CENTRO UNIVERSITÁRIO MULTIVIX - VITÓRIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

DANIEL PADOVANI ALCANTARA (2213964)
LUIZ EDUARDO BARBOSA DE LIMA DOS SANTOS (2311397)
MELISSA MEL FREITAS VANNI (2313960)

AVALIAÇÃO PROCESSUAL 2º BIMESTRE - PARTE 2

Relatório Técnico: Comunicação Indireta Pub/Sub + Eleição e Coordenação

Vitória
2025

CENTRO UNIVERSITÁRIO MULTIVIX - VITÓRIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

DANIEL PADOVANI ALCANTARA (2213964)
LUIZ EDUARDO BARBOSA DE LIMA DOS SANTOS (2311397)
MELISSA MEL FREITAS VANNI (2313960)

AVALIAÇÃO PROCESSUAL 2º BIMESTRE - PARTE 2

Relatório Técnico: Comunicação Indireta Pub/Sub + Eleição e Coordenação

Avaliação Processual do 2º bimestre apresentada ao professor Breno Aguiar Krohling, como parte dos requisitos parciais para atribuição de nota na disciplina de Programação Paralela e Distribuída.
Vitória, 01 de dezembro de 2025.

Vitória
2025

1. INTRODUÇÃO

Este trabalho apresenta o desenvolvimento de um sistema distribuído baseado em comunicação MQTT para execução de tarefas de mineração computacional em um ambiente descentralizado. O sistema é composto por múltiplos nós independentes que se coordenam para formar uma rede cooperativa. Cada nó pode assumir dois papéis principais:

- **Controlador:** Responsável por coordenar desafios de mineração, validar soluções e publicar resultados;
- **Minerador:** responsável por receber desafios, executar a busca paralela por soluções válidas e enviar respostas ao controlador.

A infraestrutura de comunicação é viabilizada pelo broker MQTT Mosquitto, que atua como servidor central responsável por receber e encaminhar mensagens entre os nós. No ambiente de testes, o broker foi executado dentro do Windows Subsystem for Linux (WSL), permitindo que o sistema distribuído fosse avaliado em um ambiente Linux integrado ao Windows. A conexão entre os nós e o broker ocorre pela porta 1883, que é a porta padrão do protocolo MQTT sem criptografia.

Para garantir robustez e tolerância a falhas, o sistema implementa um protocolo distribuído de detecção de participantes, eleição de líder utilizando uma adaptação do algoritmo Bully, e um mecanismo de publicação e assinatura para sincronização entre os nós. A resolução de cada tarefa é modelada como um problema de proof-of-work, no qual os mineradores procuram valores que satisfaçam uma determinada dificuldade baseada em prefixos de zeros no hash SHA-1.

2. METODOLOGIA DE IMPLEMENTAÇÃO

O desenvolvimento do sistema foi estruturado em módulos lógicos que representam as principais etapas do funcionamento de cada nó distribuído. A primeira etapa refere-se à comunicação MQTT, na qual cada nó conecta-se a um broker e assina tópicos específicos para coordenar suas ações. Os tópicos utilizados são:

- **sd/init:** Responsável pela identificação dos participantes;
- **sd/voting:** Destinado à troca de votos durante a eleição;
- **sd/coordinator:** Para o anúncio do líder;
- **sd/challenge:** Publica novos desafios;
- **sd/solution:** Utilizado pelos mineradores para enviar soluções;

- **sd/result:** Valida as soluções submetidas.

Na fase INIT, todos os nós enviam periodicamente mensagens de presença ao broker. Cada nó mantém um registro dos participantes conhecidos e, quando o número de participantes atinge o limite esperado, a fase é concluída. Esse mecanismo assegura que todos os nós estejam sincronizados antes do início da eleição de líder.

A etapa seguinte corresponde à eleição Bully, na qual cada nó gera um voto aleatório. O nó que obtiver o maior valor torna-se o controlador da rede. Caso o tempo máximo seja atingido sem que todos os votos sejam recebidos, a eleição prossegue com os votos disponíveis, garantindo que o sistema avance mesmo em cenários de falha ou atraso na comunicação.

