FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

**Departamento de Informática**

**Licenciatura em Engenharia Informática**

**2º TESTE– Redes de Computadores**

**1º Semestre, 2007/2008 (30/Abril/2008)**

**NOTAS:**

Leia com atenção cada questão antes de responder. A interpretação do enunciado de cada pergunta é um factor de avaliação do teste. A duração do teste é 1 hora e 15 minutos sem tolerância. **NÃO SE ACEITAM TESTES DESAGRAFADOS !**

Responda às primeiras 4 questões assinalando com um círculo a resposta certa. Se o círculo estiver mal feito ou abarcar mais do que uma resposta, a resposta será considerada nula.

NOME:_____ Nº Aluno: _____

**Questão 1** Abaixo está apresentado o resultado de uma query DNS feita através de dig a um servidor DNS do domínio "stanford.edu", sobre os RRs de qualquer tipo do domínio.

```
jalm$ dig  @Argus.stanford.edu  stanford.edu any

; <<>> DiG 9.4.1-P1 <<>> @Argus.stanford.edu stanford.edu any
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45599
;; flags: qr aa rd; QUERY: 1, ANSWER: 18, AUTHORITY: 0, ADDITIONAL: 15
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;stanford.edu.                  IN     ANY

;; ANSWER SECTION:
stanford.edu.  172800 IN    SOA    Argus.stanford.edu. hostmaster.stanford.edu. 2008042812 .....
stanford.edu.        172800 IN     NS     authdns4.netcom.duke.edu.
stanford.edu.        172800 IN     NS     Argus.stanford.edu.
stanford.edu.        172800 IN     NS     Avallone.stanford.edu.
stanford.edu.        172800 IN     NS     Atalante.stanford.edu.
stanford.edu.        3600   IN     MX     20 mx3.stanford.edu.
stanford.edu.        3600   IN     MX     20 mx4.stanford.edu.
stanford.edu.        3600   IN     MX     20 mx1.stanford.edu.
stanford.edu.        3600   IN     MX     20 mx2.stanford.edu.

;; ADDITIONAL SECTION:
Argus.stanford.edu. 3600   IN     A      171.64.7.115
Avallone.stanford.edu.     3600   IN     A      171.64.7.88
Atalante.stanford.edu.     3600   IN     A      171.64.7.61
mx1.stanford.edu.   3600   IN     A      171.67.22.29
mx2.stanford.edu.   3600   IN     A      171.67.22.30
mx3.stanford.edu.   3600   IN     A      171.67.20.23
mx4.stanford.edu.   3600   IN     A      171.67.20.24
;; Query time: 200 msec
;; SERVER: 171.64.7.115#53(171.64.7.115)
;; WHEN: Mon Apr 28 20:11:59 2008
;; MSG SIZE  rcvd: 500
```

**a)** Diga quantos servidores DNS tem o domínio:

1, 2, 3, 4, 5, 6, 7, 8, 9 , 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, nenhum destes valores.

**b)** Diga quantos servidores DNS primários tem o domínio:

1, 2, 3, 4, 5, 6, 7, 8, 9 , 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, nenhum destes valores.

**c)** Diga quantos servidores de SMTP tem o domínio:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 , 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, nenhum destes valores.

**d)** Diga qual o endereço IP do servidor DNS de que o host jalm obteve a resposta:

**e)** Quantas interacções request/reply teve o host jalm com servidores DNS admitindo que não tinha cached o endereço IP do servidor que lhe enviou a resposta acima:

1, 2, 3, 4, 5, 6, 7, 8, 9 , 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, nenhum destes valores.

