VOLTAR

Modelo Cliente/Servidor e Sockets em Java

Apresentar os conceitos sobre o modelo Cliente/Servidor e um exemplo de Socket em Java

NESTE TÓPICO

- Modelo Cliente/Servidor (Client/Server)
- Comunicação entre Cliente e Servidor
- Comunicação Orientada por Mensagem
- A Classe Servidor

Modelo Cliente/Servidor (*Client/Server*)

O modelo Cliente/Servidor é uma arquitetura de rede distribuída com papéis bem definidos entre seus componentes. O **Servidor** é um computador ou supercomputador que hospeda (*host*) serviços que podem ser acessados através de uma rede LAN, MAN, WAN ou PAN. Estes serviços executam tarefas específicas que são gerenciadas pelo sistema operacional servidor. Alguns exemplos de serviços são:

- Serviço de Diretório
- Serviço de Impressão
- Serviço de Banco de Dados
- Serviço Web
- Serviço de Aplicação

Estes serviços podem ser consumidos pelos usuários através de computadores Clientes. Estes **Clientes**, através de aplicações locais (quem executam no computador cliente), solicitam os serviços pela rede ao **Servidor**. O Servidor recebe a solicitação, faz a execução e devolve ao Cliente a resposta. Veja na Figura 1 um exemplo do modelo Cliente/Servidor.

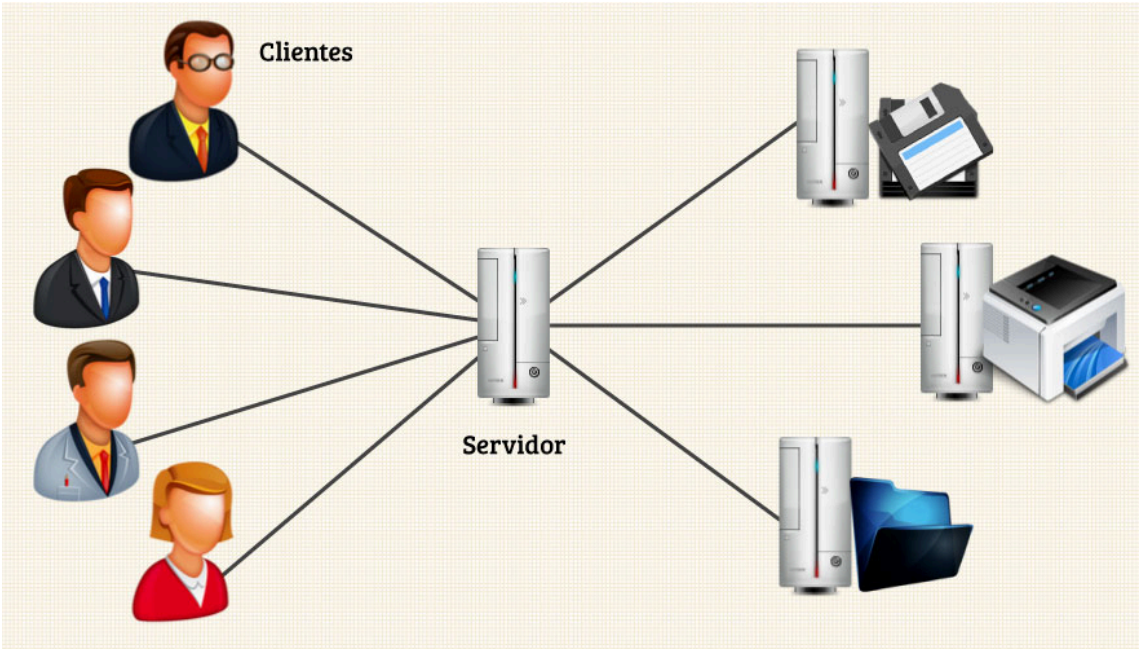

Figura 1 - Modelo Cliente/Servidor

Como pode ser visto na Figura 2, existem diversas alternativas para implementar um ambiente cliente/servidor, mas manter o cliente gordo (*fat clients*) poderá produzir muitos problemas, dentre eles o gerenciamento destes clientes. A proposta mais adequada é manter esses clientes magros (*thin clients*) e concentrar o restante no servidor. Em outras palavras, a máquina cliente deve conter somente a interface de usuário e regras básicas de validação de formulários de entrada de dados (cliente magro). No servidor, contém os serviços de processamento de aplicação e dados.

Comunicação entre Cliente e Servidor

A comunicação entre a aplicação cliente e o serviço em execução no servidor ocorre através de um terminal ou porta disponível. O servidor pode ter um ou muitos serviços em execução e cada um deles terá uma porta diferente. Os clientes se conectam nestas portas de serviço que são padronizadas mundialmente através do protocolo TCP (*Transmission Control Protocol*). Assim, o cliente faz uma requisição de conexão nesta porta, o servidor autentica este acesso (pode ser por usuário/senha) e estabelece uma conexão do cliente na porta do serviço. Alguns exemplos de porta de serviço é a porta 21 (FTP - *File Transport Protocol*) para transferência de arquivos e a porta 80 (HTTP - *Hyper Text Transport Protocol*) para os serviços Web.