O papel do controlador é central para o funcionamento do sistema. Ele é responsável por gerar um desafio, composto por um identificador de transação (transactionID) e um nível de dificuldade. Esse desafio é publicado para todos os mineradores, que passam a competir pela solução. O controlador recebe as soluções enviadas e realiza a validação, verificando se o hash SHA-1 atende ao número de zeros iniciais exigido, se a transação ainda está pendente e se já existe um vencedor registrado. Após validar, o controlador publica o resultado e registra o vencedor na tabela de transações. Passado um intervalo determinado, no caso 20 segundos, o controlador gera um novo desafio, mantendo o ciclo ativo.

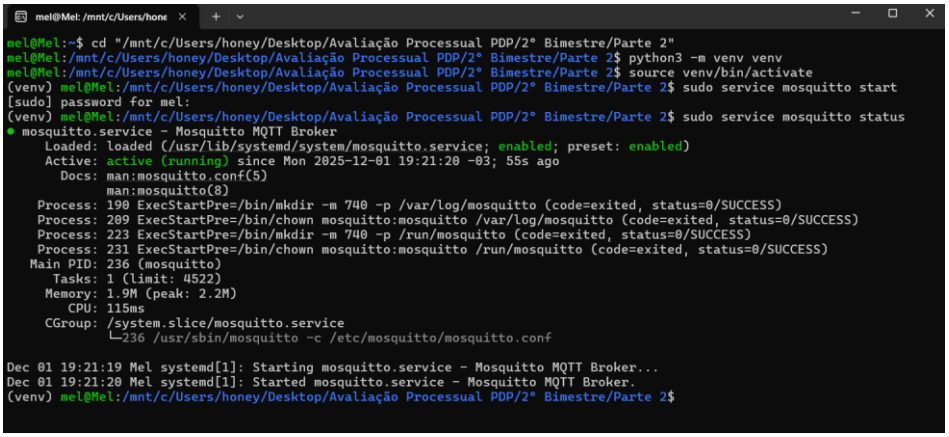
Os mineradores, por sua vez, aguardam a publicação de novos desafios e iniciam o processo de mineração utilizando processamento paralelo por meio da biblioteca ThreadPoolExecutor. A busca é interrompida assim que um dos threads encontra uma solução válida, evitando desperdício de recursos computacionais. Em seguida, o minerador envia sua solução ao controlador para validação.

Por fim, o sistema conta com uma tabela de transações, implementada com proteção por mutex para evitar condições de corrida. Essa tabela ajuda a organizar as informações passadas em cada terminal (atraso de alguma mensagem), como a dificuldade do desafio, a solução encontrada, o vencedor e o status da transação, sendo está pendente ou resolvida. O controlador exibe periodicamente o estado atualizado da tabela, permitindo acompanhar a evolução das tarefas de mineração e a distribuição dos resultados entre os nós participantes.

3. TESTES

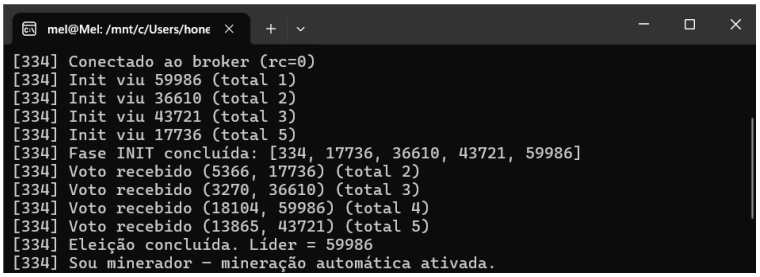
Nos testes de inicialização, cada nó passou a anunciar sua presença publicando periodicamente no tópico sd/init. O sistema mantém uma lista dos participantes que já responderam e finaliza essa etapa quando o total esperado é alcançado. Na prática, ao iniciar conjuntos com 5 ou 10 nós, sendo que no código o mínimo é 5, todos exibiram em console a lista de participantes detectados naquele ciclo. Quando houve atraso ou perda de mensagens, o envio se repetiu até o tempo limite configurado, o que permitiu ao sistema seguir adiante mesmo sem todos os nós responderem dentro do período ideal.

