Sistemas Distribuídos - Relatório Detalhado

Universidade Federal do Ceará – Campus de Quixadá
Professor: Rafael Braga
Trabalho 1: Comunicação entre processos
Questão 5: Sistema Distribuído de Votação
Dupla: Mônica Maria Rodrigues
Maryana Moraes

Enunciado Resumido

Implementar uma **aplicação distribuída** de **sistema de votação** com as seguintes características:

- **Servidor** controla a votação e os candidatos.
- **Eleitores** se conectam via **TCP**, fazem login, recebem lista de candidatos e enviam seu voto.
- **Administradores** enviam **avisos informativos** aos eleitores por meio de **multicast UDP**.
- A **votação tem tempo limitado** — após o prazo, o servidor fecha o recebimento e **calcula os resultados**.
- A **representação externa de dados** (mensagens trocadas) deve usar **Protocol Buffers (protobuf)**.
- O servidor deve ser **multithreaded**, capaz de lidar com múltiplos clientes simultaneamente.

Estrutura do Projeto (C++ / Ubuntu)

O projeto foi implementado integralmente em **C++** com **sockets POSIX**, **threads (std::thread)** e **Protocol Buffers**, mantendo a lógica exigida no enunciado.

Arquivos criados:

- voting.proto → definição das mensagens protobuf.
- common.h → funções auxiliares de envio/recebimento com prefixo de 4 bytes.
- server.cpp → servidor multithread que gerencia candidatos, votos e envia multicast.

- voter_client.cpp → cliente eleitor (login, lista e voto).
- admin_client.cpp → cliente administrador (envia avisos multicast).
- multicast_listener.cpp → cliente que escuta mensagens administrativas via UDP.
- Makefile → automação de compilação com protoc e g++.

Estrutura das Mensagens (voting.proto)

As trocas de mensagens foram modeladas com **Protocol Buffers** no arquivo voting.proto, contendo:

- LoginRequest / LoginReply
  → Login do eleitor e resposta com lista de candidatos.
- VoteRequest / VoteReply
  → Envio do voto e confirmação de sucesso ou erro.
- Candidate e CandidateResult
  → Estruturas que representam candidatos e resultados da eleição.
- AdminNotice
  → Mensagem enviada via multicast (título, corpo, timestamp).
- Results
  → Resultado final da votação (total, porcentagens, vencedor).

Lógica do Servidor (server.cpp)

O **servidor**:

- Usa **TCP** para comunicação unicast com eleitores.
- Inicia com candidatos padrão (“Alice” e “Bob”).
- Gera threads independentes para cada cliente.
- Controla o tempo de votação via chrono.
- Protege os dados (votos, candidatos, eleitores) com std::mutex.
- Ao final do prazo, **calcula resultados** (total, votos por candidato, porcentagem).
- Envia notificações multicast via **UDP** para clientes escutando (multicast_listener).

Fluxo resumido:

1. Eleitor conecta ao servidor.

2. Envia LoginRequest.
3. Recebe lista de candidatos (LoginReply).
4. Envia VoteRequest com o ID do candidato.
5. Recebe VoteReply com confirmação.
6. Após o prazo, o servidor imprime os resultados no console.

Cliente Eleitor (voter_client.cpp)

- Conecta ao servidor via **TCP** (127.0.0.1:5555).
- Envia seu identificador (LoginRequest).
- Recebe lista de candidatos e exibe na tela.
- Permite digitar o ID do candidato e envia VoteRequest.
- Recebe VoteReply confirmando o voto.

Cliente Administrador (admin_client.cpp)

- Utiliza **multicast UDP** (230.0.0.1:4446).
- Envia mensagens administrativas (AdminNotice) com título e corpo (por exemplo, “Atenção: votação encerrará em breve!”).
- As mensagens são codificadas em protobuf e transmitidas pela rede local.

Cliente Escutador (multicast_listener.cpp)

- Entra no grupo multicast 230.0.0.1:4446.
- Recebe mensagens AdminNotice enviadas pelos administradores.
- Exibe na tela o título e o corpo de cada aviso.

Compilação e Execução (Ubuntu)

Dependências:

sudo apt update
sudo apt install -y build-essential protobuf-compiler libprotobuf-dev pkg-config

Compilação:

make

Execução em múltiplos terminais:

Terminal 1 - Servidor (com duração opcional em segundos)
./server 120

Terminal 2 - Escuta multicast
./multicast_listener

Terminal 3 - Cliente eleitor
./voter_client eleitor1

Terminal 4 - Administrador envia aviso
./admin_client "Atenção" "A votação encerrará em breve!"

Resultado Final

Ao término do tempo definido, o servidor:

- Rejeita novos votos.
- Calcula o total de votos, percentual de cada candidato e exibe no console.
- Demonstra a representação externa e serialização de dados conforme a proposta do exercício.

Conclusão

A aplicação atende **todos os requisitos da Questão 5**, implementando:

- Comunicação **TCP unicast** entre clientes e servidor.
- Comunicação **UDP multicast** para mensagens administrativas.
- **Representação externa via Protocol Buffers.**
- **Servidor multithreaded** com controle de concorrência e tempo de votação.
- Estrutura modular e compatível com o ambiente **Linux/Ubuntu**.

O sistema simula de forma completa um **ambiente de votação distribuído**, demonstrando domínio dos conceitos de **sockets, threads, serialização e protocolos de comunicação** em sistemas distribuídos.