# Trabalho com conexão TCP

# Implementação de Cliente/Servidor TCP Multithread com Transferência de Arquivos e Chat

## Objetivo do Projeto:

Desenvolver uma aplicação cliente-servidor utilizando o protocolo TCP e programação de sockets. O servidor deve ser capaz de lidar com múltiplos clientes concorrentemente (usando threads) e oferecer funcionalidades como transferência de arquivos grandes com verificação de integridade (SHA) e um chat simples.



Requisitos Gerais:

1. **Linguagem de Programação:** Livre escolha do aluno (ex: Python, Java, C/C++, etc.).
2. Uso de Sockets: É obrigatório utilizar a API de sockets diretamente para toda a comunicação TCP.
3. Não é permitido o uso de bibliotecas que abstraiam ou mascarem a manipulação direta de sockets e conexões TCP.
4. Sugere-se iniciar com um exemplo simples “Hello World” cliente/servidor TCP multithread para familiarização antes de implementar as funcionalidades completas.

## Requisitos do Servidor TCP (Multithread):

1. Inicialização: Deve ser executado antes de qualquer cliente.
2. Porta de Escuta: Deve aguardar conexões em uma porta TCP específica (escolhida pelo aluno, maior que 1024).
3. Gerenciamento de Conexões:
   a. Aceitar conexões TCP de múltiplos clientes.
   b. Para cada nova conexão aceita, o servidor deve criar uma thread dedicada para lidar exclusivamente com a comunicação daquele cliente.

### Funcionalidades Dentro de Cada Thread do Servidor:

A thread dedicada a um cliente deve:

1. Loop de Requisições: Aguardar e processar requisições enviadas pelo cliente conectado até receber a requisição “Sair”.
2. Tratamento de Requisições: Implementar a lógica para as seguintes requisições (o formato exato faz parte do protocolo a ser definido pelo aluno):
   a. Requisição “Sair”:
      I. Fechar a conexão TCP com aquele cliente.
      II. Encerrar a thread correspondente de forma limpa.
   b. Requisição “Arquivo [Nome_Arquivo.ext]”:
      I. Verificar se o arquivo [Nome_Arquivo.ext] existe no servidor.
      II. Se o arquivo existir:
         A. Calcular o hash SHA (ex: SHA-256) do conteúdo completo do arquivo. (O aluno deve pesquisar como implementar/usar bibliotecas padrão para cálculo de SHA).
         B. Enviar o arquivo para o cliente, seguindo o protocolo de aplicação definido (ver seção “Protocolo de Aplicação”).
         C. A transferência deve suportar arquivos grandes (> 10 MB).
      III. Se o arquivo não existir:
         A. Enviar uma mensagem de erro/status apropriada para o cliente, conforme definido no protocolo.
   c. Requisição “Chat [Mensagem]” (Recebida do Cliente):
      I. Exibir a [Mensagem] recebida na console do servidor (indicando de qual cliente veio, se possível).
   d. Envio de Mensagens de Chat (Iniciadas pelo Servidor):
      I. Permitir que texto digitado na console principal do servidor seja enviado como mensagem de chat para todos os clientes conectados (ou implementar um mecanismo para direcionar a um cliente específico, se desejado). Nota: Esta parte requer atenção à comunicação entre a thread principal/console e as threads dos clientes.

## Requisitos do Cliente TCP:

1. Inicialização: Deve ser executado após o servidor estar ativo.
2. Conexão:
   a. Permitir ao usuário especificar o endereço IP e a porta do servidor TCP.
   b. Estabelecer uma conexão TCP com o servidor.
3. Interface do Usuário:
   a. Oferecer um meio para o usuário escolher e enviar as seguintes requisições para o servidor: “Sair”, “Arquivo [Nome_Arquivo.ext]”, “Chat [Mensagem]”.
4. Processamento de Respostas do Servidor:
   a. Resposta a “Sair”: Após enviar “Sair”, fechar a conexão do lado do cliente e encerrar o programa.
   b. Resposta a “Arquivo”:
      I. Receber os dados do arquivo (metadados e conteúdo) conforme a ordem definida no protocolo de aplicação.
      II. Salvar o conteúdo recebido em um arquivo local.
      III. Após receber todo o conteúdo, calcular o hash SHA do arquivo recebido.
      IV. Verificar a integridade: Comparar o hash calculado com o hash recebido do servidor. Informar ao usuário se o arquivo foi recebido corretamente ou se houve corrupção.
      V. Deve ser capaz de receber e salvar arquivos grandes (> 10 MB).
   c. Resposta a “Chat” (Mensagens recebidas do Servidor):
      I. Exibir as mensagens de chat recebidas do servidor na tela do cliente.

## Definição do Protocolo de Aplicação (Tarefa do Aluno):

O aluno deve definir e documentar um protocolo de aplicação simples sobre TCP para gerenciar a comunicação. Este protocolo deve especificar claramente:

1. Formato das Requisições: Como o cliente envia “Sair”, “Arquivo [Nome]”, “Chat [Msg]”.
2. Formato das Respostas/Transferência de Arquivo: A ordem e o formato para enviar:
   a. Status da operação (ex: OK, ERRO_ARQUIVO_NAO_ENCONTRADO).
   b. Metadados do arquivo (Nome do arquivo, Tamanho total).
   c. Hash SHA do arquivo completo.
   d. Os dados do arquivo (como serão segmentados/enviados sobre o stream TCP).
3. Formato das Mensagens de Chat.

Requisitos Chave a Serem Demonstrados:

1. Multithreading: O servidor tratando pelo menos dois clientes simultaneamente.
2. Funcionalidades: Demonstração bem-sucedida das requisições “Sair”, “Chat” (bidirecional) e “Arquivo”.
3. Transferência de Arquivo Grande: Transferência e salvamento correto de um arquivo > 10 MB.
4. Verificação de Integridade: Demonstração da verificação por Hash SHA no cliente, indicando sucesso ou falha.
5. Tratamento de Erro: Demonstração do tratamento de erro (ex: Arquivo não encontrado).
6. Robustez: O sistema deve lidar corretamente com as interações descritas.

Lembre-se: O vídeo deve ser uma demonstração prática, complementada por explicações claras sobre o funcionamento e as decisões de projeto.

O vídeo **deve** ser anexado como **link** nos comentários da entrega dentro da atividade do classroom. **NÃO** deve ser enviado o arquivo do video!



- O aluno deve usar as chamadas TCP e não pode usar bibliotecas que mascarem o trabalho.