1º parcial de Redes - ETSINF - 9 de junio de 2011

Apellidos, Nombre:_____

Grupo de matrícula:_____

1. ***(0,5 puntos)*** ¿Qué acción conlleva poner a *true* el parámetro ***autoFlush*** del constructor de la clase ***PrintWriter***?

A la hora de emplear flujos de salida, en algunos casos es necesario garantizar que los datos introducidos en el flujo se transmiten inmediatamente. Para ello se emplea la orden ***flush()*** , que vacía el *buffer* de transmisión. El ***autoFlush*** permite omitir la invocación de este método, ya que tras cada línea transmitida (indicada mediante el método ***println***) se procede automáticamente al vaciado del *buffer*.

2. ***(0,5 puntos)*** ¿Para qué se utiliza la orden ***PORT*** del protocolo FTP en un cliente? Detalla los parámetros de dicha orden.

Se utiliza para indicar al servidor FTP la direccion IP y puerto con los que tiene que establecer la conexión de datos. Tiene 6 parámetros enteros de 8 bits: n1, n2, n3, n4, n5 y n6. Los 4 primeros son la dirección IP, mientras que los dos últimos indican el puerto, definido como n5*256+n6

3. ***(0,5 puntos)*** Si ***ds*** es un objeto del tipo **DatagramSocket**, ¿Cómo es posible conocer la longitud del campo de datos del datagrama UDP recibido tras ejecutar ***ds.receive(dp)***?

Empleando el método ***getLength()*** de la clase ***DatagramSocket***, es decir, mediante la sentencia:

```
int longitud = ds.getLength()
```

4. ***(0,5 puntos)*** ¿Cómo determina un servidor web implementado en Java el valor que acompaña a la cabecera ***Content-Length*** en su respuesta? Indica método y clase a la que pertenece el método.

La cabecera ***Content-Length*** marca la longitud del objeto contenido en el cuerpo de la respuesta. Puesto que este objeto generalmente se almacena en forma de fichero, será necesario emplear el método ***length*** de la clase ***File***. Así, si el nombre del fichero está contenido en el *String* ***objeto***:

```
File f = new File (objeto);
int longitud = f.length()
```

5. ***(1 punto)*** Los computadores A y B están conectados mediante una red de conmutación de paquetes. La ruta que siguen los paquetes que viajan de A a B atraviesa 20 *routers*. Calcula el tiempo necesario para transmitir completamente (sin control de flujo ni error) un fichero de $5x10^6$ bytes desde A hasta B. Considera que el tamaño de los paquetes es de $5x10^3$ bytes de datos más una cabecera de 100 bytes. Todos los enlaces tienen una longitud de 500 metros. La velocidad de transmisión de todos los enlaces es 1 Gbps, y la velocidad de propagación es de $2x10^8$ m/s. Supón que el único tráfico en la red es el que se especifica anteriormente, y que el tiempo de procesamiento en los *routers* es despreciable.

Número de paquetes: $N = 5x10^6 / 5x10^3 = 10^3$ paquetes
Tamaño de un paquete: $L = (5000+100)*8 = 40800$ bits

Distancia de cada enlace: D = 500 m; Número de enlaces a atravesar: 21 enlaces
v_{trans}= 1 Gbps = 10^9 bits/s; v_{prop}= $2x10^8$ m/s

Calculamos los tiempos de transmisión y propagación por paquete:
$t_{trans} = L / v_{trans} = 40800 \times 10^{-9}$ s = 40800 ns;
$t_{prop} = D / v_{prop} = 2500$ ns

El tiempo necesario para que llegue a B el primer paquete:
$21 \cdot (40800 + 2500) = 909300$ ns = 0,909 ms

