FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Informática

Licenciatura em Engenharia Informática

3º TESTE– Redes de Computadores

2º Semestre, 2007/2008 (30/Maio/2008)

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 3 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 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=449 ms
64 bytes from 64.233.187.99: icmp_seq=2 ttl=235 time=467 ms
64 bytes from 64.233.187.99: icmp_seq=3 ttl=235 time=490 ms
64 bytes from 64.233.187.99: icmp_seq=4 ttl=235 time=461 ms
64 bytes from 64.233.187.99: icmp_seq=5 ttl=235 time=486 ms
64 bytes from 64.233.187.99: icmp_seq=6 ttl=235 time=465 ms
64 bytes from 64.233.187.99: icmp_seq=7 ttl=235 time=587 ms
64 bytes from 64.233.187.99: icmp_seq=8 ttl=235 time=448 ms
64 bytes from 64.233.187.99: icmp_seq=9 ttl=235 time=461 ms
64 bytes from 64.233.187.99: icmp_seq=10 ttl=235 time=469 ms
64 bytes from 64.233.187.99: icmp_seq=11 ttl=235 time=472 ms
64 bytes from 64.233.187.99: icmp_seq=12 ttl=235 time=496 ms
64 bytes from 64.233.187.99: icmp_seq=13 ttl=235 time=461 ms
64 bytes from 64.233.187.99: icmp_seq=14 ttl=235 time=470 ms
64 bytes from 64.233.187.99: icmp_seq=15 ttl=235 time=471 ms

