Este documento apresenta o desenvolvimento e os desafios enfrentados na implementação do servidor e do cliente para uma rede de sensores sem fio (RSSF). O sistema foi projetado para coletar medições de sensores e gerenciar a comunicação entre múltiplos dispositivos, utilizando sockets e protocolo TCP. O objetivo era simular um ambiente de sensores, onde os clientes enviam e recebem dados, atualizando suas medições com base nos sensores vizinhos.

Desafios e Dificuldades

1. Implementação Inicial do Sistema

O trabalho exigia o desenvolvimento de um servidor multithread capaz de gerenciar múltiplas conexões simultâneas e transmitir mensagens para todos os sensores de um mesmo tipo. A lógica inicial incluiu:

- Gerenciamento de clientes conectados ao servidor.
- Implementação do protocolo de comunicação entre cliente e servidor.
- Encaminhamento de mensagens para sensores do mesmo tipo.

Dificuldade: Garantir que as mensagens fossem corretamente propagadas para todos os sensores sem sobrecarregar o servidor ou introduzir erros de sincronização entre threads.

Solução: A utilização de mutexes para proteger as estruturas compartilhadas, como a lista de clientes, garantiu a consistência dos dados e o funcionamento correto do sistema.

2. Testes com Múltiplos Clientes

A maior dificuldade enfrentada foi realizar testes realistas com múltiplos clientes conectados ao servidor, simulando medições e atualizações. O enunciado exigia que cada cliente fosse inicializado com parâmetros específicos (tipo, coordenadas e intervalo de medições), o que aumentou a complexidade do processo.

Problemas Encontrados:

- A necessidade de iniciar vários clientes manualmente, com os valores descritos no enunciado, tornou o processo de testes demorado e sujeito a erros.
- Diferenças nos comportamentos dos clientes conectados dificultaram a validação dos resultados esperados.

Solução: Para facilitar os testes, foram criados executáveis personalizados, como o client20, que inicializa sensores de temperatura com medições fixas, permitindo maior controle durante a validação do sistema.

3. Lógica de Desconexão e Mensagens de Erro

O servidor deveria detectar automaticamente quando um cliente se desconectava e propagar essa informação para os sensores do mesmo tipo, utilizando uma mensagem de medição com valor -1.0. Inicialmente, o servidor imprimia mensagens de desconexão no formato inadequado.

Solução: A lógica foi ajustada para exibir as mensagens no mesmo formato do cliente, garantindo consistência no log:

log:

temperature sensor in (5,6)

measurement: -1.0000

action: removed

Soluções e Lições Aprendidas

Soluções Adotadas:

1. **Automação para Testes:** O desenvolvimento de clientes com comportamentos fixos, como o client20, facilitou os testes em cenários complexos com múltiplos sensores.
2. **Logs de Depuração:** Mensagens detalhadas foram adicionadas ao servidor e ao cliente para identificar rapidamente falhas de comunicação ou inconsistências na lógica de atualização.
3. **Gerenciamento de Threads:** O uso cuidadoso de mutexes para proteger estruturas compartilhadas evitou problemas de concorrência.

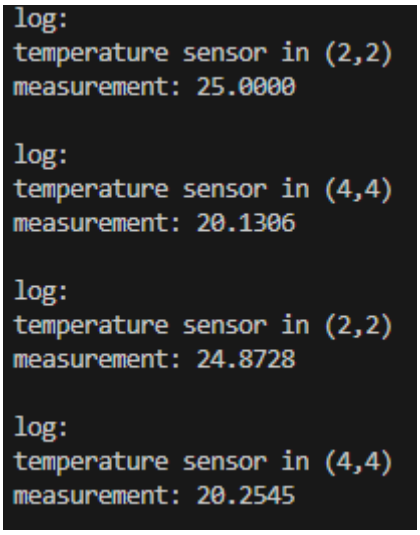
Lições Aprendidas:

- **Complexidade de Testes:** A simulação de cenários realistas com múltiplos clientes requer planejamento cuidadoso e ferramentas auxiliares para simplificar o processo.
- **Padronização de Protocolos:** Um formato de mensagem consistente entre cliente e servidor é essencial para evitar erros de interpretação.

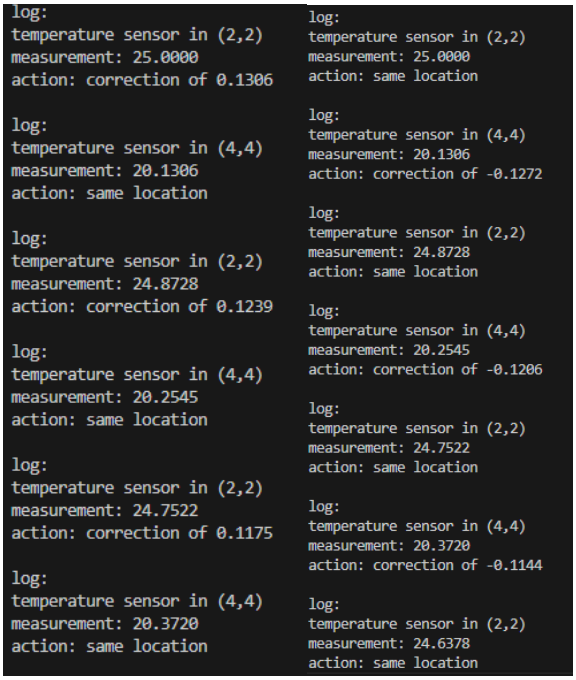
- **Resiliência a Erros:** Mensagens claras e padronizadas ajudam a depurar problemas durante a execução, especialmente em sistemas distribuídos.

Exemplo Prático de Execução

1. **Servidor:**



2. **Clientes:**



Conclusão

Este trabalho proporcionou uma compreensão aprofundada de conceitos fundamentais de redes, como sockets, gerenciamento de múltiplas conexões e sincronização de threads. Apesar das dificuldades, especialmente relacionadas aos testes com múltiplos clientes, o sistema demonstrou robustez e funcionalidade conforme os requisitos. A experiência reforça a importância de planejamento, padronização e documentação para o sucesso de projetos colaborativos.

Além disso, comparando com o primeiro TP da disciplina, foi possível reutilizar boa parte do código, então a implementação foi mais rápida. Porém, no TP do labirinto era só tentar rodar o código e jogar para ver se respondia como o esperado, nesse TP de RSSF é bem mais complexo, principalmente porque no funcionamento padrão do código os valores gerados dos sensores são aleatórios, então complicou para saber se as contas estavam corretas.