FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Informática

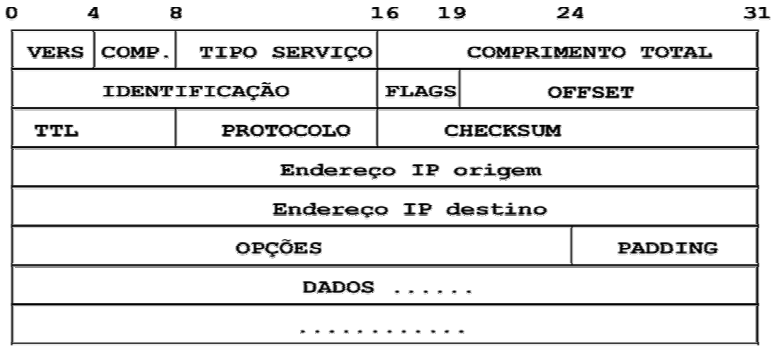
Licenciatura em Engenharia Informática
3º TESTE– Redes de Computadores – Versão 1
1º Semestre, 2008/2009 (3/Junho/2009)

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 (Questões 1 e 2) DESCONTAM**
- Pode responder a lápis
- Não é permitido o uso de máquina de calcular ou telemóvel
- O enunciado contém 7 páginas que devem ser entregues com a resposta ao teste.

NOME:_____ Nº Aluno: _____

Questão 1) O desenho apresenta um diagrama de um pacote IPv4. Assinale os campos que são modificados quando o pacote atravessa (sem fragmentação) um *router*. Escolha uma e uma só das opções.



0. Os campos TTL, Tipo de Serviço e Opções
1. Os campos TTL e OFFSET
2. Os campos TTL, CHECKSUM e FLAGS
3. Os campos TTL e CHECKSUM
4. Os campos TTL e Endereço Origem
5. Os campos TTL, CHECKSUM e Endereço Origem
6. Os campos Endereço Origem e Identificação
7. Os campos Identificação e Padding
8. Os campos CHECKSUM e Endereço Origem

Questão 2 Um *router* recebe um pacote IPv4 com endereço origem 130.45.3.3 e endereço de destino 201.23.4.6. Não existe nenhum prefixo de rede na tabela de encaminhamento do router que contenha esse endereço (nem sequer uma entrada por defeito). Como deve o *router* tratar esta situação? Assinale uma e uma só das opções abaixo.

1. Devolver o pacote ao endereço 130.45.3.3
2. Enviar o pacote para o router associado à rota por defeito
3. Enviar o pacote para a Internet
4. Destruir o pacote e avisar o router anterior
5. Destruir o pacote e avisar o router seguinte
6. Destruir o pacote e enviar um ICMP do tipo "Port Unreachable" para o endereço 201.23.4.6
7. Enviar o pacote para o endereço 201.23.4.6
8. Enviar o pacote para a rede a que pertence o endereço de destino
9. Avisar o host no endereço 130.45.3.3 de que não sabe encaminhar o pacote
10. Destruir o pacote e enviar um ICMP do tipo "Host Unreachable" para o endereço 201.23.4.6
11. Destruir o pacote e enviar um ICMP do tipo "Host Unreachable" para o endereço 130.45.3.3
12. Devolver o pacote ao router anterior

Questão 3 Um *router* tem uma interface Ethernet com o endereço IP 192.168.0.1/24 e recebe um pacote com endereço de destino 192.168.0.230. A tabela de ARP do *router* tem o seguinte conteúdo:

192.168.0.1	23:45:A0:4F:67:CD
192.168.0.4	23:45:AB:2F:67:AD
192.168.0.10	23:45:AB:2F:60:CD
192.168.0.67	23:45:CD:4A:67:2D

Quais os endereços origem e destino do(s) *frame(s)* Ethernet e respectivo tipo que o *router* deve enviar para encaminhar o pacote até ao seu destino final.