--- bob.domain.com ping statistics ---
15 packets transmitted, 15 received, 0% packet loss
rtt min/avg/max/mdev = 448 / 476 / 587  / 29.052 ms
```

a) 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 75 100 ou nenhum destes valores

b) Admita agora que está a usar um CODEC que tolera 15% de percas de pacotes sem degradação significativa da qualidade. Relativamente ao valor da alínea **a),** o valor de playout delay a usar deve ser (indique uma das opções):

muito maior, maior, igual, menor , muito menor

Questão 2 Admita, por hipótese, que o RTT médio dentro da rede interna da FCT é de 1 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 50 ms. Admita que os PCs internos só usam o protocolo HTTP 1.0. e que o tempo de transmissão de páginas WEB pelos servidores é desprezável.

a) Qual o tempo necessário para aceder a uma página WEB externa à FCT/UNL por um PC interno?

1 2 3 4 5 6 7 8 9 10 20 25 26 27 28 29 30 50 51 52 53 54 55 100 101 102 103 104 105 106 107 108 109 110 120 130 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

b) Decidiu-se instalar um servidor proxy / cache HTTP na FCT que todos os computadores internos passaram a usar sempre que pretendem aceder a uma página externa. Constatou-se que em 50% dos casos, uma página HTTP externa solicitada estava já no proxy (*cache hit ratio* = 50%). Admita que o proxy só usa o protocolo HTTP 1.0. e que o tempo de transmissão de páginas WEB pelos servidores continua a ser desprezável. Qual dos seguintes valores passou a ser o tempo médio aproximado às unidades que dura cada acesso a páginas externas pelos clientes situados na rede interna da FCT?

1 2 3 4 5 6 7 8 9 10 20 25 26 27 28 29 30 50 51 52 53 54 55 100 101 102 103 104 105 106 107 108 109 110 120 130 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

c) Admita agora que os computadores internos só usam o proxy para acesso a páginas externas e fazem acesso directo aos servidores HTTP da rede da FCT/UNL (páginas de disciplinas por exemplo). O peso dos acessos internos é de 50% do total de acessos. O cache hit ratio continua a ser o mesmo que na **b)**. Qual o tempo médio de acesso a uma página (externa ou interna) por um PC interno?

1 2 3 4 5 6 7 8 9 10 20 25 26 27 28 29 30 50 51 52 53 54 55 100 101 102 103 104 105 106 107 108 109 110 120 130 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

Questão 3 Responda às seguintes questões fazendo um círculo envolvendo a opção certa (Verdade ou Falsidade da afirmação). Uma resposta errada desconta 50% da cotação. A ausência de resposta não desconta nada.

a) Num protocolo de janela deslizante versão "Go Back N" com ACKs cumulativos, a existência de um mecanismo de *timeout* é dispensável pois o emissor acabaria sempre por detectar que um pacote se perdeu através dos números de sequência.

Verdade Falsidade

b) Num protocolo de janela deslizante a utilização de um mecanismo de Negative ACK é completamente opcional pois o mesmo só serve para acelerar a recuperação de erros no caso de perca de um pacote.

Verdade Falsidade

c) Num protocolo de janela deslizante para tirar rendimento do mecanismo de pipelining (poder ter mais do que um pacote em trânsito por RTT), é imprescindível que a janela do emissor seja superior a 1.

