

## Tema 5 Capa de aplicación

### Fundamentos de Redes

Doble Grado en Ingeniería Informática y Matemáticas  
Doble Grado en Ingeniería Informática y ADE  
Curso 2021 – 2022

#### Jorge Navarro Ortiz

Departamento de Teoría de la Señal, Telemática y Comunicaciones  
E.T.S. Ingenierías Informática y Telecomunicación – Universidad de Granada  
C/ Periodista Daniel Saucedo Aranda, s/n - 18071 – Granada (Spain)  
Teléfono: +34-958 241000, ext 20042 - Fax: +34-958 243032 - Email: [jorgenavarro@ugr.es](mailto:jorgenavarro@ugr.es)

© 2022



1



### Tema 2. Capa de aplicación

## Esquema

1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
3. La navegación web
4. El correo electrónico
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

2






2



Tema 2. Capa de aplicación

## Objetivos del tema

-  Conocer las aplicaciones y servicios estándar en Internet, identificando los protocolos y servicios de usuario más relevantes a nivel de red, transporte y aplicación.
-  Conocer el funcionamiento del modelo cliente/servidor.
-  Desarrollar programas básicos de transmisión de datos

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

3



3



Tema 2. Capa de aplicación

## Bibliografía



Capítulo 2 (2.1, 2.2, 2.4, 2.5), James F. Kurose y Keith W. Ross. **COMPUTER NETWORKING. A TOP-DOWN APPROACH**, 5ª Edición, Addison-Wesley, 2010, ISBN: 9780136079675.

 Para saber más: capítulos 7 y 8



Capítulo 11, Pedro García Teodoro, Jesús Díaz Verdejo y Juan Manuel López Soler. **TRANSMISIÓN DE DATOS Y REDES DE COMPUTADORES**, Ed. Pearson, 2ª Edición, 2014, ISBN: 9788490354612.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

4



4



Tema 2. Capa de aplicación

## Esquema

1. **Introducción a las aplicaciones de red**
2. Servicio de Nombres de Dominio (DNS)
3. La navegación web
4. El correo electrónico
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

5



5

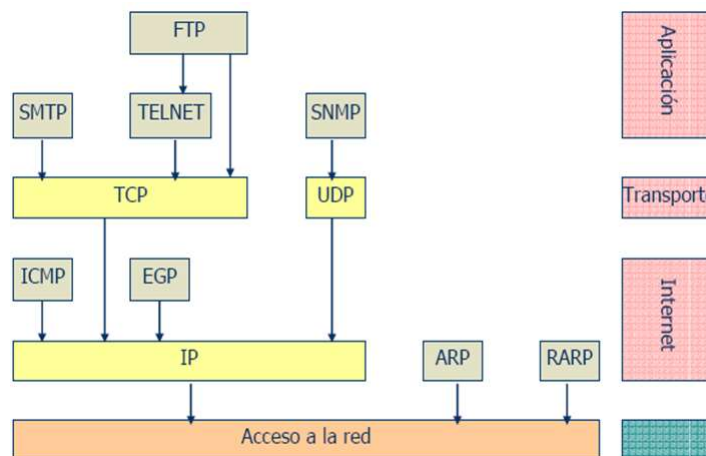


Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

PROTOCOLOS TCP/IP

### Estructura de protocolos



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

6



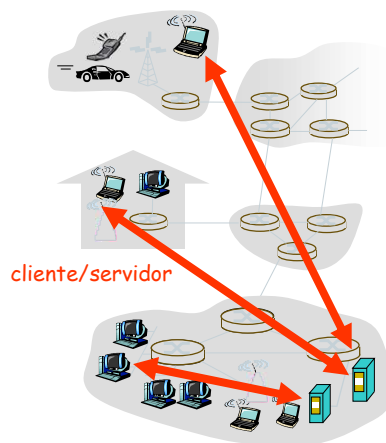
Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### INTERACCIÓN CLIENTE/SERVIDOR

#### Arquitectura cliente-servidor

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



#### Servidor:

- Siempre en funcionamiento
- IP permanente & pública
- Agrupados en "granjas"
- <http://www.xatakandroid.com/mundo-android/la-imagen-de-la-semana-google-muestra-el-corazon-de-internet>
- <https://www.youtube.com/watch?v=zRwPSFpLX8I>

#### Clientes:

- Funcionando intermitentemente
- Pueden tener IP dinámica & privada
- Se comunican con el servidor
- No se comunican entre sí



7



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### INTERFAZ SOCKET

**Proceso Cliente** : proceso que inicia la comunicación

**Proceso Servidor**: proceso que espera a ser contactado

→ IP permanente & pública

➤ Proceso envía/recibe mensajes a/desde su **socket**

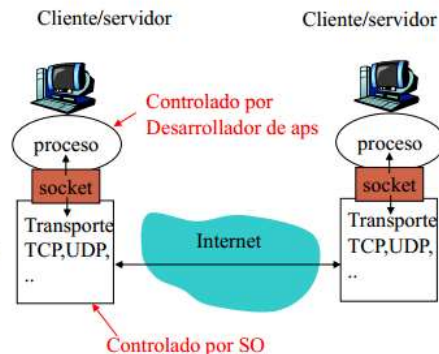
➤ Para recibir mensajes un proceso debe tener un **identificador (IP + puerto)**

Ej: servidor web [gaia.cs.umass.edu](http://gaia.cs.umass.edu):

Dirección IP: 128.119.245.12

Número de puerto: 80

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



8



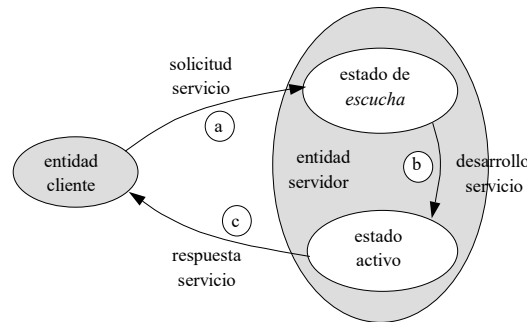
Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

INTERFAZ SOCKET

La mayoría de las transacciones entre aplicaciones se basan en el paradigma cliente-servidor:

- **Servidor:** programa que ofrece un servicio accesible a través de la red.
- **Cliente:** programa que envía peticiones y espera respuestas del servidor a través de la red.



### Diferencias:

- El servidor comienza antes (apertura pasiva).
- El servidor se ejecuta de forma permanente.
- El servidor usa puertos reservados.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

9



9



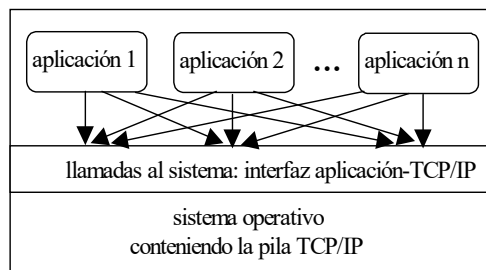
Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

INTERFAZ SOCKET

- El software TCP/IP es parte del S.O.
- Las aplicaciones lo usan a través de una **API** consistente en **llamadas al sistema**:

"la interfaz socket"



La definición de la **interfaz socket** no es parte de ningún protocolo.

Distintas implementaciones:

- Berkeley Socket Distribution,
- Winsock,
- Transport Layer Interface, etc.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

10



10



Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## INTERFAZ SOCKET

### La interfaz socket (BSD):

- Punto de partida: la gestión de I/O del S.O. se basa en:  
abrir-leer/escribir-cerrar
- Implementado mediante llamadas al sistema:  
open, read, write, close
- Concepto importante: el **descriptor de fichero**.
- La interfaz socket extiende la I/O a conexiones en red.
- Se necesita añadir la identificación de los puntos finales:  
IP local, IP remota, puerto local y puerto remoto
- Definimos **SOCKET** como un **descriptor** de una transmisión a través del cual la aplicación puede enviar y/o recibir información hacia y/o desde otro proceso de aplicación.
- Es una "puerta" de acceso entre la **aplicación** y los servicios de **transporte**.

