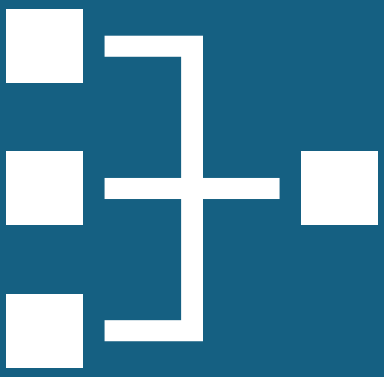
ALGORITMO DE BULLY

SIMULADOR DE USINAS



Igor Rochelle Ferreira

Lucas Righetto Correa

Matheus Hortiz de Moraes

EEP Escola de Engenharia de Piracicaba

Sumário

Sumário de Imagens

1.Introdução

Como parte da atividade proposta pelo professor Martins, na disciplina de Sistemas Distribuídos, foi apresentada aos alunos uma lista de temas para suporte à elaboração de um projeto prático. O objetivo dessa atividade é desenvolver uma aplicação capaz de simular situações reais envolvendo comunicação cliente-servidor, seguindo regras específicas definidas pelo docente, de modo a reforçar a compreensão dos conceitos estudados em sala de aula.

O tema escolhido pelo grupo foi a implementação do algoritmo de Bully, um dos mecanismos de eleição de líder discutidos pelo professor durante o conteúdo teórico da disciplina. A escolha baseia-se tanto na relevância do algoritmo dentro do contexto de sistemas distribuídos quanto na possibilidade de demonstrar, na prática, como ocorre a eleição de um coordenador em cenários com múltiplos processos ou usuários.

Inspirados em uma atividade realizada no semestre anterior, o grupo optou por aplicar o algoritmo em um simulador industrial. Assim, o projeto consiste no desenvolvimento de um sistema de monitoramento e controle de mosto em usinas, no qual usuários interagem com um painel que exibe valores fornecidos pelo servidor referente a variáveis como temperatura, pH, adição de água e adição de cal. Contudo, apenas o usuário eleito como coordenador — escolhido pelo algoritmo de Bully — possui permissão para manipular os controles do tanque. Os demais usuários têm acesso apenas ao monitoramento das informações, sem autorização para alterar os parâmetros do sistema.

Dessa forma, este projeto visa promover não apenas a capacitação técnica dos integrantes, mas também o desenvolvimento de competências relacionadas ao planejamento, implementação, documentação e validação de soluções distribuídas. Além disso, busca reforçar a compreensão prática dos mecanismos de coordenação, tolerância a falhas e gerenciamento de concorrência dentro de um ambiente controlado, aproximando o aprendizado teórico da realidade de processos industriais automatizados.

2. Fundamentação Teórica

2.1. Algoritmo de Bully

O algoritmo de Bully, proposto por Héctor García-Molina (em 1982), é um método de eleição de líder utilizado em sistemas distribuídos. Ele assume que cada processo possui um identificador único (ID) e conhece os IDs de todos os demais processos. Quando o coordenador atual falha ou deixa de responder, qualquer processo pode iniciar uma eleição enviando mensagens para os processos com IDs maiores que o seu.

Se nenhum processo com ID superior responder, o iniciador assume o papel de coordenador. Caso contrário, o processo de maior ID ativo torna-se automaticamente o novo administrador. A lógica central do algoritmo garante que sempre exista um líder válido e operacional, evitando falhas de coordenação no sistema.

2.2. Sistemas distribuídos

Sistemas distribuídos são formados por vários computadores independentes que, por meio de comunicação via troca de mensagens, atuam de forma integrada e se apresentam ao usuário como um único sistema. Essa integração geralmente é realizada por um middleware, que funciona como uma camada de software responsável por unificar e coordenar a comunicação entre as máquinas.

