Universidade Federal de Uberlândia - UFU
Faculdade de Computação - FACOM
Bacharelado em Sistemas de Informação - Campus Monte Carmelo
Disciplina: Sistemas Distribuídos

Nome: Lucas Dornelles
Matrícula: 31811BSI026

Sistema de Gerenciamento de Tarefas Distribuído (To-Do List)

Descrição do Tema

O projeto consiste no desenvolvimento de um **Sistema de Gerenciamento de Tarefas Distribuído (To-Do List)**, onde múltiplos clientes podem interagir com uma lista compartilhada de tarefas. O sistema possui:

- Um **Servidor Central (Líder)** que coordena as operações
- **Nós Secundários** (réplicas) que mantêm cópias sincronizadas da lista

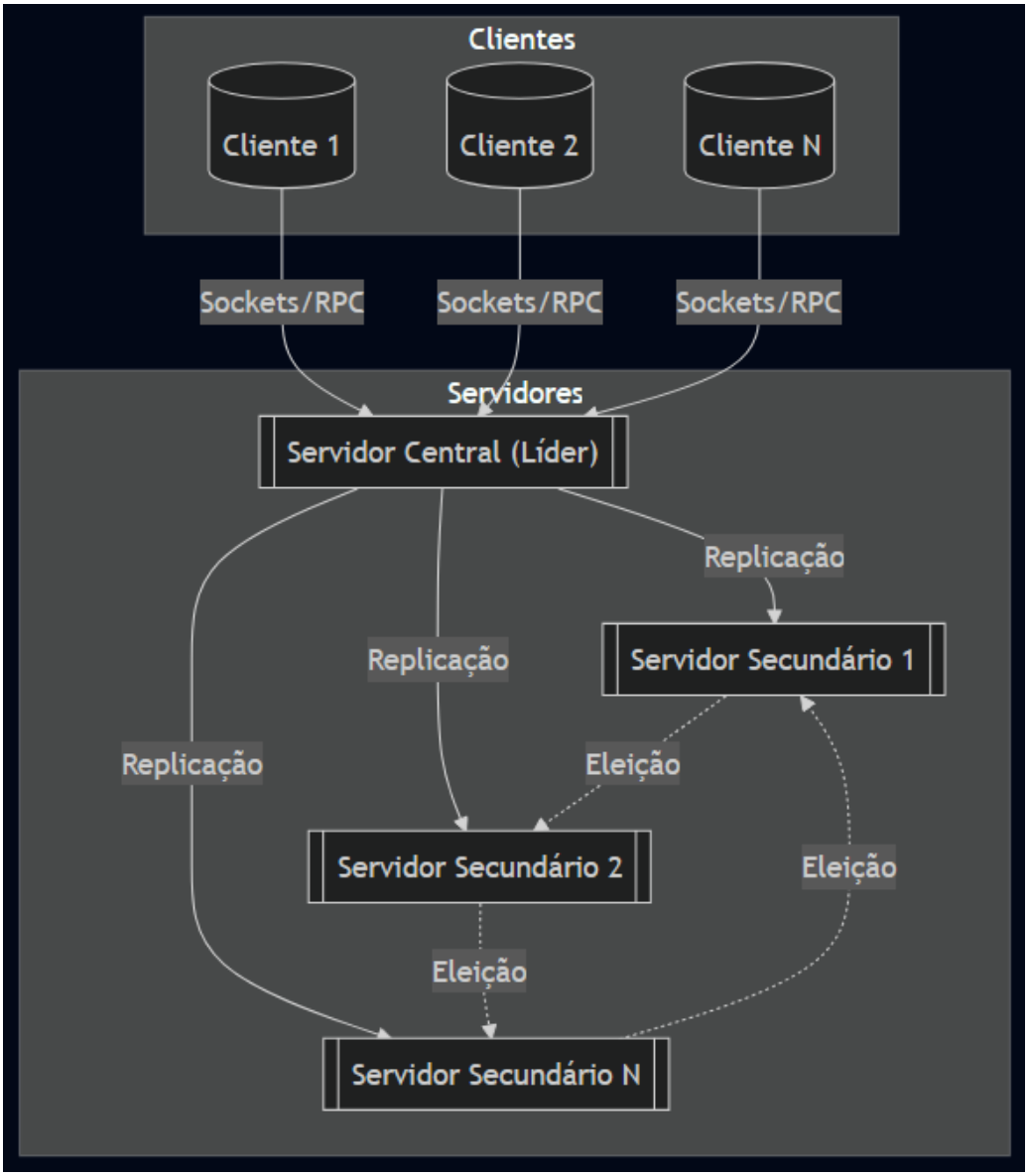
Fluxo Principal:

1. Cliente envia requisição (add/remove/edit/list) ao líder via **Sockets/RPC**
2. Líder processa e replica a mudança para os nós secundários
3. Se o líder falhar, os nós iniciam um **protocolo de eleição**

Metas de Sistemas Distribuídos

Meta	**Implementação**
Escalabilidade	Adição dinâmica de nós secundários
Disponibilidade	Failover automático para réplicas
Tolerância a Falhas	Eleição de líder + heartbeat
Consistência	Modelo eventualmente consistente
Exclusão Mútua	Timestamps para conflitos

Diagrama da Arquitetura



Componentes

Cliente: Processo que envia operações via Sockets/RPC. Interface pode ser CLI ou GUI.

Servidor Central (Líder): Processo principal que gerencia a lista global, coordena réplicas e implementa lógica de eleição.

Servidores Secundários: Processos que mantêm réplicas atualizadas e participam da eleição.

Etapa 2: Implementação com Threads e Comunicação via Sockets

Descrição do Estado Atual do Projeto

Nesta etapa, o projeto está funcional com três componentes principais: servidor líder, servidores secundários e clientes. Os clientes são capazes de se conectar simultaneamente ao servidor líder e enviar comandos para adicionar, remover, editar e listar tarefas. O líder processa essas operações e replica as mudanças para os servidores secundários via comunicação socket. Cada secundário mantém uma cópia atualizada da lista de tarefas.

Justificativa: Processos vs. Threads

A implementação foi feita utilizando **threads**, em vez de processos, pelos seguintes motivos:

- **Baixo custo de criação e gerenciamento**: Threads são mais leves e rápidas para serem criadas.
- **Compartilhamento de memória**: As threads compartilham o mesmo espaço de memória, o que facilita o acesso e atualização da lista de tarefas entre diferentes conexões simultâneas.
- **Sincronização simplificada**: Com o uso de locks, é possível garantir exclusão mútua de forma simples e eficiente.

Descrição da Comunicação via Sockets

Toda a comunicação entre os componentes do sistema ocorre via **Sockets TCP**. A escolha do protocolo TCP se deu por garantir:

- **Confiabilidade na entrega dos dados**
- **Manutenção da ordem das mensagens**
- **Correção de erros automática no nível de transporte**

Conexões implementadas:

- **Clientes <−> Servidor Líder**: os clientes enviam comandos (add, remove, edit, list) e recebem respostas.
- **Servidor Líder <−> Servidores Secundários**: o líder replica alterações para as réplicas sempre que a lista é modificada.

Formato das mensagens:

- `add;Tarefa`
- `remove;Tarefa`
- `edit;Antiga;Nova`
- `list`

Instruções de Execução

1. Certifique-se de ter o Python 3 instalado no sistema.
2. No terminal, inicie o servidor líder:

   ```
   python servidor_lider.py
   ```

3. Em outro terminal, inicie um ou mais servidores secundários:

   ```
   python servidor_secundario.py
   ```

4. Em outro terminal, execute um cliente:

   ```
   python cliente.py
   ```

5. No cliente, utilize os seguintes comandos:

   - `add;Comprar pão`
   - `remove;Comprar pão`
   - `edit;Comprar pão;Comprar café`
   - `list`
   - `exit` ou `quit` para encerrar