FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Informática

Licenciatura em Engenharia Informática
1º TESTE– Redes de Computadores
1º Semestre, 2007/2008 (26/Março/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.

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

NOME:_____ Nº Aluno: _____

1) Suponha que um canal ponto a ponto foi estabelecido entre a Terra e um veículo de exploração em Marte, com a capacidade de transmissão de 1 Mbps. Assuma, por hipótese, que a distância da Terra a Marte quando estão mais perto é de 45 Gm = 45 x 10^9 metros. Os dados percorrem este canal à velocidade de 3 x 10^8 metros por segundo.

a) Calcule o tempo de ida e volta do canal ou RTT do canal.
   a. 2 segundos
   b. 100 segundos
   c. 150 segundos
   d. 200 segundos
   e. 280 segundos
   f. 300 segundos
   g. 310 segundos
   h. 400 segundos
   i. nenhum dos valores acima

b) Calcule quantos bits podem ser transmitidos por este canal antes do primeiro chegar ao destino ou, por outras palavras, quantos bits cabem dentro do canal.
   a. 2 x 10^6 bits
   b. 10^8 bits
   c. 1,5 x10^8 bits
   d. 2 x 10^8 bits
   e. 2,8 x 10^8 bits
   f. 3 x 10^8 bits
   g. 3,1 x 10^8 bits
   h. 4 x 10^8 bits
   i. nenhum dos valores acima

c) O veículo em Marte tira fotografias com 5 Mbytes (≈ 4 x 10^7 bits). Quanto tempo levam estas mensagens a chegar à estação de controlo na Terra, admitindo que o RTT é de 400 segundos. Admita que uma fotografia = um mensagem , ou seja, uma grande sequência com ≈ 4 x 10^7 bits sem erros.
   a. 20 segundos
   b. 40 segundos
   c. 120 segundos
   d. 140 segundos
   e. 150 segundos
   f. 190 segundos
   g. 240 segundos
   h. 440 segundos
   i. nenhum dos valores acim

d) Calcule o mesmo tempo que na c) admitindo que o canal tem globalmente o mesmo comprimento e que o sinal leva o mesmo tempo a propagar-se da Terra a Marte, mas que agora atravessa um *router* instalado num estação em órbita terrestre a 35.000

Km de altitude. Esse *router* funciona em *store & forward* da mensagem (sempre de 5 M bytes). O canal da Terra ao *router* tem a capacidade de 5 M bps e o do *router* a Marte 1 Mbps.

a. 50 segundos
b. 70 segundos
c. 140 segundos
d. 155 segundos
e. 175 segundos
f. 200 segundos
g. 230 segundos
h. 248 segundos
i. nenhum dos valores acima

Questão 2

A **iBei** é uma empresa de leilões que pretende expandir a sua área de negócios para as licitações online através de dispositivos móveis. Com esse objectivo, a iBei criou uma subsidiária, a **AuctionsOnTheGo(tm)**, que liderá o processo para desenvolver o software que suportará as licitações online em variados dispositivos móveis, tais como telemóveis 3G e PDAs equipados com Java e conectividade IP. A sua tarefa consistirá em supervisonar a criação do protótipo inicial da plataforma, consituída pelo software cliente instalado no dispositivo móvel e pelo servidor, alojado na iBei.

O diálogo entre os clientes e os servidor faz-se através de mensagens de texto. Correntemente, são suportados os comandos seguintes:

LIST

- *Permite ao cliente listar todos os items actualmente em licitação.*

SELL <item> <valor>

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

BUY <item> <valor>

- *Permite ao cliente licitar um artigo por um dado valor. Em caso de falha indica o valor da licitação mais alta.*

A resposta do servidor consiste numa mensagem de texto começada por **OK** ou **FAIL**, seguida pelo **comando que lhe deu origem** e, finalmente, pelo **resultado da operação,** eventualmente vazio.

O código em anexo representa o estado acutal 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) O código do cliente e do servidor contém alguns erros graves. Assinale esses erros, indicando as linhas de código incorrectas através de (→). Apresente, ao lado, a sua correcção.

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

c) A implementação corrente não avisa os clientes quando as respectivas licitações são superadas por ofertas maiores. Corrija esse problema, notificando um cliente assim que a sua licitação sobre um artigo é ultrapassada, enviando uma mensagem com o seguinte formato: **FAIL <item> <valor>**

d) 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, representado o 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 ser atingido o limite dado pelo utilizador.

```
public class Bid {
    // Bid significa Licitação. Esta classe auxiliar é partilhada pelo cliente e pelo servidor.
    // No servidor guarda o cliente que fez a melhor licitação até ao momento para um dado item.
   // No cliente, apenas relevante para a alínea d), guarda o valor limite máximo de licitação
   // que o utilizador introduziu para o item.
      String item ; double value ; InetSocketAddress owner ;
      Bid( String item, double value ) {
      this( item, value, null) ;
      }
      Bid( String item, double value, InetSocketAddress owner ) {
      this.item = item ; this. value = value ; this.owner = owner ;
      }
      public String toString() {
      return item + " " + value;
      }
}
```

NOTA: A resolução correcta das alíneas a) e b) **não está depende** da resolução das alíneas c) e d). Para o efeito, as zonas código marcadas com (alínea c) e (alínea d) **podem ser ignoradas**.