Entre as características principais desse tipo de sistema estão: concorrência, permitindo a execução simultânea de processos; inexistência de um relógio global, já que a sincronização entre máquinas depende apenas da troca de mensagens; e falhas independentes, pois cada componente pode falhar sem interromper o funcionamento dos demais. Esses conceitos são essenciais para o entendimento de algoritmos de coordenação e tolerância a falhas utilizados em ambientes distribuídos.

2.3. Controle e simulação de processos industriais

O controle de mosto em processos industriais envolve a análise e o monitoramento de variáveis químicas fundamentais, como temperatura, acidez (pH) e proporções de reagentes. Essas variáveis influenciam diretamente a eficiência do processamento e a qualidade do produto, exigindo métodos técnicos precisos para manter o sistema dentro de parâmetros adequados.

Ajustes inadequados podem gerar reações indesejadas, afetando a estabilidade da mistura e comprometendo a segurança operacional. Nesse contexto, o uso de modelos matemáticos e equações baseadas em princípios da química aplicada permite representar de forma mais fiel o comportamento real do sistema, contribuindo para simulações e análises mais confiáveis.

Além disso, ambientes industriais que manipulam substâncias de alto valor ou potencialmente perigosas demandam rigorosos protocolos de segurança. Por isso, a modelagem das variáveis deve considerar limites operacionais seguros, mecanismos de estabilização e respostas a falhas. A implementação de sistemas de monitoramento baseados em variáveis químicas contribui, assim, para o controle preventivo de riscos, garantindo maior precisão e segurança em todas as etapas do processo.

2.4. Comunicação cliente-servidor

3.Desenvolvimento

3.1. Front-End

Com o objetivo de conferir maior realismo ao projeto, analisamos diversas imagens de painéis de controle — tanto de usinas açucareiras, que serviram de referência principal, quanto de outros tipos de usinas, como usinas de geração de energia elétrica. Essa análise revelou padrões recorrentes e singularidades: interfaces simples, voltadas estritamente para a utilidade, com poucos botões e displays; priorização dos elementos necessários à operação das máquinas; e ênfase no monitoramento para verificar se os processos estavam ocorrendo conforme esperado.

Figura 1 - Painel de Usina 1

Figura 2 - Painel de Usina 2

https://www.kdmsteel.com/pt/painel-de-controle-da-usina-de-energia/

A partir dessas observações definimos a estética e o comportamento da interface. Optamos por um fundo padrão em tons de cinza; valores numéricos são exibidos em um fundo preto com fonte verde; e sinais de alerta/estado são representados por LEDs, de modo a permitir identificação rápida de situações de atenção. Essas escolhas visuais foram pensadas para reforçar clareza e legibilidade em ambientes operacionais.

Figura 3 - Painel de Usina 3

https://www.youtube.com/watch?v=xryMaF0eoI0

No que diz respeito à apresentação das informações ao usuário, a interface foi projetada para exibir os valores de controle essenciais: volume do mosto (em litros), temperatura em graus Celsius, pH do mosto e um prompt de comandos que informa ao usuário — especialmente ao administrador (ADM) — as ações que estão sendo executadas pelo sistema. A organização dos elementos visa facilitar a leitura rápida dos dados críticos durante a operação.

Por se tratar de um sistema que incorpora o algoritmo de Bully, também foi necessário representar visualmente regras de acesso e permissões. Quando um usuário não está apto a utilizar determinadas funções (por exemplo, por não ser administrador), isso é indicado visualmente: os botões perdem parte de sua saturação e ficam indisponíveis para uso. Além dessa sinalização visual, o sistema exibe uma mensagem na área inferior da interface informando que o usuário não possui privilégios de administrador.

O desenvolvimento do protótipo seguiu etapas iterativas. A primeira versão foi criada em Microsoft PowerPoint — ferramenta escolhida pela rapidez para prototipação visual e capacidade de edição gráfica. Esse protótipo contemplava as ações básicas do usuário e o padrão de cores inicial, que foi posteriormente ajustado; embora a versão em PowerPoint possuísse recursos visuais limitados, ela já consolidava o padrão definido pelo grupo.

