UNIVERSIDADE VEIGA DE ALMEIDA

CURSO DE ENGENHARIA DA COMPUTAÇÃO

LUIZ FELIPE MONTEIRO PINTO - 1210103026

ATIVIDADE AVALIATIVA (A1)

REDES DE COMPUTADORES

RIO DE JANEIRO

2023

LUIZ FELIPE MONTEIRO PINTO - 1210103026

REDES DE COMPUTADORES

Trabalho acadêmico apresentado a Universidade Veiga de Almeida, desenvolvido para a obtenção de grau avaliativo na disciplina de Redes de Computadores.

Orientador: Fabio Contarini Carneiro

RIO DE JANEIRO

2023

INTRODUÇÃO

Neste trabalho, é mostrado um programa Java simples para transferências de arquivos baseadas em TCP (Transmission Control Protocol) entre hosts na Internet. Os usuários deste programa podem informar ao servidor o nome do arquivo que desejam enviar, e o servidor enviará o arquivo e o salvará com esse nome. Comentários explicativos estão incluídos no código para ajudar os leitores a compreender cada etapa da implementação. Esses comentários abordam questões como escolha de TCP em vez de UDP, operação do programa, cabeçalhos de protocolo de transferência, estimativa de rendimento, bem como instruções de compilação e execução. Neste estudo, a comunicação de rede para transferência segura de arquivos entre sistemas é ilustrada de forma prática.

UTILIZAÇÃO DE TCP AO EM VEZ DE UDP

O TCP (Transmission Control Protocol) foi escolhido em vez do UDP (User Datagram Protocol) para essa aplicação de transferência de arquivos devido às seguintes razões:

Confiabilidade: O TCP oferece uma entrega confiável de dados, garantindo que os dados sejam entregues na ordem correta e sem perda de pacotes. Isso é essencial para garantir que o arquivo seja transferido com precisão, sem corrupção de dados.

Controle de Fluxo: O TCP possui mecanismos de controle de fluxo integrados, o que significa que ele pode ajustar a taxa de transferência de acordo com a capacidade da rede e a disponibilidade do destino. Isso evita a sobrecarga da rede e garante uma transferência de arquivo eficiente.

Controle de Congestionamento: O TCP também inclui recursos de controle de congestionamento para evitar congestionamentos na rede. Ele pode ajustar a taxa de transferência com base na situação atual da rede, garantindo que a transferência seja justa com outras comunicações na mesma rede.

CÓDIGO DO SERVIDOR QUE IRÁ RECEBER O ARQUIVO

```
import java.io.*;
import java.net.*;

public class FileTransferServer {
    public static void main(String[] args) {
        // Porta em que o servidor estará ouvindo conexões
        int port = 12345;

        try {
            // Criação de um servidor Socket para aceitar conexões
            ServerSocket serverSocket = new ServerSocket(port);
            System.out.println("Servidor aguardando conexões na porta " + port);

            while (true) {
                // Aguarda a conexão de um cliente
                Socket clientSocket = serverSocket.accept();
                System.out.println("Cliente conectado: " +
clientSocket.getInetAddress().getHostAddress());

                // Criação de fluxos de entrada e saída para transferência de dados
                InputStream inputStream = clientSocket.getInputStream();
                OutputStream outputStream = new FileOutputStream("arquivo_recebido.txt");

                // Buffer para ler os dados do arquivo
                byte[] buffer = new byte[1024];
                int bytesRead;

                // Lê os dados do cliente e os escreve no arquivo
                while ((bytesRead = inputStream.read(buffer)) != -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }

                System.out.println("Transferência de arquivo concluída.");

                // Fecha os fluxos e a conexão com o cliente
                outputStream.close();
                inputStream.close();
                clientSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

CODIGO DE QUEM IRÁ ENVIAR O ARQUIVO

