Programação Paralela e Distribuída 2025/2
Laboratório III – Comunicação Indireta Pub/Sub + Eleição e Coordenação

Prof Breno Krohling - brenokrohling@professor.multivix.edu.br

Objetivo

- Utilizar a implementação de sistemas de comunicação indireta por meio de middleware Publish/Subscribe (Pub/Sub) com Filas de Mensagens;
- Realizar uma eleição de coordenador em sistemas distribuídos por meio da troca de mensagens entre os participantes do sistema;
- Realizar votação sobre o estado de transações distribuídas por meio da troca de mensagens entre os participantes do sistema.

1 Especificação do Sistema

Você precisará construir um protótipo similar a resolvedor de provas de trabalho em um minerador de criptomoedas, com implementação baseada em comunicação indireta usando o modelo Publish/Subscriber e Filas de Mensagens. A implementação do minerador deverá ser realizada em Python com a biblioteca Paho[1] MQTT e o broker utilizado deverá ser o EMQX. Todo participante do sistema deverá implementar, simultaneamente, o papel do minerador e do controlador.

O processo controlador deverá ter o seguinte funcionamento:

1. Manter, enquanto estiver em execução, uma tabela com os seguintes registros:
   - TransactionID: Identificador da transação, representada por um valor inteiro;
   - Challenge: Valor do desafio criptográfico associado à transação, representado por um número [1..20], onde 1 é o desafio mais fácil. Gere desafios aleatórios ou sequenciais (experimente as diferentes abordagens);
   - Solution: String que, se aplicada a função de hashing SHA-1, solucionará o desafio criptográfico proposto;

[1] https://www.eclipse.org/paho/

- Winner: ClientID do usuário que solucionou o desafio criptográfico para a referida TransactionID (mesma linha da tabela). Enquanto o desafio não foi solucionado, considere que o ClientID = -1 (indicador de que o desafio ainda está pendente);

2. Ao carregar, deverá gerar um novo desafio com TransactionID = 0;

3. Assinar a seguinte fila de mensagens:

   - Nome: *sd/solution*
   - dados: ClientID <int>, TransactionID <int>, Solution <string>.
   - Solution usada pelo ClientID para resolver o desafio associado à TransactionID. Importante: sempre enviar/ receber em formato string codificada em JSON

4. Sempre que receber uma proposta de solução, o controlador deverá:

   - cerificar se a TransactionID está pendente e se a solução atende aos requisitos do desafio proposto;
   - Caso atenda, atualiza a tabela de estado das transações e publica o resultado na fila *sd/result*;
   - Caso não atenda, publica o resultado na fila sd/result informando que a solução foi rejeitada.

O papel de minerador deverá ter o funcionamento igual ao descrito no laboratório II, e seguir o seguintes passos:

1. Manter, enquanto estiver em execução, uma tabela com os seguintes registros:

   - TransactionID: Identificador da transação, representada por um valor inteiro;
   - Challenge: Valor do desafio criptográfico associado à transação, representado por um número [1..20], onde 1 é o desafio mais fácil. Gere desafios aleatórios ou sequenciais (experimente as diferentes abordagens);
   - Solution: String que, se aplicada a função de hashing SHA-1, solucionará o desafio criptográfico proposto;
   - Winner: ClientID do usuário que solucionou o desafio criptográfico para a referida TransactionID (mesma linha da tabela). Enquanto o desafio não foi solucionado, considere que o ClientID = -1 (indicador de que o desafio ainda está pendente);

2. Ao iniciar, essa tabela deverá estar vazia e o minerador deverá bloquear até receber uma mensagem na fila sd/challenge. Ao receber essa mensagem, deverá imprimir em tela a recepção da mensagem, com o respectivo TransactionId e o desafio correspondente;

3. Assinar as seguinte filas de mensagens no broker Pub/Sub:

   - Nome: *sd/challenge*
   - dados: TransactionID <int>, Challenge <int>.

- Publicado pelo controlador! Contém o novo TransactionID e o Challenge associado a ele <int>[1..20]. Importante: sempre enviar/ receber em formato string codificada em JSON!