11



11

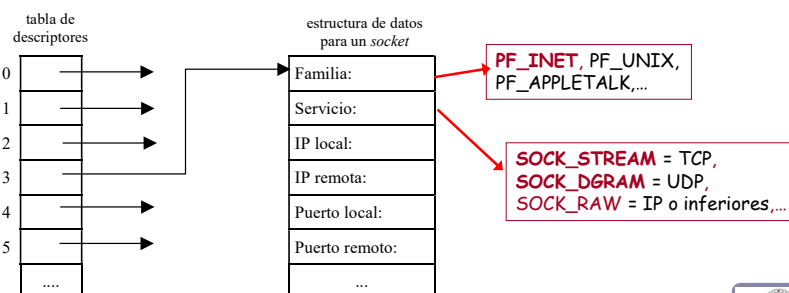


Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## INTERFAZ SOCKET

- Definimos **SOCKET** como un **descriptor** de una transmisión a través del cual la aplicación puede enviar y/o recibir información hacia y/o desde otro proceso de aplicación.
- Es una "puerta" de acceso (metafóricamente) entre la **aplicación** y los servicios de **transporte**.
- En la práctica un **socket** es una variable tipo puntero a una estructura:



12



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### INTERFAZ SOCKET

- Especificar la dirección IP y puerto LOCAL:

```
int bind (int socket, struct sockaddr *myaddr, int addresslen)
```

- Para pasar direcciones se define la estructura:

```
struct sockaddr_in {           /* INET socket addr info */
    short sin_family;           /* familia: AF_INET */
    u_short sin_port;           /* puerto: 16 bits, nbo */
    struct in_addr sin_addr;     /* dir IP de 32 bits */
    char sin_zero[8];           /* no usada */
};

struct in_addr {
    u_long s_addr;              /* dir IP de 32 bits, nbo */
};
```

...es necesario hacer un casting :

0	16	31	0	16	31
familia	dirección <sub>0-1</sub>		AF_INET(2)	puerto	
dirección <sub>2-5</sub>			dirección IP		
dirección <sub>6-9</sub>			00...00		
dirección <sub>10-13</sub>			00...00		

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

13



13



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### INTERFAZ SOCKET

- Especificar la dirección IP y puerto REMOTOS:

```
int connect (int socket,
             struct sockaddr *toaddr,
             int addresslen)
```

- Si el socket es **SOCK\_STREAM**, **connect** envía un SYN (TCP hand-shaking) a través de la red. Exige simultaneidad de los dos procesos.
- Si el socket es **SOCK\_DGRAM**, **connect** NO envía nada a través de la red. Sólo especifica las dirección IP y puerto remoto. Esto facilita su utilización posterior.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

14



14





Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### INTERFAZ SOCKET

- Para poner un socket en modo pasivo en el servidor y definir el número de solicitudes de conexión pendientes que se encolarán:

```
int listen (int sockfd, int maxwaiting)
```

no es una llamada "bloqueante".

- Para detener el flujo del programa y esperar hasta que llegue una solicitud de conexión en el servidor:

```
int accept (int sockfd,  
            struct sockaddr *fromaddrptr,  
            int *addresslen)
```

devuelve un socket nuevo conectado para comunicarse con los clientes.

15



15



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### INTERFAZ SOCKET

- Para enviar (escribir)...

```
int sendto (int sockfd, char *buff, int buflen, int flags,  
            struct sockaddr *toaddrptr, int addresslen)
```

o si el socket está previamente "conectado"...

```
int send (int sockfd, char *buff, int buflen, int flags)
```

- Para recibir (leer)...

```
int recvfrom (int sockfd, char *buff, int buflen,  
              int flags, struct sockaddr *fromaddrptr,  
              int *addresslen)
```

o si el socket está previamente conectado...

```
int recv (int sockfd, char *buff, int buflen, int flags)
```

- Para cerrar ...

```
int close (int sockfd)  
int shutdown (int sockfd, int how)
```

16



16







Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

INTERFAZ SOCKET

## Otras llamadas...

- `read()` / `readv()` / `recvmsg()`
- `write()` / `writv()` / `sendmsg()`
- `gethostbyname()`
- `getservbyname()`
- `getprotobyname()`
- `htons()` / `htonl()`
- `ntohs()` / `ntohl()`
- Ficheros cabecera típicos:
  - `netinet/in.h`
  - `sys/types.h`
  - `sys/socket.h`

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

17



17



Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

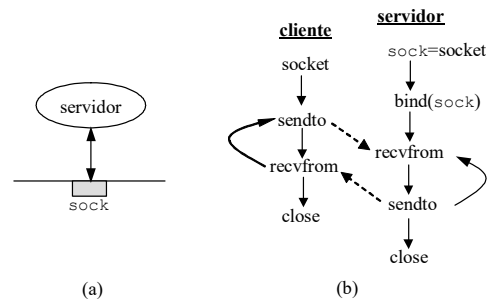
TIPOS DE SERVIDORES

## Tipos de servidores:

### Criterios clasificación:

- Orientados a Conexión - No Orientados a Conexión
- Iterativos - Concurrentes

## Servidor iterativo no orientado a conexión:



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

18



18

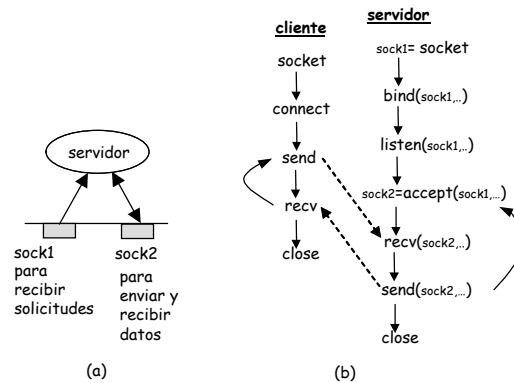


Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## TIPOS DE SERVIDORES

### Servidor iterativo orientado a conexión:



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

19



19

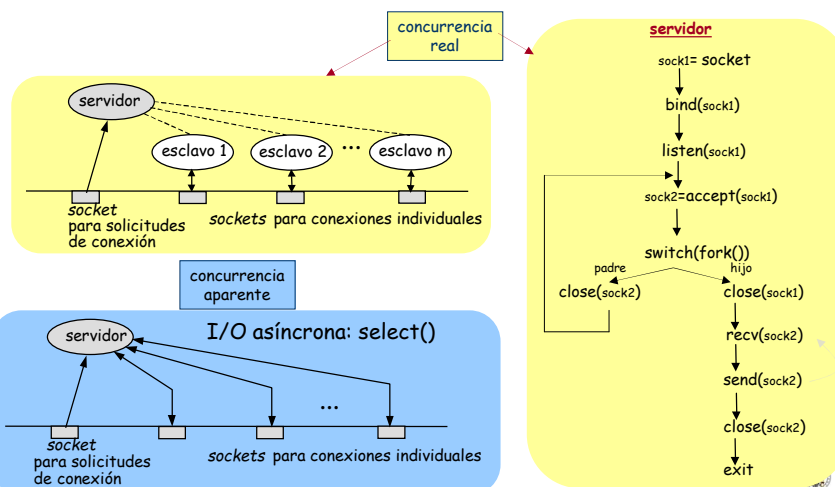


Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## TIPOS DE SERVIDORES

### Servidores concurrentes orientados a conexión:



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

20



20



Tema 2. Capa de aplicación

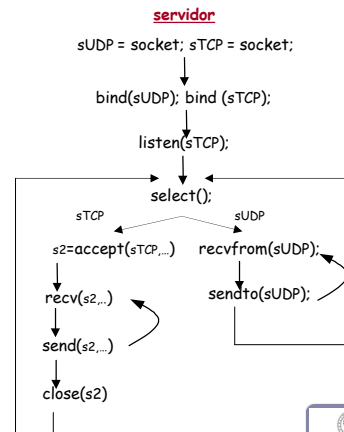
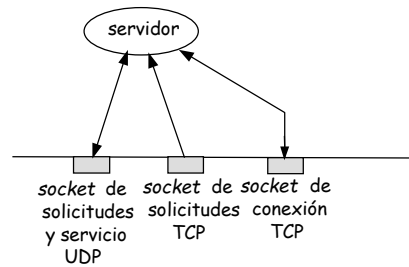
## 1. Introducción a las aplicaciones de red

