista precisamos de 7. Por isto, há 3 tipos de requests

Add new music

REQUEST: type | add_music ; username | username que vai ficar associado à adição ; groups | lista de grupos com quem é partilhada esta informação ; title | title; artist | artist ; genre | genre ; duration | duration

Add new artist

REQUEST: type | add_artist ; username | username que vai ficar associado à adição ; groups | lista de grupos com quem é partilhada esta informação ; name | name; description | description ; concerts | lista de concertos próximos* ; genre | genre

a lista de concertos deve conter os concertos separados por vírgulas, e cada concerto deve ser "concertVenue-city-country-year-month-day-hour"

Add new album

REQUEST: type | add_album ; username | useername que vai ficar associado à adição ; groups | lista de grupos com quem é partilhada esta informação ; title | title ; artist | artist ; musiclist | lista de músicas do álbum ; year | ano de publicação ; publisher | editora ; genre | genre ; description | description

ANSWER: type | add ; operation | succeeded/failed ; error | message

Consultar detalhes sobre álbum e sobre artista

Consultar detalhes sobre um álbum

REQUEST: type | get_info ; object | album ; title | album_title ; artist | artist_name

ANSWER: type | get_info ; info | toda a informação numa String

Consultar detalhes sobre um artista

REQUEST: type | get_info ; object | artist ; title | artist_name

ANSWER: type | get_info ; info | toda a informação numa String

Consultar detalhes sobre uma musica

REQUEST: type | get_info ; object | music ; title | music_title

ANSWER: type | get_info ; info | toda a informação numa String

Escrever críticas a um álbum

REQUEST: type | review ; album_title | album title ; artist_name | artist_name ; username | username ; text | texto até 300 carateres ; rate | rate

ANSWER: type | review ; review | successful/failed ; error | descrição do erro

Dar privilégios de editor ou owner a um user

REQUEST: type | grant_perks ; perk | (editor / user) ; username | username proprio ; new_user | username do novo editor ; group | groupID

ANSWER: type | grant_perks ; status | succeeded/failed ; error | descrição do erro

(Pode ser success ou fail, dependendo se o user que estiver a dar privilégios seja ou não editor ou owner

Join group

REQUEST: type | join_group ; username | username ; group | group

ANSWER: type | join_group ; username | username ; group | group ; status | succeeded ; owners | owner_list

ou

ANSWER: type | join_group ; username | username ; group | group ; status | failed ; error | descrição do erro

Upload de ficheiros para um servidor

REQUEST: type | upload ; username | username ; music_title | music title ; artistName | artistName

ANSWER: type | upload ; port | port where the server is listening

ou

ANSWER: type | upload ; operation | failed

Download de ficheiros para um servidor

REQUEST: type | download ; username | username ; music_title | musicTitle ; artistName | artistName

ANSWER: type | download ; port | 5500

Partilha de um ficheiro musical de forma a permitir o seu download

Obter a lista de músicas que o user tem no servidor associadas ao seu nome

REQUEST: type | get_musics ; username | username ;

ANSWER: type | get_musics; item_count | nº de músicas ; music_list |
<musica1:nomeArtista,musica2:nomeArtista,musica2:nomeArtista,...>

Enviar a música e a lista de grupos que passarão a ter acesso ao ficheiro

REQUEST: type | share_music ; username | username ; musicTitle | musicTitle ; artistName | artistName ;
groupIDs | <ID1,ID2,ID3,...>

O servidor multicast retorna também a lista de users que têm acesso ao ficheiro pela primeira vez, para que o RMI possa enviar uma notificação para eles.

ANSWER: type | share_music ; item_count | count ; user_list | user1,user2,...

Guardar notificação porque o user não está loggado

REQUEST: type | notification ; username | username ; message | mensagem da notificação

ANSWER: type | status ; operation | succeeded

Aceder às notificações quando o usar logga

REQUEST: type | get_notifications ; username | username

ANSWER: type | get_notifications ; item_count | n ; notifications | String com as notificações todas

Funcionalidades não implementadas

No que toca aos requisitos funcionais e à semelhança de failovers, tudo foi implementado segundo a checklist fornecida pelo professor. No caso do tratamento de exceções, ficou por implementar as seguintes funcionalidades:

- Avarias temporárias (<30s) dos N servidores são invisíveis para clientes;

- Pedidos de leitura são respondidos apenas por um servidor multicast;

- Não se perde/duplica músicas se os servidores RMI falharem.

Em termos de extras, a pesquisa pode ser feita por qualquer parametro de qualquer objeto (artista, musica, album) e desenvolvemos detalhes de artista.