ACH147 - Desenvolvimento de Sistemas de Informação Distribuídos

Relatório Exercício Programa: parte 1

João Paulo Santos Torres
Rodrigo Dorneles

1. Introdução

O objetivo deste Exercício Programa é criar um sistema de compartilhamento de arquivos peer-to-peer simplificado, e nesta parte foi desenvolvida a funcionalidade de gerenciamento de peers conhecidos.

O EP foi desenvolvido em Java e a principal dificuldade encontrada foi utilizar multi-threads e sockets devido a falta de prática com essas ferramentas, para lidar com essas dificuldades o gemini foi consultado.

O arquivo “instrucoes.txt” explica como compilar e executar o arquivo.

2. Orientação a Objetos

O paradigma utilizado foi programação orientada a objetos, devido a linguagem escolhida ser java, porque ele facilita a organização do código e é a única maneira que conheço de trabalhar com multi-threads. As classes escolhidas foram: eachare, comandos, mensagens, relogio e vizinhos.

- A classe eachare é a classe principal que inicializa o programa e que cria as threads;
- A classe comandos agrupa os comandos do menu, com um método para cada opção;
- A classe mensagens é a responsável por mandar as mensagens para os outros peers, o conteúdo das mensagens é definido pela classe comandos ou pelo método handleConnection da classe eachare ao responder uma mensagem GET_PEERS;
- A classe relogio é responsável pelo funcionamento do relógio local e garante que uma uma mensagem apareça para o usuário sempre que o relógio é atualizado;
- A classe vizinhos é utilizada para agrupar os dados (endereço e estado) dos vizinhos.

A classe eachare é a única a manipular threads e possui três métodos:

- O método main inicializa o programa, lendo os argumentos e abrindo os arquivos, caso tudo ocorra bem, ele cria a thread com o método menu e fica em loop esperando conexões com o serverSocket, quando uma conexão ocorre ele inicia uma nova thread com o método handleConnection;
- O método menu apresenta os comandos para o usuário e espera uma resposta, para atender a cada comando ele utiliza um método da classe comandos;
- o método handleConnection recebe as mensagens do outro peers e executa as ações necessárias, como mudar o estado de um peer na lista de vizinhos ou responder a mensagem GET_PEERS com a mensagem PEER_LIST.

3. Outros aspectos da implementação

A estrutura de dados utilizada foi o ArrayList presente no Java, devido a facilidade de uso.

Para enviar e receber dados são usadas operações bloqueantes, porque elas são mais fáceis de gerenciar e com o uso de multi-thread cada operação bloqueia apenas uma thread permitindo o funcionamento das outras threads do sistema.

4. Testes

Os testes feitos foram concentrados em cada comando implementado, a cada comando desenvolvido era feito um teste para verificar se ele estava funcionando. Para isso foi usado a pasta "compartilhar" e o arquivo "vizinhos.txt" presentes no arquivo zip enviado, e todos os testes foram realizados em um único computador utilizando o endereço 127.0.0.1:<porta>.

Foi observado um erro quando a porta usada para executar o programa estava presente no arquivo "vizinhos.txt", esse erro aparece ao fechar o programa com o comando 9, ele ocorre porque o programa tenta acessar um socket após ele ter sido fechado pelo comando.

O programa não valida a entrada, portanto se o usuário digitar uma entrada inválida o programa dará erro, além disso, se o programa não for encerrado com o comando 9, é possível que o serverSocket fique aberto e não será possível executar o programa em seguida com a mesma porta.