### TIPOS DE SERVIDORES

#### Servidores multiprotocolo con I/O asíncrona (select):

- Un solo proceso iterativo ofrece varios protocolos.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



21



21



Tema 2. Capa de aplicación

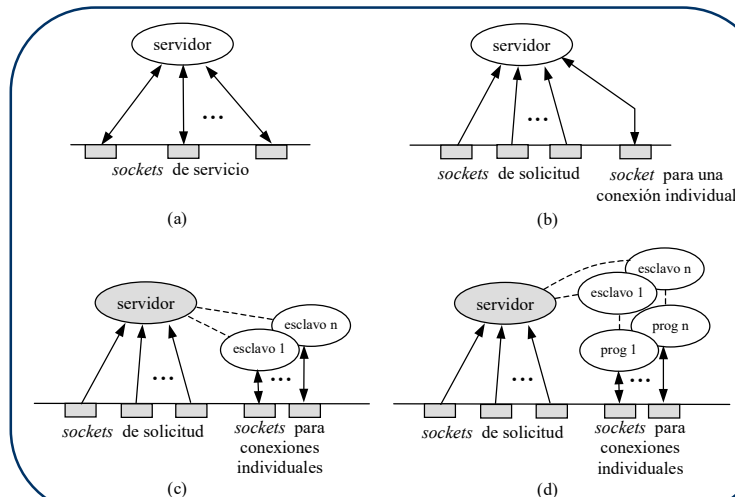
## 1. Introducción a las aplicaciones de red

### TIPOS DE SERVIDORES

#### Servidores multiservicio:

- Ofrecen varios servicios con concurrencia.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



22



22



Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## CLIENTE TCP - SOCKETS BSD

## SOCKETS - EJEMPLOS

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Saler y Jorge Navarro Ortiz

```
/*
 * File: cliente.c
 */

#include <stdio.h>
#include <malloc.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/time.h>

#include <stdlib.h>
#include <stdio.h>

/*
 * Cliente TCP
 */
int
main(int argc, char** argv) {

    int socket_datos;
    struct sockaddr_in sockname;
    char buffer[82];

    // Obteniendo parámetros de la línea de comandos...
    if (argc!=3)
        perror("Sintaxis: cliente IP_servidor puerto_servidor"),
        exit(1);

    char *servidor = argv[1]; // Dirección IP del servidor
    int port = atoi(argv[2]); // Puerto de escucha del servidor

    // Creando el socket de datos...
    if((socket_datos=socket(AF_INET,SOCK_STREAM,0))!=-1)
        perror("Cliente: error en la llamada a la función socket"),exit(1);

    // Asignando puerto y dirección...
    sockname.sin_family=AF_INET;
    sockname.sin_addr.s_addr=inet_addr(servidor);
    sockname.sin_port=htons(port);

    // Conectándose con el servidor
    if(connect(socket_datos,(struct sockaddr *)&sockname,
        sizeof(sockname))!=-1)
        perror("Cliente: error en la llamada a la función connect"),exit(1);

    // Bucle para enviar mensajes hasta introducir "FIN"...
    do{
        printf("Teclee el mensaje a transmitir:\n");
        gets(buffer);

        printf ("Has tecleado: %s\n", buffer);

        // Mandando datos a través del socket...
        if(send(socket_datos,buffer,80,0)!=-1)
            perror("Cliente: error en la llamada a la función send"),exit(1);
        }while(strcmp(buffer,"FIN")!=0);

    // Cerrando el socket de datos...
    close(socket_datos);

    exit (0);
}
```

23

Universidad de Granada



Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## SERVIDOR ITERATIVO TCP - SOCKETS BSD

## SOCKETS - EJEMPLOS

```
/*
 * File: servidor.c
 */

#include <stdio.h>
#include <malloc.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/time.h>

#include <stdlib.h>

/*
 * Servidor TCP iterativo
 */
int
main(int argc, char** argv) {

    int socket_control, from_len, socket_datos;
    struct sockaddr_in from, sockname;
    char buffer[82];
    int salir=0;

    // Obteniendo parámetros de la línea de comandos...
    if (argc!=2)
        perror("Sintaxis: servidor puerto_servidor"),exit(1);

    int port = atoi(argv[1]);

    // Creando el socket de control (para aceptar conexiones)...
    if((socket_control=socket(AF_INET,SOCK_STREAM,0))!=-1)
        perror("Servidor: error en la llamada a la función socket"),exit(1);

    // Asignando dirección y puerto...
    sockname.sin_family=AF_INET;
    sockname.sin_addr.s_addr=INADDR_ANY;
    sockname.sin_port=htons(port);

    // Anunciándose como servidor...
    if(bind(socket_control,(struct sockaddr *)&sockname,sizeof(sockname))!=-1)
        perror("Servidor: error en la llamada a la función bind"),exit(1);

    // Diciendo que será un socket de escucha...
    if(listen(socket_control,1)!=-1)
        perror("Servidor: error en la llamada a la función listen"),exit(1);

    // Bucle infinito para aceptar peticiones...
    do{

        // Aceptando conexiones de los diferentes clientes
        // (creándose un nuevo socket de datos)...
        from_len=sizeof(from);
        socket_datos=accept(socket_control,(struct sockaddr *)&from,&from_len);
        if(socket_datos!=-1)
            perror("Servidor: error en la llamada a la función accept"),exit(1);

        // Recibiendo un mensaje y escribiéndolo en pantalla (hasta recibir FIN)...
        do{
            int nbytes = recv(socket_datos,buffer,80,0);
            if(nbytes!=-1)
                perror("Servidor:Recv"),exit(1);

            if (nbytes==0)
                perror("El cliente se ha desconectado"),exit(1);

            printf("El mensaje recibido fue:\n%s\n",buffer);
        }while(strcmp(buffer,"FIN")!=0);

        // Cerrando el socket de datos...
        close (socket_datos);
    }while(!salir);

    // Cerrando el socket de control...
    close (socket_control);

    // Terminando el programa...
    exit(0);
}
```

24

Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

SERVIDOR CONCURRENTE TCP - SOCKETS BSD

SOCKETS - EJEMPLOS

```

/*
 * File: servidorconcurrente.c
 */

#include <stdio.h>
#include <malloc.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/time.h>

#include <stdlib.h>

/*
 * Servidor TCP concurrente
 */
int
main(int argc, char** argv) {
    int socket_control, from_len, socket_datos;
    struct sockaddr_in from, sockname;
    char buffer[B2];
    int salir=0;

    // Obteniendo parámetros de la línea de
    // comandos...
    if (argc!=2)
        perror("Sintaxis: servidor
        puerto_servidor"),exit(1);

    int port = atoi(argv[1]);

    // Creando el socket de control (para aceptar
    // conexiones)...
    socket_control=socket(AF_INET,SOCK_STREAM,0);
    if((socket_control==1)
        perror("Servidor: error en la llamada a la
        función socket"),exit(1);

        // Asignando dirección y puerto...
        sockname.sin_family=AF_INET;
        sockname.sin_addr.s_addr=INADDR_ANY;
        sockname.sin_port=htons(port);

        // Anunciándose como servidor...
        if(bind(socket_control,(struct sockaddr *)
            &sockname,sizeof(sockname))!=-1)
            perror("Servidor: error en la llamada a la
            función bind"),exit(1);

        // Diciendo que será un socket de escucha...
        if(listen(socket_control,1)==-1)
            perror("Servidor: error en la llamada a la
            función listen"),exit(1);

        // Bucle infinito para aceptar peticiones...
        do{

            // Aceptando conexiones de los diferentes
            // clientes (creándose un nuevo socket de
            // datos)...
            from_len=sizeof(from);
            if((socket_datos=accept(socket_control,
                (struct sockaddr *) &from,&from_len))!=-1)
                perror("Servidor: error en la llamada a la
                función accept"),exit(1);

            // Creación de un proceso hijo para atender al
            // cliente que se conectó...
            int pid; // Identificador del proceso padre
            if ((pid=fork())==0){
                // Proceso hijo =
                // PROCESO DE ATENCIÓN AL CLIENTE
            }

        } while(1);
    }
    
```

```

// Recibir un mensaje y almacenarlo en
// pantalla (mientras no se reciba FIN)...
do{
    int nbytes=recv(socket_datos,buffer,80,0);
    if(nbytes==1)
        perror("Servidor: error en la llamada a la
        función recv"),exit(1);

    if (nbytes==0)
        perror("El cliente se ha desconectado"),
        exit(1);

    printf("El mensaje recibido fue:\n%s\n",
        buffer);

}while(strcmp(buffer,"FIN")!=0);

// Cerrando el socket de datos...
close(socket_datos);

// Finalización del proceso de atención al
// cliente...
exit(0);

} else {
    // Proceso padre =
    // PROCESO QUE ESPERA PETICIONES

    // No hace nada, el bucle hará que vuelva a
    // esperar una petición (accept()).
}

}while(salir);

// Cerrando el socket de control...
close(socket_control);

// Terminando el programa...
exit(0);
    
```

