Desenvolvimento de um Sistema Distribuído para Monitoramento de Temperatura com Controle e Visualização Interativa

Aluno: Thiago Lima Bahia Santos
Curso: Engenharia de Computação – CEFET-MG
Disciplina: Sistemas Distribuídos
Período Letivo: 1º Semestre de 2025

1. Contextualização e Justificativa

O crescimento de sistemas computacionais distribuídos nas mais diversas aplicações, como monitoramento ambiental, automação industrial e Internet das Coisas (IoT), demanda a formação de profissionais capacitados a compreender os princípios e os desafios dessas arquiteturas. Assim, este projeto foi desenvolvido como prática de consolidação dos conceitos estudados na disciplina de Sistemas Distribuídos, com foco na criação de uma solução funcional e didática para simulação de sensores de temperatura operando de forma descentralizada.

O projeto explora uma estrutura cliente-servidor com múltiplos sensores simulados, integrando conceitos de redes, concorrência, tolerância a falhas, persistência de dados e exibição gráfica por meio de uma interface web. O sistema visa representar um cenário prático realista, como o monitoramento de ambientes industriais, laboratórios ou data centers.

2. Objetivos

2.1 Objetivo Geral

Desenvolver um sistema distribuído de monitoramento de temperatura, com sensores simulados comunicando-se com um servidor central para coleta, registro e análise dos dados obtidos.

2.2 Objetivos Específicos

- Criar sensores simulados com comportamento semi-realista de variação de temperatura;
- Implementar um servidor TCP capaz de gerenciar conexões simultâneas e validar sensores autorizados;
- Persistir as leituras de temperatura em um arquivo estruturado (CSV);
- Detectar e responder automaticamente a condições críticas (temperaturas elevadas);
- Fornecer uma interface web clara, responsiva e didática para exibição de dados, alertas e tendências;
- Automatizar a execução de múltiplos sensores de maneira controlada.

3. Metodologia e Arquitetura

3.1 Arquitetura Geral

O sistema foi projetado com base em uma arquitetura cliente-servidor. Os sensores simulados (clientes) comunicam-se via socket TCP com o servidor central, enviando periodicamente leituras de temperatura. O servidor valida os sensores, registra os dados recebidos e toma decisões sobre o estado de cada sensor. Em paralelo, um servidor web (Flask) acessa os dados registrados e apresenta-os em uma interface interativa para fins de análise e acompanhamento.

3.2 Tecnologias Utilizadas

- **Python 3.13**: linguagem de programação base do projeto;
- **Sockets TCP**: para comunicação entre sensores e servidor;
- **CSV**: persistência leve de dados;
- **Flask + Jinja2**: construção da interface web;
- **Chart.js**: gráficos interativos de temperatura;
- **Script de automação**: execução paralela de sensores por meio de subprocessos.

4. Funcionalidades Implementadas

4.1 Sensores Simulados

Cada sensor é uma aplicação cliente que:

- Gera uma temperatura inicial aleatória entre 35 e 37°C;
- Realiza pequenas variações ao longo do tempo, respeitando limites e mantendo um comportamento semelhante ao de sensores físicos;
- Envia leituras periodicamente ao servidor;
- Recebe comandos do servidor (ex: desligamento ou reajuste).

4.2 Servidor Central

O servidor TCP possui as seguintes responsabilidades:

- Aceitar múltiplas conexões simultâneas de sensores;
- Validar sensores autorizados a partir de um arquivo de configuração;
- Registrar todas as leituras recebidas no arquivo registros.csv;
- Verificar se a temperatura ultrapassa um limite crítico (38°C);
- Enviar comandos de **ajuste térmico** ao sensor, simulando um sistema de resfriamento automatizado.

4.3 Interface Web

O painel desenvolvido com Flask apresenta:

- Status de atividade de cada sensor (ativo, inativo, crítico);
- Gráfico dinâmico com séries temporais de temperatura por sensor;
- Listagens das últimas leituras recebidas;
- Histórico de alertas recentes;
- Atualização automática a cada 5 segundos, dispensando recarga manual.

5. Estrutura do Projeto

O projeto foi dividido em três diretórios principais:

- /cliente: código-fonte dos sensores e script de execução paralela (iniciar_sensores.py);
- /servidor: código do servidor TCP, servidor Flask e arquivos HTML;
- Arquivos auxiliares na raiz:
    - registros.csv: armazenamento persistente das leituras;
    - sensores_autorizados.txt: lista de sensores permitidos no sistema.

6. Resultados Obtidos

O sistema demonstrou boa estabilidade mesmo com múltiplos sensores operando simultaneamente, e permitiu visualizar claramente as leituras e alertas por meio da interface web. A resposta automatizada a condições críticas mostrou-se funcional, simulando o desligamento ou o resfriamento dos sensores conforme projetado.

As principais melhorias realizadas ao longo do projeto incluem:

- Substituição do desligamento dos sensores por um mecanismo mais realista de reajuste de temperatura;
- Reformulação do painel gráfico com maior apelo visual e informativo;
- Inclusão de atualização automática dos dados via JavaScript;
- Criação de script auxiliar para facilitar a execução em massa dos sensores;
- Reorganização dos diretórios e arquivos para facilitar a manutenção e a apresentação do projeto.

7. Desafios e Aprendizados

Durante o desenvolvimento, foram enfrentados desafios como a sincronização entre múltiplas conexões de sensores, a manipulação eficiente do arquivo de registros, a integração entre backend e frontend, além da necessidade de garantir clareza na apresentação dos dados.

Esses desafios proporcionaram um aprendizado aprofundado nas seguintes áreas:

- Programação orientada a eventos com sockets em Python;
- Manipulação de dados estruturados com CSV;
- Desenvolvimento web com Flask e integração com bibliotecas JavaScript;
- Planejamento modular de software com foco em manutenção e extensibilidade.

8. Considerações Finais

O projeto permitiu consolidar, de forma prática, diversos conceitos discutidos ao longo da disciplina, além de promover o desenvolvimento de habilidades técnicas transversais, como organização de código, design de interface e comunicação entre sistemas heterogêneos. O sistema desenvolvido pode ser estendido futuramente para abranger novos sensores, protocolos de comunicação e armazenamento em banco de dados relacional.