Protocolo de Comunicação para Transferência de Arquivos Cliente/Servidor.

Introdução

Este documento descreve o protocolo de comunicação cliente/servidor para transferência de arquivos via TCP, especificando os comandos suportados na comunicação com o nome de cada função seguido por sua explicação, além dos mecanismos implementados para garantir a segurança e a integridade dos dados transmitidos.

Estabelecimento de Conexão Cliente/Servidor

• Cliente - conectaAoServidor()
Cria um socket TCP e conecta ao servidor, permitindo o envio e recebimento de dados.

• Servidor - escutaPorta()
Configura um socket TCP na porta informada e passa a aguardar conexões de clientes.

Formatação e Recepção de Mensagens com Cabeçalho de Tamanho

• Cliente/Servidor - adicionaTamanho(dados)
Prefixa cada payload com 4 bytes que representam o tamanho da mensagem (big-endian), garantindo que o receptor saiba exatamente quantos bytes ler em cada envio.

Formato: [4 bytes de tamanho N, big-endian] [N bytes de payload]

• Servidor - leComando(sockCon)
Lê os 4 bytes iniciais para descobrir o comprimento da mensagem que o cliente enviou e, em seguida, faz leituras sucessivas até acumular todos esses bytes, retornando o comando completo recebido.

Segurança de Acesso a Arquivos e Resposta a Comandos Inválidos

• Servidor - pastaPermitida(caminhoArquivo, pastaBase)
Essa função garante que o caminho do arquivo acessado esteja dentro da pasta permitida, prevenindo que o servidor acesse arquivos fora do diretório base definido. Ela faz isso comparando os caminhos absolutos e reais dos diretórios.

• Servidor - respondeComandoNulo(sockCon)
O servidor responde ao cliente com uma mensagem vazia, formatada com cabeçalho de tamanho, indicando que não há dados a serem enviados.

Processamento de Comandos no Servidor e Interação do Cliente com o Menu

• Servidor - processaComando(sockCon, comando)
Analisa os três primeiros bytes do comando recebido pelo servidor e, conforme o prefixo (como DIR, DOW, DMA, etc.), chama a função correspondente para processar a solicitação do cliente; caso o comando não seja reconhecido, responde com uma mensagem nula.

• Cliente - mostraOpcoes()
Exibe um menu com seis opções numeradas, solicita ao usuário que escolha uma delas e retorna o valor digitado, repetindo em caso de erro até receber um número válido.

• Cliente - main()
Inicia a conexão com o servidor e então entra em um loop que exibe o menu, lê a opção do usuário e chama a ação correspondente.

Interação entre Cliente e Servidor: Fluxo de Comandos e Respostas

• Cliente - listaArquivos() | Servidor - respondeComandoDir(sockCon)
A função do cliente inicia o processo enviando o comando DIR ao servidor; o servidor interpreta esse comando, lista os arquivos da pasta permitida, monta uma resposta com os nomes e tamanhos, e a envia de volta ao cliente; o cliente então lê até receber tudo e ao final exibe a listagem formatada dos arquivos disponiveis no servidor.

• Cliente - obterArquivo(nomeArquivo)
A função garante que a pasta downloads exista, verifica se o arquivo solicitado já está presente localmente e, se estiver, pergunta ao usuário se deseja sobrescrevê-lo; em seguida, envia ao servidor uma resposta indicando se deve prosseguir ou pular o envio; se autorizado, o cliente inicia a leitura dos dados em blocos até completar o total, salvando o conteúdo no disco.

• Cliente - downloadArquivo() | Servidor - respondeComandoDow(sockCon, nomeArquivo)
A função do cliente envia ao servidor o comando DOW seguido do nome do arquivo desejado; o servidor verifica se o caminho é permitido e se o arquivo existe, respondendo com uma mensagem vazia se não for válido; se válido, envia o conteúdo do arquivo em blocos, que será recebido pelo cliente na função obterArquivo() para salvar localmente.

• Cliente - downloadMascara() | Servidor - respondeComandoDMA(sockCon, mascara)
A função do cliente envia ao servidor o comando DMA junto com a máscara desejada (como *.txt), e o servidor busca arquivos que combinam com a máscara dentro da pasta permitida e envia ao cliente a lista desses nomes; o cliente então separa os arquivos recebidos e para cada um chama obterArquivo() para baixá-lo, enquanto o servidor aguarda a resposta OK (2 Bytes) ou SKIP (4 Bytes) antes de enviar o conteúdo do arquivo com respondeComandoDow().

• Cliente - obterMd5() | Servidor - respondeComandoMd5(sockCon, dados)
A função do cliente envia ao servidor o nome do arquivo e a posição até onde o hash deve ser calculado, e o servidor, ao receber, valida se o caminho é permitido e se o arquivo existe; em seguida, lê os primeiros n bytes do arquivo, calcula o hash MD5 e envia esse valor de volta ao cliente, que então o imprime na tela.

• Cliente - retomarDownload() | Servidor - respondeComandoDra(sockCon, dados)
A função do cliente verifica o quanto já foi baixado de um arquivo e envia ao servidor um comando DRA com o nome, quantidade de bytes e o hash local, e o servidor valida se o caminho é permitido, se o arquivo existe e se o hash confere com os bytes iniciais; se tudo estiver correto, envia os dados restantes do arquivo, que são recebidos pelo cliente e anexados ao arquivo local para concluir o download.

Gerenciamento de Conexões e Execução do Servidor

• Servidor - trataCliente(sockConexao, cliente)
Gerencia a comunicação com um cliente, adicionando-o à lista de conexões ativas, lendo comandos em loop e processando-os; se a conexão for encerrada ou ocorrer um erro, ela remove o cliente da lista e fecha o socket.

• Servidor - main()
Inicia o servidor chamando escutaPorta, imprime que está escutando conexões e entra em um loop aceitando novos clientes, criando uma thread para tratar cada conexão.