UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO - UFERSA
Departamento de Computação - DC
Graduação em Ciência da Computação
Disciplina: Sistemas Distribuídos
Prof.: Paulo Henrique Lopes Silva

Prática 3.4 – Cliente-Servidor TCP *multithread*

1. Cliente.

Java

```java
public class Cliente {

	Socket socket;
	InetAddress inet;
	String ip;
	int porta;

	public Cliente(String ip, int porta) {

		this.ip = ip;
		this.porta = porta;
		this.rodar();

	}

	private void rodar() {

		/*
		 * Para se conectar ao servidor,
		 * cria-se objeto Socket.
		 * O primeiro parâmetro é o
		 * IP ou endereço da máquina que
		 * se quer conectar e o segundo é
		 * a porta da aplicação.
		 * Neste caso, usa-se o IP da
		 * máquina local (127.0.0.1) e a porta da
		 * aplicação Servidor de Eco (54321).
		 */

		try {

			socket = new Socket(ip, porta);
			inet = socket.getInetAddress();

			System.out.println("HostAddress = " + inet.getHostAddress());
			System.out.println("HostName = " + inet.getHostName());

			/*
			 * Criar um novo objeto Cliente
			 * com a conexão socket para que
			 * seja executado em
			 * um novo processo.
			 * Permitindo assim a conexão de
			 * vários clientes com o
			 * servidor.
```

```
                     */

                    ImplCliente c = new ImplCliente(socket);
                    Thread t = new Thread(c);
                    t.start();

             } catch (IOException e) {
                    e.printStackTrace();
             }
      }

      public static void main(String args[]) {

             new Cliente("127.0.0.1", 54321);

      }

}
```

2. Implementação do cliente.

Java

```
public class ImplCliente implements Runnable {

      private Socket cliente;
      private boolean conexao = true;
      private PrintStream saida;

      public ImplCliente(Socket c) {
             this.cliente = c;
      }

      public void run() {

             try {

                    System.out.println("O cliente conectou ao servidor");

                    // Prepara para leitura do teclado

                    Scanner teclado = new Scanner(System.in);

                    // Cria objeto para enviar a mensagem ao servidor

                    saida = new PrintStream(cliente.getOutputStream());

                    // Envia mensagem ao servidor
                    String mensagem;

                    while (conexao) {

                           System.out.println("Digite uma mensagem: ");
                           mensagem = teclado.nextLine();
```

```
                        if (mensagem.equalsIgnoreCase("fim"))
                                conexao = false;
                        else
                                System.out.println(mensagem);

                        saida.println(mensagem);

                }

                saida.close();
                teclado.close();
                cliente.close();
                System.out.println("Cliente finaliza conexão.");

        } catch (IOException e) {
                e.printStackTrace();
        }
    }

}
```

3. Servidor.

Java

```
public class Servidor {

    ServerSocket socketServidor;
    Socket cliente;
    int porta;

    public Servidor(int porta) {
        this.porta = porta;
        this.rodar();
    }

    private void rodar() {

        /*
         * Cria um socket na porta 54321
         */
        try {
                socketServidor = new ServerSocket(porta);

                System.out.println("Servidor rodando na porta " +
                        socketServidor.getLocalPort());
                System.out.println("HostAddress = " +
                        InetAddress.getLocalHost().getHostAddress());
                System.out.println("HostName = " +
                        InetAddress.getLocalHost().getHostName());

                /*
                 * Aguarda alguém se conectar.
                 * A execução do servidor fica bloqueada na chamada
                 * do método accept da classe ServerSocket.
```

```
                     * Quando alguém se conectar ao
                     * servidor, o método desbloqueia e
                     * retorna com um objeto da classe Socket, que
                     * é uma porta da comunicação.
                     */

                    System.out.println("Aguardando conexão do cliente...");

                    while (true) {

                          cliente = socketServidor.accept();

                          // Cria uma thread do servidor para tratar a conexão
                          ImplServidor servidor = new ImplServidor(cliente);

                          Thread t = new Thread(servidor);
                          // Inicia a thread para o cliente conectado

                          ImplServidor.cont++;

                          t.start();
                    }

             } catch (IOException e) {
                    e.printStackTrace();
             }
      }

      public static void main(String[] args) throws Exception {

             new Servidor(54321);

      }

}
```

4. Implementação do servidor.

Java

```
public class ImplServidor implements Runnable {

      public Socket socketCliente;
      public static int cont = 0;
      private boolean conexao = true;
      private Scanner s = null;

      public ImplServidor(Socket cliente) {
             socketCliente = cliente;
      }

      public void run() {
             String mensagemRecebida;

             System.out.println("Conexão " +
```

```
                        ImplServidor.cont +
                        " com o cliente "+
                        socketCliente.getInetAddress().getHostAddress() +
                        "/" +
                        socketCliente.getInetAddress().getHostName()
                        );

            try {

                  s = new Scanner(socketCliente.getInputStream());

                  // Exibe mensagem no console
                  while (conexao) {
                        mensagemRecebida = s.nextLine();

                        if (mensagemRecebida.equalsIgnoreCase("fim"))
                              conexao = false;
                        else
                              System.out.println(mensagemRecebida);
                  }

                  // Finaliza scanner e socket
                  s.close();
                  System.out.println("Fim do cliente " +
                              socketCliente.getInetAddress().getHostAddress());

                  socketCliente.close();

            } catch (IOException e) {
                  e.getMessage();
            }
      }

}
```