Departamento de Informática
Faculdade de Ciências e Tecnologia
UNIVERSIDADE NOVA DE LISBOA

27-07-2001 - 0010

Licenciatura em Engenharia Informática

Redes de Computadores, 1º Semestre 2002/2003

Teste/Avaliação, 27/Novembro//2002

Notas:

- Duração: 1h30m
- Leia completamente e com atenção cada questão antes de responder.
- A correcta interpretação do enunciado é um factor de avaliação do teste.
- O enunciado tem **3 questões,** divididas em diferentes alíneas.

---------- A preencher pelos alunos ----------

Nº de aluno: _____ Nome: _____

Classificação (a preencher pelo docente):

Questão 1	Questão 2	Questão 3

Questão 1)

Para as seguintes perguntas, indique a resposta que julga ser a mais correcta com uma seta → . Tenha atenção que respostas erradas podem retirar pontuação.

1. **A programação de aplicações baseadas nos protocolos da pilha TCP/IP segundo o modelo cliente/servidor pressupõe que um das componentes arranca em primeiro lugar:**
   a) Não. Tanto o cliente ou servidor podem arrancar em primeiro lugar.
   b) Sim. O servidor deve ser o primeiro a executar.
   c) Sim. O cliente é sempre o primeiro a executar.
   d) Sim. O servidor, mas apenas para aplicações programadas na linguagem JAVA.

2. **No modelo cliente/servidor para aplicações baseadas nos protocolos da pilha TCP/IP é necessário que:**
   a) Os clientes conheçam o endereço IP e o process id (PID) do servidor.
   b) O endereço IP e o porto do servidor sejam dinâmicos.
   c) O endereço IP e o porto do cliente sejam fixos.
   d) A localização do servidor (endereço IP e o porto) sejam conhecidos previamente pelos clientes.

3. **Nas aplicações baseadas no protocolo TCP segundo o modelo cliente/servidor, a aplicação que estabelece a conexão é:**
   a) O cliente, ou o servidor se este criar o socket e imediatamente depois fizer *accept*.
   b) Sempre o cliente.
   c) Sempre o servidor.
   d) Nunca o servidor, pois depois de criar o socket fica bloqueado no accept à espera das conexões originadas pelos clientes.

4. **Nas aplicações baseadas no protocolo TCP, implementadas segundo o modelo cliente/servidor, envolvendo um cenário com 1 servidor e 1 cliente, em que ambos já estão a receber e/ou a enviar dados, o número mínimo de sockets abertos é:**
   a) 2, um no cliente e um no servidor.
   b) 3, dois no cliente e um no servidor.
   c) 3, um no cliente e dois no servidor.
   d) 4, dois sockets em cada componente autónoma.

5. **Nas aplicações baseadas no protocolo TCP segundo o modelo cliente/servidor, qual a aplicação que deve começar a escrever dados para o canal:**
   a) O cliente ou servidor, mas sempre o primeiro a criar o canal.
   b) O cliente.
   c) O servidor.
   d) Qualquer um deles, inclusivamente ao mesmo tempo.

6. **Nas aplicações baseadas no protocolo TCP segundo o modelo cliente/servidor, a aplicação que deve começar a ler dados do canal deverá ser:**
   a) O cliente ou servidor, vai depender do protocolo aplicacional.
   b) Sempre o cliente.
   c) Sempre o servidor.
   d) Qualquer um deles, inclusivamente ao mesmo tempo.

7. **Nas aplicações baseadas no protocolo UDP segundo o modelo cliente/servidor, a comunicação realiza-se à custa de:**
   a) Troca de mensagens num canal bi-direccional.
   b) Troca fiável de um fluxo (stream) de bytes num canal bi-directional, mas onde a troca da ordem de entrega e sua duplicação de mensagens pode acontecer.
   c) Troca não fiável de mensagens avulsas de bytes, cuja ordem pode não ser garantida.
   d) Troca não fiável de um fluxo (stream) de bytes, cuja ordem pode não ser garantida, mas geralmente é preservada.

8. **Nas aplicações baseadas no protocolo UDP, implementadas segundo o modelo cliente/servidor, envolvendo um cenário com 1 servidor e 1 cliente, em que ambos já estão a receber e/ou a enviar dados, o número mínimo de sockets abertos é:**
   a) 2, um no cliente e um no servidor.
   b) 3, dois no cliente e um no servidor.
   c) 3, um no cliente e dois no servidor.
   d) 4, dois em cada componente, igual ao TCP.

