FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Informática

Licenciatura em Engenharia Informática
1º TESTE– Redes de Computadores – (ABC-1011)
2º Semestre, 2010/2011 (16/Abril/2011)

NOTAS: Leia com atenção cada questão antes de responder. A interpretação do enunciado de cada pergunta é um factor de avaliação do teste. As respostas erradas às alíneas das questões 1, 2 e 3 **descontam 25%** da respectiva cotação. **A duração do teste é 2 horas sem tolerância.**
O enunciado contém **6 páginas** que devem ser entregues como resposta ao teste.

NOME:_____ **Nº Aluno:** _____

Questão 1) Considere um canal bidirecional que liga dois computadores A e B, com uma velocidade de transmissão de 10 Mbit/s e 500 Km de comprimento. Assuma que a velocidade de propagação é de $2 * 10^5$ Km/s. Sobre este canal usam-se pacotes com 10000 bits. Ignore os tempos de processamento ou possíveis tempos em filas de espera. Assinale com um círculo a sua resposta às seguintes questões:

a) Quanto tempo leva um bit a propagar-se desde o emissor até ao receptor? (em milissegundos)

0,5 1 1,5 2 2,5 3 3,5 4 4,5 5 5,5 6 6,5 7 7,5 8 8,5 9 ou nenhum destes valores

b) Quanto tempo leva um pacote a ser transferido através deste canal, isto é, quanto tempo tem de passar desde o início da transmissão até o último bit do pacote chegar à outra extremidade do canal, admitindo que existem dois routers pelo meio que trabalham em *store and forward*? (em milissegundos)

0,5 1 1,5 2 2,5 3 3,5 4 4,5 5 5,5 6 6,5 7 7,5 8 8,5 9 ou nenhum destes valores

c) Nas condições acima enunciadas, quanto tempo leva, aproximadamente, a transferir um ficheiro com 10^7 bits (aprox. 1Mbytes) de A para B ? (aproximado em segundos)

0,5 1 1,5 2 2,5 3 3,5 4 4,5 5 5,5 6 6,5 7 7,5 8 8,5 9

d) Considere agora que, ao mesmo tempo que transfere o ficheiro de A para B, decorre outra transferência idêntica em sentido contrário, passando pelo mesmo canal e routers. Como será o novo tempo necessário para transferir o mesmo ficheiro?

Menos de metade, metade, igual, o dobro, mais do dobro

e) Admita agora que o emissor só pode passar a transmitir o pacote seguinte quando recebe a confirmação vinda do receptor da recepção do último pacote transmitido. Essa confirmação vem através de um pacote especial, com tempo de transmissão desprezável, transmitido logo que o receptor recebe o último bit de cada pacote. Quanto tempo é agora necessário para transferir o ficheiro ? (aproximado em segundos)

0,5 1 1,5 2 2,5 3 3,5 4 4,5 5 5,5 6 6,5 7 7,5 8 8,5 9 9,5 10

f) Que fracção aproximada da velocidade de transmissão do canal está a ser usada na alínea anterior ?

1/10 1/9 1/8 1/7 1/6 1/5 1/4 1/3 1/2 1/1 nenhum destes valores

Questão 2) Um cliente HTTP (*browser*) acede à informação disponível no seguinte URL: http://serv1.domi.com/doc.html. Esta página inclui ainda duas imagens com URLs: http://serv1.domi.com:80/img1.jpg e http://host2.domi.com/img2.jpg. Considere que a página e cada uma das imagens pode ser transmitida em 3milisegundos e um RTT médio de 20milisegundos. Assinale com um círculo a sua resposta às seguintes questões:

a) Quantos pedidos são efectuados pelo cliente ao DNS para ser possível transferir toda a informação indicada?

1, 2, 3, 4, 5, 6, 7, 8, 9, nenhum destes valores

b) Nas condições anteriores, quantos pedidos HTTP são efectuados pelo cliente para obter toda informação necessária para mostrar correctamente a informação (incluindo imagens)?

1, 2, 3, 4, 5, 6, 7, 8, 9, nenhum destes valores

c) Nas condições anteriores, quantas conexões TCP são estabelecidas pelo cliente com os servidores para obter toda a informação, supondo que é usado o protocolo HTTP/1.0 e que é possível estabelecer conexões paralelas?

