

Projeto SD

Sistemas Distribuidos



UNIVERSIDADE DE COIMBRA

Francisco Barão 2016238992

Pedro Ribeiro 2016218753

Index

1. Software Architecture	2
2. UDP	
2.1 Protocol.....	3
2.2 One Answer Only.....	3
3. RMI	
3.1 Interface.....	4
3.2 FailOver.....	4
3.3 Callback.....	5
4. Chores.....	6
5. Software Testing.....	7

Software Architecture

On our project, the client through the RMI Client is asked to give some input, which can be a vary of functions. After receiving an input, the program either tells the user the input is not correct or proceeds to call a remote method on the RMIServer sending the necessary arguments. Here, the program will act accordingly and translates the request into an UDP packet which will ultimately be sent to the multicast network to be received by the multicast servers. Before sending this packet, the RMIServer randomizes the multicast server which will respond and sends the packet to the network. Each Multicast Server will check if the packet contains its id and if it does, the multicast will decipher the packet into an SQL string which will then access the database using JDBC and retrieve some information. After that, multicast will rebuild the UDP packet and resend it to the RMI server which will build the objects and send them through RMI.

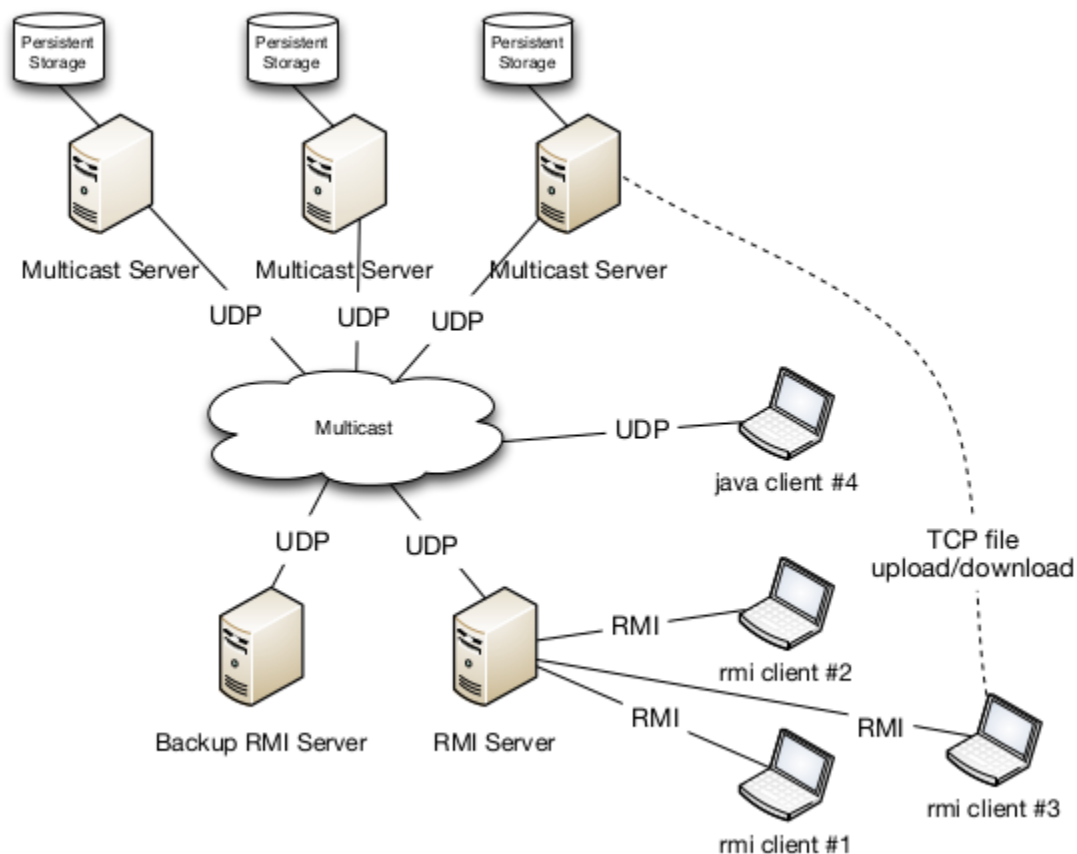


Fig1.Architecture

UDP Functionalities

Protocol

Function -> Create/Update/Delete/Search (Fig)

What -> Type (artist/album/..) (Fig)

Where -> clauses of SQL that specifies what the query will look for (Fig)

Set -> What will change on DB (Fig)

