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

Licenciatura em Engenharia Informática
PROVA DE TESTE PRÁTICO – Redes de Computadores
1º Semestre, 2003/2004

NOTAS:
Leia com atenção cada questão antes de responder. A interpretação do enunciado de cada pergunta é um factor de avaliação do teste.
Pode utilizar elementos pessoais de consulta. A duração do teste é 2h00.

O enunciado contém 7 páginas que devem ser entregues com a resposta ao teste.

NOME:_____ Nº Aluno: _____

1) Considere que pretende criar, usando o protocolo UDP, um sistema cliente/servidor que permita verificar a validade de um par nome de utilizador/senha.

O cliente deve receber como parâmetros o nome do utilizador, a senha respectiva e o nome da máquina na qual funciona o servidor. Após contactar o servidor, deve apresentar no terminal a palavra "Correcto" ou "Incorrecto" caso o par nome de utilizador/senha seja válido ou inválido, respectivamente.

O servidor deve fornecer o serviço pretendido no porto 6162, recorrendo, sempre que necessário, à função **validLogin** disponível na classe **MySystem**. No servidor, as excepções ocorridas durante o processamento de um pedido do cliente não devem levar ao encerramento do servidor, mas antes à falha do pedido respectivo: em consequência, o servidor não deve enviar nenhuma mensagem para o cliente, aguardando a recepção do próximo pedido.

a. Complete o anexo A.1 com o código do cliente, não se preocupando com eventuais perdas de mensagens.

b. Complete o anexo A.2 com o código do servidor.

c. Complete o anexo A.3 com o código do cliente, tratando eventuais perdas de mensagens através da técnica habitual de retransmissão de mensagens. Se não conseguir obter o resultado através da retransmissão da mensagem três vezes, o cliente deve desistir a apresentar no terminal a palavra "Incorrecto".

d. Indique as modificações que deve introduzir no código do servidor para que o código apresentado na alínea anterior funcione correctamente.

2) Considere que existe um programa cliente que envia uma sequência de mensagens para um servidor usando o protocolo UDP. Suponha que o servidor recebe todos os pacotes enviados pela ordem que o cliente enviou. Nesse caso, complete a seguinte tabela indicando

Conteúdo do pacote enviado	Dimensão escrita	Conteúdo do pacote recebido	Dimensão do buffer de leitura
12345	5	12345	5
1234567	7		5
12	2		5
123456	6		5

3) Indique se as afirmações seguintes são verdadeiras ou falsas, justificando as afirmações falsas:

a. Um socket datagrama pode ser usado para enviar mensagens para múltiplos destinos.

Verdadeiro Falso porque...

b. Para enviar uma mensagem multicast para um dado endereço multicast, o seu programa deve-se juntar ao grupo antes de enviar a mensagem.

Verdadeiro Falso porque...

c. Após ter efectuado a operação "joinGroup" num dado socket multicast apenas é possível enviar mensagens para o grupo ao qual se juntou.

Verdadeiro Falso porque...

d. Na recepção de mensagens UDP num socket datagrama, a execução do método receive() bloqueia-se sempre, independentemente das operações executadas sobre o socket anteriormente.

Verdadeiro Falso porque...

4) No âmbito do segundo trabalho prático referente ao desenvolvimento de um programa de chat que permite a transferência de ficheiros a partir de um servidor, indique se as afirmações seguintes são verdadeiras ou falsas, justificando as afirmações falsas (considere uma implementação correcta e completa do enunciado e não a sua própria implementação):

a. O servidor pode funcionar em qualquer máquina desde que se junte a um IP multicast pré-definido: os clientes conseguem contactar o servidor enviando mensagens para esse endereço multicast.

Verdadeiro Falso porque...

b. Na fase 2, em que os pacotes do ficheiro são enviados sequencialmente (modo burst) sem nenhum feedback dos clientes, considera-se que a transferência foi bem sucedida caso o número de pacotes recebidos seja igual ao número de pacotes esperado.

Verdadeiro Falso porque...

c. Na fase 3, em que a transferência do ficheiro é efectuada recorrendo a um protocolo de stop&wait,

é possível usar apenas dois números de sequência (zero e um) para identificar alternativamente os pacotes a enviar.

Verdadeiro Falso porque...

Ainda relativamente ao trabalho prático, responda às seguintes perguntas.

d. Na fase 4, em que cada janela era retransmitida completamente, explique a condição ou condições que despoletavam a retransmissão da janela.

e. Explique brevemente uma solução para aumentar a janela do emissor e discuta os problemas que podem surgir.