Tipo do conteúdo do frame 1:
Endereço origem:
Endereço destino:

Tipo do conteúdo do frame 2:
Endereço origem:
Endereço destino:

Tipo do conteúdo do frame 3:
Endereço origem:
Endereço destino:

Questão 4 Um *router* a executar o algoritmo de encaminhamento "Bellman-Ford", também conhecido por "Vector de Distâncias", tem uma tabela de encaminhamento com as seguintes entradas (da forma [destino, custo até ao destino, interface por onde enviar o pacote para este chegar a destino]):

Rede	**Custo**	**Interface**
A	4	eth2
B	3	eth3
C	1	eth6
D	5	eth7

Este recebe pela interface eth3 o seguinte anúncio vindo de um router directamente ligado e à distancia 1: A com custo 2, B com custo 3, C com custo 3, D com custo 7, E com custo 2. Indique de que forma fica a tabela de encaminhamento do router após o processamento do anúncio:

Rede	**Custo**	**Interface**
A		
B		
C		
D		
E		

Questão 5 Dadas as tabelas de encaminhamento dos routers R1 e R2:

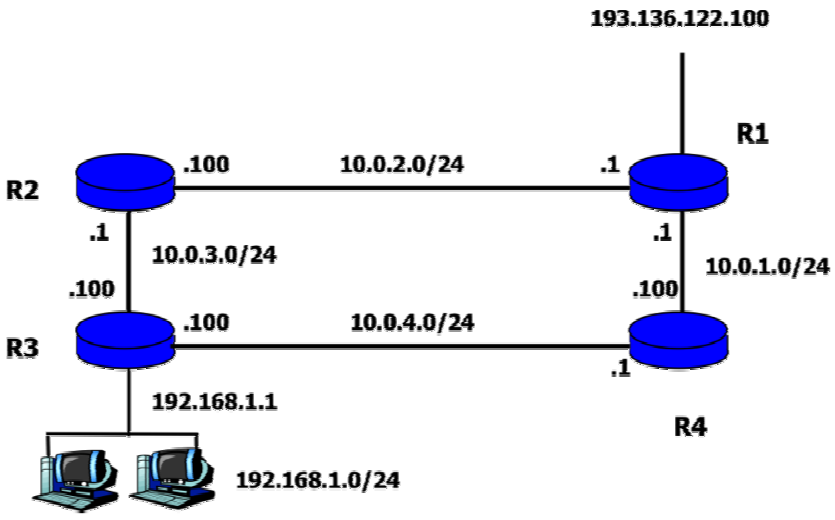
R1	**Rede/Máscara**	**Gateway ou Interface**
	10.0.1.0/24	Encaminhamento directo via a interface local usando ARP
	10.0.2.0/24	Encaminhamento directo via a interface local usando ARP
	10.0.3.0/24	10.0.2.100
	10.0.4.0/24	10.0.1.100
	192.168.1.0/24	10.0.1.100
	default	193.136.122.100

R2	**Rede/Máscara**	**Gateway ou Interface**
	10.0.1.0/24	10.0.2.1
	10.0.2.0/24	Encaminhamento directo via a interface local usando ARP
	10.0.3.0/24	Encaminhamento directo via a interface local usando ARP
	10.0.4.0/24	10.0.3.100
	192.168.1.0/24	10.0.3.100
	default	10.0.2.1

indique as tabelas de encaminhamento dos routers R3 e R4 (ignorando as interfaces excepto se forem locais):

R3	**Rede/Máscara**	**Gateway ou Interface**
	10.0.1.0/24	
	10.0.2.0/24	
	10.0.3.0/24	
	10.0.4.0/24	
	192.168.1.0/24	
	default	

R4	**Rede/Máscara**	**Gateway ou Interface**
	10.0.1.0/24	
	10.0.2.0/24	
	10.0.3.0/24	
	10.0.4.0/24	
	192.168.1.0/24	
	default	



Questão 6. Imagine uma unidade de produção industrial onde, por questões de segurança, existe a necessidade de monitorizar a temperatura dos equipamentos em funcionamento. Para tal, existem espalhados pela fábrica pequenos computadores, ligados em rede e munidos de sensores de temperatura, que designaremos genericamente por *sensores*. No sistema existem também computadores que funcionam como monitores (operados automaticamente ou por pessoas) e cujo papel é colectar periodicamente as medidas de temperatura dos sensores.