25

Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

CLIENTE TCP - JAVA

SOCKETS - EJEMPLOS

```

/**
 * <p>Title: Mínimo cliente TCP</p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2007</p>
 * <p>Company: UGR</p>
 * <p>Author not attributable
 * <p>@version 1.0
 */

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

public class MinimoClienteTCP {

    // Atributos de la clase:
    static Socket socket_datos;
    static String direccionServidor; // Nombre o dirección IP
    static int puerto;
    static PrintWriter out;
    static BufferedReader in;

    public MinimoClienteTCP() {

    }

    public static void main (String args[]) {
        boolean error=false;
        String mensajeSolicitud;
        String mensajeRespuesta = "";
    }
    
```

```

// Se piden 3 argumentos: dirección del servidor, puerto y mensaje a enviar.
if (args.length<3) {
    System.err.println("Sintaxis: MinimoClienteTCP <direccion-servidor>
    <puerto> <mensaje a enviar>");
    System.exit(-1);
}

// Dirección (IP o nombre) del servidor
direccionServidor = args[0];
// Puerto
puerto = Integer.parseInt(args[1]);
// Mensaje a enviar
mensajeSolicitud = args[2];

// 1 - Se abre el socket y se conecta a la dirección y puerto del servidor.
try {
    socket_datos = new Socket (direccionServidor, puerto);

    // Se obtienen los flujos de lectura y escritura para recibir y enviar
    // mensajes.
    out = new PrintWriter (socket_datos.getOutputStream(), true);
    in = new BufferedReader (new
        InputStreamReader(socket_datos.getInputStream()));
} catch (UnknownHostException e) {
    System.err.println ("Error: no se pudo encontrar al servidor " +
        direccionServidor);
    System.exit(-2);
} catch (IOException e) {
    System.err.println("Error: no se pudo establecer la conexión con el
    servidor");
}

// (Continúa en la siguiente transparencia ...)
    
```

26



Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## SOCKETS - EJEMPLOS

### CLIENTE TCP - JAVA (continuación)

```
// (...) continuación de MinimoClienteTCP

// 2 - Se escribe el mensaje de solicitud y leemos la respuesta:
out.println(mensajeSolicitud);
try {
    mensajeRespuesta = in.readLine();
} catch (IOException e) {
    System.err.println("Error: no se pudo leer la respuesta.");
}

// 3 - Se cierra la conexión.
try {
    in.close();
    out.close();
    socket_datos.close();
} catch (IOException e) {
    System.err.println("Error: no se pudo cerrar la conexión.");
}

// Se muestra la respuesta:
System.out.println("El mensaje enviado fue: " + mensajeSolicitud);
System.out.println("El mensaje recibido fue: " + mensajeRespuesta);
}
```

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

27



27



Tema 2. Capa de aplicación

# 1. Introducción a las aplicaciones de red

## SOCKETS - EJEMPLOS

### SERVIDOR ITERATIVO TCP - JAVA

```
/**
 * <p>Title: Mínimo servidor iterativo TCP</p>
 */
import java.net.*;
import java.io.*;

public class MinimoServidorTCP {

    // Atributos de la clase:
    static ServerSocket socket_control;
    static Socket socket_datos;
    static int puerto;
    static PrintWriter out;
    static BufferedReader in;

    public MinimoServidorTCP() {

    }

    public static void main (String args[]) {
        boolean salir = false;
        boolean error = false;
        String mensajeSolicitud;
        String mensajeRespuesta;

        // El argumento es el puerto donde se
        // atenderá el servicio.
        if (args.length > 1) {
            System.err.println("Sintaxis:
            MinimoServidorTCP <puerto>");
            System.exit(-1);
        }

        // Se obtiene el puerto:
        puerto = Integer.parseInt(args[0]);

        // 1 - Se abre el socket en modo "escucha"
        try {
            socket_control = new ServerSocket (puerto);
        } catch (IOException e) {
            System.err.println("Error: no se puede abrir el puerto
            indicado.");
            System.exit(-2);
        }

        // Es un servidor iterativo: acepta una conexión, la
        // procesa y la cierra. Después se acepta otra conexión
        // y así sucesivamente.
        do {
            // 2 - Se bloquea la hebra actual en "accept", y se
            // devuelve la conexión establecida con el cliente.
            try {
                socket_datos = socket_control.accept();

                // Se obtienen los flujos de entrada y salida para
                // recibir y enviar mensajes.
                try {
                    out = new PrintWriter
                    (socket_datos.getOutputStream(), true);
                    in = new BufferedReader (new InputStreamReader
                    (socket_datos.getInputStream()));
                } catch (IOException e) {
                    System.err.println("Error: no se pudo obtener un
                    canal para los flujos");
                    error = true;
                }

                // 3 - Código del servicio ofrecido.
                // 3a) Se lee una línea del cliente
                mensajeSolicitud = in.readLine();

                // 3b) Se aplica el servicio:
                mensajeRespuesta =
                procesoServicio(mensajeSolicitud);

                // 3c) Se envía la respuesta:
                out.println(mensajeRespuesta);

            } catch (IOException e) {
                System.err.println("Error: no se pudo aceptar la
                solicitud de una conexión");
                error = true;
            }

            } while (!salir);

        }

        static String procesoServicio (String mensaje) {
            // Aquí se podría poner el código que procesara el
            // mensaje recibido. Actualmente sólo lo devuelve tal
            // cual vino.
            return mensaje;
        }
    }
}
```

28

Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### SERVIDOR CONCURRENTE TCP - JAVA

