ACH147 - Desenvolvimento de Sistemas de Informação Distribuídos

Relatório Exercício Programa: parte 2

João Paulo Santos Torres
Rodrigo Dorneles

1. Implementações dos novos comandos

A escolha de deixar função do relógio implementada em uma classe separada facilitou a primeira parte do ep de alterar o funcionamento do relógio local e o fato de existir uma classe separada para vizinho facilitou o acréscimo da informação de seu relógio deixando mais simples a alteração no gerenciamento de peers conhecidos, dessa forma os itens 2 e 3 das especificações foram codificados sem dificuldade.

Mas a escolha de utilizar multi-thread gerou um problema na implementação do comando busca, que será descrita a seguir, entretanto quando a solução foi encontrada, ela era simples. Na implementação do comando busca, inicialmente a mensagem LS era enviada e se voltava para o loop do método menu que apresentava as opções, isso gerou um problema, pois havia duas threads esperando um comando do usuário, e com isso era necessário digitar o comando mais de uma vez, após receber a mensagem LS_LIST. E acabava ocorrendo um comando que não era desejado (do menu de opções). Ao reler as especificações e analisar o código, percebeu-se que após a mensagem LS ser enviada, não era para voltar para o menu, mas sim esperar a resposta dos vizinhos, mostrar os arquivos recebidos em LS_LIST, se não for cancelado enviar a mensagem DL só voltar após receber a mensagem FILE e completar o download, dessa forma, foi inserido no comando 4 um trecho de código que esperava isso tudo ocorrer e após isso voltar para o menu.

Foi adicionada a classe arquivos_vizinhos que agrupa as informações das mensagens usadas no comando busca e a classe comandos recebeu três atributos: um "int ls" usado para contar quantas mensagens do tipo LS foram mandadas e depois usados para contar quantas mensagens do tipo LI_LIST estão sendo esperadas; uma lista de arquivos_vizinhos para guardar as mensagens recebidas em LI_LIST para ser usada posteriormente; e o “boolean bloqueia” usado no comando 4 para saber até quando esperar para voltar para o menu como falado acima.

O método setRelogio foi modificado para se adequar ao novo funcionamento e o método handleConnection foi modificado para receber as novas mensagens e executar as ações necessárias.

Foi acrescentado mais um método a classe comandos o executaLS_LIST para executar os comandos necessários ao receber a mensagem LS_LIST.

Não foi necessário modificar as funções que estavam implementadas antes, mas foi necessário acrescentar alguns argumentos a mais em handleConnection para que ele tivesse as informações necessárias para responder as novas mensagens.

2. Teste

Após cada funcionalidade implementada: relógio, comando, mensagem enviada era feito um teste com dois peers para verificar se estava funcionando como o desejado. Quando o código estava finalizado, foi feito um teste em que um dos peers tinha 3 arquivos em sua pasta compartilhada e o outro não tinha nenhum, após um peer mandar uma mensagem de HELLO para o outro para que os dois atualizassem o estado um do outro para ONLINE, o comando busca foi usado no peer que não tinha arquivos na pasta compartilhada para receber os 3 arquivos do outro e isso ocorreu corretamente.

3. Observações

O gemini do google foi utilizado para descobrir como fazer algumas funções como codificar e decodificar o arquivo para a base 64 e para ajudar a entender e corrigir os erros. No primeiro caso, ao se fazer algumas perguntas para especificar o que se queria, se chegou a respostas úteis que ajudaram na implementação. No segundo caso, o gemini conseguiu ajudar em alguns erros simples, como descobrir que esqueci de inicializar a lista de arquivos_vizinhos da classe comandos. Porém como mencionado acima, para resolver o problema de concorrência entre threads foi necessário uma pausa para revisão das especificações e do código para se chegar a solução.

O código usa a classe Base64 que está presente na versão do Java 8 ou posterior, ou seja, o código precisa ser compilado e rodado a partir dessa versão.