**Questão 2** Os computadores da Alice e do Bob estão ligados através da Internet. Com o programa *ping* a Alice chegou à conclusão que o tempo de trânsito de ida e volta entre o computador dela e o do Bob seguia o seguinte padrão:

```
alice@localhost$ ping bob.domain.com
PING bob.domain.com (64.233.187.99) 56(84) bytes of data.
64 bytes from 64.233.187.99: icmp_seq=1 ttl=235 time=149 ms
64 bytes from 64.233.187.99: icmp_seq=2 ttl=235 time=167 ms
64 bytes from 64.233.187.99: icmp_seq=3 ttl=235 time=190 ms
64 bytes from 64.233.187.99: icmp_seq=4 ttl=235 time=161 ms
64 bytes from 64.233.187.99: icmp_seq=5 ttl=235 time=186 ms
64 bytes from 64.233.187.99: icmp_seq=6 ttl=235 time=165 ms
64 bytes from 64.233.187.99: icmp_seq=7 ttl=235 time=237 ms
64 bytes from 64.233.187.99: icmp_seq=8 ttl=235 time=148 ms
64 bytes from 64.233.187.99: icmp_seq=9 ttl=235 time=161 ms
64 bytes from 64.233.187.99: icmp_seq=10 ttl=235 time=169 ms
64 bytes from 64.233.187.99: icmp_seq=11 ttl=235 time=172 ms
64 bytes from 64.233.187.99: icmp_seq=12 ttl=235 time=196 ms
64 bytes from 64.233.187.99: icmp_seq=13 ttl=235 time=161 ms
64 bytes from 64.233.187.99: icmp_seq=14 ttl=235 time=170 ms
64 bytes from 64.233.187.99: icmp_seq=15 ttl=235 time=171 ms

--- bob.domain.com ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14017ms
rtt min/avg/max/mdev = 148 / 173 / 237  / 28.052 ms
```

Indique, neste contexto, qual dos valores abaixo de "playout delay" em mili segundos seria o mais conveniente de usar pelos programas de IP phone da Alice e do Bob para poderem realizar, via a Internet, uma chamada telefónica de um para o outro admitindo aqueles padrões de RTT. Por hipótese, o CODEC usado não tolera percas de pacotes sem degradação significativa da qualidade:

1 2 3 4 5 6 7 8 9 10 15 20 30 50 100 ou nenhum destes valores

**Questão 3** Admita, por hipótese, que o RTT médio dentro da rede interna da FCT é de 4 ms (mili segundo), que essa rede está bem dimensionada para o tráfego existente, não se formando em geral filas de espera nos *routers* internos. Admita, também por hipótese, que o RTT médio entre um computador da rede da FCT e um servidor HTTP externo é de 200 ms. Admita ainda que 50% dos acessos a páginas feitos na rede da FCT/UNL são a servidores internos à rede (páginas de disciplinas por exemplo).

Decidiu-se instalar um servidor proxy / cache HTTP na FCT que todos os computadores internos passaram a usar apenas quando pretendem aceder a páginas externas à FCT/UNL. Constatou-se que em 50% dos casos, uma página HTTP externa solicitada estava já no proxy (*cache hit ratio* = 50%). Admita que que os PCs internos e o proxy só usam o protocolo HTTP 1.0. e que o tempo de transmissão de páginas WEB pelos servidores é desprezável.

Qual dos seguintes valores passou a ser o tempo médio aproximado às unidades que dura cada acesso a páginas HTTP pelos clientes situados na rede interna da FCT:

1 2 3 4 5 6 7 8 9 10 20 30 50 100 101 102 103 104 105 106 107 108 109 110 120 130 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

**Questão 4** Um cliente está ligado a um servidor através de um canal a funcionar à velocidade de transmissão de 1 Mbps e está a usar um protocolo de transporte de janela deslizante para transferir para o servidor um ficheiro de grandes dimensões. Admita que não há erros no canal e que o RTT entre o cliente e o servidor é estável e com o valor de 16 ms. Os segmentos trocados entre o cliente e o servidor são de dimensão constante e têm um tempo de transmissão de 4 ms