Figura 1 - Inicialização do Broker Mosquitto no WSL.



Na fase de eleição, cada nó gerou um valor aleatório e o publicou no tópico sd/voting. O maior voto determina o controlador. Quando todos os nós foram iniciados ao mesmo tempo, o líder foi escolhido corretamente em todas as execuções. Vale registrar que, se o líder fosse encerrado manualmente, os demais nós permaneciam operando, mas não havia qualquer reeleição automática, para eleger um novo líder, era necessário reiniciar todo o conjunto.

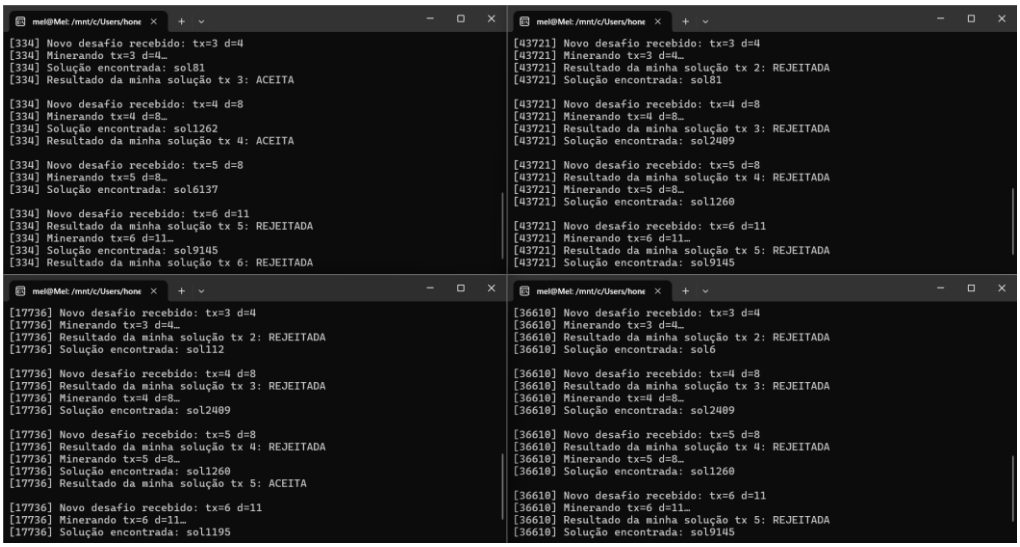
Figura 2 - Fase do Init e Eleição no minerador.



A mineração foi conduzida pelos nós mineradores, cada um utilizando múltiplas threads em paralelo. O tempo de mineração diminuiu à medida que o número de threads aumentou, mostrando vantagem clara da paralelização local. Quando mais de um minerador estava ativo, a competição era simultânea, porém apenas a primeira solução

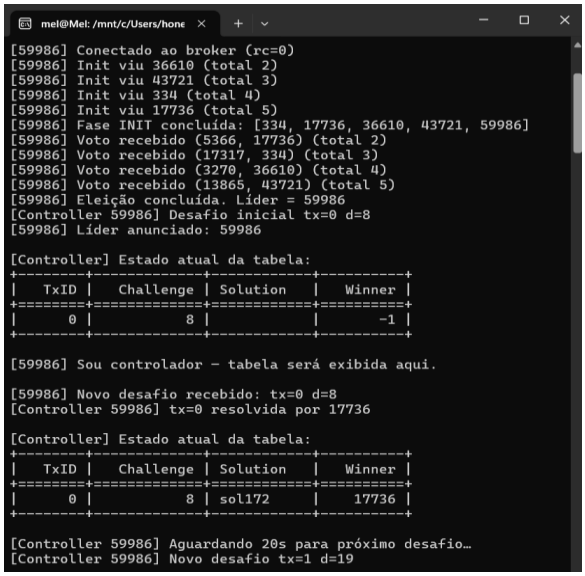
válida recebida pelo controlador era aceita. Para dificuldades mais altas, entre 15 e 20, observou-se que nenhum minerador conseguia encontrar um valor dentro do tempo limite, levando o controlador a emitir mensagens de timeout de forma recorrente.

Figura 3 - Execução dos mineradores.



