UNIVERSIDADE DE COIMBRA

iVotas: Voto Eletrónico na UC
Sistemas Distribuídos 2017/2018 Meta 1

João Rodrigues Martins
João Filipe Mendes Castilho

1. Arquitetura

A arquitetura do sistema é composta pelos seguintes elementos:

- Uma base de dados, neste caso um ficheiro de objetos, partilhada por ambos os Rmis por shared folder e lido quando estes são iniciados.

- Dois servidores Rmis, o servidor primário e o servidor backup, sendo chamados pelo mesmo ficheiro, RMIServer.java, com parâmetros de entrada diferentes. Estes comunicam entre sim, enviando pacotes para verificar o estado do seu homónimo, utilizando uma conexão UDP implementada no ficheiro UDPConnection.

Ao servidor RMI , poderão ser ligados dois tipos de programas:

- Consola de Administração, chamada pela classe AdminConsole.java

- Servidores TCP, chamados pela classe TCPServer.java que funcionam como mesa de voto, para os eleitores se identificarem. Neste servidor TCP, a main thread fica à escuta do input da consola para quando um eleitor se identifica, desbloquendo um dos terminais de voto. É também criada uma thread do tipo ThreadTCP, que possui o Server Socket e fica à espera que se conectem clientes TCP, neste caso, terminais de voto. Quando um terminal de voto se liga ao server socket, é criada uma thread do tipo ThreadConnection que trata da comunicação entre o terminal e o TCPThread.

- Terminais de voto têm como objetivo realizar a função de login e de voto.

2. Descrição do servidor TCP

O servidor TCP, desenvolvido na classe TCPServer, faz lookup ao objeto remoto do rmi para poder invocar métodos nele presentes. Entretanto, é criada uma thread do tipo ThreadTCP para a qual são passados como pârametros, o objeto da interface RMI, o número da porta onde se irão ligar os sockets, entre outros... Cada vez que um terminal é ligado à porta onde está o Server Socket , é criada uma thread do tipo ThreadConnection responsável por esse terminal. A main thread do TCPServer possui um BufferedReader à espera que o utilizador se identifique na consola, desbloqueando através da invocação de um método do objeto ThreadTCP, um dos terminais ativos e livres guardados numa variável na ThreadTCP.

O servidor TCP implementa a interface ConnectionClient, com finalidades de callback por parte do RMI server.

3. Descrição do servidor RMI

O servidor RMI implementa a interface ConnectionRMI, para o TCP poder invocar os seus métodos.

Para fins de Callback, quando um TCPServer (mesa de voto) faz o lookup e se conecta ao RMI, é imediatamente chamda a função Subscribe, onde é passada para o RMI a referência deste TCPServer, que ficará guardada num ArrayList deste tipo juntamente com todas as mesas de voto online.

Para efeitos de Fail Over, o RMI primário e secundário trocam entre sim hearthbeats via UDP. Quando o servidor primário não responder a 5 pacotes, o backup assume que o primário foi abaixo e faz a substituição. Quando o primário volta, fica como backup. Quando isto acontece, tanto as consolas de administração como o TCPServer mudam de RMI quando ao invocar um método este retorne uma exceção do tipo Remote Exception.

Os métodos aqui presentes são os seguintes:

- add_Department, para adicionar um departamento.
- add_table, para adicionar uma mesa de voto a uma determinada eleição.
- remove_table, para remover uma mesa de voto.
- create_election, para a criação de uma eleição.
- detailVote, para mostrar informações relativas ao local e momento onde o utilizador votou.
- edit_department, para editar informação de um departamento.
- edit_election, para editar informação de uma eleição.
- edit_user, para editar informação de um utilizador.

- getStatsElection, que dá a informção estatística de uma dada eleição.
- identify_user, chamada numa mesa de voto, para um utilizador desbloquear um terminal e votar.
- isUp, para verificar se o rmi está ativo, se não estiver apanha com a remoteException
- item_count, para fazer listas consoante o protocolo.
- subscribe, serve para armazenar as mesas de voto que se ligaram ao RMI, para efeitos de callback na função list_tables.
- list_tables, serve para o administrador verificar as mesas e terminais ativos. Esta função irá percorrer todas os objetos do tipo table existentes em cada eleição e chamar uma função com o seu respetivo callback. A função de callback retorna ainda informações relativas dos terminais conectados à respetiva mesa de voto.
- Outras funções iniciadas por list, são identicas entre si, apenas listam diferentes objetos (list_users,list_elections, ...).
- login, serve para verificar a existencia do utilizador na base de dados.
- loginTable, serve para logar quando uma mesa de voto é ligada (username e password da mesa são definidos aquando do registo da mesa na função add_table)
- register, para o registo de um utilizador.
- Funções iniciadas por remove servem para remover objetos da base de dados.
- votersOnElection, para listar os utilizadores que votam numa dada eleição.
- submitVote, a função para que permite um utilizador votar.
- showInfoElection, para listar informações de uma dada eleição.
- searchPossibleElections, para listar as eleições onde um dado utilizador poderá votar.

4. Distribuição de tarefas

Segue em anexo, no ficheiro "sd_checklist_meta.xlxs", informação relativa à divisão de tarefas. Optámos por dividir equitativamente as tarefas.

5. Descrição dos testes feitos à plataforma

Segue em anexo, no ficheiro "Testes.txt", informação relativa aos testes efetuados.