**a)** Qual a taxa de utilização (em %) do canal pelo protocolo, quando a janela do emissor tem o valor de 1 segmento:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 30 40 50 60 70 80 90 100 110 120 130 ou nenhum destes valores

**b)** Qual a taxa de utilização (em %) do canal pelo protocolo quando a janela do emissor tem o valor de 1 segmento e a janela do receptor tem a dimensão de 20 segmentos:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 30 40 50 60 70 80 90 100 110 120 130 ou nenhum destes valores

**c)** Qual a taxa de utilização (em %) do canal pelo protocolo quando a janela do emissor tem o valor de 4 segmentos e a janela do receptor tem a dimensão de 20 segmentos:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 30 40 50 60 70 80 90 100 110 120 130 ou nenhum destes valores

**Questão 5**

A **iBei** vem, mais uma vez, apelar para que ajude a sua subsidiária AuctionsOnTheGo(tm) na conquista do mercado de leilões móveis. Desta vez, a sua tarefa consistirá em supervisionar a criação de um segundo protótipo da plataforma, agora baseado em canais de dados, constituído pelo software cliente instalado no dispositivo móvel e pelo servidor, alojado na iBei.

O diálogo entre os clientes e o servidor faz-se através de linhas de texto sobre canais de dados. Correntemente, são suportados os seguintes comandos:

**SELL <item> <user> <value>**

- *Permite ao cliente colocar um novo artigo para venda, especificando o valor da licitação mínima.*

**BUY <item> <user> <value>**

- *Permite ao cliente licitar um artigo por um dado valor. Em caso de falha, a resposta indica o valor da licitação corrente.*

A resposta do servidor consiste numa linha de texto começada por **OK** ou **FAIL**, seguida pelo **comando que lhe deu origem** e, eventualmente, alguma informação extra.

**NOTA IMPORTANTE**: Uma ligação (canal de dados) apenas é usado para enviar um pedido e receber a respectiva resposta!

O código em anexo representa o estado actual da implementação do cliente e do servidor, ainda bastante incompletos. Como pode constatar, a **iBei** precisa da sua ajuda e para garantir a sua continuidade é imperativo que resolva as seguintes questões:

**a)** Complete o código em falta do cliente e do servidor, preenchendo as caixas. *(Apenas serão aceites respostas completas. Respostas consideradas disparatadas serão motivo de penalização).*

**b)** Correntemente, quando um cliente faz uma licitação sem sucesso, cabe ao utilizador a responsabilidade repetir a operação. Constatou-se que tal é mau para o negócio. Assim, pretende-se que, do lado do cliente, o utilizador possa especificar no comando **BUY** um valor extra, opcional, corresponte ao montante máximo que o utilizador está disposto a pagar pelo artigo em questão. O cliente deverá, então, ser melhorado para realizar automaticamente novas licitações, aumentando a oferta gradualmente *(melhor oferta+1.0)*, até ter sucesso ou ter sido atingido o limite dado pelo utilizador.

**c)** A implementação dada do servidor apenas trata um pedido de cada vez. Torne o servidor concorrente, ignorando eventuais problemas de sincronização. Responda na área reservada no código para o efeito.

**CLIENTE**

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

public class iBeiClient {

    String user;
    int server_port;
    String server_host;

    iBeiClient(String u, String s, String p ) {…}

    /*
     * Lê do canal de entrada, passado em argumento, uma linha composta por um
     * número váriavel de strings. Devolve as strings lidas num array de
     * comprimento apropriado...
     */
    private String[] parseInput( InputStream is) {
        Scanner ss = new Scanner(is);
        String cmdLine = ss.nextLine().toUpperCase();

        ArrayList<String> res = new ArrayList<String>();
        Scanner s = new Scanner(cmdLine);
        while (s.hasNext())
            res.add(s.next() );

        return res.toArray(new String[res.size()]);
    }

    /**
     * Envia um pedido ao servidor, retorna a resposta do servidor.
     */
    String[] doRequest( String request) throws IOException {
        request += "\n" ;
            [                          ] cs ;
```