Verdade Falsidade

d) Num protocolo de janela deslizante para tirar rendimento do mecanismo de pipelining (poder ter mais do que um pacote em trânsito por RTT), é imprescindível que a janela do receptor seja superior a 1.

Verdade Falsidade

e) Um computador A está a enviar um ficheiro muito grande para o computador B através de uma conexão TCP. Supondo que o computador B não tem dados para enviar para A, então B não envia ACKs para A pois não pode fazer *piggybacking* dos ACKs.

Verdade Falsidade

f) O valor do campo RcvWindow do cabeçalho TCP nunca se altera durante toda a duração de uma conexão TCP.

Verdade Falsidade

g) Um computador A está a enviar um ficheiro muito grande para o computador B através de uma conexão TCP. Supondo que o número de sequência de um segmento enviado de A para B é N, então o número de sequência do próximo segmento é sempre N+1.

Verdade Falsidade

Questão 4 Na rede a seguir indicada existem várias sub-redes interligadas pelos *routers* R1 e R2. O *router* R2 por sua vez liga à Internet através de um router de um ISP que conhece pelo endereço 10.0.1.2. Indique a tabela de encaminhamento do *router* R1.

Tabela de endereços das interfaces dos *routers*

	Eth0	Eth1	Eth2
Router R1	---	10.0.2.1	10.0.4.1
Router R2	10.0.2.2	10.0.3.1	10.0.1.1

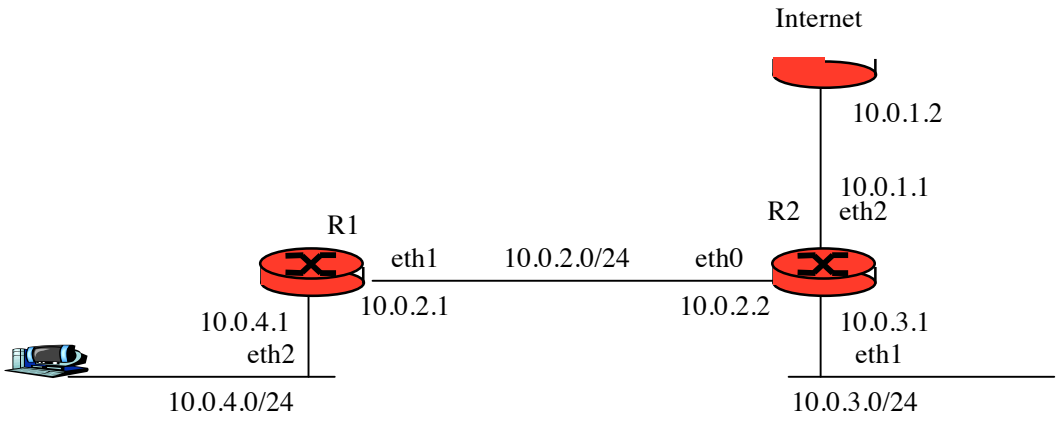


Tabela de encaminhamento do *router* R1

Prefixo IP de destino	Tipo de encaminhamento (directo, indirecto ou por defeito)	Next hop (interface ou endereço)	Número de hops
10.0.1.0/24			2
10.0.2.0/24			1
10.0.3.0/24			2
10.0.4.0/24			
0.0.0.0			

Questão 5 A **iBei** vem, uma última vez, apelar para que ajude a sua subsidiária AuctionsOnTheGo(tm) na conquista do mercado de leilões móveis. Agora, a sua tarefa consistirá em supervisionar a criação de um último protótipo da plataforma, novamente baseado na troca de mensagens, mas desta vez tendo em conta a perda de mensagens e a consequente necessidade de repetir o envio.

O diálogo entre os clientes e o servidor faz-se por trocas de mensagens contendo linhas de texto. 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.

O código em anexo representa o estado actual da implementação do cliente e do servidor. Resolva as seguintes questões:

(Apenas serão aceites respostas completas. Respostas consideradas disparatadas serão motivo de penalização).

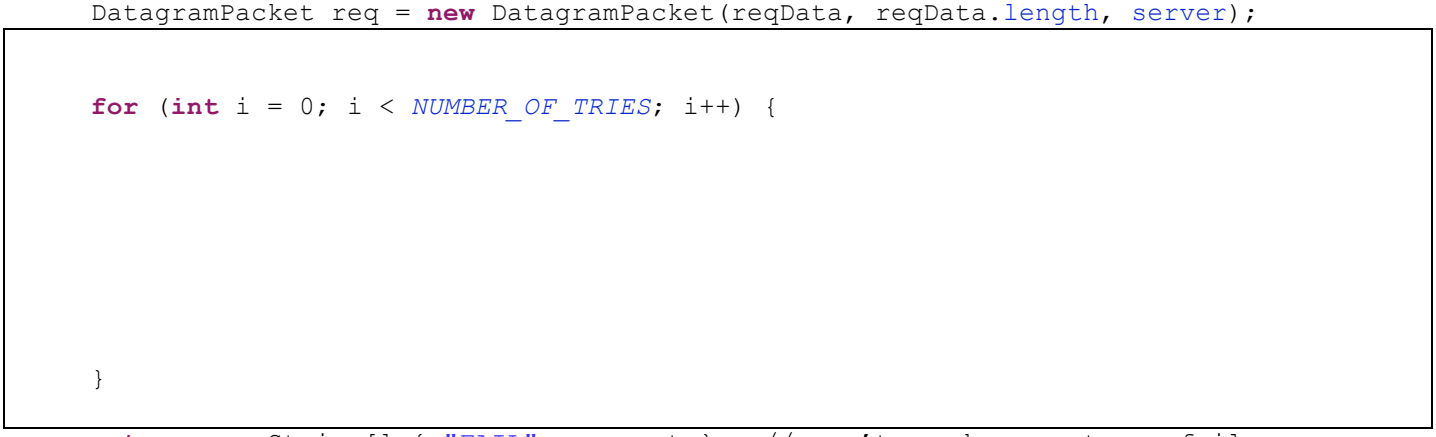
a) Complete o código em falta do cliente e do servidor, preenchendo as caixas, de forma a garantir a chegada dos pedidos ao servidor, considerando a possível perda de mensagens.

b) Complete o código em falta do cliente e do servidor, preenchendo as caixas assinaladas, de forma a permitir que o cliente descubra a localização do servidor por multicast. Para tal, o cliente deverá enviar uma mensagem contendo **"?"**, para um grupo multicast e um porto, pré-definidos, à qual o servidor responderá (directamente ao cliente) com uma mensagem contendo uma string na forma: **SERVER <IP> <port>**

CLIENT

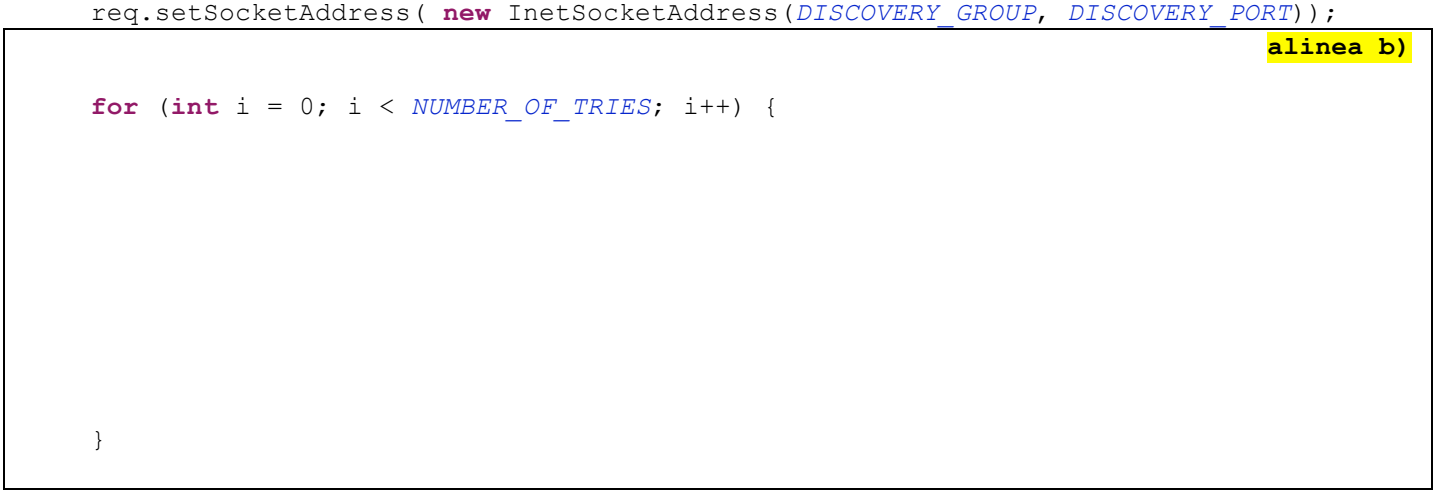
```
import java.io.*;
import java.net.*;
import java.util.*;