Cada sensor pode ser monitorizado por vários computadores ao mesmo tempo. Para isso, recorre-se a um grupo multicast para cada sensor, para onde este irá difundir as suas temperaturas. Todos os monitores interessados nesse sensor devem subscrever o respectivo grupo para receber as suas temperaturas.

Para facilitar a configuração do sistema, cada sensor, quando arranca, regista-se num computador central, sendo-lhe por este atribuído o respectivo endereço multicast e porta para onde este passará a difundir as suas temperaturas. Qualquer computador, que deseje monitorizar um sensor, pode perguntar por esse sensor junto da central, para descobrir o respectivo endereço multicast e porta. Seguidamente, pode receber as leituras de temperatura enviadas pelo sensor do respectivo grupo multicast e porta.

O código em anexo representa uma hipotética implementação da central de registo de sensores (listagem **SensorCentral.java**); dos sensores (listagem **TemperatureSensor.java**) e dos monitores (listagem **Monitor.java**). Note que o endereço IP da central pode variar não sendo por isso conhecido à partida, tendo os clientes da central (sensores ou monitores) que usar um endereço multicast (225.0.0.0) e um porto (1225), predefinidos, para contactar a central.

Neste cenário, a comunicação é toda suportada por troca de linhas de texto (strings), de acordo com o seguinte protocolo aplicacional:

- Registo do sensor na central: o sensor envia uma mensagem UDP à central contendo "REGISTER *sensor_id*"; como resposta o sensor recebe "OK *multicast_IP port*".
- Consulta da central pelo cliente monitor: o monitor envia uma mensagem UDP contendo "SENSOR *sensor_id*". A central responde ao monitor com uma mensagem UDP contendo "OK *multicast_ip port*" se este estiver registado ou "ERROR <unknown sensor>" se 'sensor_id' não estiver registado.
- Envio das leituras das temperaturas do sensor para os seus monitores: Cada valor de temperatura é enviado com uma linha na forma de "DATA *sensor_id currTemperature*", usando o respectivo grupo.

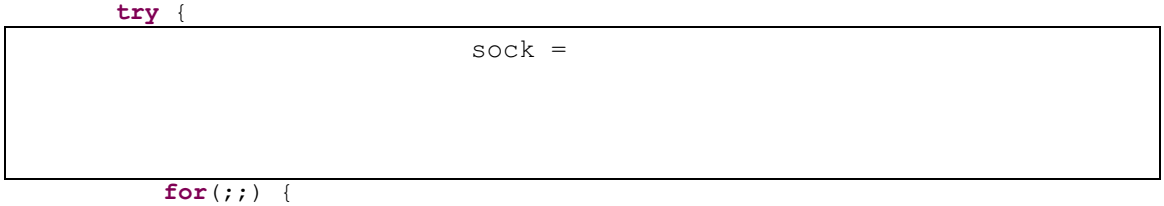
a) Complete os espaços deixados em branco na listagem **SensorCentral.java.**
b) Complete os espaços deixados em branco na listagem **TemperatureSensor.java.**
c) Implemente o monitor na listagem **Monitor.java. Note que a interacção com a central tem de ser fiável, ou seja, estar preparada para o caso de perca de mensagens UDP.**

SensorCentral

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

public class SensorCentral {

    public static final InetSocketAddress DISCOVERY_LOCATION =
                                    new InetSocketAddress("225.0.0.0", 1225) ;

    private Map<String, InetSocketAddress> groups =
                                    new HashMap<String, InetSocketAddress>() ;
    private void doIt() {
        try {
                                sock =

            for(;;) {
                byte[] tmp = new byte[65536] ;
                                req =new                (tmp,tmp.length);
                sock.                ( req ) ;