### SOCKETS - EJEMPLOS

```

/**
 * <p>Title: Mínimo servidor concurrente TCP</p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2007</p>
 * <p>Company: UGR</p>
 * <p>author not attributable</p>
 * <p>@version 1.0</p>
 */
import java.net.*;
import java.io.*;

public class MinimoServidorConcurrenteTCP {
    // Atributos de la clase
    static ServerSocket socket_control;
    static Socket socket_datos;
    static int puerto;
    static Servicio servicio;
    // Hebras para que el servicio sea concurrente

    public MinimoServidorConcurrenteTCP() {
    }

    static public void main (String args[]) {
        boolean salir = false;
        boolean error = false;

        // Al menos un argumento, el puerto.
        if (args.length > 1) {
            System.err.println ("Sintaxis:
                MinimoServidorConcurrenteTCP <puerto>");
            System.exit(-1);
        }

        // Se obtiene el puerto.
        puerto = Integer.parseInt(args[0]);

        // 1 - Se abre el socket en modo "escucha".
        try {
            socket_control = new ServerSocket (puerto);
        } catch (IOException e) {
            System.err.println ("Error: no se puede abrir el
                puerto indicado.");
            System.exit(-2);
        }
    }
        
```

```

        // Bucle para aceptar conexiones. Por cada conexión
        // aceptada se creará una hebra a la que se le pasará el
        // socket para cursar el servicio.
        do {
            // 2 - Se bloquea la hebra actual en "accept"
            // esperando una solicitud de conexión. Se devuelve
            // un nuevo socket con la conexión establecida.
            try {
                socket_datos = socket_control.accept();

                // Se lanza una hebra para que sirva a este cliente
                // por "socket_datos".
                new Servicio(socket_datos).start();
            } catch (IOException e) {
                System.err.println("Error: no se pudo aceptar la
                    solicitud de una conexión.");
                error = true;
            }
        } while (!salir);
    }

    class Servicio extends Thread {
        // Atributos de la clase
        Socket socket_datos;
        PrintWriter out;
        BufferedReader in;

        // El constructor recibirá como argumentos el socket
        // (abierto) que debe utilizar (se lo pasa la hebra
        // principal del servidor).
        public Servicio (Socket socket_datos_) {
            socket_datos = socket_datos_;
        }

        // Se obtienen los flujos de lectura y de escritura para
        // enviar y recibir mensajes.
        try {
            out = new PrintWriter
                (socket_datos.getOutputStream(), true);
            in = new BufferedReader (new InputStreamReader
                (socket_datos.getInputStream()));
        }
        
```

```

        } catch (IOException e) {
            System.err.println(this.getName() + " Error: no
                se pudo obtener un canal para los flujos.");
        }
    }

    public void run() {
        String mensajeSolicitud = "";
        String mensajeRespuesta = "";

        try {
            // 3 - Código del servicio ofrecido.
            // 3a - Se lee el mensaje del cliente.
            mensajeSolicitud = in.readLine();
        } catch (IOException e) {
            System.err.println(this.getName() + " Error: no
                se pudo leer el mensaje.");
        }

        // 3b - Se aplica el servicio.
        mensajeRespuesta = procesaServicio
            (mensajeSolicitud);

        // 3c - Se envía la respuesta.
        out.println(mensajeRespuesta);

        // 4 - Se cierra la conexión establecida con
        // el cliente.
        try {
            in.close();
            out.close();
            socket_datos.close();
        } catch (IOException e) {
            System.err.println(this.getName() + " Error: no
                se pudo cerrar la conexión.");
        }
    }

    static String procesaServicio (String mensaje) {
        // Aquí se podría poner el código que procesara
        // el mensaje recibido.
        // Actualmente sólo lo devuelve tal cual vino.
        return mensaje;
    }
        
```

29

Tema 2. Capa de aplicación

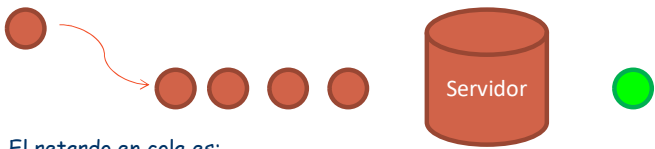
## 1. Introducción a las aplicaciones de red

### RETARDO EN COLA

### RETARDO EN COLA

➤ Para estimar los retardos (tiempos) en cola se usa la teoría de colas:

- El uso de un servidor se modela con un sistema M/M/1 (ver **TRANSMISIÓN DE DATOS Y REDES DE COMPUTADORES**)



➤ El retardo en cola es:

$$R = \frac{\lambda \cdot (T_s)^2}{1 - \lambda \cdot T_s}$$

■ donde  $T_s$  (distribución exponencial) es el tiempo de servicio y  $\lambda$  (Poisson) la ratio de llegada de solicitudes.

➤ Esta misma expresión se puede utilizar para calcular el retardo en cola en un router.

30





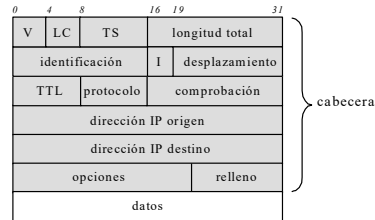
Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### ¿QUÉ DEFINEN LOS PROTOCOLOS DE APLICACIÓN?

➤ ¿Qué es y qué define un protocolo de aplicación?

- El tipo de servicio
  - - Orientado o no orientado a conexión
  - - Realimentado o no
- El tipo de mensaje
  - ej., request, response,
- La sintaxis:
  - Definición y estructura de "campos" en el mensaje
  - En aplicación generalmente son orientados a texto (HTTP)
  - Aunque hay excepciones (DNS)
  - Tendencia : usar formato Type-Length-Value
- La semántica:
  - Significado de los "campos"
- Las reglas:
  - Cuando los procesos envían mensajes/responden a mensajes



➤ Tipos de protocolos:

- Protocolos de dominio público (Definidos en RFCs (ej., HTTP, SMTP)) versus propietarios → (ej., Skype, IGRP)
- Protocolos in-band versus out-of-band
- Protocolos stateless versus state-full
- Protocolos persistentes versus no-persistentes (sobre servicios SOC)



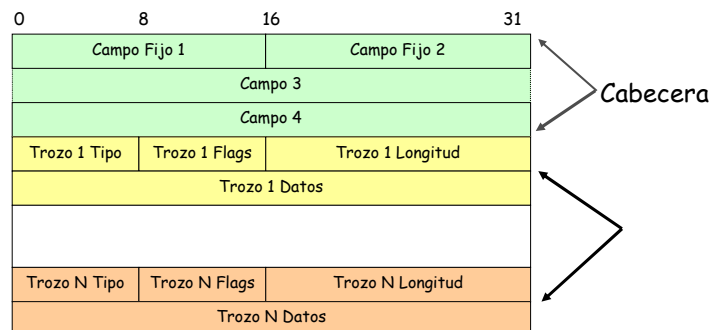
Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### ¿QUÉ DEFINEN LOS PROTOCOLOS DE APLICACIÓN?

➤ Tendencia: hacer los protocolos flexibles con

- Una cabecera fija
- Una serie de "trozos" (obligatorios y opcionales)





Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

¿QUÉ DEFINEN LOS PROTOCOLOS DE APLICACIÓN?

- **Tendencia: hacer los protocolos flexibles con**
  - **Una cabecera fija**
  - **Una serie de "trozos" (obligatorios y opcionales)**
    - Los trozos pueden incluir una cabecera específica más una serie de datos en forma de parámetros:
      - Parámetros fijos: en orden
      - Parámetros de longitud variable u opcionales.
      - Para los parámetros se usa Formato TLV (Type-Length-Variable)

0	8	16	31
Tipo de parámetro		Longitud del parámetro	
Valor del parámetro			



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

APLICACIONES DE RED: CARACTERÍSTICAS

- 📄 **Características/requisitos de las aplicaciones:**
  - **Tolerancia a pérdidas de datos (errores):** Algunas apps (ej., audio) pueden tolerar algunas pérdida de datos; otras (ej. FTP, telnet, HTTP) requieren transferencia 100% fiable
  - **Exigencia de requisitos temporales:** Algunas apps denominadas *inelásticas* (ej., telefonía Internet, juegos interactivos) requieren retardo (*delay*) acotado para ser efectivas, otras aplicaciones no
  - **Demanda de ancho de banda (tasa de transmisión o throughput)**  
Algunas apps requieren envío de datos a una tasa determinada (p. ejemplo un codec de vídeo), otras no
  - **Nivel de seguridad:** Los requisitos de seguridad para las distintas apps son muy variables (Encriptación, autenticación, no repudio, integridad...)
  - **Conclusión:** las distintas aplicaciones tienen requisitos **HETEROGÉNEOS**





Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### REQUERIMIENTOS DE ALGUNAS APLICACIONES

Application	Data loss	Throughput	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video: 10kbps-5Mbps	yes, 100's ms
stored audio/video	loss-tolerant	same as above	yes, few s
interactive games	loss-tolerant	few kbps up	yes, 100's ms
instant messaging	no loss	elastic	yes and no