```
import java.io.*;
import java.net.*;

public class FileTransferClient {
    public static void main(String[] args) {
        // Endereço IP e porta do servidor
        String serverAddress = "127.0.0.1"; // Altere para o endereço do servidor
        int serverPort = 12345;

        try {
            // Criação de um socket para conectar ao servidor
            Socket socket = new Socket(serverAddress, serverPort);

            // Nome do arquivo a ser enviado
            String fileName = "Redes_de_Computadores.txt";

            // Criação de fluxos de entrada e saída para transferência de dados
            OutputStream outputStream = socket.getOutputStream();
            FileInputStream fileInputStream = new FileInputStream(fileName);

            // Buffer para ler os dados do arquivo
            byte[] buffer = new byte[1024];
            int bytesRead;

            // Lê os dados do arquivo e os envia para o servidor
            while ((bytesRead = fileInputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }

            System.out.println("Arquivo enviado com sucesso.");

            // Fecha os fluxos e a conexão com o servidor
            outputStream.close();
            fileInputStream.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

FUNCIONAMENTO DO CÓDIGO DA APLICAÇÃO

O código da aplicação consiste em dois programas: o servidor (“FileTransferServer”) e o cliente (“FileTransferClient”).

Servidor (“FileTransferServer”):

1) O servidor cria um ‘ServerSocket’ na porta especificada e aguarda a conexão de clientes.

2) Quando um cliente se conecta, ele cria fluxos de entrada e saída para transferência de dados.

3) O servidor lê os dados do cliente (o arquivo a ser transferido) em blocos de 1024 bytes (buffer) e os escreve em um arquivo local chamado "arquivo_recebido.txt".

4) O processo de leitura e escrita continua até que todos os dados tenham sido transferidos.

5) Após a conclusão da transferência, o servidor fecha os fluxos e a conexão com o cliente e volta a aguardar por novas conexões.

Cliente (“FileTransferClient”):

1) O cliente cria um Socket para se conectar ao servidor especificado pelo endereço IP e porta.

2) Ele abre o arquivo local ("Redes_de_Computadores.txt") que deseja enviar.

3) O cliente lê os dados do arquivo em blocos de 1024 bytes e os envia para o servidor através do socket.

4) O processo de leitura e envio continua até que todo o arquivo tenha sido transferido.

5) Após a conclusão da transferência, o cliente fecha os fluxos e a conexão com o servidor.

CABEÇALHOS DO PROTOCOLO DE TRANSFERÊNCIA

A transferência de dados é realizada diretamente por meio dos fluxos de entrada e saída do Java, e os cabeçalhos TCP/IP são gerenciados pelo próprio TCP.

CÁLCULO DA TAXA DE TRANSFERÊNCIA

Para calcular a taxa de transferência, foi medido o tempo que leva para transferir o arquivo e dividido o tamanho do arquivo pelo tempo decorrido.

Código para calcular a Taxa de Transferência:

```
long startTime = System.currentTimeMillis();

// Realize a transferência de arquivo aqui

long endTime = System.currentTimeMillis();
long elapsedTime = endTime - startTime;

// Tamanho do arquivo em bytes
long fileSize = new File("arquivo_a_enviar.txt").length();

// Taxa de transferência em bytes por segundo
double transferRate = (double) fileSize / (double) elapsedTime * 1000;
// Multiplicado por 1000 para obter bytes por segundo
System.out.println("Taxa de Transferência: " + transferRate + " bytes por segundo");
```

COMPILAÇÃO E EXECUÇÃO

Para compilar e executar os programas em Java, devemos seguir essas etapas:

1) Instalar o JDK (Java Development Kit) no sistema.

2) Salvar o código do servidor em um arquivo chamado "FileTransferServer.java" e o código do cliente em um arquivo chamado "FileTransferClient.java."

3) Compilar o código do servidor com o seguinte comando:

```
javac FileTransferServer.java
```

4) Compilar o código do cliente com o seguinte comando:

```
javac FileTransferClient.java
```

5) Executar o servidor em um terminal com o seguinte comando:

```
java FileTransferServer
```

6) Executar o cliente em outro terminal com o seguinte comando:

```
java FileTransferClient
```