FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Informática

Licenciatura em Engenharia Informática
2º TESTE– Redes de Computadores – Versão 1
1º Semestre, 2008/2009 (29/Abril/2009)

NOTAS:

- Leia com atenção cada questão antes de responder pois a interpretação do enunciado de cada pergunta é um factor de avaliação do teste. (Não se esclarecem dúvidas.)
- O tempo para realização do teste, 1h30, é propositadamente limitado.
- As respostas erradas nas perguntas de escolha múltipla **NÃO descontam**
- Pode responder a lápis
- Não é permitido o uso de máquina de calcular ou telemóvel
- O enunciado contém 8 páginas que devem ser entregues com a resposta ao teste.

NOME:_____ Nº Aluno: _____

Questão 1) Admita, por hipótese, que o RTT médio dentro da rede interna da FCT é de 5 ms (mili segundo) e que o RTT médio entre um computador da rede da FCT e qualquer servidor externo é de 50 ms. Estes valores são muito estáveis em todas as situações. Admita também que os computadores só usam o protocolo HTTP 1.0, que o tempo de transmissão de páginas WEB pelos servidores é desprezável e que todos os segmentos/pacotes necessários para conter as páginas WEB são transmitidos uns a seguir aos outros, sem mais demora (isto é, o TCP usa sempre uma janela tão grande quanto necessário e não se perdem pacotes).

a) Qual o tempo necessário para aceder a uma página WEB existente num servidor externo à FCT/UNL por um computador interno que já conhece o endereço IP do servidor?

10 20 25 26 27 28 29 30 31 40 45 46 50 51 52 53 54 55 100 101 102 103 104 105 106 107 108 109 110 120 130 140 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

b) Qual o tempo necessário para aceder a uma página WEB existente num servidor externo à FCT/UNL por um computador interno que não conhece o endereço IP do servidor. No entanto, este endereço é conhecido do servidor DNS local da FCT.

10 20 25 26 27 28 29 30 31 40 45 46 50 51 52 53 54 55 100 101 102 103 104 105 106 107 108 109 110 120 130 140 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

c) 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. Os computadores internos conhecem o endereço IP do *proxy*. Constatou-se que em 80% dos casos, uma página HTTP externa solicitada estava já no proxy (*cache hit ratio* = 80%). Admita que o proxy só usa o protocolo HTTP 1.0 e que quando não tem uma página WEB na sua cache não conhece o endereço IP do servidor que a serve mas este endereço é conhecido do servidor DNS local da FCT. 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?

10 20 25 26 27 28 29 30 31 40 45 46 50 51 52 53 54 55 100 101 102 103 104 105 106 107 108 109 110 120 130 140 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

Questão 2) Admita, por hipótese, que o RTT médio dentro da rede interna da FCT é de 5 ms (mili segundo) e que o RTT médio entre um computador da rede da FCT e qualquer servidor externo é de 50 ms. Estes valores são muito estáveis em todas as situações. Constatou-se que em 20% dos casos, uma *query* DNS ao servidor DNS local já estava na sua cache (*cache hit ratio* = 20%) e que outras 20% das *queries* diziam respeito a nomes locais, servidos pelo servidor local. Admita também que em média, para resolver um nome DNS que não esteja em cache, o servidor de DNS local necessita de 3 *queries* DNS iterativas a servidores DNS distintos pertencentes a domínios externos. Qual dos seguintes valores passou a ser o tempo médio aproximado às unidades que dura cada *query* DNS de clientes situados na rede interna da FCT e que usam por defeito o servidor local?

20 25 26 27 28 29 30 31 50 51 52 53 54 55 60 65 67 70 75 80 85 90 95 100
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
130 150 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

Questão 3) Admita que existe a página WWW com o URL **http://194.1.1.5/teste.html**. Fazendo a análise do seu conteúdo, conclui-se que contém texto e referências a 2 imagens com os seguintes URLs:

http:// 194.1.1.3/fig1.gif (250 bytes)
http:// 194.1.1.3/fig2.gif (350 bytes)