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



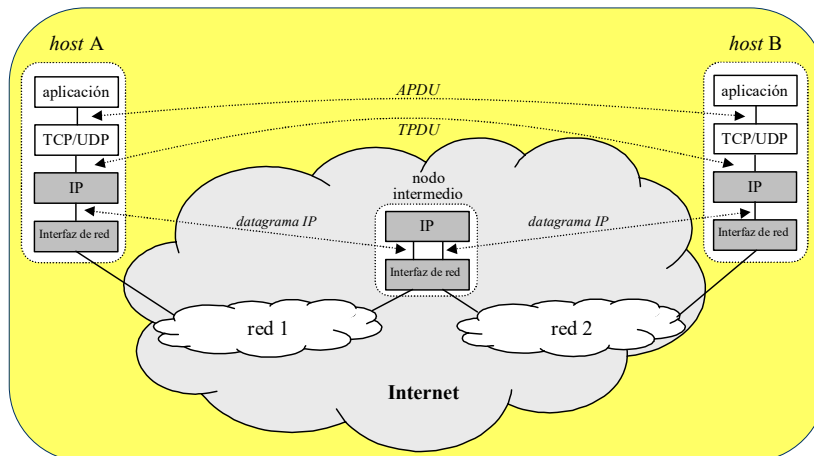
35



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### PROTOCOLOS DE TRANSPORTE



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



36



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### PROTOCOLOS DE TRANSPORTE

#### Servicio TCP:

Orientado a conexión  
Transporte fiable con control de errores  
Control de flujo  
Control de congestión

#### Servicio UDP:

No orientado a conexión  
Transporte no fiable  
Sin control de flujo  
Sin control de congestión,  
¿Para qué existe UDP?

- TCP y UDP (capa de transporte) al ser usuarios del protocolo IP (capa de red) **no garantizan Calidad de Servicio (QoS), es decir:**
  - El **retardo** NO está acotado
  - Las **fluctuaciones en el retardo** NO están acotadas
  - No hay una **velocidad de transmission** mínima garantizada
  - No hay una **probabilidad de pérdidas** acotada
- Tampoco hay garantías de seguridad.



Tema 2. Capa de aplicación

## 1. Introducción a las aplicaciones de red

### PROTOCOLOS DE TRANSPORTE

	Application	Application layer protocol	Underlying transport protocol
	e-mail	SMTP [RFC 2821]	TCP
remote	terminal access	Telnet [RFC 854]	TCP
	Web	HTTP [RFC 2616]	TCP
	file transfer	FTP [RFC 959]	TCP
streaming	multimedia	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP
	Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP





Tema 2. Capa de aplicación

## Esquema

1. Introducción a las aplicaciones de red
2. **Servicio de Nombres de Dominio (DNS)**
3. La navegación web
4. El correo electrónico
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz




39



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

- La comunicación en Internet precisa de direcciones IP
- Los usuarios prefieren usar "nombres de dominio" (más de  $300 \times 10^6$ )
- DNS: traducción de nombres a direcciones IP (resolución de nombres)  
 **goliat.ugr.es** <-----> **150.214.20.3**
- Estructura jerárquica en dominios:  
*Parte\_local.dominio\_niveln. ... .dominio\_nivel2.dominio\_nivel1.*
- Al dominio de nivel 1 se le denomina **dominio genérico** (.com .es .edu etc).
- El dominio raíz o "." está gestionado por el **ICANN** (Internet Corporation for Assigned Names and Numbers;  
<http://www.icann.org>). ICANN delega la gestión de algunos dominios genéricos a centros regionales.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



40



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

### Lectura recomendadas

Tutorial sobre "los nombres de dominios":

- <https://www.icann.org/en/system/files/files/domain-names-beginners-guide-06dec10-es.pdf>

Instrucciones para registrar un nombre de dominio en .es:

- <http://www.dominios.es/dominios/es/todo-lo-que-necesitas-saber/sobre-registros-de-dominios>

Instalación y ejemplos de ficheros configuración de *named*

- <https://www.tldp.org/HOWTO/DNS-HOWTO.html>



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

Inicialmente fueron definidos los siguientes 9 dominios genéricos (RFC 1591):

- .com -> organizaciones comerciales
- .edu -> instituciones educativas, como universidades, de EEUU.
- .gov -> instituciones gubernamentales estadounidenses
- .mil -> grupos militares de estados unidos
- .net -> proveedores de Internet
- .org -> organizaciones diversas diferentes de las anteriores
- .arpa -> propósitos exclusivos de infraestructura de Internet
- .int -> organizaciones establecidas por tratados internacionales entre gobiernos
- .xy -> indicativos de la zona geográfica (ej. es (España); pt (portugal); jp (Japón)...

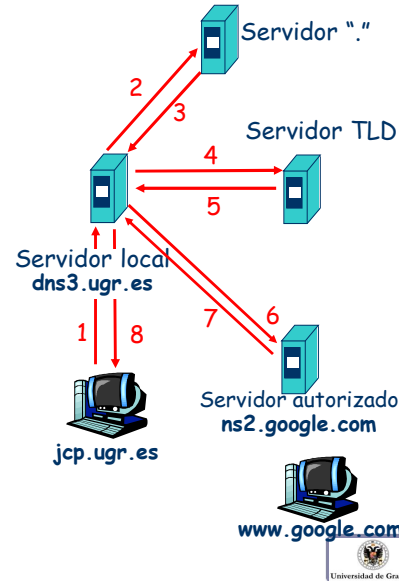




Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

- DNS es un **protocolo** de aplicación para el acceso a una base de datos distribuida con una gestión distribuida.
- 3 niveles de servidores:
  - Servidores raíz "."
  - Servidores de dominio (Top-Level domain o TLD)
  - Servidores Locales
- jcp.ugr.es → [www.google.com](http://www.google.com)
  - Consulta al "resolver" local
  - Conexión con DNS local con IP conocida: ¿cómo se conoce?
  - El DNS local realiza la "resolución" (ver página siguiente)
- Resolución iterativa o recursiva
- Para mejorar prestaciones se usan caches



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

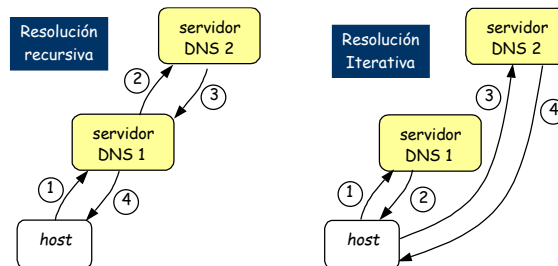
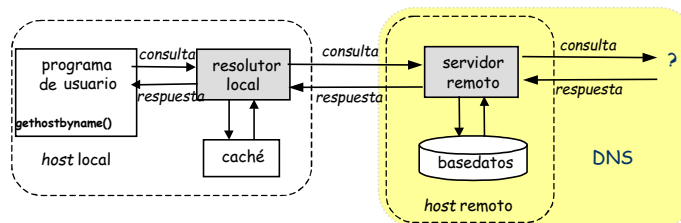


43



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz




44

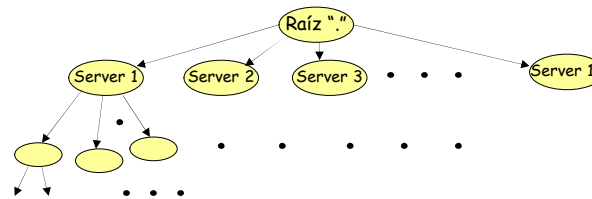




Tema 2. Capa de aplicación


## 2. Servicio de Nombres de Dominio (DNS)

-  **Gestión de la base de datos distribuida y jerárquica:**
- ❑ Está formada por un conjunto de servidores cooperativos que almacenan parcialmente la base de datos que se denomina BIND (Berkeley Internet Name Domain).
  - ❑ Cada servidor es responsable de lo que se denomina **ZONA**.
  - ❑ Una **zona** es un conjunto de nombres de dominio contiguos (por debajo de un nodo en el árbol) de los que un servidor tiene toda la información y es su **autoridad**.
  - ❑ Los **servidores autoridad** (*Start of Authority Servers*) deben contener **toda** (no "cacheada") la información de su zona.
  - ❑ La autoridad puede **delegarse** jerárquicamente a otros servidores



