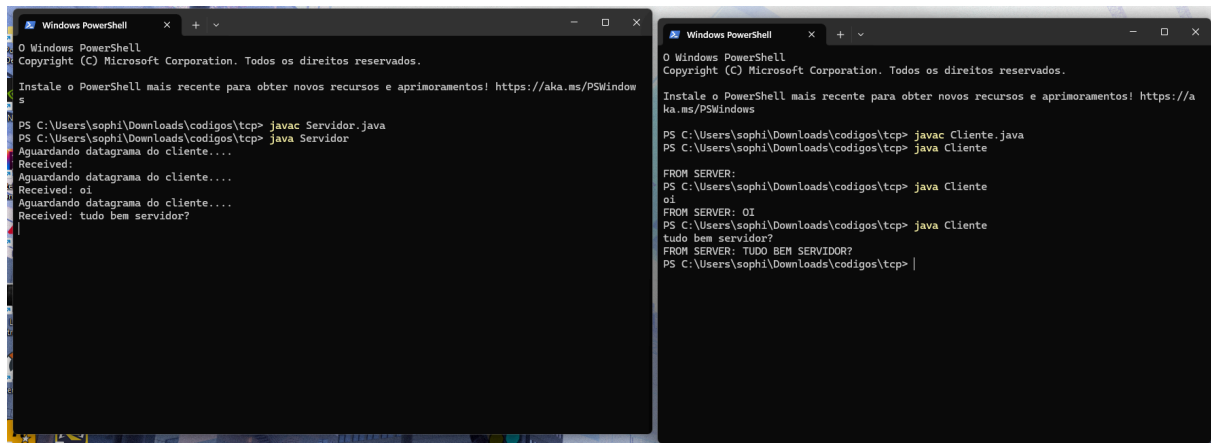


Lista 06 - Aplicação com TCP em Java

Sophia Carrazza - CC PUC Minas



```
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS C:\Users\sophi\Downloads\codigos\tcp> javac Servidor.java
PS C:\Users\sophi\Downloads\codigos\tcp> java Servidor
Aguardando datagrama do cliente....
Received:
Aguardando datagrama do cliente....
Received: oi
Aguardando datagrama do cliente....
Received: tudo bem servidor?

O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS C:\Users\sophi\Downloads\codigos\tcp> javac Cliente.java
PS C:\Users\sophi\Downloads\codigos\tcp> java Cliente

FROM SERVER:
PS C:\Users\sophi\Downloads\codigos\tcp> java Cliente
oi
FROM SERVER: oi
PS C:\Users\sophi\Downloads\codigos\tcp> java Cliente
tudo bem servidor?
FROM SERVER: TUDO BEM SERVIDOR?
PS C:\Users\sophi\Downloads\codigos\tcp> |
```

No UDP, o servidor **fica esperando datagramas desde o início**, pois não há handshake ou conexão prévia necessária. Ele simplesmente aguarda pacotes em uma porta, independentemente de quem os envie.

Diferente do UDP, o TCP fica aguardando **conexões** (solicitações de handshake) na porta configurada. Quando um cliente se conecta (via **connect()**), aí sim o servidor aceita a conexão (**accept()**) e só então começa a esperar e receber dados.

```

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

class Cliente {
    private static String ipServidor = "127.0.0.1";
    private static int portaServidor = 6790;

    public static String lerString() throws Exception {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        return in.readLine();
    }

    Run | Debug
    public static void main(String argv[]) throws Exception {
        // Efetua a primitiva socket e connect, respectivamente.
        Socket socket = new Socket(ipServidor, portaServidor);

        // Efetua a primitiva send
        DataOutputStream saida = new DataOutputStream(socket.getOutputStream());
        saida.writeBytes(lerString() + '\n');

        // Efetua a primitiva receive
        BufferedReader entrada = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        System.out.println("FROM SERVER: " + entrada.readLine());

        // Efetua a primitiva close
        socket.close();
    }
}

```

```

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

class Servidor {
    private static int portaServidor = 6790;

    Run | Debug
    public static void main(String argv[]) throws Exception {
        // Efetua as primitivas socket e bind, respectivamente
        try (ServerSocket socket = new ServerSocket(portaServidor)) {

            while (true) {
                // Efetua as primitivas de listen e accept, respectivamente
                Socket conexao = socket.accept();

                // Efetua a primitiva receive
                System.out.println("Aguardando datagrama do cliente...");
                BufferedReader entrada = new BufferedReader(new InputStreamReader(conexao.getInputStream()));

                // Operacao com os dados recebidos e preparacao dos a serem enviados
                String str = entrada.readLine();
                System.out.println("Received: " + str);

                str = str.toUpperCase() + '\n';

                // Efetua a primitiva send
                DataOutputStream saida = new DataOutputStream(conexao.getOutputStream());
                saida.writeBytes(str);
            }
        }
    }
}

```