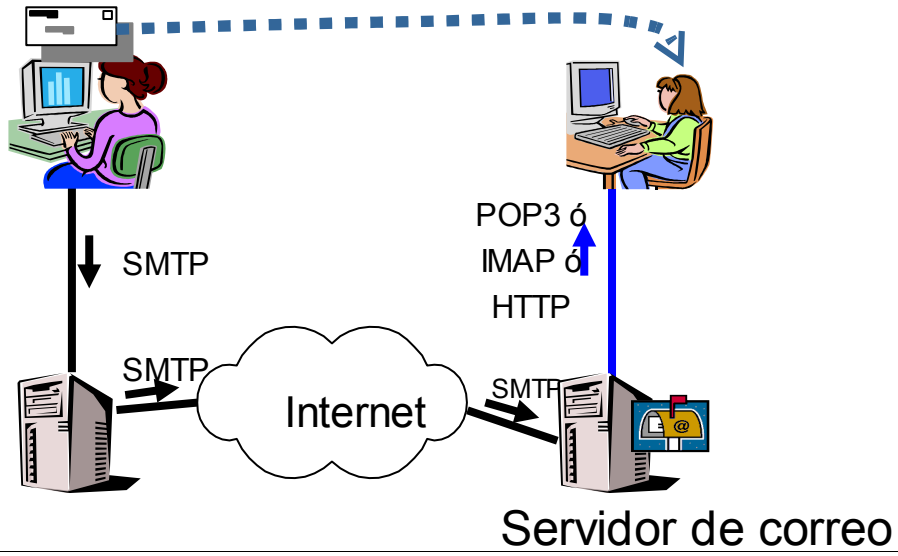
Para que lleguen los 999 restantes:
$999 \cdot 40800 = 40759200$ = 40,759 ms
Tiempo total = 0,909 + 40,759 = **41,668 ms**

6. ***(1,5 puntos)***

a) Dibuja un diagrama que muestre la arquitectura de la aplicación de correo electrónico. En el diagrama debes indicar cuáles son los roles de los sistemas que participan y los protocolos de aplicación que conozcas que puedan emplearse para intercambiar información entre ellos.

b) Describe la secuencia de pasos que se aplican sobre un correo electrónico desde que el remitente lo escribe hasta que el destinatario lo lee utilizando un servicio de *webmail* como ***Gmail***, asumiendo que el emisor y el destinatario tienen buzones de correo de servidores distintos.

a)



b)

La secuencia de pasos será la siguiente:

1.- El emisor del mensaje lo redacta mediante su agente de usuario.

2.- El agente de usuario emplea el protocolo SMTP para enviar el correo a su servidor predeterminado, mediante el protocolo SMTP

3.- El servidor predeterminado inspecciona el correo, determina que el buzón destino se encuentra en otro servidor y averigua, mediante DNS, la dirección IP del servidor destino.

4.- El servidor predeterminado actúa como cliente SMTP para hacer llegar el correo al servidor final, donde reside el buzón destino. Este servidor final actuará como servidor SMTP.

5.- El servidor final almacena el correo en el buzón correspondiente.

6.- El usuario destinatario entra, utilizando un cliente web a su proveedor de servicio de webmail.

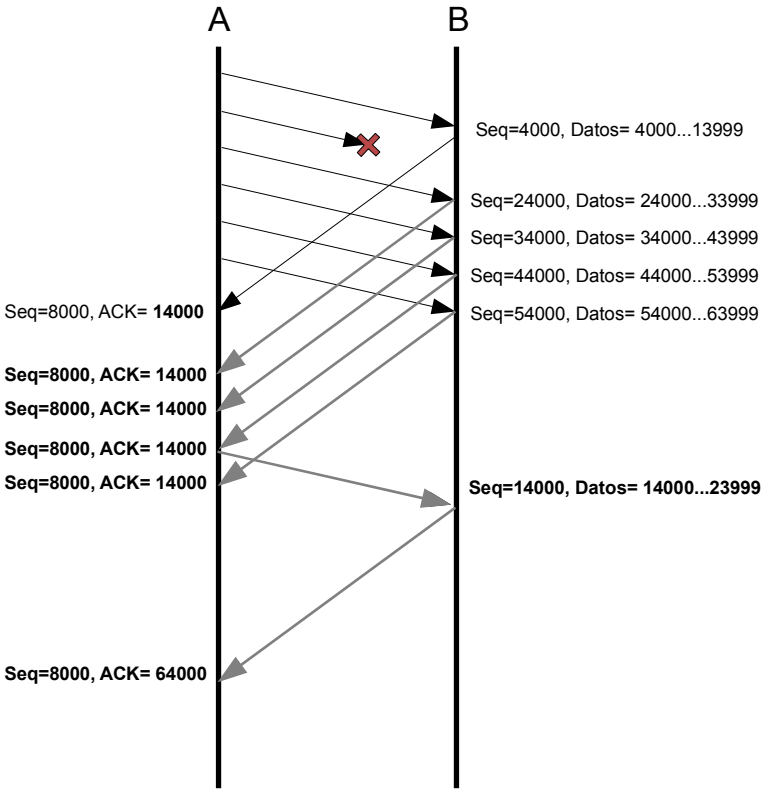
7.- El proveedor de servicio de webmail accede vía POP3 o IMAP al servidor de correo del destinatario y visualiza el correo en el cliente web del destinatario.

