Relatório Técnico

Sistema Distribuído com MQTT, Eleição e Prova de Trabalho

Autora: Gabriel Ziktnel, Kayo heleno, Roniel
Disciplina: Programação Paralela e Distribuída
Data: 01/12/2025

1. Introdução

Este projeto teve como objetivo implementar um sistema distribuído capaz de:

- Realizar descoberta entre processos usando um broker MQTT;
- Eleger um líder de forma distribuída;
- Alternar papéis entre controlador (líder) e mineradores;
- Executar um desafio de prova de trabalho (Proof-of-Work - PoW) simples.

O sistema segue o modelo definido pelo professor, utilizando um broker EMQX público e comunicação pelos tópicos `sd/init`, `sd/voting`, `sd/challenge`, `sd/solution` e `sd/result`.

Apesar de aparentar simplicidade, diversos desafios técnicos surgiram durante o desenvolvimento, principalmente devido ao comportamento assíncrono do MQTT e à sincronização entre múltiplos processos distribuídos. Nas seções seguintes descrevo a arquitetura final e os problemas enfrentados.

2. Arquitetura Geral do Sistema

Cada processo implementa simultaneamente dois papéis principais:

- Participante da eleição distribuída;
- Controlador ou minerador (dependendo do resultado obtido na eleição).

Toda a comunicação entre os nós ocorre exclusivamente por meio do protocolo MQTT, promovendo baixo acoplamento e independência entre os processos.

O sistema foi estruturado nas seguintes fases:

1. **DISCOVERY**: cada nó envia seu ClientID até identificar os demais nós;

2. **ELECTION**: cada nó envia um voto aleatório e o líder é escolhido;
3. **CHALLENGE**: o líder gera um novo desafio de mineração;
4. **MINING**: cada minerador tenta encontrar uma solução que gere um hash SHA-1 com prefixo de zeros;
5. **RESULT**: o líder valida as soluções recebidas e anuncia o vencedor.

A comunicação entre as fases é inteiramente assíncrona, aproveitando as capacidades de publicação/assinatura do MQTT.

3. Problemas Encontrados e Lições Aprendidas

Durante o desenvolvimento inicial, diversas inconsistências ocorreram. Abaixo listo os principais desafios e as soluções adotadas.

3.1. Perda de mensagens durante a inscrição MQTT

O MQTT não garante a entrega de mensagens enquanto o cliente ainda não completou a fase de *subscribe*. Isso provocava:

- nós que nunca recebiam mensagens `INIT`;
- travamentos ao esperar outros participantes;
- eleições incompletas.

Correção: todos os tópicos passaram a ser assinados imediatamente após a conexão, eliminando perda de mensagens.

3.2. Estado interno inconsistente

Em versões anteriores, o ClientID era alterado após a conexão. Isso não é permitido no Paho MQTT, pois o cliente mantém buffers e estados internos baseados no ID inicial.

Consequências:

- mensagens não eram recebidas;
- inscrições eram descartadas;
- surgiam comportamentos aleatórios e inconsistentes.

Correção: o cliente agora é criado diretamente com o ID final, evitando reconfigurações tardias.

3.3. Mineração interrompida antes do término

Alguns nós interrompiam a mineração antes do processamento do resultado. O problema estava na variável de estado da transação, que não era reinicializada adequadamente.

Correção: o estado de transação é reiniciado sempre que um novo desafio é gerado.

4. Funcionamento da Versão Final

A versão final implementa o comportamento distribuído de forma estável e organizada. Entre as principais características:

- Descoberta dos participantes usando mensagens repetidas de INIT;
- Eleição distribuída usando números aleatórios de 16 bits;
- Definição automática do líder ao comparar votos e ClientIDs;
- Mineração usando SHA-1 e busca por prefixos com zeros;
- Validação centralizada pelo líder;
- Criação automática de novos desafios após cada vencedor.

O ciclo de descoberta, eleição, desafio e mineração funciona continuamente enquanto os processos permanecem ativos.

5. Conclusão

O projeto permitiu explorar na prática diversos conceitos fundamentais em sistemas distribuídos, tais como:

- coordenação e sincronização entre processos independentes;
- desafios do modelo Publish/Subscribe;
- impacto de latência e assincronicidade sobre algoritmos distribuídos;
- importância de estados bem definidos e consistentes.

A versão final do sistema apresentou comportamento estável e aderente aos requisitos da disciplina. Após ajustes e correções, foi possível observar o funcionamento completo do sistema: descoberta, eleição, mineração e validação dos resultados.