Centro Universitário da FEI

Rede Social com Publicações e Troca de Mensagens

(Projeto de Sistemas Distribuídos)

Disciplina:

CC7261 – Sistemas Distribuídos (Prof. Leonardo Anjoletto Ferreira)

Alunos:

Ana Beatriz Tavares Malheiro (RA 24.122.019-3)

Bruno Andwele Alves Antunes (RA 24.122.030-0)

São Bernardo do Campo

1° Semestre / 2025

Rede Social com Publicações e Troca de Mensagens

1. Introdução

Este relatório apresenta o desenvolvimento e funcionamento de um Sistema Distribuído que implementa a replicação de postagens e a sincronização de relógios entre servidores, utilizando os algoritmos de Bullying para eleição de coordenador e Berkeley para ajuste de relógios. O sistema simula uma rede social distribuída, garantindo a alta disponibilidade, a tolerância a falhas e a consistência das operações.

2. Requisitos do Sistema

- Python 3.7
- Biblioteca ZeroMQ instalada

3. Arquivos do Projeto

Usuario.py, Usuario.c, Usuario.java:

- Este arquivo fornece uma interface interativa para o usuário final, permitindo que ele se cadastre, publique textos, siga outros usuários, envie e visualize mensagens privadas, consulte a timeline e veja notificações.
- Ele também se comunica com o servidor via ZeroMQ utilizando mensagens JSON, e pode registrar logs das operações realizadas.

Servidor.py:

- O Servidor.py representa o servidor principal da aplicação de rede social. Ele é responsável por processar as requisições dos clientes, aplicar as regras de negócio (como postagens, seguidores, envio de mensagens privadas) e se comunicar com o banco de dados para armazenar e recuperar informações.

- O servidor também gerencia as notificações, controlando tópicos e eventos, e pode realizar sincronização ou coordenação com outros servidores, dependendo da arquitetura escolhida.

Proxy.py:

- Atua como um intermediário entre os servidores.
- Executa o escalonamento de atividades, direcionando mensagens e coordenando a distribuição de tarefas entre os servidores para balancear a carga e garantir disponibilidade.

ReturnCodes.py:

- Arquivo que define códigos de retorno padronizados, facilitando a comunicação entre processos e o tratamento de erros ou sucessos em operações diversas.

BancoDeDados.py :

- Este arquivo implementa o servidor de banco de dados do sistema. Ele é responsável por armazenar todas as informações persistentes, incluindo usuários, seguidores, postagens, mensagens privadas e notificações.
- Responde às requisições dos servidores, realizando operações de leitura e escrita, e oferece funcionalidades adicionais, como replicação ou consistência, conforme o projeto

Logs (Monitoramento de atividades) :

Cada servidor presente na simulação possui um arquivo de log txt nomeado de “servido{número do servidor}”, onde são armazenado informações como data, tipo de operação que foi realizada:

```
2025-05-21 00:16:16,108 - INFO - [LOG] Log individual configurado para servidor_1_log.txt
2025-05-21 00:16:16,108 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:16,109 - INFO - [SYNC] Relógio local: 1747797376.11
2025-05-21 00:16:16,109 - INFO - [DRIFT] Drift aplicado ao relógio local: -0.35s. Novo valor: 1747797375.75
2025-05-21 00:16:18,114 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:20,118 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:21,111 - INFO - [DRIFT] Drift aplicado ao relógio local: -0.82s. Novo valor: 1747797374.93
2025-05-21 00:16:22,123 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:22,656 - INFO - [SYNC] Sincronização recebida. Ajustando relógio local de 1747797374.93 para 1747797382.66
2025-05-21 00:16:24,125 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:26,112 - INFO - [DRIFT] Drift aplicado ao relógio local: +0.41s. Novo valor: 1747797383.07
2025-05-21 00:16:26,114 - INFO - [SYNC] Relógio local: 1747797383.07
2025-05-21 00:16:26,115 - INFO - [Atualização] Servidores conectados: ['1']
2025-05-21 00:16:26,130 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:28,115 - INFO - [SYNC] Checagem de líder: líder atual é 1
2025-05-21 00:16:28,116 - INFO - [SYNC] Sou o líder (1). Enviando sincronização de relógio (1747797388.12)
2025-05-21 00:16:28,117 - INFO - [SYNC] Sincronização recebida. Ajustando relógio local de 1747797383.07 para 1747797388.12
2025-05-21 00:16:28,117 - INFO - [SYNC] Resposta do proxy para sync_clock: {'status': 'clock_sync_broadcasted', 'timestamp'
2025-05-21 00:16:28,132 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:30,132 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:31,116 - INFO - [DRIFT] Drift aplicado ao relógio local: +0.24s. Novo valor: 1747797388.36
2025-05-21 00:16:32,136 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:34,142 - INFO - [HEARTBEAT] Enviado heartbeat do server 1
2025-05-21 00:16:36,119 - INFO - [SYNC] Relógio local: 1747797388.36
2025-05-21 00:16:36,119 - INFO - [DRIFT] Drift aplicado ao relógio local: -0.35s. Novo valor: 1747797388.01
2025-05-21 00:16:36,119 - INFO - [Atualização] Servidores conectados: ['1']
```