CLIENTE

```
import java.io.* ;
import java.net.* ;
import java.util.* ;
public class iBeiClient implements Runnable {

   int port ;
   DatagramSocket socket ;
   InetAddress server ;
   Map<String, Bid> myBids = new Hashtable<String, Bid>() ; // apenas relevante para a alínea d)

   iBeiClient( InetAddress server, int port ) {
     this.server = server ;
     this.port = port ;
   }

   void doIt() throws IOException {

     socket = new DatagramSocket( port ) ;
     new Thread( this).run() ;

     String cmdLine ;
     Scanner ss = new Scanner( System.in ) ;

     while( ! (cmdLine = ss.nextLine()).equals("FIM")) {
       Scanner ss2 = new Scanner( cmdLine.toUpperCase() ) ;
       String cmd = ss2.next() ;
       String request = "" ;

       if( cmd.equals("SELL")) {
          String item = ss2.next();
          double value = ss2.nextDouble() ;
          request = "SELL " + item + " " + value ;
       }
       else if ( cmd.equals("LIST") ) {
          request = "LIST" ;
       }
       else if( cmd.equals("BUY")) {
          String item = ss2.next();
          double value = ss2.nextDouble() ;
          request = "BUY " + item + " " + value ;
(alínea d)    try {  // guarda em myBids o valor da licitação máxima automática, caso o utilizador o tive dado
             double maximumAutomaticBid = ss2.nextDouble() ;
             myBids.put( item, new Bid( item, maximumAutomaticBid) ) ;
          } catch( NoSuchElementException x){ // não foi introduzido um valor de licitação máxima}

       } else {
          System.err.println("ERRO: Comando desconhecido...") ;
          continue ;
       }

       byte[] requestData = request.getBytes() ;
       DatagramPacket requestMsg = new …(, requestData.length );
       requestMsg.setAddress( server ) ;
       requestMsg.setPort( port ) ;
       socket.send( requestMsg ) ;
     }
   }
```

CLIENTE (continuação)

```
public void run() {
    try {
     for(;;) {
         byte[] buffer = new byte[ 65536 ] ;
         DatagramSocket replyMsg = new DatagramSocket ( buffer, buffer.length ) ;

         socket.read( replyMsg ) ;
         String reply = new String( replyMsg.getData(), 0, replyMsg.getLength() ) ;

         Scanner ss = new Scanner( reply ) ;
         if( ss.next().equals("OK")) { // um comando teve sucesso…
            String cmd = ss.next();
            if( cmd.equals("LIST") || cmd.equals("SELL")) {
           // código omitido
            }
            else if( cmd.equals("BUY")) {
                      String item = ss.next();
                      double value = ss.nextDouble() ;
                      System.out.println( + item + " por " + value ) ;
                 }
            }
            else if( ss.next().equals("FAIL")) { // um comando falhou…
                String cmd = ss.next();
                if( cmd.equals("LIST") || cmd.equals("SELL")) {
                 // código omitido
                       }
              else if( cmd.equals("BUY")) {
                          // um comando BUY falhou. Há uma licitação superior…

                          String item = ss.next();
                          double value = ss.nextDouble() ;
```

(alínea d)

```
                   }
               }
               else {
                   System.err.println( reply ) ;
                   continue ;
               }
     }
    } catch( IOException x ) {}
   }
   public static void main( String[] args ) throws IOException {
      if( args.length != 2 ) {
      System.err.println(" use: java EbeyClient ip_do_servidor porto_do_servidor" ) ;
      return ;
      }
      int port = Integer.parseInt( args[1]) ;
      InetAddress server = InetAddress.getByName( args[0]) ;
      new iBeiClient( server, port ).doIt() ;
   }
}
```

SERVIDOR

```
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 ) {
      this.port = port ;
   }
   public void doIt() throws IOException {
      socket = new DatagramSocket() ;
      for(;;) {
          InetSocketAddress client ;
          byte[] buffer = new byte[ 65536 ] ;
          DatagramPacket requestMsg = new DatagramPacket( buffer, buffer.length ) ;
          socket.receive( requestMsg ) ;
          String request = new String( requestMsg.getData(), 0, requestMsg.getLength()) ;
          client = new InetSocketAddress( requestMsg.getAddress(),equestMsg.getPr );
          handleRequest( request, client ) ;
      }
   }
```

(continua no verso)

SERVIDOR (continuação)

```
public void handleRequest( String request, InetSocketAddress client ) throws IOException {
     String reply = "" ;
     Scanner ss = new Scanner( request ) ;
     String cmd = ss.next().toUpperCase() ;
     if( cmd.equals("BUY")) {
        String item = ss.next();
        double value = ss.nextDouble() ;
        Bid bestBid = bids.get(item) ;
        if( bestBid == null || bestBid.value < value ) {
                 bids.put( item, new Bid( item, value, client ) ) ;
                 reply = "OK BUY " + item + " " + value + " ";
```

(alínea c)

```
        }
        else
                // já havia uma licitação superior…
                  reply = "FAIL BUY " + item + " " + bids.get(item).value ;
     }
     else if( cmd.equals("LIST")) {
        // código omitido
        reply = "OK LIST\n" ;
        for( Bid i : bids.values() )
            reply += i.toString() + "\n";
     }
     else if( cmd.equals("SELL")) {
        // código omitido
        String item = ss.next();
        double value = ss.nextDouble() ;
        bids.put( item, new Bid(item, client, value ) ) ;
        reply = "OK SELL " + item + " " + value ;
     }
     else reply = "FAIL " + cmd + " : COMANDO DESCONHECIDO";
     byte[] replyData = reply.getBytes() ;
     DatagramPacket replyMsg = new DatagramPacket( replyData, replyData.length) ;
     replyMsg.setAddress( client.getAddress() ) ;


     replyMsg.setPort( client.getPort() ) ;
     socket.send( replyMsg ) ;


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

```
}
```