Figura 4 - Esboço em Power Point

Em seguida, desenvolvemos o primeiro protótipo em Java. Essa versão incorporou um simulador de servidor, permitindo testar a eficiência do sistema e avaliar a experiência do usuário. O protótipo em Java mantinha um padrão de cores similar ao definido no PowerPoint e teve como finalidade principal o refinamento das interfaces e a validação prévia das funcionalidades antes da integração com o algoritmo de Bully em ambiente de servidor.

Figura 5 - Teste Beta do Simulador

A versão final do visualizada do algoritmo de Bully foi obtida ao integrar a interface com o back end. Nessa nova etapa, além das informações já exibidas nas versões anteriores, o usuário passou a dispor de controles adicionais: botões para conectar e desconectar do servidor; opção para acionar a eleição que define o administrador; e campos para seleção do IP e da porta do servidor, necessários para estabelecer a conexão. Essa integração consolidou tanto a usabilidade quanto as funcionalidades essenciais do sistema.

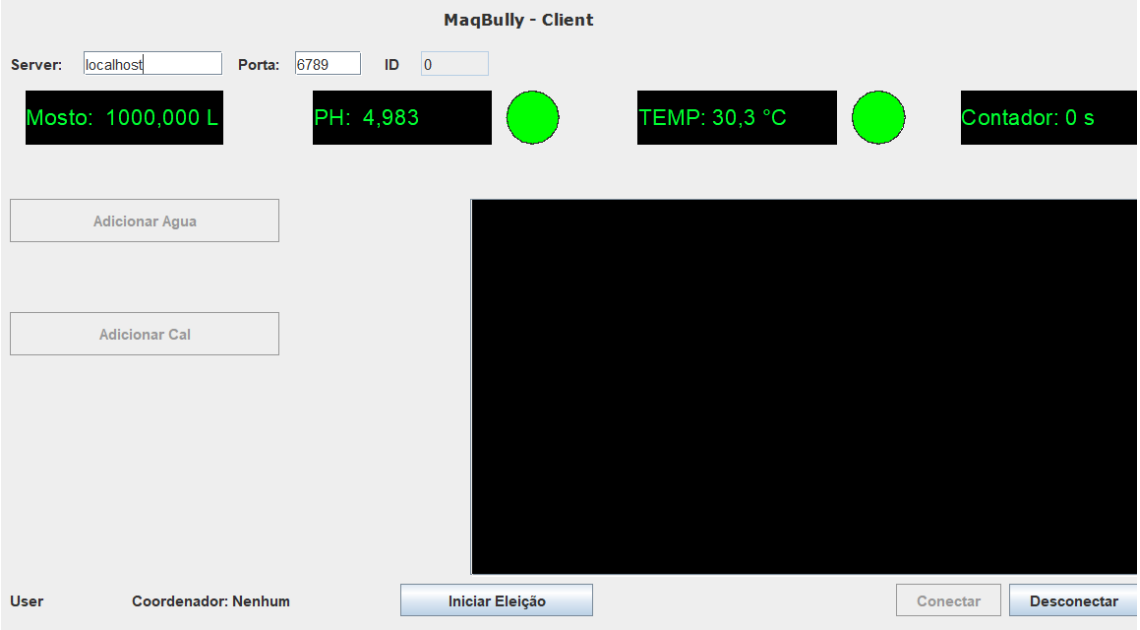


Figura 6 - Versão final do Simulador

3.2. Back-end

O back-end do sistema foi desenvolvido em linguagem Java, com o objetivo de implementar uma arquitetura de sistemas distribuídos baseada em comunicação cliente-servidor utilizando sockets TCP. O projeto simula um ambiente industrial de controle de mosto, no qual múltiplos clientes comunicam-se com um servidor central. A aplicação utiliza o algoritmo Bully para eleição de um coordenador entre os clientes, responsável pelo controle das variáveis físico-químicas do sistema, como pH, temperatura e volume do mosto.