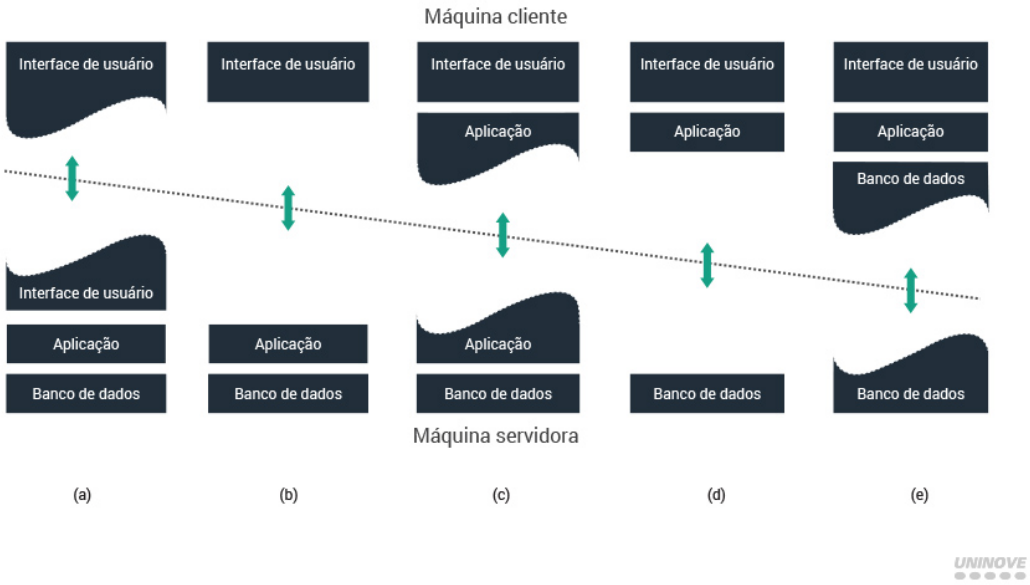

Figura 2 - Organização do modelo Cliente/Servidor e suas alternativas

Fonte: TANENBAUM, Andrew S., STEE, Maarten V. Sistemas Distribuídos - Princípios e Paradigmas. 2ª Edição. p. 25. São Paulo: 2007.

Comunicação Orientada por Mensagem

Algumas soluções distribuídas adotam o modelo de comunicação orientada por mensagem através da camada de transporte. Como pode ser visto na Figura 3, a camada de transporte utiliza os protocolos TCP e UDP (*User Datagram Protocol*) para a troca de mensagens por meio de portas de nível de serviço.

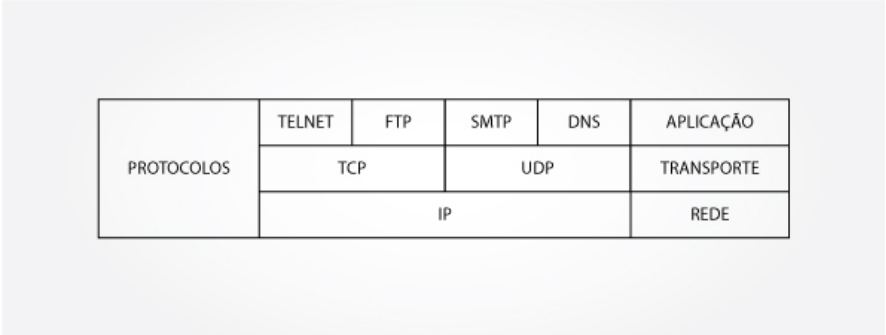

Figura 3 - Camada de Transporte e os Protocolos de Rede

O modelo de troca de mensagens a partir de portas TCP permite que desenvolvedores possam escrever seus programas em rede utilizando-se de um terminal de comunicação denominado por soquete (*sockets*). O soquete é utilizado pelo sistema operacional local (exemplo: Debian Linux) para um protocolo de transporte adotado. Assim, o sistema operacional de rede deve associar a porta com um endereço IP para que a troca de mensagens somente ocorra entre aqueles computadores e através daquela porta.

Em suma, o soquete é criado do lado da máquina cliente e da servidora (primitiva *Socket*) para estabelecer uma conexão (primitiva *Connect*) através de um endereço local (primitiva Bind). A partir da adoção destas primitivas, o cliente poderá enviar uma mensagem (primitiva *Send*) e o servidor recebe a mensagem (primitiva *Receive*).

A seguir, temos um simples exemplo em Java de programação orientada por troca de mensagens utilizando um soquete.

A Classe Servidor

A classe Servidor foi desenvolvida para criar o soquete na porta 9515. Esta porta receberá a conexão cliente e viabilizará a troca de mensagens entre o cliente e o servidor. Assim, a linha 11 instancia um novo soquete na porta 9515 no endereço 127.0.0.1 (localhost ou loopback local).