```
        String[] res = parseInput( cs.                    ());
        cs.                    ();
        return res;
    }

void doIt() throws IOException {

    for (;;) {

        // Lê um comando do canal de entrada standard...
        String[] tokens = parseInput(System.in);
        String cmd = tokens[0];
        // Processa o comando introduzido...
        if (cmd.equals("FIM")) {
            return;
        }
        else if (cmd.equals("SELL")) {
            String item = tokens[1];
            double val = Double.parseDouble(tokens[2]);
            String[] reply = doRequest(" SELL " + item + " " + user + " " + val);
            System.out.println("RESULT:> " + reply[0]);
        }
        else if (cmd.equals("BUY")) {
            String item = tokens[1];
            double bidValue = Double.parseDouble(tokens[2]);
            double maxValue = 0 ;
            if (tokens.length == 4)
                maxValue = Double.parseDouble( tokens[3]);

            String[] reply = doRequest("BUY " + item + " " + user + " " + bidValue);
            if (reply[0].equals("OK")) {
                System.out.println("Licitação aceite para " + item + " por " + bidValue);
            } else {
```

alínea b)

```
                System.out.print("Licitação rejeitada para o item: ” + item) ;
                System.out.println(“Ofereça acima de " + bestBid);
            }
```

```
        }
        else {
            System.out.println("COMANDO DESCONHECIDO");
        }
    }
 }

 public static void main(String[] args) throws IOException {
    if (args.length != 3) {
        System.err.println(" use: java iBeiClient user server port");
        return;
    }
    new iBeiClient(args[0], args[1], args[2]).doIt();
 }

}
```

## SERVIDOR

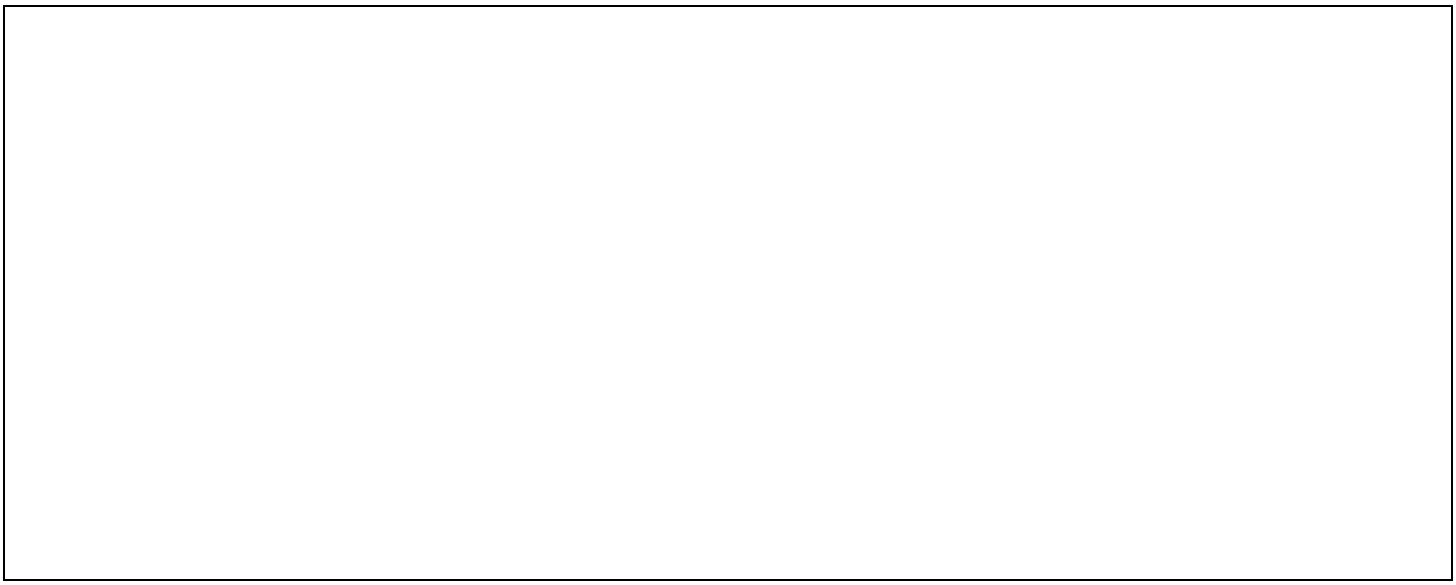
```
public class Bid {
   // Bid significa Licitação. Esta classe auxiliar é apenas usada pelo servidor,
   // onde se guarda o cliente que fez a melhor licitação até ao momento para um dado item.


