Desenvolvimento de Aplicações Distribuídas

Aula 7: Estudo de Caso: Sockets TCP em Java

Prof. Dr. Fábio Favarim
favarim@utfpr.edu.br

Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco

27 de Agosto de 2018

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Objetivos da Aula

- Estudo de Caso: API Java para Sockets TCP
- Uso de Streams: Array de Bytes, Data e Object

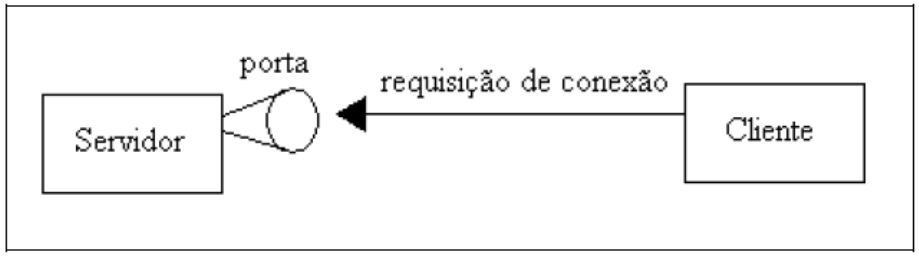
Sockets: TCP - Transfer Control Protocol

TCP: Orientado a conexão – Confiável

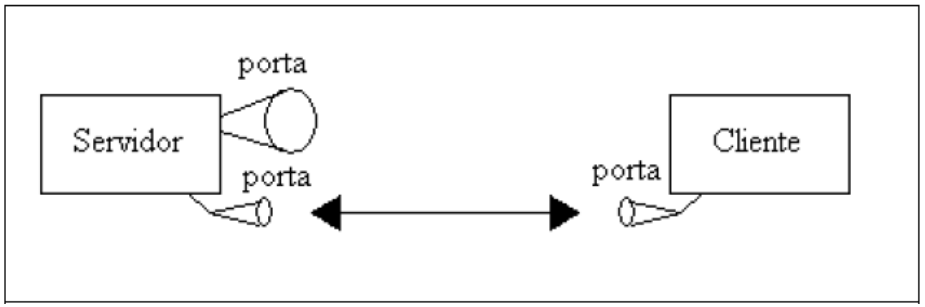
- Uma conexão (canal) deve ser estabelecida antes da transmissão dos dados;
- Dados são enviados em **fluxo contínuo (stream)** no **canal**;
- Dados não são enviados datagrama a datagrama, mas fragmentados pelo TCP automaticamente em segmentos
- A conexão deve ser encerrada após a transmissão dos dados;
  - Basta **gravar** no stream para enviar uma mensagem
  - Basta **ler** do stream para receber uma mensagem
- É **garantida a entrega** dos segmentos
- É **garantida a ordem de entrega** dos segmentos

Sockets TCP - Funcionamento Geral

Sockets de Conexão - O servidor fica aguardando **conexões** de clientes em um socket em **porta** específica.



Socket de comunicação - Quando aceita a conexão de um cliente, **outro socket** é criado para a **comunicação**.



Passos de Comunicação com Sockets TCP

Servidor

1. Criar socket de conexão
2. Fica ouviando aguardando conexões de clientes, quando conectar criar socket de comunicação
3. Cria canais de comunicação (streams) com o cliente (no Java)
4. Faz a comunicação com o cliente
5. Fecha socket com o cliente

Cliente

1. Cria socket cliente
2. Cria canais de comunicação (streams) com o servidor (no Java)
3. Faz a comunicação com o servidor
4. Fechar socket com o servidor

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

API Java para programação com Sockets

Pacote **java.net**, principais classes:

- **UDP:** DatagramPacket e DatagramSocket;
- **TCP:** Socket e ServerSocket;
- Outras: obtenção endereço IP e porta de um pacote, entre outras coisas.

