MC 714 – Sistemas Distribuídos

Trabalho 2

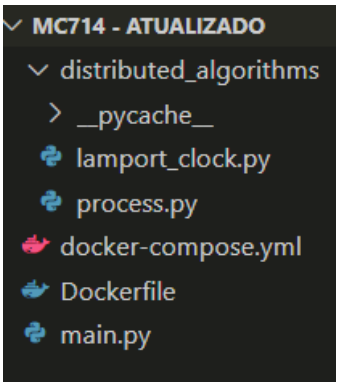
Samuel Nascimento de Souza RA:244192

1. Introdução

Esse trabalho tem como objetivo realizar um estudo sobre os algoritmos utilizados em sistemas distribuídos. Dessa forma, foram simulados ambientes distribuídos a fim de testar três implementações principais: um Relógio de Lamport, um algoritmo de exclusão mútua (Algoritmo de Ricart-Agrawala) e um algoritmo de eleição de líder (Valentão).

2. Implementação

A implementação inicial foi pensada em se utilizar python juntamente com o Docker com a finalidade de criar contêiners para emular um ambiente distribuído. A comunicação entre os processos foi pensada para ser feita utilizando MPI. Entretanto, problemas na comunicação exigiram um caminho alternativo. Sendo assim, os processos foram criados a partir de threasd, para rodar em paralelo e permitir a comunicação. O diretório geral do projeto pode ser visto na imagem abaixo:



Na pasta distributed_algotithms temos as implementações dos algoritmos propostos. O arquivo lamport_clock.py contém a implementação do relógio, que foi feita de forma bem simples, com um método de tick(), que incrementa quando ocorre algum evento, update(), que atualiza o relógio entre os processos e get_time() que retorna o valor atual.

```python
# Relógio de Lamport
class LamportClock:
    def __init__(self):
        self.time = 0

    def tick(self):
        self.time += 1
        return self.time

    def update(self, received_time):
        self.time = max(self.time, received_time) + 1
        return self.time

    def get_time(self):
        return self.time
```

No arquivo process.py temos a implementação do algoritmo de exclusão mútua, no qual foi utilizado Ricart-Agrawala e de eleição de líder, utilizando a técnica de bullying. O processo conta com as seguintes funções:

```python
class Process:

    def __init__(self, pid, n_processes):...

    def start(self):...

    def request_critical_section(self):...

    def handle_message(self, sender, timestamp, message_type):...

    def use_critical_section(self):...

    def send_message(self, recipient, message_type, timestamp):...

    def start_election(self):...

    def announce_leader(self):...

    def check_leader(self):...
```

FUNÇÃO	FINALIDADE
def __init__(self,...):	Inicializar as variáveis do processo
***def* start(*self*):**	Iniciar o processo
def request_critical_section(self):	Solicitar acesso ao recurso compartilhado
def handle_message(self,...):	Executar ações a partir do state da mensagem recebida
***def* use_critical_section(*self*):**	Entra na seção crítica e realiza saída em sequência.
***def* send_message(*self*,..):**	Função de envio de mensagens

***def* start_election(*self*):**	Iniciar eleição de processo líder
def announce_leader(self):	Anunciar que o processo atual ganhou a eleição
def check_leader(self):	Verifica se o processo líder ainda está ativo

Cada processo possui as seguintes variáveis quando o processo é instanciado:

```python
class Process:
    def __init__(self, pid, n_processes):
        self.pid = pid
        self.n_processes = n_processes
        self.clock = LamportClock()
        self.request_queue = []
        self.reply_count = 0
        self.replies = []
        self.state = 'RELEASED'
        self.timestamp = 0
        self.leader = None
        self.running = True
        self.leader_alive = True
```

A partir dessa modelagem, foram definidos states para cada tipo de mensagem que pode ser recebida ou enviada, que são tratadas na função handle_messages().Os states utilizados estão disponíveis na tabela abaixo:

STATE	FINALIDADE
RELEASED	Processo aguardando solicitações
WANTED	Pretende acessar recurso
HELD	Permitido entrar na seção crítica
REQUEST	Envia mensagem aos demais processos pedindo acesso ao recurso na área crítica
REPLY	Envia resposta ao processo que solicitou acesso caso não esteja utilizando o recurso
ELECTION	Envia mensagem aos demais processos para realizar eleição
COORDINATOR	Novo líder envia mensagens para ajustar líder em cada um dos processos
CHEAK_LEADER	Envia mensagem para ao líder para verificar se está vivo.
LEADER_ALIVE	O Líder retorna mensagem se estiver vivo

Por fim, temos os arquivos de configuração do Docker, que são explicados no vídeo e o arquivo main.py é responsável por iniciar os processos e providenciar os ambientes para os testes e vai ser abordado no próximo tópico.