1, 2, 3, 4, 5, 6, 7, 8, 9, nenhum destes valores

d) Mantendo-se as condições anteriores e ignorando a resolução dos nomes pelo DNS, quanto tempo decorre para obter toda a informação indicada, nas condições anteriores, (em milisegundos).

63 66 70 73 76 83 86 89 90 99 103 109 113 120 nenhum destes valores

e) Admitindo agora que usa o protocolo HTTP/1.1 (sem pipeline e sem conexões paralelas), diga quanto tempo decorre até obter toda a informação, ignorando o DNS. (em mili segundos)

63 66 70 73 76 83 86 89 90 99 103 109 113 120 nenhum destes valores

Questão 3) Numa empresa optou-se por instalar um proxy de HTTP com cache, que todos passaram a usar, tendo-se verificado um *hit ratio* de 30%. Assumindo um RTT interno à empresa de 5ms e para o exterior de 100ms, tempos de transmissão desprezáveis, e que o cliente não tem qualquer conexão com o proxy, responda às seguintes questões (ignore o DNS):

a) Qual o tempo para obter a resposta para um pedido HTTP que se encontre na cache do proxy? (em ms)

1 2 5 10 15 20 25 30 nenhum destes valores

b) Qual o tempo médio para obter a informação correspondente a um pedido que não se encontre em cache ? (em ms)

150 200 210 240 250 270 290 300 315 400 nenhum destes valores

c) Tendo em conta o *hit ratio*, qual o tempo médio para obter uma qualquer página? (em ms)

150 200 210 240 250 270 290 300 315 400 nenhum destes valores

Questão 4 – A corretora *EuroPrev Caparica* decidiu oferecer um serviço de notificação aos seus clientes. Este serviço consiste em enviar alertas por SMS para o telemóvel do cliente sempre que as cotações do seu portfólio de ações sofrem alterações importantes. A interação com este serviço baseia-se em UDP e consiste no envio de uma mensagem de texto para o servidor, indicando o número de telemóvel e a lista de ações a monitorizar. Na EuroPrev, o servidor regista estes pedidos e, em simultâneo, de 30 em 30 segundos, avalia todos os portfólios registados para determinar se há ou não alertas a enviar. O envio das mensagens SMS foi contratado a um fornecedor externo e realiza-se através do protocolo HTTP, por recurso a um pedido POST, onde o corpo do pedido contém os dados do alerta a enviar (telemóvel e texto do alerta). O código seguinte implementa os referidos serviços e a sua utilização. **Complete o código em falta, ignorando eventuais problemas relacionados com a perda de mensagens:**

a) Complete o código do servidor para registo e monitorização dos portfólios (`StockExchange`).

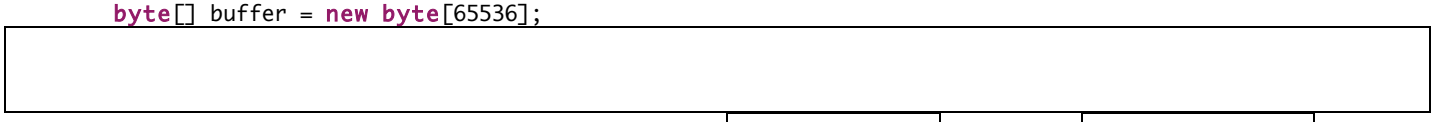
b) Complete o código do servidor do fornecedor externo que envia os SMS aos clientes (`SmsHttpGateway`).

c) Complete o código da aplicação que os clientes usam para registarem o seu portfólio no serviço (`Client`).