** -> Used to know when an object from a list of objects ends(Fig)

```
String t = "function|delete;what|notification;where|user_username='" + username + "'";
```

Fig2. UDP

```
t = "function|create;what|notification;id|null;text|"+note.text+";user_username|"+username;
```

Fig3. UDP

```
String t = "function|update;what|user;set|editor=1;where|username='"+username+"'";
```

Fig 4.. UDP

```
t = "function|search;what|artist;where|name='" + word + "'";
```

Fig5. UDP

```
= "id | 5 ; title | me ; releasedate | 2000-01-01 ; description | New Shit ; artist_id | 1 **id | 6 ; title | me ; releasedate | 1970-01-10 ; descrip
```

Fig6. UDP

```
String output = "INSERT INTO " + str.get("what") + " " + column_names + " VALUES " + values + " ;";
```

Fig7. UDP

Multicast one Answer Only

When creating a multicast server, it randomly chooses one number from 1 to 200 to use as his "id". Also, when the RMI server is starting, it asks for every connected multicast's id. So when sending a message to the multicast server, the RMI server chooses randomly on of the possible ids and adds that information to the message, making only one server respond.

RMI Functionalities

Main Menu -> Interface

/help -> Show Menu Interface

/register -> Create new User

/add type -> Allows an editor to add an artist/album/music

/search -> Allows any user to search for artist/album/music

 /review -> Allows user to write/rewrite a review

 /edit -> Allows Editor to change details about entity

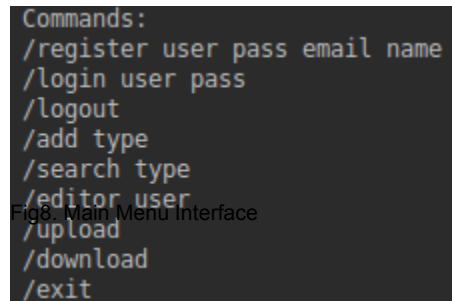
 /exit -> Exit option of this menu

/delete type -> Allows an Editor to delete an artist/album/music

/editor type -> Allows Editor to turn another non editor user into an editor

/upload -> Allows user to upload a music

/download -> Allows user to download a music from his uploaded songs or from other people shared songs



```
Commands:
/register user pass email name
/login user pass
/logout
/add type
/search type
/editor user
/upload
/download
/exit
```

Fig8. Main Menu Interface

RMI FailOver

Steps of RMIServer:

->Creates Registry

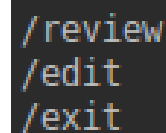
->Does lookup of server

->If no exception:

Server is already up so every second the server while works as backup will call a remote method (is.Alive) and will keep doing so while no exception is found. When one is found the program will retry 4 more times and after those it will start another server loading the online users info from the Object files making sure it is invisible to the users.

->Exception Fund:

Means no main server is up so it starts a Server.



```
/review
/edit
/exit
```

Fig9. Other Menu Interface

RMI Callback

As for callback, we create a client Interface which only method is liveNotification. When a user logs in, he then calls the method subscribe which sends the client interface and the server stores the user on the online users list(each user has their own client interface stored on the class). The user will act as a server.

Chores distribution

Function	People	Francisco Barão	Pedro Ribeiro
Register		Done	--
Login/Logout		Done	--
Add artist/album/music		Done	--
Search artist/album/music		Done	Helped
Edit artist/album/music		Done	Helped
Editor privileges		Done	--
Notifications on OnlineUsers		Helped	Done
Upload Files		--	Done
Download/Share		--	Done
RMI Server/Client Structure		Done	Helped
RMI Failover/online User management		Done	--
Multicast Structure		--	Done
UDP Protocol		Done	Done
JDBC/Database		Done	Helped
Multicast one Answer		--	Done
Several Computer		Helped	Done
Notification Offline		Done	--

Software Testing

Function	Pass/Fail
Register user	Pass
Login/Logout	Pass
Add artist	Pass
Add album	Pass
Add music	Pass
Search artist	Pass
Search album	Pass
Search music	Pass
Edit artist	Pass
Edit album	Pass
Edit music	Pass
Write Review	Pass
ReWrite Review	Pass
Delete album	Pass
Delete artist	Pass
Delete music	Pass
Editor privileges	Pass
Notifications on Online Users	Fail on other PCs
Notifications on Offline User	Pass
Upload Files	Pass
Download/Share	Pass
RMI Failover structure	Pass
RMI Failover online management	Pass
Multicast only one answer	Pass