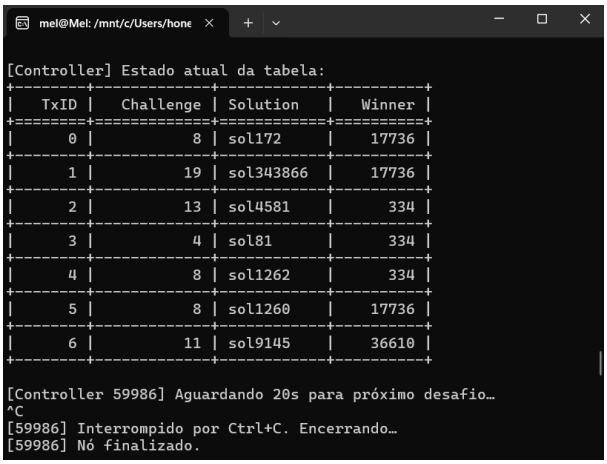
Nos testes focados na TransactionTable e na comunicação, o controlador manteve a tabela protegida por mutex, garantindo consistência durante todo o processo. Para cada desafio, apenas um vencedor era registrado. Caso diferentes mineradores enviassem soluções válidas praticamente no mesmo instante, somente a primeira era armazenada; as demais eram descartadas. A tabela exibida no controlador mostrava corretamente o status das transações, indicando winner = -1 para entradas ainda pendentes e o identificador do nó vencedor quando uma solução era aceita.

Figura 4 - Fase do Init, Eleição, lançamento de desafios e estados da tabela no controlador.



Por fim, os testes de robustez mostraram que desligar um minerador, utilizando Ctrl+C, no meio da execução não afetava o funcionamento geral, apenas interrompia a possibilidade de aquele nó enviar uma solução. Já no caso do controlador ser reiniciado, a eleição precisava ser refeita, pois o sistema ainda não conta com um mecanismo automático de recuperação da liderança.

Figura 5 - Desligando o controlador utilizando Ctrl+C.



5. RESULTADOS ENCONTRADOS

Os testes realizados mostraram que o sistema conseguiu manter uma comunicação estável entre os nós usando o broker Mosquitto, executado no WSL na porta 1883. A etapa de inicialização ocorreu sem surpresas, os nós identificaram quem deveria participar e seguiram para a eleição sem problemas relevantes. Em todas as execuções, observou-se a escolha de um único líder, o que indica que a implementação do algoritmo Bully está funcionando de maneira satisfatória. Depois disso, os mineradores passaram a receber os desafios enviados pelo controlador e trabalharam de forma coordenada pela rede MQTT.

A mineração em paralelo apresentou ganhos claros de desempenho. A cada aumento no número de threads, o tempo necessário para encontrar soluções válidas diminuía de forma evidente, mostrando que o sistema escala bem localmente. Quando havia vários mineradores atuando ao mesmo tempo, a disputa por quem chegaria primeiro foi consistente com o esperado, apenas a primeira solução válida recebida pelo controlador era aceita, conforme previsto na lógica da TransactionTable. Isso evitou conflitos e garantiu que cada transação tivesse um único vencedor.

A TransactionTable permaneceu consistente ao longo de todos os testes, registrando corretamente a dificuldade dos desafios, as soluções consideradas válidas e o

identificador do nó vencedor. Quando nenhum minerador encontrava uma solução dentro do tempo máximo, o sistema simplesmente sinalizava o timeout e continuava operando normalmente, sem prejudicar os registros nem interromper o fluxo geral do processo. Esse comportamento reforça que os mecanismos de sincronização e controle de concorrência estão funcionando adequadamente.

De modo geral, a arquitetura desenvolvida mostrou-se funcional, modular e eficiente. A adoção do MQTT revelou-se adequada para a sincronização distribuída entre os nós, oferecendo baixo acoplamento e facilidade de integração. O broker Mosquitto, executado no WSL, provou ser uma solução prática e estável para a comunicação entre clientes, garantindo interoperabilidade e operação contínua durante os testes. A implementação do algoritmo Bully forneceu um método direto e confiável para a eleição de líder, assegurando a existência de um único controlador por execução e simplificando a coordenação dos desafios de mineração.