9. **Nas aplicações baseadas no protocolo UDP segundo o modelo cliente/servidor, a aplicação que deve deve fazer/enviar a primeira comunicação é:**
   a) O cliente ou servidor, mas sempre o primeiro a criar o socket.
   b) Sempre o cliente.
   c) Sempre o servidor.
   d) Qualquer um deles, inclusivamente ao mesmo tempo.

10. **Nas aplicações baseadas no protocolo UDP segundo o modelo cliente/servidor, a aplicação que deve começar a ler dados do socket:**
   a) O cliente ou servidor, vai depender do protocolo aplicacional.
   b) Sempre o cliente.
   c) Sempre o servidor.
   d) Qualquer um deles, inclusivamente ao mesmo tempo.

11. **Os pedidos (HTTP 1.0) enviados por um browser WEB a um servidor ou a um proxy (para o mesmo URL):**
   a) Podem ser exactamente iguais
   b) São mais curtos (mensagens mais curtas) se forem do browser para o proxy.
   c) Diferem porque o documento pedido tem que ser especificado de maneira diferente.
   d) O browser nunca pode fazer pedidos directamente a um proxy.

12. **A resposta a um pedido de uma imagem (por HTTP 1.0) enviado por um browser WEB a um servidor ou a um proxy:**
   a) Implica que o browser leia do seu socket uma única string, após o que, o servidor fecha a conexão.
   b) Implica que o browser leia do seu socket uma sequência de strings até receber uma string vazia.
   c) Implica que o browser leia do seu socket uma sequência de "byte arrays" terminando a recepção com o fecho da conexão pelo servidor ou pelo proxy.
   d) Implica que o browser leia do seu socket uma sequência de strings até uma string vazia e depois uma sequência de "byte arrays" até que se dê o fecho da conexão pelo servidor ou pelo proxy.

13. **O tftp (*trivial file transfer protocol*) é um protocolo que concretiza:**
   a) um serviço de transferência de ficheiros suportado em TCP
   b) um serviço de transferência de ficheiros suportado em UDP
   c) um serviço de transferência de ficheiros em TCP ou UDP, depende das opções do cliente.
   d) para maior flexibilidade concretiza um serviço de transferência de ficheiros, directamente suportado no protocolo IP.

14. **Se num cliente usando sockets TCP na linguagem JAVA receber uma excepção "java.net.UnknownHostException:**
   a) Isso resulta do facto de o cliente ter enviado um pedido sem criar o socket.
   b) Isso resulta do facto de o cliente não ter conseguido obter o endereço IP do servidor dado o nome do mesmo.
   c) Não é possível obter essa excepção em TCP visto que essa excepção só acontece em UDP.
   d) O servidor estava numa porta diferente da porta para a qual o cliente solicitou a conexão.

15. **Se num cliente usando sockets UDP na linguagem JAVA receber uma excepção "java.net.SocketTimeoutException:**
   a) Isso resulta do facto de o cliente ter feito setSoTimeout (0) no seu socket.
   b) Isso resulta do facto de o cliente não ter conseguido obter o endereço IP do servidor dado o nome do mesmo.
   c) Não é possível obter essa excepção em UDP visto que essa excepção só acontece em TCP.
   d) Isso resulta do facto do cliente ter colocado o socket em modo não-bloqueante e ter procedido a uma opração de leitura no mesmo sem sucesso.

Questão 2) VER CÓDIGO ANEXO Q2.

Considere o código da aplicação baseada no modelo cliente/servidor e programada com *sockets*, em linguagem JAVA, usando protocolos da pilha TCP/IP.

a) Em cada componente existe um título: CÓDIGO X1 e CÓDIGO X2. Diga qual é a componente cliente e servidor:

X1 = _____
X2 = _____

b) O protocolo de transporte que está a ser usado para suportar a comunicação cliente/servidor é UDP. Verdadeiro ou Falso ? Justifique.

VERDADEIRO OU FALSO ? _____
JUSTIFICAÇÃO ?

c) Na verdade há um pequeno erro no código mostrado. Embora cliente e servidor compilem e executem, de facto as mensagens do cliente não conseguem chegar ao servidor. Descubra o erro e corrija-o (faça a correcção no próprio código).

>>> Responda utilizando o próprio código fornecido.

d) A instrução que está comentada com "**// XXXXXXXXX**" no **código do SERVIDOR** é bloqueante ou não bloqueante ? Se responder que é bloqueante diga o que tem que acontecer para se desbloquear.

A instrução comentada com "/XXXXXXX é:
bloqueante ou não bloqueante ? _____

Se é bloqueante o que é preciso acontecer para se desbloquear ?

e) A instrução que está comentada com "**// YYYYYYYYY**" no **código do CLIENTE** é bloqueante ou não bloqueante ? Se responder que é bloqueante diga o que tem que acontecer para se desbloquear.