3.2.1 Estrutura geral de comunicação

A comunicação entre os componentes é realizada por meio de mensagens de texto estruturadas, separadas pelo caractere |. Cada mensagem contém um tipo (como JOIN, ELECTION, COORDINATOR, STATE, etc.) e, opcionalmente, parâmetros adicionais. O servidor e os clientes processam essas mensagens de maneira cooperativa para manter a consistência do sistema distribuído.

A arquitetura é composta pelos seguintes módulos principais:

Servidor (BullyServer e classes TCP associadas): responsável por gerenciar conexões de clientes, coordenar a comunicação, transmitir estados do mosto e supervisionar o processo de eleição de coordenador.

Clientes (BullyClient1 e classes TCP associadas): representam os nós participantes do sistema distribuído, enviando e recebendo mensagens do servidor, participando de eleições e executando comandos de controle quando eleitos coordenadores.

3.2.2 Funcionamento do servidor

A classe Bully_TCPServerAtivosMain atua como o núcleo de execução do servidor. Ela inicializa um ServerSocket responsável por escutar conexões TCP em uma porta específica. Cada novo cliente conectado é encapsulado em um objeto Bully_TCPServerConnection, que armazena os fluxos de entrada (BufferedReader) e saída (PrintWriter) associados ao socket do cliente.

Para cada cliente, é criada uma thread dedicada da classe Bully_TCPServerAtivosHandler, que recebe e interpreta as mensagens enviadas pelos clientes. As mensagens podem ter diferentes propósitos:

JOIN: registra o cliente no servidor, associando seu identificador;

ELECTION, ALIVE, COORDINATOR: repassam mensagens do algoritmo Bully entre os nós;

ADD_WATER e ADD_LIME: comandos que ajustam o estado do mosto, alterando volume, pH e temperatura;

PONG: resposta do coordenador a um sinal de verificação de vida enviado pelo servidor.

O servidor mantém uma lista de clientes ativos e periodicamente executa uma verificação de heartbeat (batimento cardíaco) por meio do método heartbeatTick(). Caso o coordenador atual não responda dentro do intervalo definido, o servidor dispara automaticamente uma nova eleição enviando a mensagem TRIGGER_ELECTION a todos os clientes.

O estado físico-químico do mosto é atualizado periodicamente em uma thread de simulação, e o servidor transmite essas atualizações a todos os clientes com mensagens do tipo STATE. Esse processo é controlado por agendadores (ScheduledExecutorService) que garantem a atualização contínua e o monitoramento do sistema.

3.2.3 Funcionamento do cliente

Cada cliente é implementado na classe BullyClient1, que gerencia a comunicação com o servidor por meio das classes auxiliares Bully_TCPClientMain e Bully_TCPClientHandler.

A classe Bully_TCPClientMain cria a conexão TCP com o servidor (Socket), inicializa o fluxo de saída para envio de mensagens e instancia o Bully_TCPClientHandler, que atua em uma thread separada para leitura contínua das mensagens recebidas.

A thread Bully_TCPClientHandler interpreta as mensagens do servidor e executa ações conforme o tipo:

UPDATE_CLIENTS: atualiza a lista de clientes conectados;

ELECTION e ALIVE: tratam mensagens do processo de eleição Bully;

COORDINATOR: define o novo coordenador;

STATE: atualiza os valores locais de pH, temperatura e volume;

PING: recebe solicitações de heartbeat enviadas pelo servidor e as repassa para verificação;

ERROR: encerra a conexão em caso de falha crítica.

O cliente eleito como coordenador assume funções de controle, enviando mensagens ADD_WATER e ADD_LIME para o servidor, que aplica as mudanças e propaga o novo estado para todos os participantes.

3.2.4 Protocolo de mensagens

O protocolo de comunicação é textual e síncrono, sendo cada linha uma mensagem completa. A estrutura geral segue o formato:

TIPO|param1|param2|...|

Exemplos:

JOIN|3| → cliente com ID 3 se conecta;

ELECTION|5| → cliente 5 inicia eleição;

COORDINATOR|2| → cliente 2 é eleito coordenador;

STATE|980.0|5.2|29.8|timestamp| → estado físico atualizado;

ERROR|DUPLICATE_ID|ID 3 já está em uso| → erro de identificação duplicada.