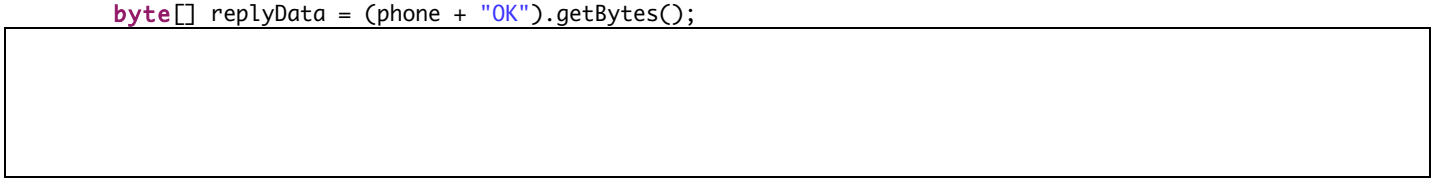
```
// código auxiliar:
public class HTTPUtilities {

 /**
  * Consome uma linha de texto terminada por CR+LF do InputStream dado e devolve-a numa String.
  */
  public static String readLine( InputStream is ) throws IOException ;

  /**
   * A partir da linha de pedido HTTP devolve um array com o metodo invocado
   * na posicao 0, o URL na posicao 1, e a versao na posicao 2 Por ex., para
   * "GET http://www.unl.pt/index.html HTTP/1.0", devovle [0]->"GET";
   * [1]->"http://www.unl.pt/index.html"; [2]->"HTTP/1.0" Caso request nao
   * seja um pedido HTTP valido, devolve null.
   *
   */
  public static String[] parseHttpRequest( String request) {}

  /**
   * A partir de um header HTTP, devolve um array com o nome do header na
   * posicao 0 e o valor indicado na posicao 1 Por ex., para "Connection:
   * Keep-alive", devovle [0]->"Connection"; [1]->"Keep-alive" Caso header nao
   * seja um header HTTP valido, devolve null.
   */
  public static String[] parseHttpHeader( String header) {}

  /**
   * A partir de uma string com o conteudo de um form submetido no formato
   * application/x-www-form-urlencoded, devolve um objecto do tipo Properties,
   * associando a cada elemento do form o seu valor
   */
  public static Properties parseHttpPostContents( String contents) {}
}

// Representa o portfolio de ações de um investidor, identificado pelo número de telemóvel
class Portfolio {
  String phone;
  List<String> stocks = new ArrayList<String>();

  public Portfolio( String phone ) {
    this.phone = phone;
  }

  void addStock( String stock ) {
    stocks.add( stock );
  }
  …
}
```

```
// Servidor de registo dos portfólios de ações
public class StockExchange implements                    {

  private int port;
  private static int smsHttpGatewayPort = 81 ;
  private static String smsHttpGateway = "sms.euronext.cc" ;

  private Set<Portfolio> portfolios = new HashSet<Portfolio>();

  StockExchange( int port ) {
    this.port = port ;
  }
  public static void main( String[] args ) throws Exception {
    if( args.length != 1 ) {
        System.err.println("use: java StockExchange port");
        System.exit(0);
    }
    new StockExchange( Integer.parseInt( args[0] ) ).doIt();
  }

  // A cada 30 segundos, verifica quais os portfólios que necessitam de alertas e requisita o seu envio
  public void            () {
    for(;;) {
       for( Portfolio p : portfolios ) {
         String alert = getStockAlerts( p ) ;
         if( ! alert.isEmpty() )
           postSmsGatewayHttpRequest( alert ) ;
       }
       try { Thread.sleep( 30000 ) ; } catch( Exception x ) {}
    }
  }
```

```
  // Aceita registos de pedidos de alerta e respectivo portfólio
  private void doIt() throws Exception {
       new                ( this ).            ();

                          ss =                                  ;
       for(;;) {
         byte[] buffer = new byte[65536];
```

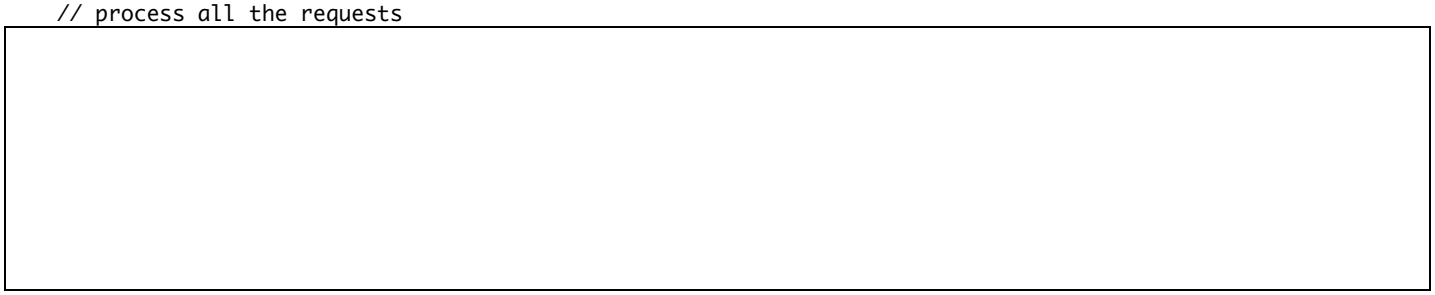


```
         Scanner scanner = new Scanner( new String( req.                ), 0, req.                ) );
         String phone = scanner.next();
         Portfolio p = new Portfolio(phone);
         while( scanner.hasNext() ) {
           String stock = scanner.next();
           p.addStock(stock);
         }
         portfolios.add( p );

         byte[] replyData = (phone + "OK").getBytes();
```

