Implementação de Algoritmos Distribuídos

Atividade 2

Brenda Martinez

Análise e Desenvolvimento de Sistemas

março/2025

Sumário

[Tecnologias 1](#_Toc192025500)

[Questão 1 – Clocks e Sincronização de Tempo 2](#_Toc192025501)

[Como Executar o Sistema (Windows) 2](#_Toc192025502)

[Explicação do Código 2](#_Toc192025503)

[Questão 2 – Estado Global e Captura de Estado 4](#_Toc192025504)

[Como Executar o Sistema (Windows) 4](#_Toc192025505)

[Explicação do Código 4](#_Toc192025506)

[Questão 3 – Algoritmos de Eleição - Bully 7](#_Toc192025507)

[Como Executar o Sistema (Windows) 7](#_Toc192025508)

[Explicação do Código 8](#_Toc192025509)

[Questão 4 – Detecção de Falhas em Sistemas Distribuídos 10](#_Toc192025510)

[Como Executar o Sistema (Windows) 10](#_Toc192025511)

[Explicação do Código 12](#_Toc192025512)

# Tecnologias

* Essa documentação foi produzida utilizando o sistema operacional **WINDOWS**.
* Linguagem: Python 3
* Bibliotecas: time, random, socket, threading, sys
* Instalando uma biblioteca python (Windows):
* Necessário ter o python previamente instalado
* Abrir o CMD e executar o comando “pip install {nome da biblioteca}”  
    
  Texto

  Descrição gerada automaticamente

# Questão 1 – Clocks e Sincronização de Tempo

## Como Executar o Sistema (Windows)

1. Abra o CMD do Windows e entre na pasta onde está o arquivo “lamport\_simulation.py”.
2. Execute o sistema utilizando o comando “python lamport\_simulation.py”  
     
   Texto

   O conteúdo gerado por IA pode estar incorreto.

## Explicação do Código

#### Relógio de Lamport

O código para esse sistema é composto de dois arquivos: o arquivo *LamportClock.py*, que define a classe LamportClock: uma representação do relógio de Lamport; e o arquivo *lamport\_simulation.py*, que realiza a simulação do sistema de disparo de eventos utilizando o relógio de Lamport.

#### LamportClock.py

Essa classe representa o relógio de Lamport. Ela simula um contador lógico em um nó de um sistema distribuído. A explicação para cada linha do código consiste nos comentários da imagem abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.

#### lamport\_simulation.py

Nesse arquivo, é executada a simulação da troca de mensagens entre 2 nós aleatórios. Seu objetivo é simular a execução de eventos em um sistema distribuído onde três nós utilizam Relógios de Lamport para manter a ordenação dos eventos.

A explicação para cada linha do código consiste nos comentários da imagem abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.

# Questão 2 – Estado Global e Captura de Estado

## Como Executar o Sistema (Windows)

1. Abra o CMD do Windows e entre na pasta onde está o arquivo “chandy\_lamport\_snapshot\_sim.py”.
2. Execute o sistema utilizando o comando “python chandy\_lamport\_snapshot\_sim.py”

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

## Explicação do Código

#### Algoritmo de Chandy-Lamport

O código para esse sistema também é composto por dois arquivos: o arquivo *JogadorProcess.py*, que define a classe JogadorProcess: uma simulação de um processo, utilizando algumas características de um jogador em um jogo, conforme exemplo dado em sala de aula; e o arquivo *chandy\_lamport\_snapshot\_sim.py*, que realiza a simulação de uma troca de mensagens entre jogadores (processos) antes e após o início de um snapshot, ou captura de estado global.

#### JogadorProcess.py

Essa classe define um processo, que, no contexto desse algoritmo, simula um jogador em um servidor de jogos online. A escolha da simulação ser feita baseada em um jogador, deve-se a melhor compreensão da troca de mensagens e o armazenamento do estado global, pois utilizando uma abstração, a compreensão e explicação do código tornou-se complicada e prolixa.

No início da classe, são definidos os atributos que aquele processo contém. Entre esses atributos estão o identificador do processo, o estado do processo, os canais de conexão, o estado global, e variáveis que armazenam as mensagens recebidas.

Após as definições dos atributos de um jogador, foram implementados os métodos necessários e comuns para o funcionamento do algoritmo de Chandy-Lamport, que propõe a captura do estado em um determinado momento por um dos processos ativos.