       String item ; double value ; String owner ;


      Bid( String item, double value, String owner ) {
        this.item = item ; this. value = value ; this.owner = owner ;
      }

      public String toString() {
        return item + " " + value;
      }
}
```

---

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

public class iBeiServer {

   int port;
   Hashtable<String, Bid> bids = new Hashtable<String, Bid>();

   iBeiServer(int port) {
     this.port = port;
   }

   public void handleRequest( [          ] cs) throws IOException {

      String reply = "";
      String[] tokens = parseInput( cs.[                    ]() );
      String req = tokens[0];

      if (req.equals("BUY")) {
          String item = tokens[1];
          String user = tokens[2];
          double value = Double.parseDouble(tokens[3]);

          Bid bestBid = bids.get(item);
          if (bestBid == null || bestBid.value < value) {
              bids.put(item, new Bid(item, user, value));
              reply = "OK BUY " + item + " " + value;
          } else
              reply = "FAIL BUY " + item + " " + bestBid.value;
      } else if (req.equals("SELL")) {
          String item = tokens[1];
          String user = tokens[2];
          double value = Double.parseDouble(tokens[3]);
          bids.put(item, new Bid(item, user, value));
          reply = "OK SELL " + item + " " + value;
      } else
          reply = "FAIL " + req + " : PEDIDO DESCONHECIDO";
```

```
    reply += "\n" ;
```

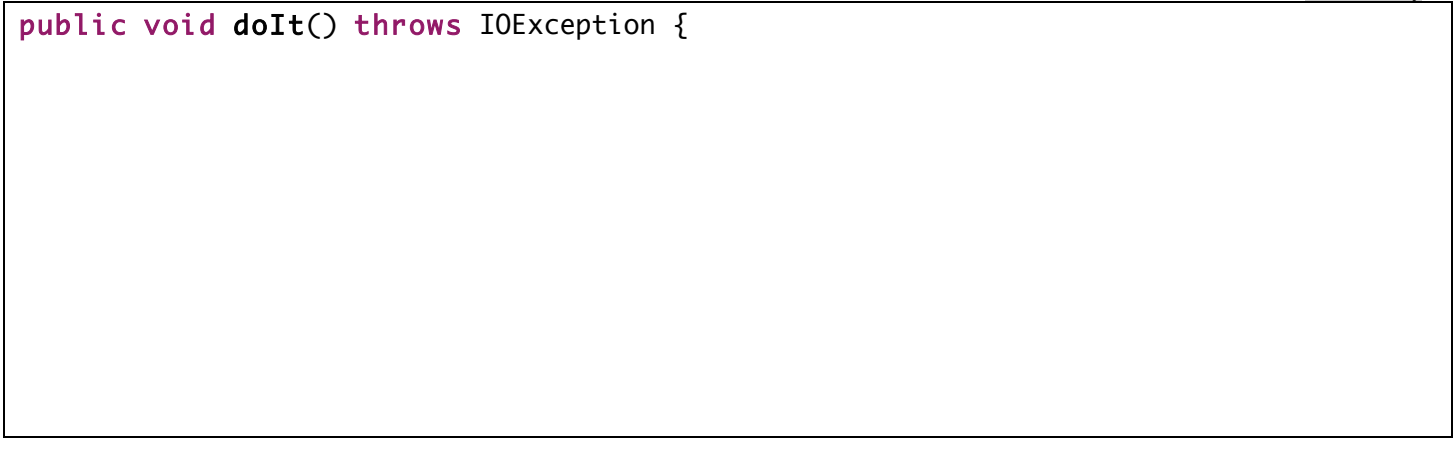
```
  }


   // … idêntico ao mesmo método no cliente …
private String[] parseInput(InputStream is) {
    … idêntico ao mesmo método no cliente…
}


public void doIt() throws IOException {
```