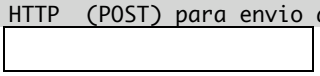


```
       }
  }

  // Obtém o alerta a enviar para um dado portfolio. Caso não necessite devolve “”.
  public String getStockAlerts( Portfolio p) { . . . < omitido > . . . }

  // Faz pedido por HTTP, através do método POST, ao servidor externo que envia os alertas por SMS
  void postSmsGatewayHttpRequest( String alert ) {
       byte[] alertData = alert.getBytes() ;

                          cs = new                                                  ;

                          is =                                  ();
                          os =                                  ();

       String request ;
       request = "POST /stockAlerts                  \r\n" ;
       request += "Content-Length:" + alertData.length + "\r\n" ;
       request +=                  ;

       os.                ( request.getBytes() ) ;
       os.                ( alertData ) ;

       String resp = HTTPUtilities.                  ( is ) ;
       while( HTTPUtilities.                                    )
         ;

       cs.                () ;

       if( resp.contains("200 OK") )
         System.out.println("Success...") ;
       else
         System.err.println("Request Failed..." + resp ) ;
  }
}
```

```
// Implementa a gateway HTTP que faz o envio de alertas por SMS
public class SmsHttpGateway {
  int port;
  public SmsHttpGateway(int port) {
    this.port = port;
  }
  public static void main(String[] args) throws Exception {
    if (args.length != 1) {
       System.err.println("usage: java SmsHttpGateway port");
       System.exit(-1);
    }
    int port = Integer.parseInt(args[0]);
    new SmsHttpGateway(port).doIt();
  }

  public void doIt() throws Exception {
    // process all the requests
```

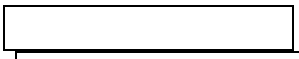


```
  }

  /* Acede ao hardware local que faz o envio de um SMS */
  void sendSmsAlert(String phone, String text) { . . . < omitido > . . . }

  /* Processa um pedido HTTP  (POST) para envio de um SMS */
  void processRequest(
```

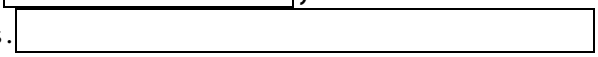


```
 cs ) throws Exception {
    String reply;
```

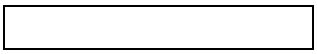
```
    String request = HTTPUtilities.
```



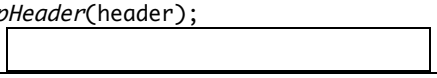
```
;
    String[] rparts = HTTPUtilities.
```



```
;
    if (rparts[0].equals("POST") && rparts[1].equals("/stockAlerts") ) {
        int contentLength = 0;
        String header;
        do {
           header = HTTPUtilities.
```

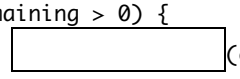


```
;
           String[] hparts = HTTPUtilities.parseHttpHeader(header);
           if (hparts != null && hparts[0].equals(
```

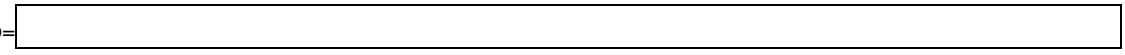


```
 ))
                      contentLength =
```

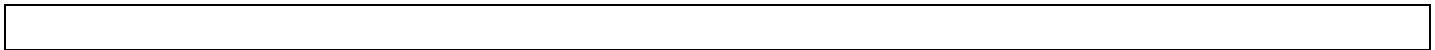
```
 ;
        } while (header.length() > 0);

        byte[] contents = new byte[contentLength];
        int remaining = contentLength;
        while (remaining > 0) {
           int n =
```



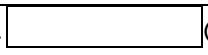
```
(contents, contentLength - remaining, remaining);
           remaining -= n;
        }
        Properties p=
```



```

        sendSmsAlert( p.getProperty("phone"), p.getProperty("alert") ) ;
        reply = "HTTP/1.0 200 OK\r\n\r\n";
    } else {
         reply = "HTTP/1.0 400 Bad Request\r\n\r\n";
    }
```