                String reqString = new String( req.getData(),0,req.getLength());
                Scanner ss = new Scanner( reqString ) ;
                String cmd = ss.next().trim() ;

                String reply ;
                if( cmd.equals("SENSOR") )
                       reply = handleSensorCMD(ss.next().trim()) ;
                else if( cmd.equals("REGISTER") )
                       reply = handleRegisterCMD(ss.next()) ;
                     else reply = "ERROR" ;

                byte[] data = reply.getBytes() ;
                                r = new                (data,data.length);
                r.
                r.
                sock.                ( r ) ;
            }
        } catch( Exception x ) {  x.printStackTrace() ;  }
    }

    private String handleSensorCMD( String id ) {
        InetSocketAddress res = groups.get( id ) ;
        if( res != null )
            return "OK "+ res.getAddress().getHostAddress()+ " "+ res.getPort();
        else   return "ERROR <Unknown sensor>" ;
    }

    private String handleRegisterCMD( String id ) {
        InetSocketAddress res = groups.get( id ) ;
        if( res == null ) {
            Random rg = new Random() ;
            String group = String.format("%d.%d.%d.%d",
                        226, rg.nextInt(255), rg.nextInt(255), rg.nextInt(255));
            res = new InetSocketAddress( group, 1000 + rg.nextInt(64535)) ;
            groups.put( id, res ) ;
        }
        return "OK "+ res.getAddress().getHostAddress() + " " + res.getPort();
    }

    public static void main( String[] args ) {
        if( args.length != 0 ) {
            System.err.println("use: java SensorCentral") ;
            System.exit(0) ;
        }
        new SensorCentral().doIt() ;
    }
}
```

TemperatureSensor

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

import static SensorCentral.DISCOVERY_LOCATION;

public class TemperatureSensor {

    static private double getTemperature() {    // retorna a temperatura lida
        return Math.random();
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("use: java TemperatureSensor sensor_id");
            System.exit(0);
        }
        String sensor_id = args[0] ;
        String r = registerInSensorCentral( sensor_id ) ;

        Scanner ss = new Scanner( r ) ;
        if( ! ss.next().equals("OK") ) {
            System.out.println("Could not register sensor. Exiting...") ;
            return ;
        }
        InetSocketAddress dst = new InetSocketAddress( ss.next(),ss.nextInt());
                                sock = new

        for (;;) {
            double val = getTemperature();
            byte []data = ("DATA " + sensor_id + " " + val).getBytes();
                          m = new                    (data, data.length );
            m.
            m.

            sock.                 ( m ) ;
            Thread.sleep(10);
        }
    }

    private static String registerInSensorCentral(String sensor_id)
                                                          throws IOException {
                                sock = new
        byte[] data = ("REGISTER " + sensor_id).getBytes() ;

                      m = new                  (data, data.length );
        m.
        m.

        sock.                 (m);

        byte[] buf = new byte[65536]
                     r = new
        sock.                 (r);
        String res = new String (r.getData(), 0, r.getLength() );
        sock.                 ();
        return res ;
    }
}
```

Monitor

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

import static SensorCentral.DISCOVERY_LOCATION;

public class Monitor {

    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("usar: java Client sensor_id");
            System.exit(0);
        }
        String sensor_id = args[0];

        String r = querySensorCentral(sensor_id);

        Scanner ss = new Scanner(r);
        if (ss.next().equals("OK")) {
            InetSocketAddress group =
                                new InetSocketAddress(ss.next(), ss.nextInt());
```

```
                                s2 =
```

```
            for (;;) {
                byte[] buf = new byte[65536];
                                  p = new                    (buf, buf.length);
                s2.                    (p);

                String data = new String(p.getData(), 0, p.getLength());
                System.out.println(data);
            }
        }
    }

    private static String querySensorCentral( String id )
                                                        throws IOException  {
        // pede ao SensorCentral o grupo que identifica o Sensor pretendido
```

```
        return res;
    }
}
```