	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 XXXXXXXXXX páginas que devem ser entregues com a resposta ao teste.

NOME:_____ **Nº Aluno:** _____

1) Suponha que se pretende implementar um sistema cliente/servidor usando o protocolo UDP no qual o cliente submete um comando para execução no servidor e apresenta o resultado da execução do mesmo.

O cliente obtém o comando a executar do utilizador, através da consola. Um comando a executar é composto por uma linha de texto. Ao iniciar o cliente, o nome do servidor a contactar deve ser o primeiro parâmetro especificado.

No servidor, a execução de um comando é efectuada através da função ***execute*** definida estaticamente na classe ***MySystem***, a qual devolve um objecto do tipo ***ExecResult*** com o resultado da execução do comando indicado. Ambos os campos do resultado devem ser transmitidos para o cliente garantindo que o tempo de execução é transmitido sem perda de precisão. A execução da função ***execute*** pode lançar uma excepção – neste caso, o deve-se considerar que o tempo de execução é "0.0" e o texto do resultado é "Erro na execução do comando indicado".

```
class ExecResult
{
    double time;     // tempo de execução
    String result;   // texto do resultado
}

class MySystem {
    /**
     * Executa o comando cmd
     */
    public static ExecResult execute( String cmd) throws Exception {
        ....
    }
}
```

a. Apresente o código do cliente e de um servidor não concorrente na linguagem Java, sem se preocupar com eventuais perdas de pacotes.

b. Suponha agora que o texto do resultado da função ***execute*** definida na classe ***MySystem*** pode ser de dois tipos: (1) uma *String* da forma "@servidor:novo_cmd", indicando que o resultado da execução do comando indicado deve ser obtido submetendo o comando "novo_cmd" no servidor "servidor" (por exemplo, caso o resultado seja "@asc.di.fct.unl.pt:ls /etc", deve-se submeter o comando "ls /etc" no servidor localizado na máquina "asc.di.fct.unl.pt"; (2) qualquer outro, representando o resultado da execução do comando indicado.

Apresente o código do servidor tratando eventuais perdas de mensagens no contacto com o novo servidor através da técnica habitual de retransmissão de mensagens. Se não conseguir obter o resultado através da retransmissão da mensagem três vezes deve-se considerar que o tempo de execução é "0.0" e o texto do resultado é "Erro na execução do comando indicado".

2) Suponha que se pretende criar um sistema para medir a poluição do ar no campus da FCT-UNL. Para tal, serão usadas estações de medição autónomas, equipadas com um computador e uma placa de rede

802.11b capaz de comunicar usando os protocolos disponíveis na pilha de protocolos TCP/IP. O sistema a desenvolver deve permitir a adição e remoção de estações de medição em qualquer momento.

a. Supondo que o objectivo do sistema é permitir obter a poluição média do ar nas estações de medida a funcionar num dado momento, explique brevemente como se deverão processar as comunicações entre o programa de controle do sistema e as estações de medida (incluindo o tipo de *sockets* e o funcionamento de cada um dos elementos do sistema). Assuma que, nas estações de medida, existe uma classe que permite obter o valor de poluição medido nessa estação. Assuma, igualmente, que o sistema funciona correctamente mesmo que o valor médio obtido não reflicta a medição em algumas das estações de medida a funcionar.

Sockets multicast – estações de medida numa endereço multicast bem-conhecido permite adicionar e remover estações de serviço facilmente.
Nas estações de serviço corre um servidor que espera pedido do sistema de controle.
Sistema de controle emite pedido por multicast e obtém todas as respostas fazendo a média. Caso alguma resposta se perca não faz mal.

b. Supondo que o objectivo do sistema é permitir notificar situações de perigo (i.e., situações em que o valor de poluição ultrapassa um dado valor limite) usando, para tal, um computador central de controle do sistema, explique brevemente como se deve organizar o sistema (incluindo o funcionamento de cada um dos elementos e tipo de *sockets* usados). Assuma que, nas estações de medida, é possível iniciar a execução de um método de uma aplicação quando se o valor da poluição medido atinge um dado valor. Assuma, igualmente, que a notificação de todas as situações de perigo é importante.

Saber IP do sistema central ou sistema central num endereço multicast bem-conhecido.
Enviar mensagem para o sistema central quando se verifica situação de perigo. Devem-se fazer retransmissões para garantir recepção no servidor.

c. (SE PERFERIREM ESTA PERGUNTA DE CÓDIGO) Apresente o código do cliente do sistema referido na alínea a, i.e., um programa que contacte as várias estações de medida e calcule a média da poluição do ar (considere que o valor da poluição medido pelas estações de medida é um valor real do tipo "double").

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 simultaneamente.

Verdadeiro ☐ Falso ☐ porque...

b. Um *socket multicast* apenas pode ser usado para enviar mensagens *multicast*, i.e., mensagens cujo destino seja um endereço *multicast*.

Verdadeiro ☐ Falso ☐ porque...

c. Supondo que todos os *routers* na Internet encaminhavam tráfego *multicast* e que nenhum pacote se perdia durante a transmissão e processamento nos *routers*, todas as mensagens enviadas num

socket multicast são sempre recebidas em todos os *sockets multicast* que se tenham juntado ao grupo *multicast* destino da mensagem.

