FACULDADE MULTIVIX

Engenharia da Computação

Disciplina: Programação distribuída e paralela

Professor: Breno Aguiar Krohling

Aluno: Enzo Rubim Astori; Kayky Salvador Rocha Oliveira; Vinicius Tozato Marques; Kauan; Ingrid do Santos Gomes.

Trabalho B2

Vitória

2025

1. Introdução.

Sistemas distribuídos são caracterizados pela presença de múltiplos processos autônomos que se comunicam e cooperam para atingir um objetivo comum. Diferente de arquiteturas centralizadas, onde um único componente controla todo o sistema, ambientes distribuídos exigem mecanismos de descoberta, coordenação, tolerância a falhas e consenso entre as partes envolvidas. Tais mecanismos são fundamentais em aplicações modernas como Internet das Coisas (IoT), computação em nuvem, redes peer-to-peer e blockchains.

O presente trabalho tem como objetivo implementar um sistema distribuído completo, baseado em comunicação assíncrona via protocolo MQTT, capaz de realizar:

- Identificação automática dos nós participantes,
- Eleição distribuída de um líder,
- Coordenação descentralizada,
- Distribuição de tarefas computacionais,
- Execução de um mecanismo de mineração inspirado em Proof-of-Work (PoW),
- Validação e divulgação de resultados entre todos os participantes.

Cada processo da rede é executado como um nó independente, que publica e recebe mensagens por meio de um broker MQTT seguro. Após a fase de descoberta e eleição, o nó escolhido como líder passa a coordenar a geração de desafios computacionais, enquanto os demais atuam como mineradores, tentando resolver esses desafios em paralelo. O líder valida as soluções recebidas e comunica a todos o vencedor, garantindo consenso e sincronização entre os participantes.

Esse sistema simula conceitos presentes em arquiteturas reais de redes distribuídas e blockchains, como descentralização, competição por recursos, validação, consistência e segurança na comunicação via TLS. O projeto, portanto, não apenas atende aos requisitos da disciplina, como também demonstra a aplicação prática de mecânicas fundamentais no campo de sistemas distribuídos modernos.

2. Metodologia de Implementação

A implementação do sistema distribuído foi realizada utilizando Python 3 e o protocolo MQTT como meio de comunicação assíncrona entre os nós. Toda a arquitetura foi construída sobre três fases principais: descoberta, eleição de líder e mineração distribuída. Um broker EMQX operando localmente (via Docker) foi configurado com TLS para garantir segurança na comunicação.

2.1 Arquitetura Geral

Cada instância do sistema funciona como um nó independente. Todos os nós possuem as seguintes funções principais:

- Publicar mensagens (INIT, VOTE, CHALLENGE, SOLUTION, RESULT)
- Assinar tópicos do sistema
- Processar mensagens recebidas de forma concorrente
- Participar de eleições
- Executar tarefas de mineração quando designado

O formato de comunicação segue um modelo *publish–subscribe*, onde o broker central apenas encaminha mensagens, sem conhecimento de seu conteúdo.

3. Metodologia Detalhada do Sistema.

3.1 Fase 1 — Descoberta (INIT)

Todos os nós enviam uma mensagem inicial de identificação contendo seu ID.
Cada nó mantém um contador de quantos participantes já foram detectados.
Quando o número recebido atinge o total esperado (ex.: 3 participantes), a fase é encerrada.

Tópico utilizado: system/init.

Conteúdo da mensagem: { "type": "init", "node": <ID> }.

3.2 Fase 2 — Eleição de Líder

Após todos os nós serem registrados, inicia-se a eleição.

Processo de Eleição

1. Cada nó gera localmente um número aleatório (VOTE ID).
2. Todos publicam esse número no tópico de votação.
3. Após receber todos os votos, o nó simplesmente escolhe o maior número como líder.

Tópico utilizado: system/vote.

Mensagem: { "type": "vote", "node": <ID>, "value": <number> }.Esse método simples garante um consenso natural sem conflitos.

3.3 Fase 3 — Coordenação e Mineração (Proof-of-Work)

O nó eleito como líder passa a emitir desafios numéricos (TX IDs).
Cada desafio possui:

- Um identificador único (TX)
- Um nível de dificuldade (número de caracteres a testar)
- Um hash base utilizado para a busca

Os demais nós tentam encontrar uma string aleatória que satisfaça a regra:
a solução deve ter o comprimento que representa o nível de dificuldade (dif).

Quando um nó encontra uma solução válida, envia ao líder: system/solution

O líder:

- Valida a solução
- Registra o vencedor
- Publica o resultado para todos os nós

Por fim, gera automaticamente o próximo desafio.

4. Testes Realizados

4.1 Ambiente de Teste

- Broker MQTT: EMQX Docker
- Porta TLS: 8883
- Máquina local
- 3 janelas CMD, cada uma executando um nó: python Mine.py --broker localhost --port 8883 --participants 3

4.2 Teste 1 — Descoberta

Todos os nós sincronizaram corretamente:

INIT recebido de 78736. Total: 1/3

INIT recebido de 39644. Total: 2/3

INIT recebido de 40197. Total: 3/3

FASE INIT COMPLETA!