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

-  **Gestión de la base de datos DNS:**
- ❑ Cada zona debe tener **al menos** un servidor de autoridad.
  - ❑ En cada zona hay servidores **primarios** (almacenan una copia *master* de la db en discos locales) y servidores **secundarios** (obtienen la db por transferencia)
  - ❑ Además, existe un servicio de **cache** para mejorar prestaciones.
  - ❑ La **topología real** de servidores es complicada: existen **13 servidores** raíz (A-M) (ver <http://www.root-servers.org>)
  - ❑ El root-server F (y otros) tiene un servidor en Madrid (**Espanix: punto neutro**)
  - ❑ Cuando un cliente (a través de un *resolver local*) solicita una resolución de nombres a su servidor, puede ocurrir:
    - **Respuesta CON autoridad:** el servidor tiene autoridad sobre la zona en la que se encuentra el nombre solicitado y devuelve la dirección IP.
    - **Respuesta SIN autoridad:** el servidor no tiene autoridad sobre la zona en la que se encuentra el nombre solicitado, pero lo tiene en la cache.
    - **No conoce la respuesta:** el servidor preguntará a otros servidores de forma recursiva o iterativa. Normalmente se "eleva" la petición a uno de los servidores raíz.



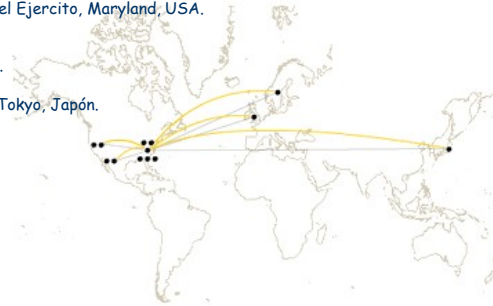


Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

Root-servers <http://www.root-servers.org/>

Servidor A: Network Solutions, Herndon, Virginia, USA.  
Servidor B: Instituto de Ciencias de la Información de la Universidad del Sur de California, USA.  
Servidor C: PSINet, Virginia, USA.  
Servidor D: Universidad de Maryland, USA.  
Servidor E: NASA, en Mountain View, California, USA.  
Servidor F: Internet Software Consortium, Palo Alto, California, USA.  
Servidor G: Agencia de Sistemas de Información de Defensa, California, USA.  
Servidor H: Laboratorio de Investigación del Ejército, Maryland, USA.  
Servidor I: NORDUnet, Estocolmo, Suecia.  
Servidor J: (TBD), Virginia, USA.  
Servidor K: RIPE-NCC, Londres, Inglaterra.  
Servidor L: (TBD), California, USA.  
Servidor M: Wide Project, Universidad de Tokyo, Japón.



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



47



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

¿Cómo es la base de datos DNS?

- ☐ Todo dominio está asociado al menos a un registro **Resource Record**.
- ☐ Cada **RR** es una tupla con 5 campos:

**Nombre del dominio:** nombre del dominio al que se refiere el RR.

**Tiempo de vida:** tiempo de validez de un registro (para la cache).

**Clase:** en Internet siempre IN.

**Tipo:** Tipo de registro.

<b>SOA</b>	Registro ( <b>Start Of Authority</b> ) con la autoridad de la zona.
<b>NS</b>	Registro que contiene un servidor de nombres.
<b>A</b>	Registro que define una dirección IPv4.
<b>MX</b>	Registro que define un servidor de correo electrónico.
<b>CNAME</b>	Registro que define el nombre canónico de un nombre de dominio.
<b>HINFO</b>	Información del tipo de máquina y sistema operativo.
<b>TXT</b>	Información del dominio.

**Valor:** Contenido que depende del campo tipo

- ☐ Existe una base de datos asociada de **resolución inversa** para traducir direcciones IP en nombres de dominio. (in-addr.arpa)

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



48



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

Ejemplo de  
fichero de zona  
para el dominio  
example.com

```
; IPv4 zone file for example.com
$TTL 2d ; default TTL for zone
$ORIGIN example.com. ; base domain-name
; Start of Authority record defining the key characteristics
; of the zone (domain)
@ IN SOA ns1.example.com. hostmaster.example.com. (
    2003080800 ; se = serial number
    12h ; ref = refresh
    15m ; ret = refresh retry
    3w ; ex = expiry
    2h ; nx = nxdomainttl
)
; name servers Resource Records for the domain
IN NS ns1.example.com.
; the second name server is
; external to this zone (domain).
IN NS ns2.example.net.
; mail server Resource Records for the zone (domain)
; value 10 denotes it is the most preferred
3w IN MX 10 mail.example.com.
; the second mail server has lower preference (20) and is
; external to the zone (domain)
IN MX 20 mail.example.net.
; domain hosts includes NS and MX records defined previously
; plus any others required
ns1 IN A 192.168.254.2
mail IN A 192.168.254.4
joe IN A 192.168.254.6
www IN A 192.168.254.7
; aliases ftp (ftp server) to an external location
ftp IN CNAME ftp.example.net.
```



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

49



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

### Formato mensajes DNS:



□ DNS se ofrece en el puerto 53 mediante UDP normalmente o TCP (para respuestas grandes > 512 bytes).

□ Más información:

- RFC 1034 y RFC 1035 (actualizados 3597 y 3658)
- /usr/doc/HOWTO/trans/es/DNS-COMO
- man named, nslookup, resolver, host.conf, dig
- DNSSEC [http://www.dominios.es/dominios/sites/dominios/files/1318333648229\\_0.pdf](http://www.dominios.es/dominios/sites/dominios/files/1318333648229_0.pdf)



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

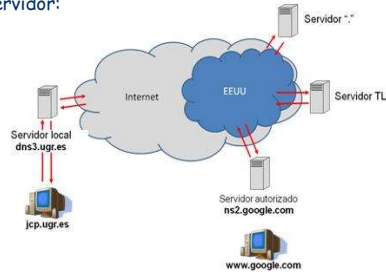
50



Tema 2. Capa de aplicación

## 2. Servicio de Nombres de Dominio (DNS)

5. En la siguiente figura se ilustra un ejemplo de acceso DNS por parte de una máquina (jcp.ugr.es) que quiere acceder a los servicios de [www.google.com](http://www.google.com). Para obtener la dirección IP del servidor, es necesario que la consulta pase por todos los servidores del gráfico. Considerando unos retardos promedio de 8  $\mu$ s dentro de una red LAN, de 12 ms en cada acceso a través de Internet (4 ms si la conexión se restringe a EEUU) y de 1 ms de procesamiento en cada servidor:



Calcule el tiempo que se tardaría si la solicitud al servidor local es recursiva, pero el propio servidor local realiza solicitudes iterativas.

Especifique una política (recursiva-iterativa) más rápida de solicitudes y el tiempo que tardaría la solicitud en ser respondida. ¿Qué desventaja tiene sobre la solución anterior?