A instrução comentada com "/YYYYYYY é:
bloqueante ou não bloqueante ? _____

Se é bloqueante o que é preciso acontecer para se desbloquear ?

f) Para que serve o valor da variável ALUNO e para que serve ?

Questão 3) VER CÓDIGO ANEXO Q3

Considere a aplicação cliente/servidor do ANEXO Q3.

a) Suponha que pretende arrancar simultaneamente 10 **clientes**, em 10 computadores diferentes para interagirem com o servidor fornecido. É necessário fazer alterações ao código do cliente para tal ser possível ? Se achar que sim faça as alterações necessárias no código do cliente.

>>> Responda utilizando o próprio código fornecido.

b) O código do servidor faz dele um servidor concorrente ou não-concorrente ? Justifique.

O servidor é _____
Porque:

c) Se achar que é concorrente transforme-o em não-concorrente e se achar que é não-concorrente transforme-o em concorrente.

>>> Responda utilizando o próprio código fornecido.

d) A alteração que produziu em c) implicam que o cliente também precise de ser alterado ? Responda apenas SIM ou NÃO e se SIM resuma o tipo de alteração (não precisa de concretizar essa eventual alteração em código).

SIM OU NÃO ? _____

Se SIM a alteração consistiria em :

ANEXO Q2

```
// CÓDIGO X1:
import java.io.*;
import java.net.*;

public class XXXsend {
   static final int aluno = 10322
   public static void main (String args[]) throws Exception {
      if (args.length !=2) {
           System.out.println ("usage: java XXXsend <host> <message>");
           System.exit(0);
      }

      InetAddress addr= InetAddress.getByName(args[0]);
      System.out.println ("HOST DESTINO: " + args[0] + " MENSAGEM: " + args[1]);

      int msglen= args[1].length();
      byte[] msg= new byte[msglen];
      args[1].getBytes(0,msglen, msg,0);

      DatagramPacket pacote= new DatagramPacket(msg, msglen, addr, aluno);

      DatagramSocket socket= new DatagramSocket ();  // XXXXXXXXX
      socket.send (pacote); // YYYYYYYY
      }
  }
-------------------------------------------------------------------------------

// CÓDIGO X2:
import java.io.*;
import java.net.*;

public class YYYreceive {
   static final int aluno=12232;
   public static void main (String args[]) throws Exception
   {
     byte[]  buffer=new byte[1024];
     String s;

      DatagramPacket pacote= new DatagramPacket(buffer, buffer.length);
      DatagramSocket socket= new DatagramSocket (aluno); // XXXXXXXXX

      for(;;)
        {
        socket.receive(pacote);  // YYYYYYYYYYY
        s = new String(pacote.getData(),0,0,pacote.getLength());
        System.out.println("Mensagem recebida do host "+ pacote.getAddress().get
HostName() + "do porto " + pacote.getPort() + ": " + s);

        }
     }
}
```

ANEXO Q3

```
// CODIGO X1

import java.io.*
import java.net.*

public class SimpleClient  {
   public static void main (String[] args ) throws Exception {
        Socket sock = new Socket (args [0], Integer.parseInt (args[1]));

   BufferedReader bris=
        new  BufferedReader (new InputStreamReader (sock.getInputStream()));

   String line=null;
   line = bris.readLine();

   System.out.println(line);

    bris.close ();
    sock.close();
   }
}
```

```
// CODIGO X2

import java.io.*
import java.net.*
import java.text.SimpleDateFormat
import java.util.Date

public class RobustServer implements Runnable {
   public static int DEFAULT = 23456 ;
   private Socket request = null ;

   public RobustServer  (Socket request) {
        this.request = request;
   }


   public void run() {
        try {
                handleRequest ();
        }       catch (IOException ioe) {
                System.out.println ("Erro no atendimento do serviço." +  "Vou parar !");
                System.exit(0);
        }
   }
```

```
    public void handleRequest () throws IOException {
        Date date = new Date (System.currentTimeMillis()));
                // cálculo da data actual em milesegundos
                SimpleDateFormat formattedDate =
                new SimpleDateFormat("EEE MMM d hh:mm:ss z yyyy");
                // formatar a data da forma   Wed Nov 27  15:45:00  GMT  2002

        PrinterWriter pwos = new PrintWriter (request.getOutputStream () );

        pwos.println (formattedDate.format (date));
        pwos.close ();
        request.close ();
}


public static void main (String[ ]  args )  throws Exception {
    ServerSocket server = new ServerSocket (DEFAULT);

    while (true) {

        Socket socket = server.accept();

        RobustServer handler = new RobustServer (socket);
        Thread thread = new Thread (handler);
        thread.start();

    }
}
```
