Implementação de algoritmos de sistemas distribuídos com comunicação gRPC

Luiz Henrique Yuji Delgado Oda, RA 247255

Julho 2024

1 Introdução

Este relatório possui como objetivo descrever o que foi implementado neste repositório do github, além de instruções de execução. Foram implementados os algoritmos: Relógio lógico de Lamport, Bully (Valentão) e Token Ring. O algoritmo Bully é um algoritmo de eleição de líder, enquanto o Token Ring é um algoritmo de exclusão mútua.

2 Algoritmos

2.1 Relógio Lógico de Lamport

Cada processo possui um contador, iniciado em zero. Ao ocorrer algum evento interno, seu contador é incrementado em um. Ao enviar uma mensagem, temos que o contador é incrementado em um e seu valor é enviado (pós-incremento). Ao receber uma mensagem, o contador incrementa seu valor em um e recebe o máximo entre seu valor e o valor da mensagem recebida. A ideia é manter a consistência causal, ou seja, se um evento a causa b, o tempo de a deve ser menor que o de b.

2.2 Bully

Cada processo possui um id, e seu objetivo é eleger um líder dinamicamente. A cada momento teremos um líder, e para um processo P, teremos o seguinte procedimento caso P se recupere de uma falha ou caso o líder falhe:

- P inicia uma eleição, enviando uma mensagem de eleição para todos os processos com id mais alto que o seu
- Caso não receba nenhuma mensagem de volta, P envia uma mensagem de vitória para todos os processos, sinalizando que ele se tornou o novo líder

- Caso receba alguma mensagem de volta, espera por uma mensagem de vitória de algum processo com id mais alto que o seu, e caso não receba retoma a eleição

É importante ressaltar que algumas modificações foram realizadas para acomodar da melhor forma o algoritmo. Uma delas foi que para descobrir se o líder falhou, todos os nós enviam regularmente mensagens para checar se o lider ainda está ativo.

2.3 Token Ring

Temos um token que é repassado de maneira circular, ou seja, os processos são arranjados em um círculo. O processo que possui o token ganha o direito de passar por uma sessão crítica, e então deve passar o token para o próximo processo. Assim, como só temos um processo com o token por vez, garantimos acesso mutualmente exclusivo ao recurso crítico.

3 Implementação

Cada algoritmo possui sua implementação independente, cada um em uma pasta com seu nome. Foi utilizado o Docker para simular um sistema distribuído, com 4 serviços, cada um simulando uma máquina. Para a comunicação foi utilizado gRPC, uma forma de comunicação moderna e muito eficaz.

No arquivo Dockerfile temos as configurações para construir a imagem que queremos em nossas máquinas virtuais. No arquivo docker-compose.yml temos as configurações do ambiente local, como quais portas cada serviço vai usar. Também temos o arquivo requirements.txt, que indica alguns requisitos que devem ser instalados no container do docker. É importante ressaltar que para todos os algoritmos foram utilizados 4 componentes, todas podendo se comunicar.

Já na pasta src temos 2 componentes, a pasta proto e o arquivo device.py. Na pasta proto temos o arquivo mensagem.proto que descreve como é o formato das mensagens a serem enviadas, assim como define as interfaces de serviços RPC. Nela também temos outros dois arquivos, que são gerados ao compilar o arquivo .proto com grpcio e grpcio-tools. Já para o arquivo device.py, ele é o programa que os containers irão executar. Ele possui nossa implementação do algoritmo daquela pasta.

4 Resultados

4.1 Testes

Foram realizados alguns testes para os algoritmos. No Relógio Lógico de Lamport, foi observado que os contadores se comportaram de forma esperada, tanto ao executar eventos internos quanto ao receber mensagens de outros containers.

Para o algoritmo Bully, foi observado que o container com maior id sempre é eleito como líder, e em caso de falha (como desligar o container), o próximo líder era eleito corretamente, e caso o container com maior id volte ele também se torna o líder novamente. Já para o algoritmo Token Ring, foi observado que o token foi passado corretamente de forma cíclica e que não ocorreu de um container acessar a região crítica junto a outro container.

4.2 Comentários

O uso do Docker para simular um sistema distribuído facilita muito a implementação dos algoritmos e sua simulação. No entanto, foram enfrentados problemas em relação a como testar o código e simular os processos, visto que temos o mesmo código para todos os containers. Uma ideia (a que foi implementada) foi a de realizar ações baseadas no número do container ou de forma randômica. Uma outra possibilidade seria o uso de alguma interface para enviar mensagens para os containers, como o gRPCurl.

4.3 Conclusão

No geral, o projeto foi proveitoso e foi possível aprender mais sobre sistemas distribuídos e algumas das dificuldades enfrentadas.

5 Referências Bibliográficas

1. Relógio Lógico de Lamport`https://en.wikipedia.org/wiki/Lamport_timestamp`.
2. Algoritmo Bully `https://en.wikipedia.org/wiki/Bully_algorithm#:~:text=In%20distributed%20computing%2C%20the%20bully,is%20selected%20as%20the%20coordinator.`
3. Algoritmo Token Ring `https://denninginstitute.com/workbenches/token/token.html#:~:text=Token%20Ring%20algorithm%20achieves%20mutual,next%20in%20line%20after%20itself.`.