O *While(true)* na linha 15 mantém o servidor aceitando conexões de um ou mais clientes (linha 17) e na linha 21, estamos instanciamento um *Buffer Reader* (leitor) da classe *BufferedReader* que recebe outro leitor através do método construtor e junta os caracateres para formar uma *String* a partir do método *ReadLine* (linha 24).

Na linha 24, estamos lendo a mensagem enviada pelo cliente e atribuindo a variável msgCliente. Por sua vez, esta variável é formatada (linha 25) para caixa alta (maiúsculo) e impressa na saída (linha 26).

```
1.  import java.io.*;
2.  import java.net.*;
3.
4.  public class Servidor {
5.
6.      public static void main(String args[]) throws Exception {
7.
8.          String msgCliente;
9.          String msgCaixaAlta;
10.
11.         ServerSocket meuSoquete = new ServerSocket(9515);
12.
13.         System.out.println("Aguardando a conexão do Cliente...\n");
14.
15.         while (true) {
16.
17.             Socket minhaConexao = meuSoquete.accept();
18.
19.             System.out.println("Aguardando a mensagem do Cliente...\n");
20.
21.             BufferedReader entradaCliente = new BufferedReader(
22.                     new InputStreamReader(minhaConexao.getInputStream()));
23.
24.             msgCliente = entradaCliente.readLine();
25.             msgCaixaAlta = msgCliente.toUpperCase() + "\n";
26.             System.out.println("Mensagem do Cliente: " + msgCaixaAlta);
27.         }
28.     }
29. }
```

A Classe Cliente

A classe Cliente foi criada para criar uma conexão ao soquete na porta 9515 e enviar uma mensagem ao servidor.

O soquete é criado em *localhost* (127.0.0.1) ou *loopback* local na porta 9515 (linha 13). Na linha 17 a classe *DataOutputStream* é instanciada para escrever dados mais comuns durante a programação, tais como int, boolean, double, entre outros tipos. Assim como na classe Servidor, na linha 11 temos a instância sendo criada de um *Buffer Reader* (leitor) da classe *BufferedReader*. Esta classe recebe outro leitor através do método construtor e junta os caracateres para formar uma *String* a partir do método *ReadLine* (linha 21). A mensagem é enviada ao servidor a partir da linha 22.

As diversas mensagens (vide *System.out.println*) enviadas para a saída das duas classes (Servidor e Cliente), foram colocadas logo após a classe Cliente. Além disso, as mensagens foram colocadas na ordem que são transmitidas pelo Servidor (o primeiro a ser executado) e pelo Cliente.

```
1.  import java.io.*;
2.  import java.net.*;
3.
4.  class Cliente {
5.
6.      public static void main(String args[]) throws Exception {
7.          String mensagem;
8.
9.          System.out.println("Iniciando a conexão...\n");
10.
11.         BufferedReader entradaUsuario =  new BufferedReader(
12.                 new InputStreamReader(System.in));
13.         Socket soqueteCliente= new Socket("127.0.0.1", 9515);
14.
15.         System.out.println("Conectado!\n");
16.
17.          DataOutputStream saidaServidor = new DataOutputStream(soqueteCliente.getOutp
    utStream());
18.
19.         System.out.print("Digite a Mensagem: ");
20.
21.         mensagem = entradaUsuario.readLine();
22.         saidaServidor.writeBytes(mensagem + "\n");
23.
24.         System.out.println("Enviado ao Servidor: " + mensagem + "\n");
25.         System.out.println("Finalizando a conexão Cliente. \n");
26.
27.         soqueteCliente.close();
28.     }
29. }
```

Saída do Servidor - 1

Aguardando a conexão do Cliente...

Saída do Cliente - 1

Iniciando a conexão...

Conectado!

Digite a Mensagem:

Saída do Servidor - 2

Aguardando a mensagem do Cliente...

Saída do Cliente - 2

Digite a Mensagem: teste

Saída do Servidor - 3

Mensagem do Cliente: TESTE

Saída do Cliente - 3

Enviado ao Servidor: teste

Finalizando a conexão Cliente.

Quiz

Exercício

Modelo Cliente/Servidor e Sockets em Java

INICIAR

Referências

TANENBAUM, Andrew S., STEE, Maarten V. Sistemas Distribuídos - Princípios e Paradigmas. 2ª Edição. São Paulo: 2007.

DA COSTA, Celso Maciel; STRINGHINI, Denise; CAVALHEIRO, Gerson Geraldo Homrich. Programação Concorrente: Threads, MPI e PVM. Escola Regional de Alto Desempenho, II ERAD, captulo, v. 2, 2002.

SILBERSCHATZ, Abraham. Sistemas operacionais com Java. Elsevier Brasil, 2008.

Avalie este tópico

ANTERIOR
Problemas de Concorrência e Compartilhamento

Índice

PRÓXIMO
Remote Procedure Call: RMI, CORBA e SOAP

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda? (https://ava.un idCurso=)