Universidade Federal do Vale do São Francisco – UNIVASF
Curso: Engenharia da Computação
Disciplina: Redes de Computadores I **Data de entrega:** 05/12/08
Professor: Leonardo Barreto Campos **Valor:** 2,0

Projeto - IV[1]

Neste projeto, o aluno irá estudar um simples servidor de Ping da Internet escrito em linguagem Java e implementar um **cliente** correspondente. A funcionalidade provida por esses programas é similar à dos programas de Ping padrão disponíveis nos sistemas operacionais modernos, exceto aqueles que usam o UDP em vez do ICMP (Internet Control Message Protocol) para se comunicar. (Java não provê meios diretos para interagir com o ICMP.)
O protocolo Ping permite a uma máquina cliente enviar um pacote de dados para uma máquina remota, a qual retornará o dado para o cliente sem modificações (uma ação conhecida como eco). Entre outros usuários, o protocolo Ping permite aos hospedeiros determinarem o tempo de resposta de outras máquinas.
Complete o código para o servidor de Ping abaixo. Seu trabalho será escrever o cliente Ping.

Código do servidor

O código a seguir implementa por completo o servidor de Ping. Você precisará compilar e executar este código. Estude-o cuidadosamente, pois ele irá ajudá-lo a escrever seu cliente de Ping.

```
import.java.io.*;
import.java.net.*;
import.java.util.*;

/*
 * Servidor para processar as requisições de Ping sobre UDP.
 */
public class PingServer
{
private static final doublé LOSS_RATE = 0.3;
private static final int AVERAGE_DELAY = 100; //milliseconds

public static void main(String[]args) throws Exception
{
// Obter o argumento da linha de comando.
if (args.length !=1) {
System.out.pritln(“Required arguments: port”);
return;
}
int port = Integer.parseInt(args[0]);

// Criar um gerador de números aleatórios para uso em simulação de perda de
pacotes e atrasos na rede.
Random random = new Random();

// Criar um socket de datagrama para receber e enviar pacotes UDP através
da porta especificada na linha de comando.
DatagramSocket socket = new DatagramSocket(port);
```

[1] Laboratório retirado do site
http://wps.aw.com/wps/media/objects/2592/2654687/tarefas_programacao.zip

```
// Loop de processamento.
while(true);
// Criar um pacote de datagrama para comportar o pacote UDP // de chegada.
DatagramPacket request = new DatagramPacket(new byte[1024],1024);
// Bloquear até que o hospedeiro receba o pacote UDP.
Socket.receive(request);

// Imprimir os dados recebidos.
printData(request);

// Decidir se responde, ou simula perda de pacotes.
if(random.nextDouble() < LOSS_RATE) {
System.out.println("Reply not sent.");
Continue;
}

// Simular o atraso da rede.
Thread.sleep((int)(random.nextDouble) * 2 * AVERAGE_DELAY));

// Enviar resposta.
InetAddress clientHost = request.getAddress();
Int clientPort = request.getPort();
Byte[]buf = request.getData();
DatagramPacket reply = new DatagramPacket(buf, buf.length, clientHost,
clientPort);
Socket.send(reply);

System.out.println("Reply sent.");
}
}

/*
 * Imprimir o dado de Ping para o trecho de saída padrão.
 */

private static void printData(DatagramPacket request) throws Exception
{
// Obter referências para a ordem de pacotes de bytes.
byte[] buf = request.getData();

// Envolver os bytes numa cadeia de entrada vetor de bytes, de modo que
você possa ler os dados como uma cadeia de bytes.
ByteArrayInputStream bais = new ByteArrayInputstream(buf);

// Envolver a cadeia de saída do vetor bytes num leitor de cadeia de
entrada, de modo que você possa ler os dados como uma cadeia de caracteres.
InputStreamReader isr = new InputStreamReader(bais);

// Envolver o leitor de cadeia de entrada num leitor com armazenagem, de
modo que você possa ler os dados de caracteres linha a linha. (A linha é
uma seqüência de caracteres terminados por alguma combinação de \r e \n.)
BufferedReader br = new BufferedReader(isr);

// O dado da mensagem está contido numa única linha, então leia esta linha.
String line = br.readLine();

// Imprimir o endereço do hospedeiro e o dado recebido dele.
System.out.println(
"Received from" +
request.getAddress().getHostAddress()+
":" +
new String(line));
}
```

```
}
```

O servidor fica num loop infinito de escuta pela chegada de pacotes UDP. Quando um pacote chega, o servidor simplesmente envia o dado encapsulado de volta para o cliente.

