

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: _____ N° Aluno: _____

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

```
jalm$ dig @bitsy.mit.edu mit.edu any
; <<>> DiG 9.4.1-P1 <<>> @bitsy.mit.edu mit.edu any
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62895
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 12, AUTHORITY: 0, ADDITIONAL: 4

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

;; ANSWER SECTION:
mit.edu.      120     IN      MX      100 M24-004-BARRACUDA-3.mit.edu.
mit.edu.      120     IN      MX      100 W92-130-BARRACUDA-1.mit.edu.
mit.edu.      120     IN      MX      100 W92-130-BARRACUDA-2.mit.edu.
mit.edu.      120     IN      MX      100 W92-130-BARRACUDA-3.mit.edu.
mit.edu.      120     IN      MX      100 M24-004-BARRACUDA-1.mit.edu.
mit.edu.      120     IN      MX      100 M24-004-BARRACUDA-2.mit.edu.
mit.edu.      21600  IN      SOA     BITSY.mit.edu. NETWORK-REQUEST.mit.edu.4878 3600 .....
mit.edu.      21600  IN      NS      STRAWB.mit.edu.
mit.edu.      21600  IN      NS      W20NS.mit.edu.
mit.edu.      21600  IN      NS      BITSY.mit.edu.
mit.edu.      60      IN      A       18.7.22.69

;; ADDITIONAL SECTION:
W92-130-BARRACUDA-1.mit.edu. 21600 IN      A       18.7.21.220
W92-130-BARRACUDA-2.mit.edu. 21600 IN      A       18.7.21.223
W92-130-BARRACUDA-3.mit.edu. 21600 IN      A       18.7.21.224
M24-004-BARRACUDA-1.mit.edu. 21600 IN      A       18.7.7.111

;; Query time: 125 msec
;; SERVER: 18.72.0.3#53(18.72.0.3)
;; WHEN: Sat Apr 26 11:51:44 2008
;; MSG SIZE rcvd: 509
```

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 e-mail 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.

d) O host jalm obteve a resposta final do servidor DNS do DI-FCT/UNL que era o seu servidor por defeito ?

sim / não.

e) Quantas interações 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 211.233.19.83: icmp_seq=0 ttl=50 time=370.685 ms
64 bytes from 211.233.19.83: icmp_seq=1 ttl=50 time=371.774 ms
64 bytes from 211.233.19.83: icmp_seq=2 ttl=50 time=348.741 ms
64 bytes from 211.233.19.83: icmp_seq=3 ttl=50 time=371.124 ms
64 bytes from 211.233.19.83: icmp_seq=4 ttl=50 time=346.136 ms
64 bytes from 211.233.19.83: icmp_seq=5 ttl=50 time=374.455 ms
64 bytes from 211.233.19.83: icmp_seq=6 ttl=50 time=345.567 ms
64 bytes from 211.233.19.83: icmp_seq=7 ttl=50 time=346.547 ms
64 bytes from 211.233.19.83: icmp_seq=8 ttl=50 time=348.305 ms
64 bytes from 211.233.19.83: icmp_seq=9 ttl=50 time=370.345 ms
64 bytes from 211.233.19.83: icmp_seq=10 ttl=50 time=371.141 ms
64 bytes from 211.233.19.83: icmp_seq=11 ttl=50 time=345.174 ms
64 bytes from 211.233.19.83: icmp_seq=12 ttl=50 time=372.469 ms
64 bytes from 211.233.19.83: icmp_seq=13 ttl=50 time=345.496 ms
64 bytes from 211.233.19.83: icmp_seq=14 ttl=50 time=371.352 ms
64 bytes from 211.233.19.83: icmp_seq=15 ttl=50 time=377.104 ms
^C
--- bob.domain.com ping statistics ---
16 packets transmitted, 16 packets received, 0% packet loss
round-trip min/avg/max/stddev = 345.174/360.901/377.104/12.726 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 que a página WWW com o URL <http://194.1.1.3:5689/teste.html> tem 100 bytes no total. Fazendo a análise do seu conteúdo, conclui-se que contém texto e referências a 2 imagens com os URLs:

[http:// 194.1.1.3/fig1.gif](http://194.1.1.3/fig1.gif) (fig1.gif com 250 bytes)

[http:// 194.1.1.3/fig2.gif](http://194.1.1.3/fig2.gif) (fig2.gif com 350 bytes)

Todas as máquinas estão a RTT mili segundos do cliente. Despreze o tempo de processamento e o tempo de transmissão de todos os pacotes envolvidos e admita também que cada mensagem HTTP de pedido ou resposta cabe dentro de um segmento TCP que cabe dentro de um único pacote IP.

a) Indique justificadamente o tempo espetável em número de RTTs que o cliente leva a obter toda a informação necessária para mostrar a página completa (com imagens) ao utilizador usando o protocolo HTTP 1.0:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 20 21 22 23 24 25 nenhum destes valores

b) Idem mas em HTTP 1.1, com pipelining:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 20 21 22 23 24 25 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 7 ms. Os segmentos trocados entre o cliente e o servidor são de dimensão constante e têm um tempo de transmissão de 3 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 3 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, correspondente 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 variável 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...  
}
```

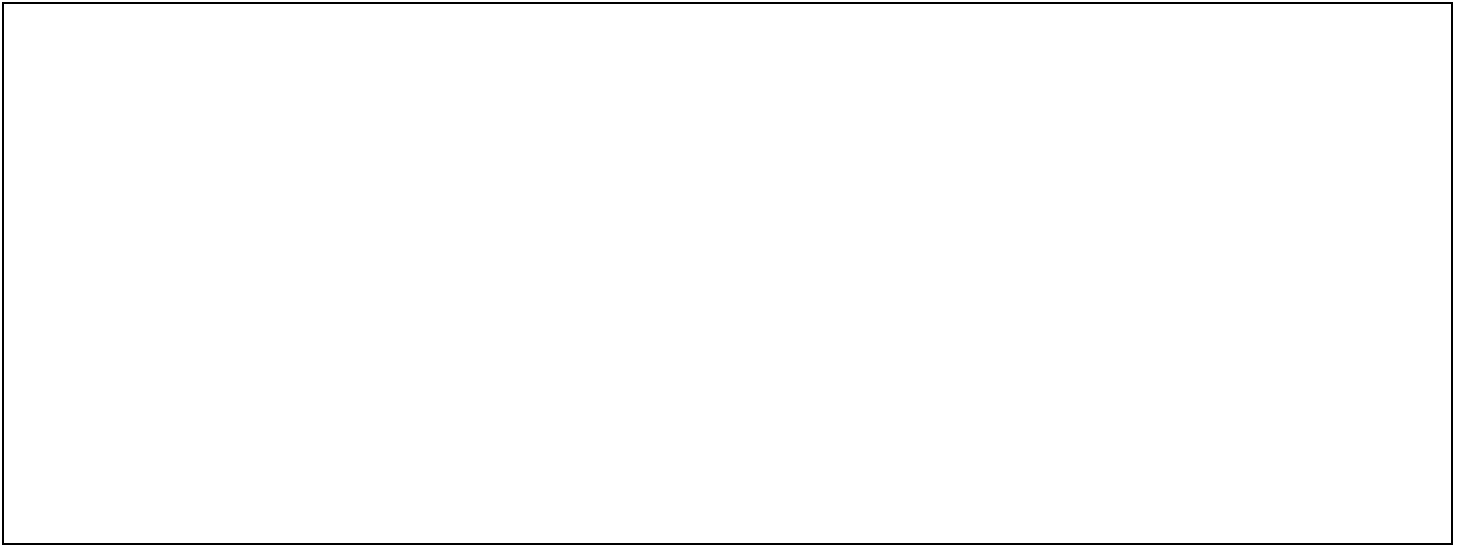
```
public void doIt() throws IOException {
```

```
}
```

```
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 {
```



}