Entre o cliente e todos os servidores, o RTT é de 20 mili segundos e este valor é estável. Despreze o tempo de processamento e o tempo de transmissão de todos os pacotes envolvidos. Admita, também, que cada mensagem ou mensagens HTTP de pedido ou resposta cabe(m) num segmento TCP que cabe dentro de um único pacote IP de 1500 bytes (MTU). Ignore os cabeçalhos do HTTP.

a) Indique o tempo que o cliente demora 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 20 25 26 27 28 29 30 31 50 51 52 53 54 55
60 65 70 75 80 85 90 95 100 101 102 103 104 105 106 107 108 109 110 120 130 150
200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

b) Idem, mas em HTTP 1.1 com *pipelinning*:
1 2 3 4 5 6 7 8 9 10 20 25 26 27 28 29 30 31 50 51 52 53 54 55
60 65 70 75 80 85 90 95 100 101 102 103 104 105 106 107 108 109 110 120 130 150
200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

c) Quantos pacotes são trocados entre o cliente e o conjunto dos servidores no caso da **a)** admitindo que a abertura de uma conexão custa 3 pacotes, cada objecto cabe num pacote que tem de ser ACKed através de outro e que o fecho de uma conexão requer 4 pacotes ?

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 48 49 50 60 70 80 90 100 ou nenhum destes valores

d) Quantos pacotes são trocados entre o cliente e o conjunto dos servidores no caso da **b)** nas mesmas condições ?

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42

Questão 4) Entre um cliente e um servidor WWW o RTT é de 20 mili segundos e este valor é estável. Despreze o tempo de processamento e o tempo de transmissão de todos os pacotes envolvidos e admita também que cada mensagem HTTP de pedido cabe dentro de um segmento TCP que cabe dentro de um único pacote IP. O cliente conhece o endereço IP do ou dos servidores WWW envolvidos.

a) Indique o tempo espectável que o cliente leva a obter uma página usando o protocolo HTTP 1.0 sabendo que a página requer a transmissão de 7 segmentos e que o TCP do lado do servidor tem uma janela de emissão com a dimensão máxima de um segmento?

1 2 3 4 5 6 7 8 9 10 20 25 26 27 28 29 30 31 50 51 52 53 54 55
60 65 70 75 80 85 90 95 100 101 102 103 104 105 106 107 108 109 110 120 130 150
160 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

b) Indique o tempo espectável que o cliente leva a obter uma página usando o protocolo HTTP 1.0 sabendo que a página requer a transmissão de 7 segmentos e que o TCP do lado do servidor permanece sempre na fase de *slow start*?

1 2 3 4 5 6 7 8 9 10 20 25 26 27 28 29 30 31 50 51 52 53 54 55
60 65 70 75 80 85 90 95 100 101 102 103 104 105 106 107 108 109 110 120 130 150
160 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

c) Indique o tempo espectável que o cliente leva a obter uma página usando o protocolo HTTP 1.0 sabendo que a página requer a transmissão de 7 segmentos e que o TCP do lado do cliente tem sempre uma *receiver window size* da dimensão de 1 segmento

1 2 3 4 5 6 7 8 9 10 20 25 26 27 28 29 30 31 50 51 52 53 54 55
60 65 70 75 80 85 90 95 100 101 102 103 104 105 106 107 108 109 110 120 130 150
160 200 201 202 203 204 205 206 207 208 209 210 ou nenhum destes valores

---

(rascunho)

(rascunho)