A explicação para cada linha do código consiste nos comentários da imagem abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

#### chandy\_lamport\_snapshot\_sim.py

Nesse arquivo, é executada a simulação da troca de mensagens entre os 3 jogadores. Seu objetivo é simular a troca de mensagens (Estado Global) antes e após o início do snapshot, em um sistema distribuído onde três processos comunicam-se.

A explicação para cada pedaço do código consiste nos comentários da imagem abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

# Questão 3 – Algoritmos de Eleição - Bully

## Como Executar o Sistema (Windows)

1. Abra o CMD do Windows e entre na pasta onde está o arquivo “bully\_sim.py”.
2. Execute o sistema utilizando o comando “python bully\_sim.py”

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

## Explicação do Código

#### Algoritmo de Eleição de Bully

O código para esse sistema também é composto por dois arquivos: o arquivo *Process.py*, que define a classe Process: uma representação de um processo em um sistema distribuído; e o arquivo *bully\_sim.py*, que realiza a simulação de uma eleição entre processos, bem como as simulações de falha e recuperação do nó líder.

#### Process.py

Essa classe define um processo, que possui um ID único e armazena uma referência da lista de processos em um sistema distribuído. Dentro dessa classe, estão definidos também os métodos necessários para a eleição, falha e recuperação do nó.

A explicação para cada linha do código consiste nos comentários das imagens abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

#### bully\_sim.py

Nesse arquivo acontece a simulação do algoritmo de eleição de bully. Primeiro, 5 processos são criados, para eficiência na visualização dos resultados. Após a criação dos processos, um nó aleatório convoca uma eleição e o nó com maior ID é eleito.

Para simular a falha e a recuperação do nó líder, uma variável “leader” foi criada para armazenar a informação de quem ganhou a eleição. Então, são chamadas as funções “fail” e “recover”, a fim de completar a simulação dos erros no nó líder, onde ambas funções convocam novas eleições.

A explicação para cada pedaço do código consiste nos comentários da imagem abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.

# Questão 4 – Detecção de Falhas em Sistemas Distribuídos

## Como Executar o Sistema (Windows)

1. Abra o CMD do Windows e entre na pasta onde está o arquivo “monitor.py”.  
     
   Interface gráfica do usuário, Texto

   O conteúdo gerado por IA pode estar incorreto.
2. Execute o sistema de monitoramento utilizando o comando “python monitor.py”  
     
   Texto

   O conteúdo gerado por IA pode estar incorreto.
3. Abra novas abas no terminal para cada novo nó que será monitorado  
     
   Interface gráfica do usuário, Texto

   O conteúdo gerado por IA pode estar incorreto.
4. Para ativar um nó execute no terminal: “python node.py <node\_id>”, onde node\_id é um número inteiro que será o ID daquele nó.  
     
   Texto

   O conteúdo gerado por IA pode estar incorreto.
5. Para interromper um nó aperte os botões “Ctrl” + “C” do teclado. O monitor exibirá uma mensagem de falha no nó interrompido.  
     
   Interface gráfica do usuário, Texto

   O conteúdo gerado por IA pode estar incorreto.  
     
   Caso todos os nós sejam interrompidos, o monitor enviará uma mensagem de falha crítica.  
     
   Texto

   O conteúdo gerado por IA pode estar incorreto.

## Explicação do Código

#### Detecção de Falhas utilizando Heartbeat

O código para esse sistema também é composto por dois arquivos: o arquivo *monitor.py*, que simula um sistema de detecção de falhas em um sistema distribuído utilizando o mecanismo “Heartbeat”; e o arquivo *node.py*, que simula um nó desse sistema.

#### monitor.py

Esse arquivo define os métodos para receber os heartbeats, detectar e tratar as falhas, simulando um servidor de pedidos de compra, conforme exemplo dado em sala de aula. A explicação para cada linha do código consiste nos comentários das imagens abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

#### node.py

Já o arquivo node.py busca simular um nó em um sistema distribuído, esse nó envia os heartbeats para o monitor a cada 5 segundos e pode ser instanciado diversas vezes, bem como interrompido a qualquer momento. A explicação para cada linha do código consiste nos comentários da imagem abaixo.

Texto

O conteúdo gerado por IA pode estar incorreto.