Perda de pacotes

O UDP provê aplicações com serviço de transporte não confiável, pois as mensagens podem se perder pela rede devido a um overflow na fila do roteador ou por outras razões. Em contraste a isso, o TCP fornece aplicações com um serviço de transporte confiável, e cada pacote perdido é retransmitido até que ele seja recebido com sucesso. Aplicações que usam o UDP para comunicação precisam implementar alguma segurança separadamente no nível de aplicação (cada aplicação pode implementar uma política diferente, de acordo com necessidades específicas).
Devido ao fato de a perda de pacotes ser rara, ou até mesmo inexistente, em uma rede típica, o servidor neste laboratório injeta perda artificial para simular os efeitos da perda de pacotes na rede. O servidor possui um parâmetro LOSS_RATE, que determina qual a porcentagem de pacotes deve ser perdida.
O servidor também possui outro parâmetro, AVERAGE_DELAY, que é usado para simular o atraso de transmissão ao enviar um pacote pela Internet. Você deve ajustar o AVERAGE_DELAY com um valor positivo quando o cliente e o servidor forem estar na mesma máquina, ou quando as máquinas estiverem muito perto fisicamente na rede. Você pode ajustar o AVERAGE_DELAY em 0 (zero) para encontrar o tempo de transmissão verdadeiro dos seus pacotes.

Compilando e executando o servidor

Para compilar o servidor, faça o seguinte:

```
javac PingServer.java
```

Para executar o servidor, faça o seguinte:

```
java PingServer port
```

onde port é o número da porta que o servidor escuta. Lembre que você deve usar um número de porta maior do que 1024, pois apenas os processos executando no modo root (administrador) possuem privilégio de usar portas menores que 1024.
Nota: Se você obtiver um erro de classe não encontrada quando executar o comando acima, você precisará dizer para o Java olhar no diretório atual para resolver as referências de classe. Nesse caso, os comandos são:

```
java –classpath . PingServer port
```

Sua tarefa: o cliente

Você deve escrever o cliente de modo que ele envie 10 requisições de Ping para o servidor, separadas por aproximadamente 1 segundo. Cada mensagem contém uma carga útil de dados que inclui a palavra PING, um número de seqüência, e uma marca de tempo. Após enviar cada pacote, o cliente espera um segundo para receber a resposta. Se um segundo se passar sem uma resposta do servidor, então o cliente pode supor que esse pacote ou o pacote de resposta do servidor se perdeu pela rede.

Dica: Copie e cole o PingServer, renomeie o código para PingClient e então modifique-o.
Você deve escrever o cliente de modo que ele inicie com o seguinte comando:

```
java PingClient host port
```

onde hospedeiro é o nome do computador em que o servidor está sendo executado e port é o número da porta que ele está escutando. Note que você pode executar o cliente e o servidor em diferentes máquinas ou na mesma.
O cliente deve enviar 10 Pings para o servidor. Como o UDP é um protocolo não confiável, alguns dos pacotes enviados pelo cliente ou pelo servidor podem ser perdidos. Por essa razão, o cliente não pode esperar indefinidamente pela resposta a uma mensagem de Ping. Você deve fazer com que o cliente espere até um segundo por uma resposta; se nenhuma resposta for recebida, ele presume que o pacote foi perdido durante a transmissão. Você precisará pesquisar a API para o DatagramSocket de modo a descobrir como se ajusta o valor de tempo de expiração num socket de datagrama.
Ao desenvolver seu código, você deve executar o servidor de Ping em sua máquina e testar seu cliente enviando pacotes para o hospedeiro local (ou, 127.0.0.1). Após o completo debug do seu código, você deve ver como sua aplicação se comunica através da rede com um servidor de Ping. **(Valor = 1,0)**

Formato das mensagens

As mensagens de Ping neste laboratório são formatadas de modo simples. Cada mensagem contém uma seqüência de caracteres terminados por um caracter de retorno (r) e um caráter de mudança de linha (n). A mensagem contém a seguinte string:

```
PING sequence_number time CRLF
```

onde sequence_number começa em 0 (zero) e progride até 9, para cada mensagem sucessiva de Ping enviada pelo cliente; time é o tempo do momento em que o cliente enviou a mensagem e CRLF representa o retorno e linha de caracteres que finalizam a linha.

Exercícios

Quando você terminar de escrever seu código, crie um novo programa com as seguintes atualizações:
1) No ponto atual, o programa calcula o tempo de transmissão de cada pacote e os imprime individualmente. Modifique isso para corresponder ao modo de funcionamento do programas de Ping padrões. Você deverá informar os RTTs mínimo, máximo e médio. **(Valor = 0,5)**
2) O programa básico envia um novo Ping imediatamente quando recebe uma resposta. Modifique-o de modo que ele envie exatamente 1 Ping por segundo, similar ao modo como programas de Ping padrões funcionam. Dica: Use as classes Timer e TimerTask em java.util. **(Valor = 0,5)**

OBS: Caso exista dois ou mais códigos semelhantes, os desenvolvedores serão convocados para explicar criteriosamente a lógica e os conceitos contidos no programa.