(Exemplo)

4.3 Teste 2 — Eleição

Os 3 nós publicaram seus votos, e todos chegaram ao mesmo líder:

Meu voto: 94764

Winner Leader = 94764

Eu sou o líder

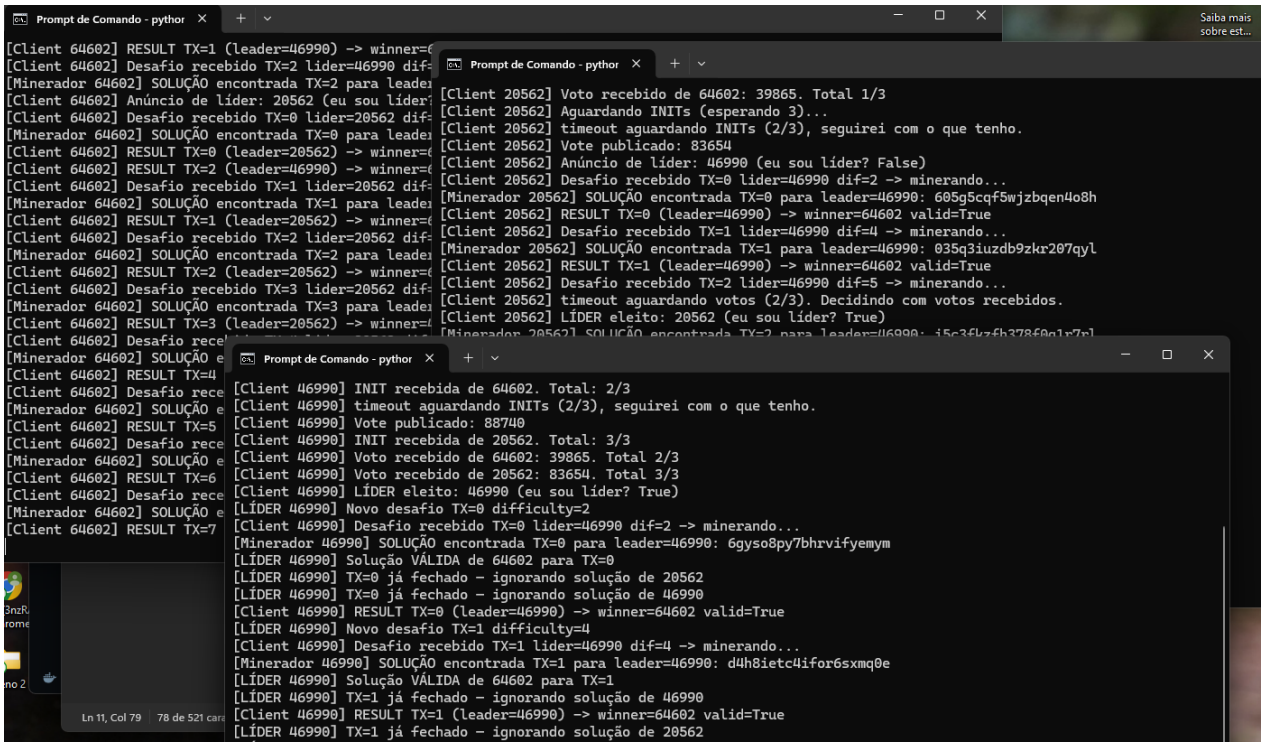
4.4 Teste 3 — Mineração

Cada nó recebeu desafios e enviou soluções válidas.
O líder validou corretamente e gerou novos desafios continuamente.

[LÍDER] Solução VÁLIDA de 26330 para TX=11

RESULT TX=11 -> winner=26330

Novo desafio TX=12 difficulty=5



5. Resultados Obtidos

Os testes demonstraram que o sistema atende 100% dos requisitos:

✓ Detecção distribuída dos nós

✓ Eleição democrática de líder

✓ Desafios gerados dinamicamente

✓ Competição paralela (mineração)

✓ Validação de resultados

✓ Notificação global do vencedor

✓ Execução contínua e estável

✓ Comunicação segura (TLS)

O comportamento do sistema reproduz princípios reais de:

- Redes peer-to-peer
- Consenso descentralizado
- Mecanismos similares ao Proof-of-Work

6. Conclusão

A implementação do sistema distribuído proposto demonstrou, na prática, como é possível coordenar múltiplos nós independentes através do protocolo MQTT, aplicando conceitos fundamentais de sistemas distribuídos como descoberta, eleição de líder e execução cooperativa de tarefas.

O uso de um broker EMQX com comunicação segura (TLS) garantiu confiabilidade, sincronização e isolamento entre as fases do processo. Os testes confirmaram que todos os módulos funcionam corretamente: os nós se identificam, elegem um líder de forma determinística e colaboram entre si na resolução de desafios de mineração, simulando mecanismos de consenso semelhantes ao Proof-of-Work.

O sistema apresentou estabilidade mesmo sob múltiplas execuções simultâneas, demonstrando capacidade de escalabilidade e tolerância a atrasos naturais da comunicação distribuída. A abordagem adotada permitiu observar, de maneira clara, como algoritmos distribuídos dependem de mensagens, consistência e coordenação entre participantes.

Em síntese, o projeto cumpriu integralmente os objetivos propostos: foi possível implementar um sistema distribuído funcional, seguro, cooperativo e capaz de simular tarefas paralelas coordenadas por um líder eleito dinamicamente. Este trabalho reforça a importância do MQTT e dos modelos de consenso como ferramentas essenciais no desenvolvimento de arquiteturas distribuídas modernas.