Sockets TCP (Java) - Comandos Básicos

Criação de Socket de Conexão (Para aguardar por conexões)

```
1   ServerSocket conexao = new ServerSocket(porta, [tamanho_fila]);
2   // tamanho_fila = maximo de pedidos de conexoes que e mantido para o
        socket
```

Criação do Socket de Comunicação / Estabelecer canal comunicação

```
3   // Lado Servidor
4   Socket socket = conexao.accept();
5
6   // Lado Cliente
7   Socket socket = new Socket(host, porta);
```

Fechar Socket de Comunicação

```
9   socket.close(); // conexao.close();
```

Sockets TCP (Java) - Comandos Básicos

Criar fluxos (streams) de entrada e saída de dados, conforme o tipo de dados.

- Streams são canais de comunicação entre um programa e uma fonte de dados
- As fontes de dados podem ser: arquivos, trechos de memória e conexões de rede
- Em Java não importa qual é a fonte, a forma de usar é a mesma
- Em Java há vários classes para manipulação de **streams**, vamos usar três:
  - Tipos básicos (int, String, double...)
  - Objetos
  - Array de Bytes

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sockets TCP (Java) - Comandos Básicos

Criar streams com tipos básicos (DataInputStream/DataOutputStream)

```
10      DataInputStream in = new DataIntputStream(socket.getInputStream());
11      DataOutputStream out = new DataOutputStream(sockeet.getOutputStream
            ());
```

Enviar dados para o stream

```
12      out.writeUTF("Alo pessoal");
13      out.writeDouble(15.5);
14      out.writeInt(15);
```

Receber dados do stream

```
15      String resposta = in.readUTF();
16      double resposta = in.readDouble();
17      int resposta = in.readInt();
```

Sockets TCP (Java) - Comandos Básicos

Criar streams de Objetos (ObjectInputStream/ObjectOutputStream)

- Um objeto só pode ser gravado/lido se implementar a interface Serializable
- Um objeto serializável é aquele cuja representação pode ser transformado em bytes e poderá ser armazenado em disco ou transmitido via stream.

```
18      ObjectInputStream in = new ObjectIntputStream(socket.getInputStream
            ());
19      ObjectOutputStream out = new ObjectOutputStream(socket.
            getOutputStream());
```

Enviar dados para o stream

```
20      out.writeObject(objeto);
```

Receber dados do stream

- Deve-se fazer o cast para o tipo de objeto específico

```
21      TipoObjeto resposta = (TipoObjeto) in.readObject();
```

Sockets TCP (Java) - Programa Servidor

```
22      //Passo 1: aguarda conexao
23      ServerSocket conexao = new ServerSocket(porta, tamanho_fila);
24      do{
25        //Passo 2: aguarda conexao do cliente
26        Socket socket = conexao.accept();
27
28        //Passo 3: obtem o stream de entrada e saida
29        DataInputStream in = new DataInputStream(socket.getInputStream());
30        DataOutputStream out = new DataOutputStream(socket.getOutputStream
              ());
31
32        //Passo 4: realiza a comunicacao conforme o protocolo
33        String data = in.readUTF();
34        out.writeUTF(data);
35
36        //Passo 5: fecha o socket com o cliente
37        socket.close();
38      } while(notExit());
39
40      //Passo 6: fecha o socket servidor
41      serverSocket.close();
```

Sockets TCP (Java) - Programa Cliente

```
42      InetAddress address = InetAddress.getbyName(name);
43
44      //Passo 1: Conecta o socket a porta do servidor
45      Socket socket = new Socket(address,port);
46
47      //Passo 2: cria streams entrada e saida
48      DataOutputStream out = new DataOutputStream(socket.getOutputStream()
            );
49      DataInputStream in = new DataInputStream(socket.getInputStream());
50
51      //Passo 3: realiza a comunicacao conforme protocolo
52      out.writeUTF(request);
53      String resposta = in.readUTF();
54
55      //Passo 4: fecha o socket
56      socket.close();
```

Código Exemplo

- Pegar no moodle os códigos de exemplo
- A aplicação exemplo trata-se de uma aplicação onde o cliente envia uma mensagem para o servidor e o servidor devolve (faz eco) a mesma mensagem ao cliente.

Exercícios

- Lista de Exercícios Sockets TCP

Próxima aula

- Threads em Sistemas Distribuídos