Atividade de Programação Comunicação Simples em Rede

Victória Tomé Oliveira

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

1. RMI

O RMI (*Remote Method Invocation*) é uma interface de programação que permite a execução de chamadas remotas no estilo RPC em aplicações desenvolvidas em Java. É uma das abordagens da plataforma Java para prover as funcionalidades de uma plataforma de objetos distribuídos. O funcionamento de RMI trabalha com a arquitetura cliente-servidor. Consiste em dois programas, onde um seria o cliente e outro o servidor.

1.1. Servidor

O servidor instancia objetos remotos, o referencia com um nome e faz um ”vinculo”dele numa porta, onde este objeto espera por clientes que invoquem seus métodos.

1.2. Cliente

O cliente referencia remotamente um ou mais métodos de um objeto remoto. RMI fornece os mecanismos para que a comunicação entre cliente e servidor seja possível.

2. HTTP

O HTTP é um protocolo baseado em requisição e resposta. O cliente faz uma requisição e o servidor envia uma resposta. O código do servidor esta online, o cliente abre uma conexão com o servidor. A partir disto, o servidor responde perguntando a operação que o cliente deseja solicitar, o cliente envia os parâmetros. Quando recebe a resposta do servidor, a conexão é encerrada.

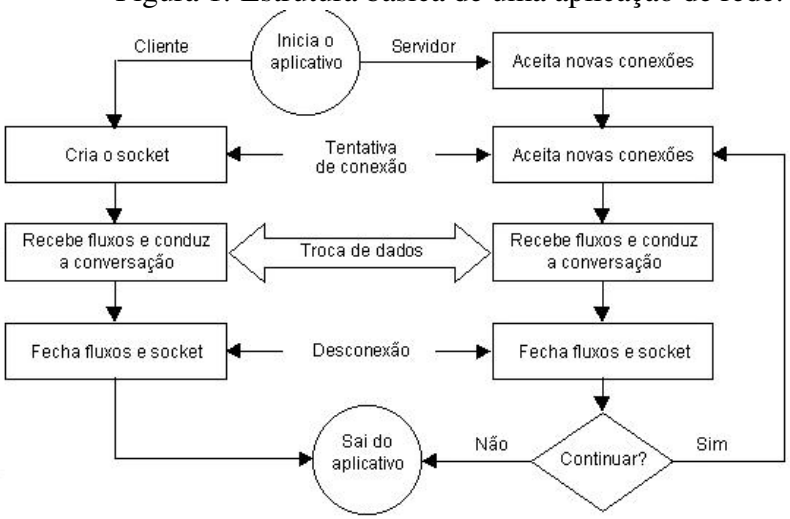
3. Sockets

Os *sockets* são compostos por um conjunto de primitivas do sistema operacional e foram originalmente desenvolvidos para o BSD Unix. Podem ser utilizados nos mais variados sistemas operacionais com recursos de comunicação em rede, sendo suportados pela maioria das linguagens de programação. *Sockets* são suportados em Java desde o JDK 1.0, para sua utilização devemos fazer uso das classes contidas no pacote java.net.

3.1. Estrutura básica de uma aplicação de rede

Uma aplicação que utiliza sockets normalmente é composta por uma parte servidora e diversos clientes. Um cliente solicita determinado serviço ao servidor, o servidor processa a solicitação e devolve a informação ao cliente. Muitos serviços podem ser disponibilizados numa mesma máquina, sendo então diferenciados não só pelo endereço IP, mas também por um número de porta. Porém, o mais comum é termos uma máquina dedicada oferecendo apenas um ou dois serviços, evitando assim a concorrência.

Figura 1. Estrutura básica de uma aplicação de rede.



3.1.1. Servidor

O serviço do computador que funciona como base deve, primeiro, abrir uma porta e ficar ouvindo até alguém tentar se conectar. Se o objeto for realmente criado, a porta é aberta. Após abrir a porta, precisamos esperar por um cliente fazer a conexão, assim que um cliente se conectar, o programa continuará. Por fim, basta ler todas as informações que o cliente nos enviar.

3.1.2. Cliente

O cliente abre uma conexão com o servidor, no IP 192.168.0.11 e porta 9090. O servidor responde perguntando a operação e os parâmetros. O cliente envia os parâmetros, quando recebe a resposta do servidor a conexão é encerrada. Usa-se os conceito de java.io para leitura do teclado e envio de mensagens para o servidor. Para a classe *Scanner*, tanto faz de onde que se lê ou escreve os dados, o importante é que esse *stream* seja um *InputStream*/*OutputStream*. É o poder das interfaces, do polimorfismo, aparecendo novamente.

4. GitHub

`https://github.com/victoriatome/SistemasDistribuidos_ComunicacoSimplesRede.git`