Sistemas Distribuídos

Projeto de Programação - Napster com RMI

Nome: Grazielle Reis Mascaro
Link do vídeo: https://youtu.be/EKXLCFRmt1U

OBSERVAÇÃO SOBRE O SCREENCAST:
O sinal de internet estava muito instável quando o screencast foi gravado e, por isso, não foi possível baixar um arquivo .mp4 de 1GB para usar na demonstração.
O arquivo .mp4 usado tem um tamanho um pouco menor, de 836KB (0,8 GB).
Entretanto, foram feitos testes com outros arquivos (que não são .mp4) muito maiores, e todas as funcionalidades parecem estar funcionando bem.

1. Estrutura do código

O projeto é composto:

- Pelo pacote *NapsterRMI.model*, onde foi implementado o RMI, com a interface ServicoNapster.java e as classes ServicoNapsterImpl.java e Search.java
- Pelo pacote *NapsterRMI.peer*, com a classe Peer.java
- Pelo pacote *server*, com a classe Servidor.java

2. Funcionalidades do Servidor

2.1. Servidor.class e ServicoNapster.java

Servidor.class: O servidor possui um ServerSocket na porta 900 [linha 17] e tudo o que é necessário para o RMI [linhas 21-24].
ServicoNapster.java: Interface que define os objetos remotos: reqJoin, reqSearch e reqUpdate.

2.2. ServicoNapsterImpl.java

Aqui, está toda a implementação relacionada ao RMI:

2.2.1. Informações dos arquivos de cada Peer [linhas 17-20]

Cada arquivo tem um ID, um inteiro chamado IDArquivo. Os detalhes sobre ele estão armazenados em três Hashtable:

- nomeArquivo, contendo ID e nome;
- IPPeer, contendo ID e IP do Peer;
- portaPeer, contendo ID e porta do Peer.

Assim, a partir de seu ID, é possível recuperar as outras informações rapidamente.

2.2.2. reqJoin [linhas 36-55]
Recebe IP, porta e um array de String com os nomes dos arquivos. As informações são armazenadas nas três Hashtable. Retorna uma String JOIN_OK.

2.2.3. reqSeach [linhas 61-104]
Recebe IP, porta e uma String que representa o nome do arquivo a ser buscado. Retorna um objeto Seach, que contém um array de IP e um array de portas que representam os Peers que possuem o arquivo desejado.

O reqSearch confere se a Hashtable nomeArquivo contém a String a ser buscada [linha 66]. Se não contém, retorna o objeto Seach com array nulo.

Caso o arquivo exista, cria uma HashSet que contém todos os IDs dos arquivos com o nome buscado [linhas 74-80]. O HashSet é transformado em um array [linhas 85-90]. Estes trechos são baseados nos códigos disponíveis aqui e aqui.

Com o comando .get, é possível relacionar os IDs obtidos com os IPs e portas dos Peers armazenadas nas outras Hashtable [linhas 92-98], retornando um objeto Search com estas informações.

2.2.4. reqUpdate [linhas 108-117]
Recebe o nome do arquivo a ser adicionado junto ao IP e porta do Peer. Depois de colocar estas informações nas Hashtable, retorna a String UPDATE_OK.

3. Funcionalidades do Peer

3.1. Um esclarecimento sobre o menu
O peer pode requisitar um JOIN, requisitar um SEARCH ou requisitar DOWNLOAD/UPDATE. O SEARCH só pode ser requisitado após ser feito o JOIN, e o DOWNLOAD e UPDATE apenas depois do SEARCH. **Por isso, foi decidido não usar nenhum tipo de menu interativo.**

O usuário, ao compilar o arquivo, só pode digitar o nome da pasta que quer compartilhar - então, é enviada uma requisição JOIN automaticamente. Depois disso, tudo o que ele digitar será enviado como uma requisição SEARCH - caso haja resultados, ele pode decidir pelo DOWNLOAD. O UPDATE é feito imediatamente depois que o DOWNLOAD é concluído.

Uma demonstração pode ser vista no vídeo.

3.2 Comunicação TCP
O Peer cria um socket (TCP) e se conecta com o servidor da porta 9000 [linha 109]. Assim, o Sistema Operacional aloca para o peer uma porta qualquer que esteja disponível. O número da porta usada é armazenado [linha 156].

Numa comunicação Peer-to-Peer, um Peer deve atuar tanto como cliente quanto como servidor. Como a comunicação com o servidor é feita por RMI e não por TCP, a conexão TCP com o servidor é fechada [linha 161], servindo apenas para que o SO encontre uma porta disponível. Com esta mesma porta, ele cria um ServerSocket. Assim, a thread pode ser executada, aguardando que outro Peer faça contato com o ServerSocket.

3.3. JOIN [linhas 115-177]
O usuário deve digitar o endereço da pasta que deseja compartilhar. O programa aguarda até que receba um caminho válido [linhas 132-145].

Ao receber um caminho válido, armazena o nome de todos os arquivos da pasta em uma array de String [linhas 147-152], que é enviada (junto com o IP e a porta do ServerSocket do Peer) em uma reqJoin. Aguarda JOIN_OK.

Em seguida, inicia uma Thread.

3.4. THREAD E DOWNLOAD [linhas 36-99]
A thread é usada apenas para comunicação TCP. Sua função é apenas aguardar até que outro Peer faça uma requisição de download, agindo de modo semelhante a um servidor. Possui um ServerSocket.

Quando um Peer se conecta à Thread, a Thread aguarda receber o nome do arquivo desejado. Com o nome do arquivo, cria o caminho para ele baseado no endereço da pasta digitado durante o JOIN [linha 62]. A partir daí, envia o arquivo [linhas 64-80]. Como o programa deve ser capaz de enviar arquivos gigantes, o arquivo não é enviado inteiro de uma vez - é enviado em pequenas quantidades de bytes por vez [linhas 76-80].

3.5. SEARCH, DOWNLOAD E UPDATE [linhas 185-255]
O usuário digita o nome do arquivo a ser buscado e o nome é enviado em uma reqSearch. Aguarda o objeto Search. Em seguida:

- caso o objeto Search seja nulo, o usuário pode digitar outro nome para ser buscado.
- caso o objeto Search seja não nulo, lista todos os IPs e portas dos Peers que possuem o arquivo desejado. Imprime no console:
  " Requisitar download? [S / N] " [linha 206]
  - caso o usuário responda "S", o programa cria um Socket usando as informações contidas no Search para se conectar ao ServerSocket do último Peer da lista [linha 211]. Cria tudo o que for necessário para a comunicação TCP, como OutputStream e DataOutputStream [linhas 213-125], envia por TCP o nome do arquivo desejado e se prepara para receber o arquivo. Após o download, é enviado um reqUpdate ao servidor por RMI para adicionar as informações do arquivo baixado. Aguarda UPDATE_OK. O socket é fechado.
  - caso o usuário responda com qualquer outra coisa, a requisição de download não é feita, e o usuário pode digitar outro nome para ser buscado.