public class iBeiClient {

  String user;
  DatagramSocket socket;
  InetSocketAddress server;
  public static final int NUMBER_OF_TRIES = 5;

  Map<String, Bid> myBids = new Hashtable<String, Bid>();

  iBeiClient(String u) {
      this.user = u;
  }

  iBeiClient(String u, String s, String p) {
      this(u);
      this.server = new InetSocketAddress(s, Integer.parseInt(p));
  }

  // Breaks 'is' string into several strings, one string for each word;
  // returns the array of those strings
  private String[] parseInput(String 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());

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

  // Reads one line from the input stream. Returns an array of strings, each string with one word
  private String[] parseInput(InputStream is) {
      return parseInput(new Scanner(is).nextLine());
  }


  void doIt() throws IOException {
      if (server == null)
          server = discoverServer();

      if (server == null) {
          System.out.println("Fatal error: server not found...");
          return;
      } else  System.out.println("Found server at:" + server);

      socket = [                              ];
      for (;;) {
          // Reads a user command ...
          String[] tokens = parseInput(System.in);
          String cmd = tokens[0];
          // Process the command ...
          if (cmd.equals("FIM")) {
```

```
            return;
        } else if (cmd.equals("BUY")) {
            String item = tokens[1];
            double bidValue = Double.parseDouble(tokens[2]);

            String[] reply = doRequest("BUY " + item + " " + user + " " + bidValue);
            if (reply[0].equals("OK")) {
                System.out.println("Bid accepted for " + item      + " with value " + bidValue);
            } else {
                System.out.println("Bid rejected for " + item + ", offer more than " + reply[3]);
            }
        } 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 {
            System.out.println("UNKNOWN COMMAND");
        }
    }
}

/*  Sends a request to the server and returns the reply   */
String[] doRequest(String request) throws IOException {
    request += "\n";

    byte[] reqData = request.getBytes();
    DatagramPacket req = new DatagramPacket(reqData, reqData.length, server);
```



```
    for (int i = 0; i < NUMBER_OF_TRIES; i++) {









    }

    return new String[] { "FAIL", request };  // can't send, so returns fail
}

/* Discovers the server address/port using multicast  */
private static final int DISCOVERY_PORT = 1234;
private static final String DISCOVERY_GROUP = "224.224.224.224";

private InetSocketAddress discoverServer() throws IOException {

    [                    ] ms = new [                    ]();     // alinea b)

    byte[] reqData = "?".getBytes();
    DatagramPacket req = new DatagramPacket(reqData, reqData.length);
    req.setSocketAddress( new InetSocketAddress(DISCOVERY_GROUP, DISCOVERY_PORT));
```

alinea b)



```
    for (int i = 0; i < NUMBER_OF_TRIES; i++) {









    }

    ms.close();
    return null;  // can't find server, returns null
```

```
  }
}

  /***************** MAIN *****************/
  public static void main(String[] args) throws IOException {
      if (args.length == 1)
          new iBeiClient(args[0]).doIt();
      else if (args.length == 3)
          new iBeiClient(args[0], args[1], args[2]).doIt();
      else
          System.err.println(" use: java iBeiClient user_name [server_ip  server_port]");
  }
```

SERVER

```
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;
  DatagramSocket socket ;

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

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

  public void handleRequest( String reqLine, InetSocketAddress client) throws IOException {
      String reply = "";
      String[] tokens = parseInput( reqLine );
      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 )
              reply = "FAIL BUY " + item + " ??";
          else if (bestBid.value < value || (bestBid.value == value && bestBid.owner.equals(user)) ) {
              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 + " : UNKNOWN REQUEST";

      reply += "\n" ;
      byte[] replyData = reply.getBytes() ;
```

```
}
// Breaks 'cmdLine' string in several strings, one string for each word
// returns the array of those strings
private String[] parseInput(String cmdLine) {
    // the same code as found in the same method of the client
}


public void doIt() {
    socket = [                                ];
    new DiscoveryServer().start();

    for (;;) {
        try {
            DatagramPacket req = new DatagramPacket( new byte[65536], 65536) ;
            [                                        ] ;
            String cmdLine = new String( req.getData(), 0, req.getLength() ) ;
            InetSocketAddress client = new InetSocketAddress( req.getAddress(), req.getPort() ) ;
            [                                ]( cmdLine, client ) ;
        } catch (IOException x) {
            x.printStackTrace();
        }

    }
}

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

// Discovery Server: replies to client requests trying to discover
// the server by multicast
private static final int DISCOVERY_PORT = 1234 ;
private static final String DISCOVERY_GROUP = "224.224.224.224" ;

class DiscoveryServer extends Thread {

    DiscoveryServer() {
        super.setDaemon(true) ;
    }

    public void run() {
        try {
            [                    ] ms = new [                    ](DISCOVERY_PORT);// alinea b)
```

alinea b)

```
            for(;;) {









            }


        } catch( Exception x ){
            x.printStackTrace() ;
        }
```

```
        }
    }
}
```