UNIVERSIDADE DE SÃO PAULO

Sistemas de Informação - EACH

DAVI FERNANDES MENESES DA SILVA, N. USP: 14760241

GABRIEL GONÇALVES DE SOUZA RIBEIRO, N. USP: 14691679

Desenvolvimento de Sistemas de Informação Distribuídos

Relatório da parte I do trabalho semestral

SÃO PAULO, SP

2025

1. Introdução

Este relatório descreve o desenvolvimento da Parte 1 do Exercício Programa, que consiste na criação de um sistema de compartilhamento de arquivos do tipo peer-to-peer (P2P) chamado EACHare. Esta primeira parte do trabalho tem como foco o gerenciamento dos peers conhecidos na rede, permitindo a descoberta de novos peers e a troca de mensagens de controle entre eles.

2. Paradigma de Programação Escolhido

Utilizamos o paradigma de programação imperativo. A escolha da linguagem Python se deu pela sua facilidade de uso, simplicidade na manipulação de sockets e boa documentação para implementações de rede.

3. Divisão em Threads

A aplicação possui uma thread principal, responsável pela interface de menu com o usuário, e uma thread secundária dedicada para escutar conexões TCP (servidor). Além disso, para cada conexão recebida, é criada uma nova thread para tratar a mensagem, garantindo que o servidor permaneça responsivo para novas conexões.

4. Operações Bloqueantes vs. Não Bloqueantes

Optamos por utilizar operações bloqueantes com timeouts curtos para envio de mensagens via socket TCP. Isso simplifica o código e facilita o controle de fluxo. O recebimento de conexões é feito em uma thread separada, o que evita o travamento da aplicação.

5. Estruturas de Dados Utilizadas

- Dicionário `peers`: utilizado para mapear cada peer conhecido para seu status atual (ONLINE ou OFFLINE).

- Variável global `clock`: representa o relógio lógico do peer (versão simplificada do Relógio de Lamport).

6. Classes Utilizadas

Optamos por não utilizar classes nesta primeira parte para manter a estrutura simples e focar na funcionalidade do protocolo. Todo o controle foi implementado com funções e variáveis globais. Em etapas futuras, a refatoração para uma abordagem orientada a objetos é recomendada.

7. Maiores Dificuldades Enfrentadas

- Garantir que o peer que responde a um `GET_PEERS` atualize corretamente o status do remetente.

- Sincronizar o uso do relógio lógico de forma correta antes de enviar e ao receber mensagens.

- Gerenciar o paralelismo de maneira segura com o uso de threads simultâneas para conexões.

8. Testes Realizados

- Testes com 2, 3 e 4 peers simulando uma rede em localhost com diferentes portas.

- Testes de envio e recebimento das mensagens `HELLO`, `GET_PEERS`, `PEER_LIST` e `BYE`.

- Testes com peers offline para verificar se os status eram atualizados corretamente.

- Validação da leitura correta de diretório e arquivo de peers.

9. Execução do Programa

Para executar o programa:

python main.py <endereco:porta> <vizinhos.txt> <diretorio_compartilhado>

Exemplo:

python main.py 127.0.0.1:9001 vizinhos.txt diretorio

Onde:

- `vizinhos.txt` contém uma lista de peers conhecidos (um por linha)

- `compartilhados/` é o diretório com arquivos que serão listados