7. ***(1,5 puntos)*** Escribe un programa en java que, dada la URL de un servidor web, visualice en la pantalla qué software está utilizando (por ejemplo, Apache) gracias a la cabecera ***Server:***

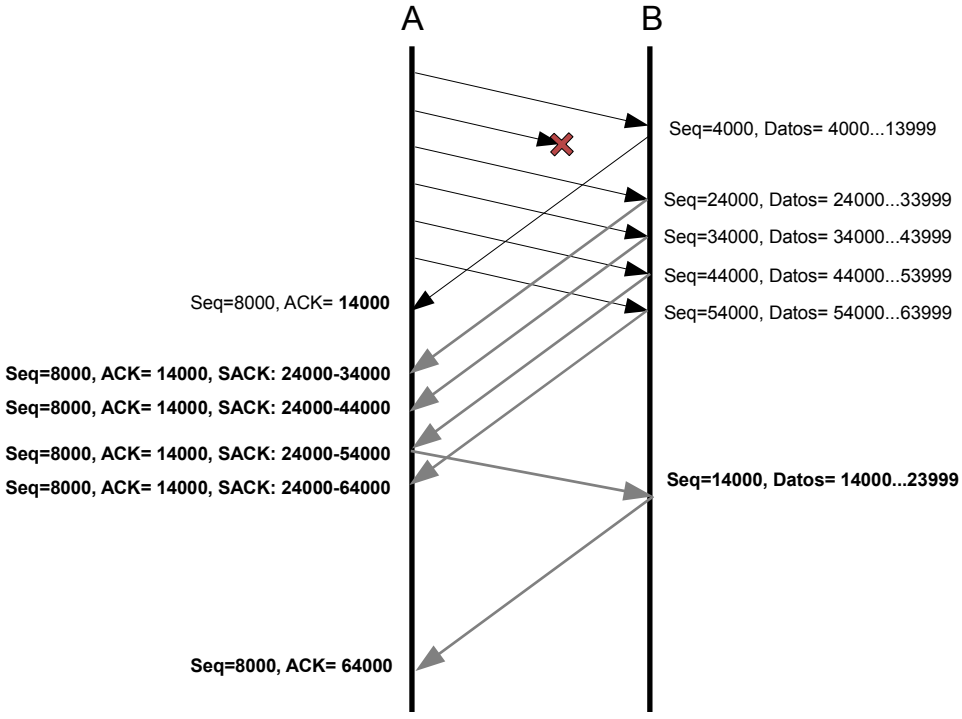
```
import java.net.*;
import java.io.*;
import java.util.*;
class queWebServer {
  public static void main(String args[]) {
    try {
       String host = args[0].substring(7);
       host = host.substring(0,host.indexOf('/'));
       int port = 80;
       Socket s=new Socket(host,port);
       Scanner entrata=new Scanner(s.getInputStream());
       PrintWriter uscita=new  PrintWriter(s.getOutputStream());
       uscita.print("GET / HTTP/1.0\r\n\r\n");
       uscita.flush();
       String linea=entrata.nextLine();
       while(linea.length()!=0) {
              if (linea.startsWith("Server:"))
                     System.out.println(linea);
        linea=entrata.nextLine();
       }
      s.close();
    }
    catch (Exception e) {
      System.err.println(e);
      System.err.println("Uso: java queWebServer <URL>");
    }
  }
}
```