Esse formato simples e padronizado permite a interoperabilidade entre diferentes componentes e facilita o rastreamento das mensagens em logs de depuração.

3.2.5 Integração e controle do mosto

A parte física simulada — volume, pH e temperatura — é controlada no servidor (Bully_TCPServerAtivosMain) por meio de funções que modelam reações químicas e variações térmicas. Quando o coordenador adiciona água (ADD_WATER) ou cal (ADD_LIME), o servidor recalcula as propriedades do mosto e envia o novo estado aos clientes.

Esse comportamento simula um processo industrial supervisionado, onde apenas o coordenador (eleito via Bully) tem permissão para ajustar as variáveis do sistema.

3.2.6. Variáveis Físico-Químicas do Mosto

No contexto do projeto, o mosto é tratado como um meio líquido cuja qualidade depende do equilíbrio entre **volume**, **pH** e **temperatura** — variáveis físico-químicas que são

monitoradas e ajustadas dinamicamente pelo servidor, sob comando do coordenador eleito.

As variáveis simuladas são:

a) Volume (L)

O **volume** representa a quantidade total de líquido presente no tanque do mosto, medido em litros. Essa variável é alterada quando o coordenador envia o comando `ADD_WATER`, indicando a adição de água ao sistema.
O método `applyAddWater(double aguaL)` recalcula o novo volume total, levando em conta o volume anterior e a quantidade adicionada. O aumento do volume implica também em uma **diluição da acidez** e **redução proporcional da temperatura**, devido à mistura com água fria.

O cálculo da nova temperatura média, por exemplo, é realizado pela equação:

$$T_{novo} = \frac{V_{antigo} \cdot T_{antigo} + V_{água} \cdot T_{água}}{V_{total}}$$

onde $T_{água}$ é fixado em 15 °C.

b) pH

O **pH** é o principal indicador da acidez do mosto, representando a concentração de íons hidrogênio (H^+) na solução. No projeto, o pH é calculado a partir de uma variável auxiliar chamada `simAcidez`, que representa a **acidez total em gramas por litro**.
Quando o coordenador adiciona **cal hidratada ($Ca(OH)_2$)** por meio do comando `ADD_LIME`, ocorre uma reação química de neutralização que **reduz a acidez** e, consequentemente, **aumenta o pH** do mosto. Essa alteração é modelada matematicamente pela expressão:

$$\Delta A = \frac{m_{cal}}{0.617 \cdot V}$$

onde m_{cal} é a massa de cal em gramas, V o volume total e o fator 0.617 representa a relação estequiométrica aproximada entre a cal e o ácido presente.

O valor final de pH é limitado entre 2,0 e 7,0, simulando condições realistas de fermentação e evitando valores fisicamente impossíveis.

c) Temperatura (°C)

A **temperatura** do mosto é um parâmetro crítico para o processo de fermentação, pois influencia diretamente a atividade enzimática e a sobrevivência das leveduras.
No sistema, a temperatura sofre **variações lentas e contínuas** ao longo do tempo,

simuladas pela variável `TEMP_DRIFT_PER_SEC`. Essa deriva representa o aumento natural de temperatura devido à fermentação e reações exotérmicas.

A adição de cal, por outro lado, provoca uma **redução leve na temperatura**, modelada pelo coeficiente `CAL_TEMP_COEFF`, que diminui a temperatura proporcionalmente à quantidade de cal adicionada.

O sistema também implementa uma **verificação de segurança térmica** por meio dos métodos `checkTempRed()` e `checkPHRed()`. Caso a temperatura ultrapasse os limites de 45 °C ou caia abaixo de 20 °C, ou se o pH sair da faixa segura (3,8–6,0), o servidor gera **alertas de risco** e contabiliza o tempo em que o sistema permanece em estado crítico (`riscoCounter`). Após 60 s de condição adversa, é registrado um aviso de **possível dano ao mosto**, simulando um cenário de falha operacional em um ambiente industrial real.