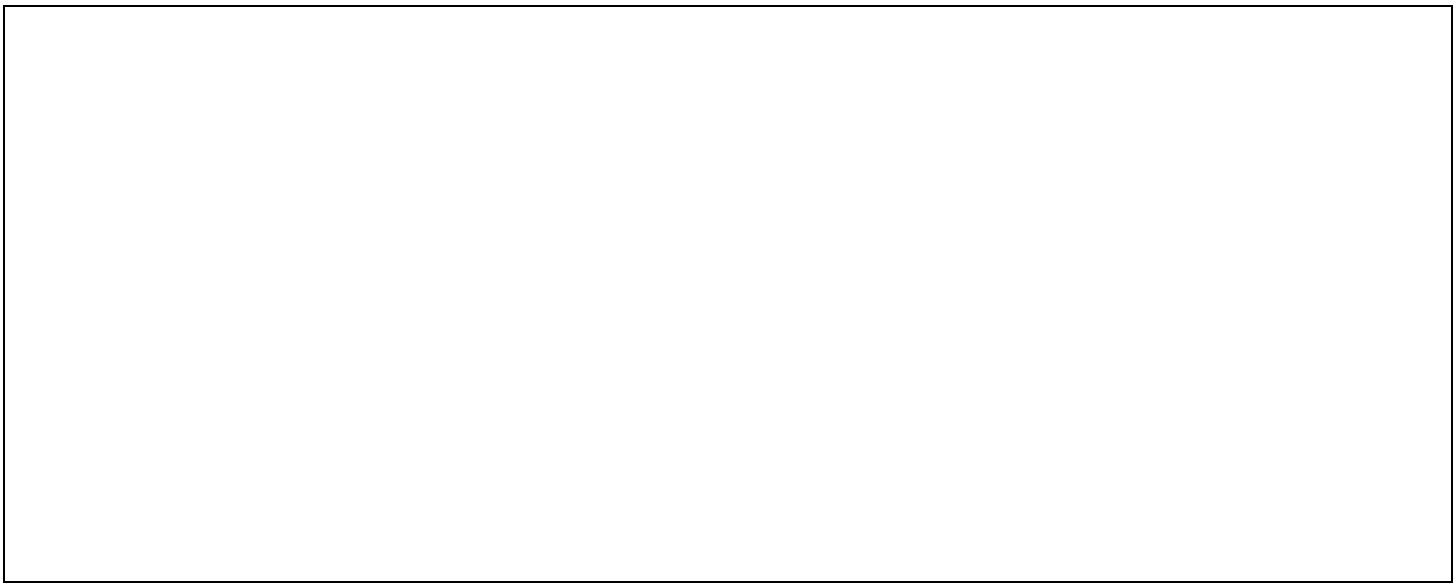


```
}

public static void main(String[] args) throws Exception {
    if (args.length != 1) {
        System.err.println("usar: java iBeiServer porto_do_servidor");
        return;
    }
    int port = Integer.parseInt(args[0]);
    new iBeiServer(port).doIt();
}
```

**alínea c)**

```
public void doIt() throws IOException {
```





}