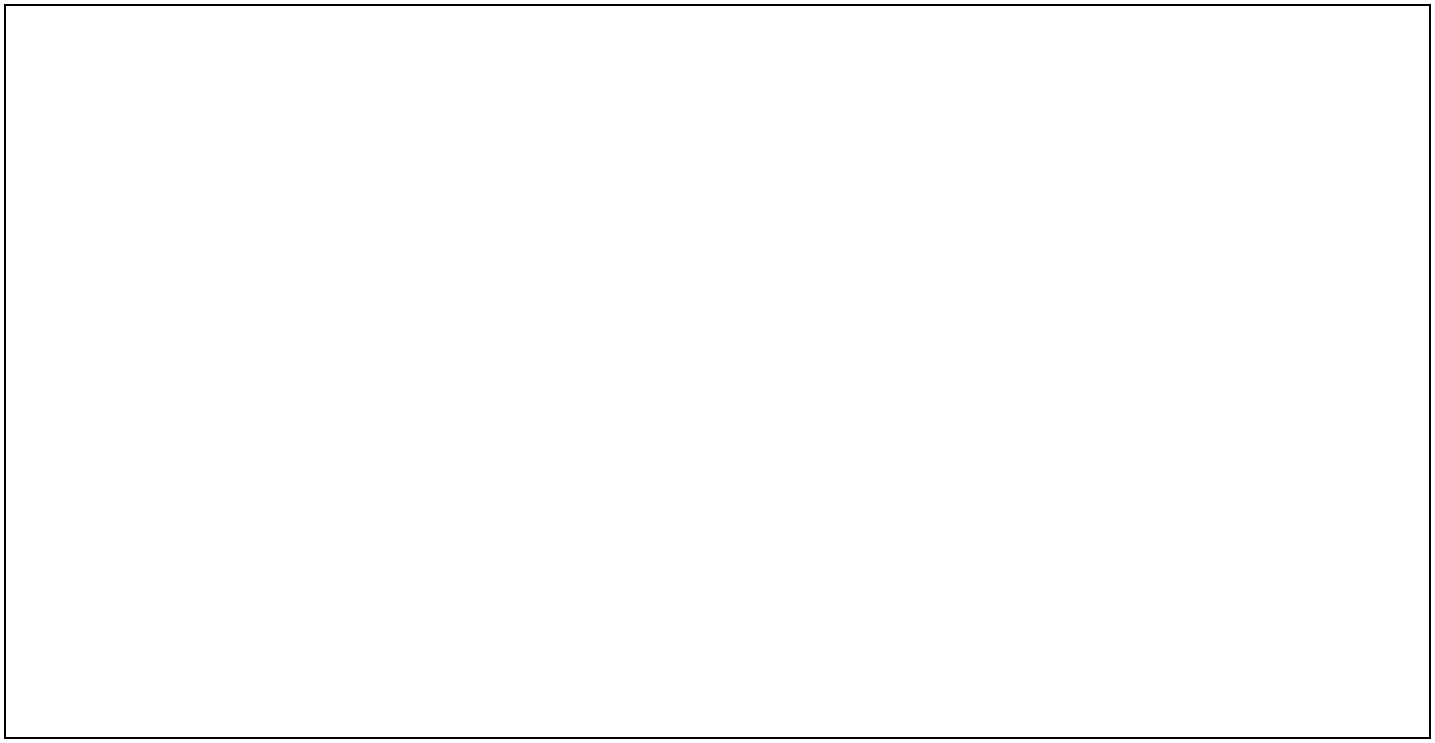


```
    cs.
```



```
() ;
  }
}
```

```
// Pede ao servidor para receber alertas sobre o portfólio passado na linha de comando
public class Client {

  public static void main(String[] args ) throws Exception {

    if( args.length < 4 ) {
       System.err.println("usar: java Client StockExchangeHost StockExchangePort phone stock1 [stock2 ...]") ;
       System.exit(0) ;
    }

    int port = Integer.parseInt( args[1]) ;
                    server =                          ( args[0] ) ;

    String phone = args[2] ;

    String request = phone ;
    for( int i = 3 ; i < args.length ; i++ )
       request += " " + args[i] ;
```



```
    System.out.println( reply ) ;  // print server reply
  }
}
```