Sistema de Exclusão Mútua Distribuído com Impressão Coordenada

Projeto no GitHub: https://github.com/pdMiranda/CD/tree/main/TP_01

1. Introdução

Este projeto simula um sistema distribuído onde múltiplos nós acessam um recurso compartilhado de forma concorrente e segura. O objetivo é garantir exclusão mútua, com sincronização precisa, controle de acesso e rastreabilidade total através de logs. A seção crítica é representada por um servidor de impressão que imprime uma sequência numérica controlada.

2. Visão Geral do Sistema

O controle da exclusão mútua é feito entre os próprios nós, utilizando o algoritmo distribuído de Ricart-Agrawala. O orquestrador atua como um ponto de entrada que libera acesso ao recurso compartilhado (print_server), mas não interfere na decisão de entrada na CS. A comunicação usa protocolos TCP/IP via Socket

3. Componentes Principais

• distributed_node.py – Nó que implementa o algoritmo de Ricart-Agrawala.

• orquestrador.py – Libera o acesso ao print_server após o nó obter permissão da rede.

• print_server.py – Imprime k números com intervalo de 0.5s, a partir do timestamp do nó.

4. Comunicação entre Componentes

1. Nó → Outros Nós: REQUEST / REPLY
2. Nó → Orquestrador: ENTER:<node_id>:<timestamp>
3. Orquestrador → Print Server: START:<node_id>:<last_number>
4. Print Server → Orquestrador: DONE:<node_id>:<last>
5. Orquestrador → Nó: ENTER_OK
6. Nó → Orquestrador: EXIT

5. Questões de Implementação

5.1 Docker e Docker Compose

Todos os componentes são containerizados com Docker. O Docker Compose é usado para orquestrar os serviços. Cada nó, orquestrador e print_server roda em seu próprio container.

5.2 Adição ou Remoção de Nós

Os nós são definidos no docker-compose.yml. Para adicionar um novo nó:

```
nodeX:
  <<: *default-config
  container_name: nodeX
  command: python3 distributed_node.py --id X --port 500X
```

Para remover, basta deletar o bloco correspondente. O código ajusta os vizinhos automaticamente.

5.3 Watchdog

Cada nó possui um watchdog. Se travar dentro da CS, ele força a saída após tempo limite para evitar deadlocks.

5.4 Controle de Tempo

O nó permanece na CS até que o print_server termine de imprimir. O orquestrador só envia ENTER_OK após o DONE do print_server.

6. Logs e Monitoramento

- node_<id>.log – Logs dos nós (entrada, saída, requests, replies)
- orquestrador.log – Acesso ao recurso, conflitos
- print_service.log – Impressões realizadas e confirmações

7. Conclusão

A exclusão mútua é garantida de forma distribuída pelos próprios nós. O orquestrador atua apenas como uma porta de acesso ao recurso físico (print_server). A sincronização é exata: o nó só sai da CS após concluir a impressão. O sistema é modular, extensível e robusto, com logs completos e controle de falhas.

8. Prints de Execução

PRINT SERVER

```
2025-05-25 16:55:45,502 - INFO - Print Server service started

2025-05-25 16:55:47,453 - INFO - Node 4 started printing numbers. | time: 7 | k = 10
2025-05-25 16:55:47,453 - INFO - Node 4 >> 8 | 1
2025-05-25 16:55:47,767 - INFO - Node 4 >> 17 | 10
2025-05-25 16:55:47,954 - INFO - Node 4 >> 9 | 2
2025-05-25 16:55:48,455 - INFO - Node 4 >> 10 | 3
2025-05-25 16:55:48,956 - INFO - Node 4 >> 11 | 4
2025-05-25 16:55:49,456 - INFO - Node 4 >> 12 | 5
2025-05-25 16:55:49,957 - INFO - Node 4 >> 13 | 6
2025-05-25 16:55:50,457 - INFO - Node 4 >> 14 | 7
2025-05-25 16:55:50,958 - INFO - Node 4 >> 15 | 8
2025-05-25 16:55:51,459 - INFO - Node 4 >> 16 | 9
2025-05-25 16:55:55,795 - INFO - Finished printing for Node 4


2025-05-25 16:55:56,864 - INFO - Node 5 started printing numbers. | time: 31 | k = 4
2025-05-25 16:55:56,864 - INFO - Node 5 >> 32 | 1
2025-05-25 16:55:57,365 - INFO - Node 5 >> 33 | 2
2025-05-25 16:55:57,865 - INFO - Node 5 >> 34 | 3
2025-05-25 16:55:58,366 - INFO - Node 5 >> 35 | 4
2025-05-25 16:55:58,866 - INFO - Finished printing for Node 5
```

NODE

```
2025-05-25 16:55:46,113 - INFO - Node 4 started on port 5004
2025-05-25 16:55:47,442 - INFO - Requesting CS with timestamp 1
2025-05-25 16:55:47,448 - INFO - Received REPLY from Node 1 (1/5)
2025-05-25 16:55:47,449 - INFO - Received REPLY from Node 6 (2/5)
2025-05-25 16:55:47,449 - INFO - Received REPLY from Node 5 (3/5)
2025-05-25 16:55:47,450 - INFO - Received REPLY from Node 2 (4/5)
2025-05-25 16:55:47,450 - INFO - Received REPLY from Node 3 (5/5)
2025-05-25 16:55:47,450 - INFO - === ENTERING CRITICAL SECTION ===
2025-05-25 16:55:47,534 - INFO - Granted access to Node 5
2025-05-25 16:55:47,839 - INFO - Granted access to Node 2
2025-05-25 16:55:47,957 - INFO - Granted access to Node 6
2025-05-25 16:55:48,366 - INFO - Granted access to Node 1
2025-05-25 16:55:50,399 - INFO - Granted access to Node 3
2025-05-25 16:55:55,805 - INFO - === IN CRITICAL SECTION ===
2025-05-25 16:55:55,821 - INFO - === EXITING CRITICAL SECTION ===
2025-05-25 16:55:55,837 - INFO - Time spent in CS: 8.386841535568237s
2025-05-25 16:55:55,837 - INFO - === LEFT CRITICAL SECTION ===
```