3. Testes

Após as falhas na utilização do Docker, o teste foi realizado na própria função main(), onde os processos eram instanciados e rodavam durante 20 segundos, onde vericamos o funcionamento do relógio e do algoritmo de exclusão mútua. Após esse período, paramos o líder e verificamos o algoritmo de eleição, por mais 20 segundos. Logo após, retornamos o processo 2 e deixamos os processos estabilizarem, verificando o comportamento encerrando os processos ao final. Seguem alguns logs:

Início:

```
ss 0 is starting an election
Process 1 is starting an election
Process 2 is starting an election
Process 2 announces itself as leader at time
Process 0 requests critical section at time 4
No leader is currently elected. Process 0 is starting a new election
Process 0 is starting an election
Process 0 received COORDINATOR from Process 2 at time 5
Process 0 recognizes Process 2
Process 2 requests critical section at time 7
Process 2 received ELECTION from Process 0 at time 8
Process 2 received ELECTION from Process 1 at time 10
Process 2 received REQUEST from Process 0 at time 12
Process 2 received ELECTION from Process 0 at time 14
Process 1 requests critical section at time 9
No leader is currently elected. Process 1 is starting a new election
Process 1 is starting an election
Process 1 received ELECTION from Process 0 at time 10
```

Falha do líder (Processo 2):

```
Leader 2 is not responding. Process 0 is starting a new election
Process 0 is starting an election
Process 0 received COORDINATOR from Process 1 at time 621
Process 0 recognizes Process 1
Process 0 received REQUEST from Process 1 at time 623
Process 1 requests critical section at time 626
Process 1 received REPLY from Process 0 at time 627
Process 1 received ELECTION from Process 0 at time 629
Process 1 received REPLY from Process 0 at time 631
Process 0 requests critical section at time 628
Process 0 received REQUEST from Process 1 at time 629
Process 1 is in critical section at time 636
Process 1 exits critical section at time 638
Process 1 received REQUEST from Process 0 at time 639
Process 1 received CHECK_LEADER from Process 0 at time 641
Process 1 received REPLY from Process 0 at time 643
Process 1 received CHECK_LEADER from Process 0 at time 645
Process 1 received CHECK_LEADER from Process 0 at time 647
Process 0 received REPLY from Process 1 at time 650
Process 0 received LEADER_ALIVE from Process 1 at time 652
Process 0 received LEADER_ALIVE from Process 1 at time 654
Process 0 received LEADER_ALIVE from Process 1 at time 656
```

Retorno Líder (Processo 2) e estabilização:

```
Process 0 is in critical section at time 983Process 1 requests critical section a

Process 2 is starting an election
Process 2 announces itself as leader at time
Process 0 exits critical section at time 986
Process 0 received REQUEST from Process 1 at time 987
Process 0 received LEADER_ALIVE from Process 1 at time 989
Process 0 received REQUEST from Process 1 at time 991
Process 0 received COORDINATOR from Process 2 at time 993
Process 0 recognizes Process 2
Process 1 requests critical section at time 994
Process 1 received COORDINATOR from Process 2 at time 995
Process 1 recognizes Process 2
Process 1 received REPLY from Process 0 at time 997
Process 1 received CHECK_LEADER from Process 0 at time 999
Process 1 received REPLY from Process 0 at time 1001
Process 1 received REPLY from Process 0 at time 1003
Process 2 requests critical section at time 1
Process 0 requests critical section at time 1003
Process 0 received REQUEST from Process 1 at time 1004
```

4.Principais Dificuldades

As principais dificuldades encontradas durante a execução desse trabalho foram a comunicação entre os processos e o algoritmo de exclusão mútua que apresentou algumas falhas consideráveis por conta de alguns stats que estavam sendo esquecidos ou não considerados no início. Além disso, apresentou alguns erros inicialmente ao se incorporar a eleição de líder, sendo necessário ajustar os stats quando ocorria alguma eleição para que houvesse solicitação de acesso ao recurso compartilhado por parte dos projetos restantes.

5. Fontes

O trabalho utilizou alguns tutorias de Docker, além de ajuda do GPT-4 para auxiliar na montagem dos testes e correção de algumas partes do código feito incialmente.

6. Links Úteis:

- Repositório Git
- Docker
- MPI