Questão 5) Genericamente, um proxy tem a função de servir de intermediário entre dois sistemas. O código que se segue implementa um proxy HTTP/1.0 concorrente sem cache. Neste caso concreto, o proxy é capaz de processar quaisquer pedidos HTTP (GET, POST ou outros), solicitados pelos browsers, actuando de modo semelhante a um servidor, e efectuar esses pedidos aos verdadeiros servidores, encaminhando seguidamente as respostas de volta aos browsers. Complete o código preenchendo os espaços em falta e socorrendo-se das classes auxiliares Url e HTTPUtilities já suas conhecidas.

```
/**
 * Classe com metodos auxiliares para processar URL
 */
public class Url {

    /* Construtor que processa uma string contendo um URL
     * Exemplo de URL: "http://asc.di.fct.unl.pt/index.html"
     */
    public Url(String url) {

    /* Devolve o protocolo indicado no URL.
     * Para o exemplo anterior: "http"
     */
    public String getProtocol() {

    /* Devolve a maquina/ip.
     * Para o exemplo anterior: "asc.di.fct.unl.pt"
     */
    public String getHost() {

    /* Devolve o porto indicado no URL. Caso nao conste nenhum porto
     * devolve o valor por defeito passado como argumento.
     * Para o exemplo anterior: defaultPort
     */
    public int getPort( int defaultPort) {

    /* Devolve o pedido correspondente ao URL.
     * Para o exemplo anterior: "/index.html"
     */
    public String getRequest() {

}


/* Classe com metodos auxiliares para processar pedidos HTTP
 */
public class HTTPUtilities {

   /* Consome uma linha de texto do canal de entrada dado,
    * terminada por CR LF e retorna-a como uma String
       */
      public static String readLine( InputStream is ) throws IOException {

       /* Processa uma string contendo um pedido HTTP e 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",
    * devolve: [0]->"GET"; [1]->"http://www.unl.pt/index.html";
    * [2]->"HTTP/1.0
       */
      public static String[] parseHttpRequest( String request) {


       /* Processa uma string contendo um header HTTP e devolve um
    * array com o nome do header na posicao 0 e o valor em 1.
    * Por ex., para "Connection: Keep-alive", devolve:
    * [0]->"Connection"; [1]->"Keep-alive"
       */
      public static String[] parseHttpHeader( String header) {

}
```

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

public class ProxyHttpServer {

    int serverPort;

    protected ProxyHttpServer( int port ) {
        this.serverPort = port ;
    }

    /**** MAIN
     */
    public static void main(String[] args) throws IOException {
        if (args.length > 1) {
            System.err.println("Use: java ChatHttpServer [porto]");
            System.exit(0);
        }

        int port = 8080;  // default port
        if ( args.length==1 ) port = Integer.parseInt(args[0]);

        new ProxyHttpServer( port ).doIt();
    }

    /* Copia o conteudo do stream in para o stream out
     */
    protected void dumpStream(InputStream in, OutputStream out) throws IOException {
        byte[] arr = new byte[1024];
        for (;;) {
            int n = in.read(arr);
            if (n == -1) break;
            out.write(arr, 0, n);
        }
    }

    /* Processa o pedido que chega pela conexao 'x'
     */
    public void processRequest(                          x ) {
        try {
                                bin = x.                          ;
                                bout = x.                         ;

            String request = HTTPUtilities.readLine(bin);
            String[] tokens = HTTPUtilities.parseHttpRequest(request);

            Url theurl = new Url( tokens[1] );

                          y = new
                                                                  ;
            sout.write( (tokens[0] + " " +
                                                  + " HTTP/1.0\r\n").getBytes() );
            int bodySize = 0;
            String head =                                         ;
            while (                                          ) {
                String[] tk = HTTPUtilities.parseHttpHeader(head);
                if ( tk[0].equals( "Keep-Alive") || tk[0].equals( "Connection")
                        || tk[0].equals( "Proxy-Connection") ) continue;
                if (tk[0].equalsIgnoreCase("Content-Length"))
                                                              ;
                sout.write( (head+"\r\n").getBytes() );
                head =                                        ;
            }
```

```
        sout.write( [                                        ] );

        if ( [                              ] ) {
           byte[] buf = new byte[1024];
           while( bodySize > 0) {
              int n=[                                   ]
              sout.write(buf, 0, n);
              bodySize -= n;
           }
        }

        // resposta para o cliente
        dumpStream( [                                      ], bout);

        y.close();
        x.close();
     } catch (IOException e) {
        System.err.println("IO ERROR! "+e);
     }
  }

  /* Funcao principal do servidor - responsavel atender conexoes e
   * criar o thread que trata o pedido de cada conexao
   */
  void doIt() {
     try {
[                                                                          ]
     } catch (IOException e) {
        System.err.println("IO ERROR! "+e);
     }
  }


  /* classe auxiliar para tratamento de pedidos em concorrencia
   */
  class RequestHandler extends Thread {
     [                    ] s;

     RequestHandler( [                     ] s ) {
        this.s = s;
     }

     public void [                     ]() {
        try {
           processRequest( s );
        } catch ( Exception e ) {
        }
     }
  }
}
```