8. ***(1 punto)***

a) Suponiendo que se emplean ACK's retrasados y que B no tiene datos para transmitir, completa el intercambio de segmentos y los campos que faltan en los segmentos mostrados (ACK) hasta que A retransmite el segmento perdido.



b) Lo mismo pero suponiendo además que A y B han acordado el uso de reconocimientos selectivos (SACK's). Indica expresamente en cada segmento el valor de los campos necesarios para implementar esta opción



9. ***(1 punto)*** En un protocolo de la capa de transporte: ¿Cuál es la diferencia entre Control de Flujo y Control de Congestión? ¿Cómo detecta TCP la existencia de congestión en la red? ¿La pérdida de paquetes siempre implica congestión?

El control de flujo se realiza entre los *hosts* (extremo a extremo) para evitar que el transmisor envíe más datos de los que el receptor pueda gestionar, en cuyo caso, el receptor los descartaría.

El control de congestión se realiza individualmente en cada *host*. Intenta evitar que la red tenga que descartar los paquetes que no es capaz de gestionar como consecuencia de un exceso de tráfico.

Otra diferencia importante es que, mientras el control de flujo se realiza entre dos dispositivos reales *(hosts)*, el control de congestión se basa en suposiciones del estado de la red asumidos por un único *host.*

TCP asume la existencia de congestión en la red cuando suceden eventos relacionados con la pérdida de segmentos, es decir, el vencimiento de un temporizador, o se reciben 3 ACKs duplicados, considerándose en este último caso que la congestión es menos severa.

No obstante, la pérdida de un segmento puede ser consecuencia de otros eventos (errores de transmisión, aumento de la latencia de la red, caída de dispositivos intermedios o del otro host, etc), que no necesariamente están ligados a la congestión. Sin embargo, son sucesos menos frecuentes que la pérdida por congestión.

10. ***(2 puntos)*** Empleando el método ***SetSoTimeout()***, y sabiendo que genera la excepción ***SocketTimeoutException,*** implementa un programa en Java que muestre por pantalla qué puertos UDP **bien conocidos** de un computador responden a un datagrama vacío. El nombre de dicho computador se indica como primer parámetro de entrada. Si este nombre es incorrecto, el programa debe generar un mensaje de error. No es necesario comprobar el **número** de parámetros de entrada. Descarta la posibilidad de pérdida de datos enviando tres datagramas a cada puerto estudiado. Considera 100ms como tiempo de espera. Es necesario verificar el origen de los datagramas de respuesta.

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

public class PortScannerUDP
{
 public static void main(String[] args) throws IOException
 {
   DatagramSocket s = new DatagramSocket();
   InetAddress dir = InetAddress.getByName(args[0]);
   String msg = "";
   int respuesta=0;
   byte[] buf = new byte[256];
   buf = msg.getBytes();
   s.setSoTimeout(100);
   System.out.println ("Listado de puertos UDP que contestan a un datagrama sin datos");
   System.out.println ("============================================================");
```

```
    for (int port=1;port < 1025; port++ )
    {
        DatagramPacket p = new DatagramPacket(buf, 0 , dir, port);
        DatagramPacket pr = new DatagramPacket(buf, buf.length);
        for (int i=0; i<3; i++)
            s.send(p); // Se envian tres datagramas seguidos.
        Respuesta=0; // No hay respuesta todavía.
        try
        {
            while (true)
            {
                s.receive(pr);  // Si el paquete corresponde con la petición
                if ((pr.getPort() == port) && (dir.equals(pr.getAddress())) && (respuesta==0))
                {
                    System.out.print(port + ", "); // El puerto responde.
                    Respuesta=1; // Los duplicados no muestran otra vez este puerto.
                }
            }
        }
        catch (Exception e)
        { // Salida del while(true)
        }
    }
    System.out.println ("Fin.");
   }

}
```