5) Considere o seguinte pedaço de código que cria e parametriza um socket multicast. Diga qual o efeito de executar as instruções comentadas com X e Y. Justifique.

```
int port = 5656;
InetAddress group= InetAddress.betByName ("238.255.10.10");
MulticastSocket ms= new MulticastSocket(port);
ms.setSoTimeout (10000);                    // X
ms.setTimeToLive (1);                       // Y
ms.joinGroup (group);
...
```

6) Considere a família de protocolos do tipo "janela deslizante" para canais de comunicação ponto a ponto do nível transporte. Responde breve, mas justificadamente, às seguintes questões:

a. Como é que um valor de "time-out" demasiado curto pode prejudicar a correcção do protocolo, isto é, introduzir erros na transmissão ?

b. A introdução de confirmações de falha ("Negative Acknowledges") pode servir para melhorar o protocolo "Stop&Wait" ?

ANEXO A.1

```
import java.io.*;
```

```
import java.net.*;

class Client
{
   public static void main( String[] args) throws Exception {
      if( args.length != 3)
         return;
      String user = args[0];     // nome do utilizador
      String pwd = args[1];      // senha
      String server = args[2];   // nome do servidor
      boolean result;  // variável para guardar resultado


      ByteArrayOutputStream baos = new ByteArrayOutputStream();
      DataOutputStream dos = new DataOutputStream( baos);
      dos.writeUTF(user);
      dos.writeUTF(pwd);
      dos.close();
      byte []buf = baos.toByteArray();

      InetAddress serverIp = InetAddress.getByName( server);
      DatagramPacket packet = new DatagramPacket( buf, buf.length, serverIp,
6162);

      DatagramSocket sock = new DatagramSocket();
      sock.send( packet);

      buf = new byte[100];
      packet = new DatagramPacket( buf, buf.length);
      sock.receive( packet);

      ByteArrayInputStream bais = new ByteArrayInputStream( packet.getData(), 0,
packet.getLength());
      DataInputStream dis = new DataInputStream( bais);
      boolean result = dis.readBoolean();




      if( result) System.out.println( "Correcto");
      else System.out.println( "Incorrecto");
   }
}
```

ANEXO A.2

```
import java.io.*;
import java.net.*;
public class Server
{
   public static void main( String argv[]) throws Exception {
      DatagramSocket sock = new DatagramSocket( 6162);
      byte []buf = new byte[1024];
      DatagramPacket packet = new DatagramPacket( buf, buf.length);
      for( ; ; ) {
         packet.setLength( buf.length);
         sock.receive( packet);
         try {
            ByteArrayInputStream bais = new
ByteArrayInputStream( packet.getData(), 0, packet.getLength());
            DataInputStream dis = new DataInputStream( bais);
            String user = dis.readUTF();
            String pwd = dis.readUTF();
            System.out.println(user + ":" + pwd);

            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            DataOutputStream dos = new DataOutputStream( baos);
            dos.writeBoolean( MySystem.validLogin( user, pwd));
            dos.close();
            byte []buf2 = baos.toByteArray();

            DatagramPacket packet2 = new DatagramPacket( buf2, buf2.length,
                     packet.getAddress(), packet.getPort());
            sock.send( packet2);
         }catch( Exception e) {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            DataOutputStream dos = new DataOutputStream( baos);
            dos.writeBoolean( false);
            dos.close();
            byte []buf2 = baos.toByteArray();

            DatagramPacket packet2 = new DatagramPacket( buf2, buf2.length,
                     packet.getAddress(), packet.getPort());
            sock.send( packet2);
         }
      }
   }
}

class MySystem
```

```
{
    /**
     * Devolve "true" se "pwd" for a password do utilizador "user".
     * Caso contrário devolve "false".
     */
    public static boolean validLogin( String user, String pwd) {
        return user.equals(pwd);
    }
}
```

ANEXO A.3

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

class Client
{

    public static void main( String[] args) throws Exception {
        if( args.length != 3)
            return;
        String user = args[0];     // nome do utilizador
        String pwd = args[1];      // senha
        String server = args[2];   // nome do servidor
        boolean result;
```

```
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream( baos);
        dos.writeUTF(user);
        dos.writeUTF(pwd);
        dos.close();
        byte []buf = baos.toByteArray();

        InetAddress serverIp = InetAddress.getByName( server);
        DatagramPacket packet = new DatagramPacket( buf, buf.length, serverIp,
6162);

        DatagramSocket sock = new DatagramSocket();
        sock.send( packet);

        buf = new byte[100];
        packet2 = new DatagramPacket( buf, buf.length);
        int count = 0;
        sock.setSoTimeout( 3000);
        for( ; count < 3; count++) {
           try {
              sock.receive( packet2);

              ByteArrayInputStream bais = new
ByteArrayInputStream( packet2.getData(), 0, packet2.getLength());
              DataInputStream dis = new DataInputStream( bais);
              boolean result = dis.readBoolean();
              break;
           } catch( SocketTimeoutException e) {
              // do nothing
           }
        }
        if( result) System.out.println( "Correcto");
        else System.out.println( "Incorrecto");
     }
}
```