3.2.7 Conclusão

O back-end do projeto demonstra uma implementação prática dos conceitos de sistemas distribuídos, comunicação via sockets TCP e coordenação descentralizada por eleição. O uso do algoritmo Bully assegura a resiliência do sistema, permitindo a substituição automática do coordenador em caso de falha. Além disso, a integração entre threads, mensagens padronizadas e simulação de estado proporciona um ambiente funcional para experimentação e estudo de tolerância a falhas, concorrência e consistência em redes distribuídas.

4.Resultados e Análises

Durante a execução do projeto, foi possível simular de forma eficiente o funcionamento do algoritmo de Bully dentro do contexto de um sistema distribuído voltado ao controle de processos industriais. O sistema desenvolvido permite a interação de múltiplos usuários conectados a um servidor, representando diferentes processos concorrentes que disputam a coordenação do sistema.

A aplicação demonstrou corretamente o mecanismo de eleição de líder. Quando o coordenador principal é desconectado ou sofre uma falha simulada, o algoritmo entra em ação, identificando os processos ativos e elegendo aquele com maior identificador (ID) como novo coordenador. Esse comportamento foi validado por meio de testes práticos, observando-se que as mensagens de eleição e confirmação foram trocadas de acordo com as etapas previstas no algoritmo original de García-Molina.

No contexto do simulador de usina, o coordenador eleito obteve controle exclusivo sobre os parâmetros de operação do tanque, podendo alterar valores de temperatura, pH, adição de água e cal. Os demais usuários permaneceram em modo de monitoramento, recebendo atualizações em tempo real sem capacidade de intervenção. Essa estrutura reproduziu fielmente o comportamento esperado em sistemas distribuídos reais, nos quais o controle centralizado evita conflitos de comando e inconsistências nos dados.

Além disso, a integração entre front-end e back-end demonstrou boa estabilidade, permitindo comunicação contínua entre cliente e servidor. A interface apresentou os valores de forma dinâmica, facilitando a visualização dos estados e alterações provocadas pelo coordenador. O tempo de resposta do sistema mostrou-se adequado, e o mecanismo de eleição ocorreu de forma rápida e consistente, confirmando a eficiência do algoritmo implementado.

Por fim, os testes evidenciaram a importância do controle de concorrência e da recuperação de falhas em sistemas distribuídos. Mesmo após interrupções simuladas, o sistema foi capaz de restabelecer a coordenação automaticamente, reforçando a robustez do modelo implementado.

5.Conclusão

A implementação do algoritmo de Bully no simulador de usinas proporcionou uma compreensão prática e aprofundada sobre os mecanismos de coordenação em sistemas distribuídos. O projeto evidenciou a relevância de algoritmos de eleição para garantir a continuidade e a consistência de operações em ambientes com múltiplos processos concorrentes.

A simulação demonstrou, de forma clara, como o algoritmo identifica e substitui coordenadores em caso de falhas, assegurando que o sistema mantenha um controle central ativo. A aplicação prática dos conceitos estudados em sala, especialmente comunicação cliente-servidor, tolerância a falhas e sincronização, que contribuiu significativamente para o aprendizado e fixação do conteúdo teórico.

Além do aprendizado técnico, o trabalho promoveu o desenvolvimento de competências colaborativas entre os integrantes do grupo, envolvendo planejamento, divisão de tarefas e integração de diferentes camadas de software.

Resumidamente, o projeto atingiu seus objetivos ao aliar teoria e prática, comprovando a eficiência do algoritmo de Bully e sua aplicabilidade em cenários de controle industrial e sistemas distribuídos.

Referências

GARCIA-MOLINA, Hector. Bully Algorithm. Disponível em: https://homepage.divms.uiowa.edu/~ghosh/Bully.pdf . Acesso em: 6 jan. 2025.