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. Caso pretendesse manter a informação sobre os elementos que estavam numa sala de *chat* **no servidor**, o programa cliente teria de notificar o servidor necessariamente quando iniciava o seu funcionamento (indicando a entrada na sala de *chat*), quando terminava o seu funcionamento (indicando a saída da sala de *chat*) e quando recebia a mensagem de JOIN de outro cliente (para permitir actualizar a lista deste último).

Verdadeiro ☐ Falso ☐ porque...

b. Na fase 2, em que os pacotes do ficheiro são enviados sequencialmente (modo *burst*) sem nenhum *feedback* dos clientes, ...

Verdadeiro ☐ Falso ☐ porque...

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

Verdadeiro ☐ Falso ☐ porque...

Ainda relativamente ao trabalho prático, responda às seguintes perguntas.

d. Na fase 4,

e. .

5) 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.

b.

ANEXO A.1

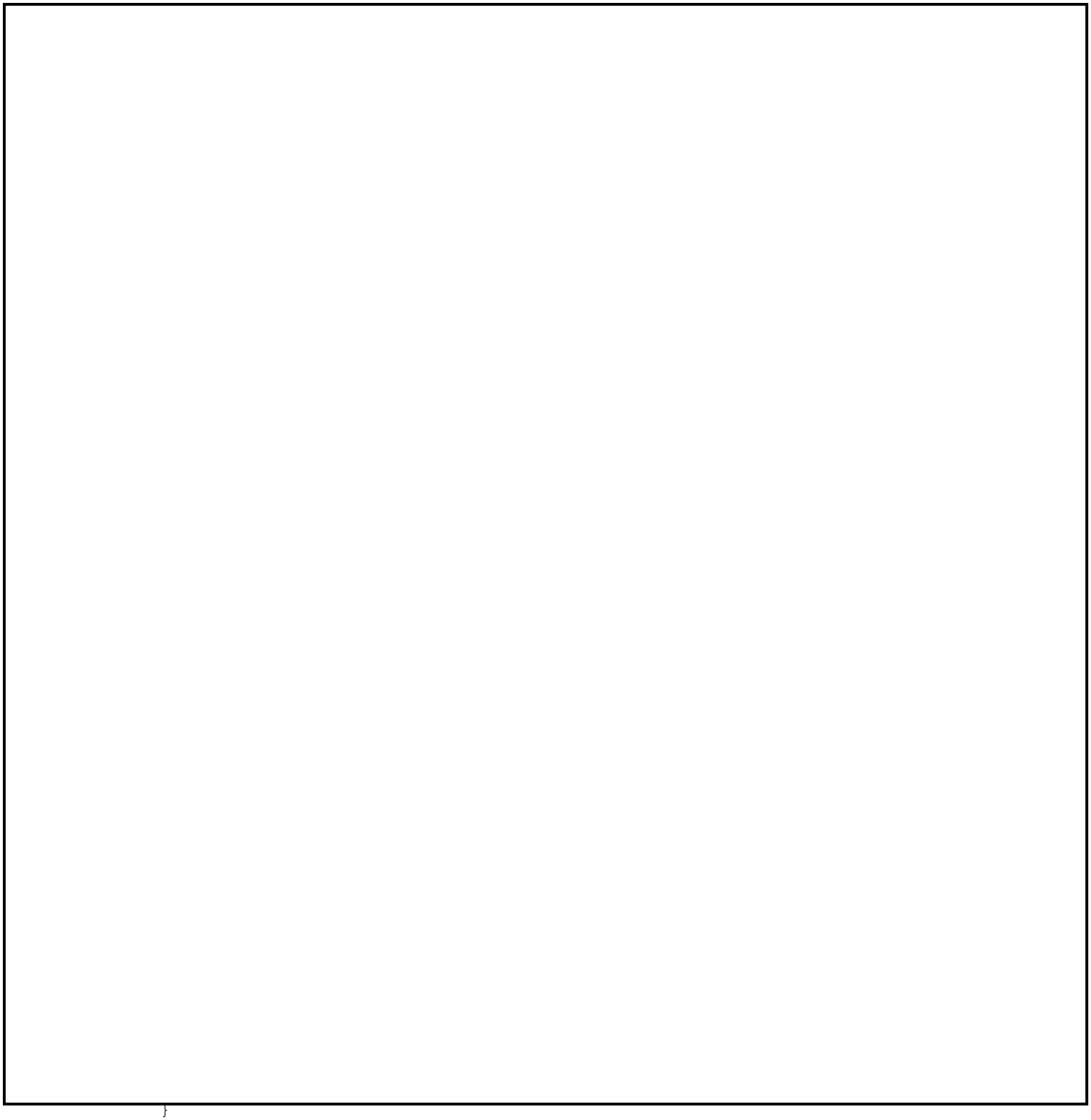
```
import java.io.*;
import java.net.*;

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
```

```
      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 {
```



```
            }
        }
    }
}
```

```
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;
```

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