4. Como Executar o Sistema

1. Seguir os pré-requisitos
2. Abrir o terminal.
3. Inicializar o Proxy executando o arquivo “Proxy.py”.
4. Inicializar os Servidores executando o arquivo “Servidores.py”:

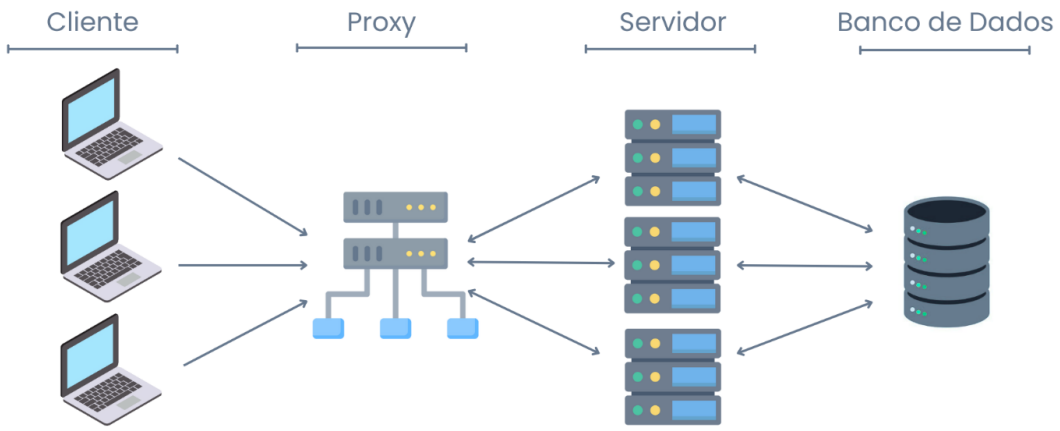
    - Abrir 3 terminais separados.
    - Em cada terminal, executar o arquivo Servidor.py informando um identificador (1, 2 ou 3).

5. Inicializar os Usuários executando o arquivo “Usuario.py”, ou o compilado do Usuario.c ou o Usuario.java:

    - Abrir 5 terminais separados.

- Em cada terminal, executar o arquivo Usuario.py informando o nome do usuário.

5. Arquitetura do Sistema



6. Mensagens entre sistemas:

O sistema utiliza o padrão de mensagens JSON para a comunicação entre clientes, servidor e banco de dados. Todas as mensagens trocadas seguem uma estrutura padronizada, onde o campo principal é "action", responsável por indicar a operação desejada. Os demais campos variam conforme a funcionalidade.

Mensagens:

- Cadastro de usuário:

  O cliente envia uma mensagem JSON contendo a ação "add_user" e o nome de usuário desejado. O servidor retorna o resultado da operação, juntamente com o identificador do usuário e o tópico de notificação correspondente.

  - Entrada: { "action": String , "username": String }

- Retorno: { "ret": Int, "id": Int, "topic": String}

- Publicação de Texto:

  O cliente envia uma mensagem JSON com a ação "post_text", incluindo nome de usuário, ID, texto e horário de envio. O servidor retorna apenas o status da operação.

  - Entrada: { "action": String, "username": String, "id": Int, "texto": String, "tempoEnvioMensagem": String ou Int }
  - Retorno: { "ret": Int }

- Seguir Usuário:

  O cliente envia a ação "add_follower", com o ID do usuário atual e o nome do usuário a ser seguido. O servidor confirma a operação com um código de retorno.

  - Entrada: { "action": String, "id": Int, "to_follow": String }
  - Retorno: { "ret": Int }

- Envio de Mensagem Privada:

  O cliente envia "add_private_message" com remetente, destinatário, mensagem e timestamp inteiro. O servidor responde apenas com o status da operação.

  - Entrada: { "action": String, "remetente": String, "destinatario": String, "mensagem": String, "timestamp": Int }
  - Retorno: { "ret": Int }

- Consulta de Conversa Privada:

  O cliente solicita o histórico de mensagens privadas com a ação "get_private_messages", informando remetente e destinatário. O servidor retorna a lista das mensagens entre os dois.

  - Entrada: { "action": String, "remetente": String, "destinatario": String }
  - Retorno: { "mensagens": [[String, Int, String], ...] }

- Consulta da Timeline:

  O cliente solicita as postagens públicas enviando "get_timeline". O servidor retorna uma lista das postagens com autor, texto e data/hora.

  - Entrada: { "action": String }
  - Retorno: [{ "username": String, "texto": String, "tempoEnvioMensagem": String ou Int }, ...]

- Notificações:

  As notificações são enviadas do servidor para o cliente via PUB/SUB em JSON, contendo o tópico e a mensagem.

  - Entrada: { "topico": String, "mensagem": String }