UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

SISTEMAS DISTRIBUÍDOS

DOCENTE: RENAN CERQUEIRA AFONSO ALVES

RELATÓRIO: EXERCÍCIO PROGRAMA

Parte 2

Júlia Du Bois Araújo Silva N° USP: 14584360

SÃO PAULO 2024

SUMÁRIO

1. Execução do programa

A execução do programa segue a mesma da parte 1 do exercício, descrita no relatório 1. Para acesso fácil, as instruções para a execução estão replicadas nesta seção.

Primeiramente, é necessária uma instalação da linguagem de programação Python na máquina utilizada. As bibliotecas utilizadas no programa são os, sys, socket e threading. Para executar um peer, utilize o seguinte comando:

python localDoRepositorio/eachare.py <endereco>:<porta> <vizinhos.txt> <diretorio_compartilhado>

Se na sua máquina há outra versão do Python instalado - como, por exemplo, Python3 - , modifique o comando anterior para, em vez de “python”, a palavra-chave da sua versão.

2. Refatoração e escolhas feitas na implementação da parte 1

As principais escolhas realizadas na parte 1 foram em relação à linguagem (Python), ao paradigma de programação (orientação a objetos), às classes criadas e ao uso de threads. Nas próximas subseções, cada uma dessas categorias é abordada e, em seguida, são apresentadas as refatorações e expansões feitas. Algumas dessas refatorações foram mencionadas como propostas de melhorias no relatório 1. Por último, é abordado o *feedback* enviado após a entrega da parte 1 do exercício e como foi implementado no código.

1.1 Linguagem

A linguagem escolhida na parte 1 do exercício facilitou a realização da parte 2 por conta das características pelas quais ela foi escolhida: suas bibliotecas. Como adição na parte 2, também foi utilizada a biblioteca base64 para a codificação e decodificação dos arquivos.

1.2 Orientação a objetos

Agora mais familiar, a escolha desse paradigma se mostrou vantajosa nessa expansão por conta do alto grau de modularização do código, que facilitou sua

expansão. Para essa entrega, foi criada uma nova classe file, que representa um arquivo. Além dessa, foi criada a classe messageParser, mencionada como possibilidade no relatório 1, para organizar o parseamento de mensagens. Não foi implementada a classe para a lógica de envio e recebimento de mensagens descrita no relatório 1 pois não foi encontrada a necessidade dessa modularização para essa entrega, embora ainda seja uma boa ideia para o futuro.

1.2.1 file

Essa classe representa um arquivo. Ela possui os atributos filename, size, peerIP, e peerPort (no qual peerIP e peerPort representam o endereço IP e porta do peer que possui o arquivo).

1.2.2 messageParser

Essa classe é responsável por analisar e extrair informações de uma mensagem recebida, de acordo com uma gramática pré-definida. Os atributos da classe são:

- msg: (não usado) parece ser destinado a armazenar a mensagem original, mas não está em uso atualmente;
- senderIP: o endereço IP de quem enviou a mensagem;
- senderPort: a porta de quem enviou a mensagem;
- senderClock: o relógio lógico do remetente;
- messageType: o tipo da mensagem.

Esse parser é utilizado na função handleRemoteCommand da classe commandHandler.

1.3 Threads

As threads usadas no programa seguem as mesmas da entrega 1, pois não houve a necessidade de criar novas threads. Agora, estão melhor manejadas, pois foram criadas funções para garantir o fechamento delas ao receber o comando 9 e ao finalizar uma comunicação. Essas funções são openListening e closeListening para a thread receiveConnections; e startReceiving e stopReceiving para a thread de receiveCommands.

2. Testes

Foram realizados testes locais manuais, utilizando apenas uma máquina. Além disso, para simular o uso de múltiplas imagens, foram utilizados containers do Docker, simulando diferentes tempos de lag nos peers (tanto naqueles enviando as mensagens quanto naqueles recebendo). Também foram realizados testes com envios de arquivos de diferentes tamanhos e formatos. Não foram encontrados problemas inesperados durante a execução desses testes.

3. Dificuldades, desafios e *feedback* da entrega 1

Essa seção do relatório está escrita em primeira pessoa por ser de caráter pessoal.

Não encontrei grandes dificuldades nessa entrega. Quanto ao *feedback* da entrega 1:

- Crash ao tentar enviar mensagem HELLO para peer offline
    - Estava aguardando o erro errado. Isso agora está corrigido.
- Prints um pouco fora do padrão: ao receber mensagem sempre imprime "resposta recebida"
    - Me confundi com os exemplos do EP, creio que esse erro agora está corrigido.
- Programa trava ao [9] Sair, sem enviar nenhum BYE
    - As threads não estavam sendo corretamente encerradas e estava usando a função errada para encerrar o programa. Isso agora está corrigido.
- Às vezes ocorre um erro 'str object cannot be interpreted as integer' ao utilizar a opção Obter Peers e às vezes ocorre um erro 'list index out of range' ao utilizar a opção Obter Peers
    - Não consegui replicar estes erros, então seguem sem correção. Vou tentar novamente na entrega 3.
- Recv(1024)
    - Foi implementada a função recvUntilNewline para evitar o limite de bytes em na função recv. Essa função recebe continuamente de 1024 em 1024 bytes até encontrar o caractere “\n”.