- Nome: *sd/result*
- dados: ClientID <int>, TransactionID <int>, Solution <string>, Result <int>
- Publicado pelo controlador! Contém a resposta do controlador a respeito da submissão de uma solução para o respectivo TransactionID. Result = 0 caso a solução seja inválida, e Result != 0 caso a solução seja válida. Importante: sempre enviar/ receber em formato string codificada em JSON!

Na inicialização do sistema, todos os participantes do (processos) deverão realizar uma eleição para definição do controlador (1) e dos mineradores (2).

2 Eleição Distribuída

A Figura 1 mostra os 3 (três) estados iniciais do sistema, que deverá ser implementado em cada participante.

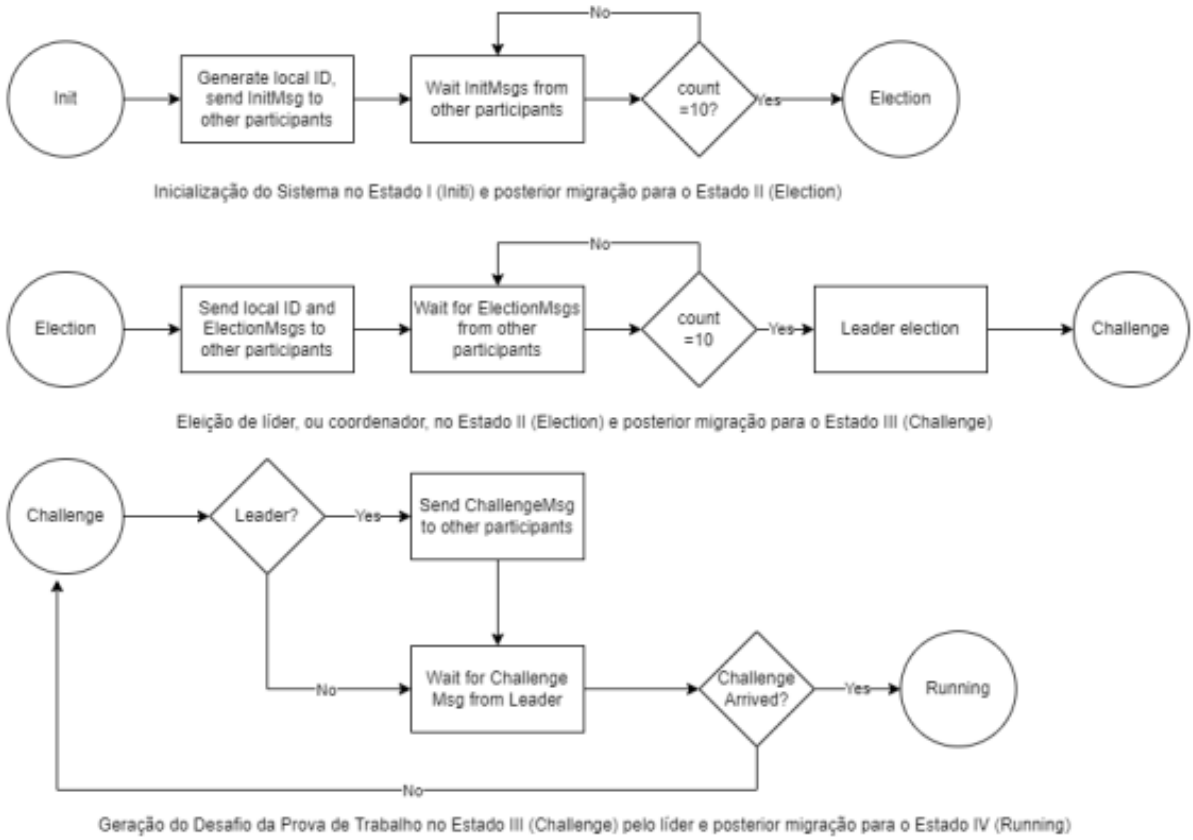

Figura 1: Figura 1: Eleição e Coordenaçao distribuída

De acordo com a Figura 1, o sistema deverá inicializar sempre no estado Init, onde cada participante terá o seu identificador local (ClientID). Esse valor deverá ser gerado aleatoriamente num intervalo de 16 bits [0..65335]. Após gerar o seu ClientID, o nó participante deverá enviá-lo para os demais por meio da publicação de uma mensagem de registro InitMsg na fila sd/init no broker de comunicação. A mensagem InitMsg tem a formatação a seguir:

- sd/init
- ClientID <int>
- Contém o ClientID de 16 bits gerado aleatoriamente pelo processo participante do sistema no estado de inicialização (Init). Importante: sempre enviar/ receber em formato string codificada em JSON!

Ainda de acordo com a Figura 1, após o envio da sua própria mensagem de inicialização, o processo deverá aguardar a chegada de um total de n (n=10 na figura) mensagens de inicialização distintas, provenientes dos demais participantes – importante, n deverá ser passado como parâmetro de inicialização do processo. Ao receber um total de n-1 mensagens do tipo InitMsg na fila sd/init, encerra-se a fase de inicialização. O processo poderá reenviar sua própria mensagem de inicialização enquanto as n demais mensagens não forem recebidas – esse reenvio é de definição livre.
Após concluir a fase de inicialização (Init), tem-se inicio a fase de eleição do coordenador (ou líder) do sistema (Election). Nesta fase cada processo deverá gerar um número aleatório de 16 bits (VoteID) e enviar ao grupo por meio de uma mensagem de eleição (ElectionMsg) na fila sd/election, contendo seu ClientID e seu "voto" (número aleatório) na mensagem de eleição. Após enviar a ElectionMsg com seu VoteID, cada nó deverá aguardar o recebimento dos votos dos demais participantes do sistema (count=10 na Figura 1). Após receber todos os votos, deve-se escolher o participante com maior VoteID como líder do grupo e encerrar essa fase. Use uma combinação de VoteID + ClientID para decidir sobre eventuais desempates entre os participantes (maior ClientID terá prioridade em caso de empate). Assuma que todos os participantes são honestos e gerarão números realmente aleatórios e aceitarão o resultado da eleição.

- sd/voting
- ClientID <int>; VoteID <int>
- Contém o ClientID de 16 bits gerado aleatoriamente pelo processo participante do sistema no estado de inicialização (Init). Contém, também, o voto VoteID, de 16 bits, gerado aleatoriamente. Importante: sempre enviar/ receber em formato string codificada em JSON!

Considerando que não há perda de mensagens no sistema, todos os participantes estarão sincronizados após a eleição e assumirão o estado de desafio (Challenge). O líder eleito pode assumir o papel de controlador e, então, definir um desafio e enviar para os demais participantes do sistema.
Como requisitos e premissas, em seu projeto assuma que:

- O número de participantes é fixo e conhecido;
- Não haverá entrada/saída dinâmica de nós no sistema (churn);
- O broker Pub/Sub é único, infalível, e de conhecimento prévio por todos os participantes;
- Os nós participantes do sistema são honestos, não havendo comportamento malicioso para atrapalhar o funcionamento do sistema;

3 Instruções de envio

- O trabalho pode ser feito em grupos de 2 ou 3 alunos.
- Email: **brenokrohling@professor.multivix.edu.br**
- Assunto: **PPD-CI-{CODIGO_TURMA}**, se atentando para substituir **{CODIGO-TURMA}** pelo código da sua turma.
- Deve-se enviar via email apenas o link para o repositório virtual da atividade (Github, Bitbucket, Google Colaboratory ou similares) contendo:
  1. Códigos-fonte;
  2. Instruções para execução;
  3. Relatório técnico (.pdf ou README)
  4. Vídeo curto (máx 5 min) mostrando uma execução, resultado e análise e/ou explicação dos resultados encontrados quando necessário;
- O relatório técnico deverá conter: a metodologia de implementação, testes e resultados encontrados;
- O relatório não deverá conter: trechos explícitos de código, apenas explicações gerais quando necessário.
- Data de Entrega: 01 de Dezembro de 2025.

Bom trabalho!