

Pontifícia Universidade Católica de Minas Gerais

Instituto de Ciências Exatas e Informática - ICEI MG

Curso de Graduação em Engenharia de Computação

Leonardo Piuzana Pizani

Tiago Carvalho Moscoso

Pedro Henrique Caillaux

**Rede de Computadores I**

Trabalho Prático 1

Belo Horizonte 2024

Sumário

[1. Introdução 2](#_Toc182165683)

[2. Estrutura do Projeto 2](#_Toc182165684)

[3. Descrição dos Componentes 2](#_Toc182165685)

[3.1 ClientTCP 2](#_Toc182165686)

[3.2 ServerTCP 2](#_Toc182165687)

[3.3 TcpUtils 3](#_Toc182165688)

[4. Compilação e Execução 3](#_Toc182165689)

[5. Conclusão 3](#_Toc182165690)

# 1. Introdução

O projeto foi desenvolvido como uma coleção de ferramentas para facilitar a comunicação via protocolo TCP. O projeto contém implementações de um cliente e servidor TCP, além de diversos utilitários auxiliares. As ferramentas foram desenvolvidas na linguagem **C++**, conhecida por seu desempenho e controle de baixo nível sobre recursos do sistema, o que é particularmente útil em projetos de rede que exigem eficiência e controle sobre o gerenciamento de conexões e fluxos de dados.

# 2. Estrutura do Projeto

A estrutura do projeto é organizada em diferentes diretórios, conforme descrito abaixo:

- ClientTCP: Contém a implementação do cliente TCP.

- ServerTCP: Contém a implementação do servidor TCP.

- TcpUtils: Contém utilitários para suportar a comunicação TCP, incluindo funções de log e manipulação de mensagens.

- CMakeLists.txt e CMakePresets.json: Arquivos de configuração do CMake para facilitar a compilação e configuração do projeto.

# 3. Descrição dos Componentes

## 3.1 ClientTCP

O diretório ClientTCP contém os arquivos necessários para a implementação de um cliente TCP. Os principais arquivos são:  
- ClientTCP.cpp: Implementação das funcionalidades do cliente TCP.  
- ClientTCP.h: Declaração das funções e estruturas de dados utilizadas pelo cliente TCP.

## 3.2 ServerTCP

O diretório ServerTCP contém os arquivos relacionados à implementação de um servidor TCP. Os arquivos principais são:  
- ServerTCP.cpp: Implementação das funcionalidades do servidor TCP.  
- ServerTCP.h: Declaração das funções e estruturas de dados utilizadas pelo servidor TCP.

## 

## 3.3 TcpUtils

O diretório TcpUtils contém utilitários que auxiliam no funcionamento do cliente e servidor TCP, incluindo funções de logging e manipulação de mensagens. Os principais arquivos são:  
- Logger.h: Funções para registro de logs.  
- MessageContent.cpp: Manipulação de mensagens enviadas e recebidas via TCP.  
- TCPClient.cpp e TCPServer.cpp: Implementações específicas para gerenciamento de conexão cliente e servidor.  
- TcpUtils.cpp e TcpUtils.h: Funções gerais de suporte para comunicação TCP.

# 4. Compilação e Execução

Para compilar e executar o projeto, é necessário ter o CMake instalado. Os passos para compilação e execução são:  
1. Navegue até o diretório do projeto.  
2. Execute o comando 'cmake .' para gerar os arquivos de compilação.  
3. Use 'make' para compilar o projeto.  
4. Após a compilação, os binários do cliente e do servidor estarão disponíveis para execução.

# 5. Conclusão

O projeto fornece uma implementação robusta e funcional de um cliente e servidor TCP, juntamente com um conjunto de utilitários essenciais para o estabelecimento e manutenção de comunicação confiável via protocolo TCP. Esse conjunto de ferramentas demonstra, na prática, conceitos fundamentais de redes de computadores, como a conexão orientada a dados, controle de fluxo, e garantia de entrega — todos aspectos cruciais para aplicações que exigem comunicação segura e eficiente.

Os módulos de cliente e servidor, desenvolvidos em C++, são acompanhados por utilitários que tornam o processo de comunicação e depuração mais transparente e acessível, facilitando o monitoramento e o registro de logs. A estrutura modular do projeto possibilita que desenvolvedores utilizem TCPtools como uma base flexível para desenvolver novas funcionalidades ou adaptar o código para casos específicos de uso em comunicação de redes. Além disso, a utilização do **CMake** como ferramenta de configuração e compilação permite que o projeto seja facilmente portado para diferentes sistemas e ambientes, facilitando a experimentação e a integração em projetos maiores.

Este projeto pode servir como base para aplicações que demandam comunicação em rede ou como material de estudo para alunos interessados em redes de computadores e programação em C++.