Tema 2. Capa de aplicación

## Esquema

1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
3. **La navegación Web**
4. El Correo electrónico
5. Seguridad y protocolos seguros
6. Aplicaciones multimedia
7. Cuestiones y ejercicios





Tema 2. Capa de aplicación

### 3. La navegación web

- Una página Web es un fichero (HTML) formado por objetos:
  - ficheros HTML, imágenes JPEG, Java applets, ficheros de audio, vídeo, etc
- Cada objeto se direcciona por una URL (o URI):

esquema:[//[user[:password]@]dominio[:puerto][/path][/recurso][?solicitud][#fragment]

Name	Used for	Example
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
news	Newsgroup	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Sending e-mail	mailto:JohnUser@acm.org
telnet	Remote login	telnet://www.w3.org:80



Tema 2. Capa de aplicación

### 3. La navegación web

- Las páginas se sirven con el protocolo HTTP: Hyper Text Transfer Protocol
  - Modelo cliente-servidor
  - **cliente**: browser que solicita, recibe y muestra objetos web
  - **servidor**: envía objetos web en respuesta a peticiones
- Las páginas web pueden ser **estáticas** (contenido invariable) o **dinámicas** (con contenido variable).
- Las páginas dinámicas pueden proporcionar contenido variable:
  - Usando lenguajes de **scripting en el cliente**: JavaScript o Flash etc
  - Usando lenguajes de **scripting en el servidor**: Perl, PHP, Ruby, Python etc. Se utilizan incrustando etiquetas dentro de la página web. Cuando el cliente solicita esa página web, el servidor web interpreta estas etiquetas para realizar acciones en el servidor generando contenido dinámico. Por ejemplo, insertando información de una base de datos.





Tema 2. Capa de aplicación

### 3. La navegación web

#### ➤ Características HTTP:

- Usa los servicios de TCP (S.O.C.) en el puerto 80
  - Inicio de conexión TCP, envío HTTP, cierre de conexión TCP
- HTTP es "stateless" → Cookies
  - El servidor no mantiene información sobre las peticiones de los clientes (su estado) y así ahorra recursos aunque hace más compleja la interacción
- Existen dos tipos de servidores
  - No persistente → Se envía únicamente un objeto en cada conexión TCP.
  - Persistente → Pueden enviarse múltiples objetos sobre una única conexión TCP entre cliente y servidor

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

55



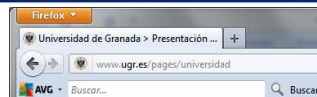
55



Tema 2. Capa de aplicación

### 3. La navegación web

#### MENSAJES HTTP



1. El cliente HTTP (navegador) solicita un objeto identificado por su URL, en el ejemplo [www.ugr.es/pages/Universidad](http://www.ugr.es/pages/Universidad). Según la configuración del servidor, si no se especifica nada, por defecto se sirve el fichero index.html
2. El cliente consulta al resolver de DNS por la dirección IP de [www.ugr.es](http://www.ugr.es)
3. DNS contesta 150.214.204.231
4. El cliente abre una conexión TCP al puerto 80 de 150.214.204.231 (3 bandas)
5. El cliente envía una petición "GET /pages/universidad/ ..." (más otra información adicional: cabeceras, cookies, variables, etc)
6. El servidor responde enviando el fichero "index.html" por la misma conexión TCP
7. Al usar TCP el cliente y servidor de HTTP reciben un servicio orientado a conexión, fiable, sin errores, con control de flujo, con control de congestión, etc. Es decir una comunicación TRANSPARENTE y FIABLE.
8. Si es persistente se siguen solicitando objetos de la página ("GET...") por la conexión
9. Se cierra la conexión TCP y se liberan recursos en el servidor y cliente
10. El cliente visualiza el contenido

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



56





Tema 2. Capa de aplicación

### 3. La navegación web

#### MENSAJES HTTP



- 1a. El Cliente HTTP inicia conexión TCP al servidor HTTP (proceso) en `www.ugr.es` en el puerto 80 (segmento SYNC de TCP)
- 1b. El Servidor HTTP acepta la conexión y solicita al cliente abrir la conexión (SYNC+ACK)
2. El Cliente HTTP envía *request message* para el objeto
- 1c. El cliente confirma (ACK)
3. El servidor HTTP devuelve la respuesta
4. Si es persistente → Envío de más objetos por la misma conexión TCP
5. Cierre de conexión TCP (liberación de recursos)
6. Nuevas conexiones TCP

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

57



57



Tema 2. Capa de aplicación

### 3. La navegación web

#### TIPOS DE MENSAJES HTTP

- HTTP define dos tipos de mensajes (*request*, *response*):
1. *HTTP request message* (solicitudes del cliente al servidor):

Línea de petición  
(GET, POST, HEAD)

Líneas de cabecera

Carriage return + line feed

Indican fin del mensaje

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

(extra carriage return, line feed)

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

58



58





Tema 2. Capa de aplicación

### 3. La navegación web

TIPOS DE MENSAJES HTTP ■

HTTP define dos tipos de mensajes (*request, response*):

2. *HTTP response message*: (respuestas del servidor al cliente):

Línea de estado

Líneas de cabecera

59 Datos,  
ej. fichero html

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 .....
Content-Length: 6821
Content-Type: text/html
data data data data data ...
```

200 OK  
301 Moved Permanently  
400 Bad Request  
404 Not Found  
505 HTTP Version Not  
Supported



59



Tema 2. Capa de aplicación

### 3. La navegación web

8. Compare el rendimiento en términos temporales de HTTP persistente y no persistente considerando los siguientes parámetros:

- Descarga de una página web con 10 objetos incrustados
- Tiempo de Establecimiento de conexión TCP → 5 ms
- Tiempo de Cierre de conexión TCP → 5 ms
- Tiempo de solicitud HTTP → 2 ms
- Tiempo de respuesta HTTP (página web u objeto) → 10 ms

60



60



Tema 2. Capa de aplicación

### 3. La navegación web

PROTOCOLO HTTP 1.1 (RFC 2616) ■

- **MÉTODOS** (acciones solicitadas por los clientes en los *request messages*):
  - **OPTIONS**: solicitud de información sobre las opciones disponibles
  - **GET**: solicitud de un recurso (puede ser condicional)
  - **HEAD**: igual que **GET** pero el servidor no devuelve el "cuerpo" sólo cabeceras
  - **POST**: solicitud al servidor para que acepte y subordine a la URI especificada, los datos incluidos en la solicitud,
  - **PUT**: solicitud de sustituir la URI especificada con los datos incluidos en la solicitud.
  - **DELETE**: solicitud de borrar la URI especificada.
- **CÓDIGOS DE RESPUESTA** (para los *response messages del servidor*):
  - **1xx** indican mensajes exclusivamente informativos
  - **2xx** indican algún tipo de éxito
  - **3xx** redirección al cliente a otra URL
  - **4xx** indican un error
  - **5xx** indican un error
- **CABECERAS** (47 *request headers* y 49 *response headers*)

61

From: , User-Agent:, Content-Type:, Content-Length: ,.....  
[http://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_header\\_fields](http://en.wikipedia.org/wiki/List_of_HTTP_header_fields)



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

61



Tema 2. Capa de aplicación

### 3. La navegación web

PROTOCOLO HTTP 1.1 (RFC 2616) ■

- <https://developer.mozilla.org/es/docs/Web/HTTP/Headers>
- [https://www.tutorialspoint.com/http/http\\_quick\\_guide.htm](https://www.tutorialspoint.com/http/http_quick_guide.htm)

✓ **Cabeceras comunes para peticiones y respuestas**

- **Content-Type**: descripción MIME de la información contenida en este mensaje.
- **Content-Length**: longitud en bytes de los datos enviados, expresado en base decimal.
- **Content-Encoding**: formato de codificación de los datos enviados en este mensaje. Sirve, por ejemplo, para enviar datos comprimidos o encriptados.
- **Date**: fecha local de la operación. Las fechas deben incluir la zona horaria en que reside el sistema que genera la operación. Por ejemplo: Sunday, 12-Dec-96 12:21:22 GMT+01. No existe un formato único en las fechas.

62



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

62



Tema 2. Capa de aplicación

### 3. La navegación web

PROTOCOLO HTTP 1.1 (RFC 2616) ■

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

✓ **Cabeceras sólo para peticiones del cliente**

- **Accept:** campo opcional que contiene una lista de tipos MIME aceptados por el cliente.
- **Authorization:** clave de acceso que envía un cliente para acceder a un recurso de uso protegido o limitado. La información incluye el formato de autorización empleado, seguido de la clave de acceso propiamente dicha. La explicación se incluye más adelante.
- **From:** campo opcional que contiene la dirección de correo electrónico del usuario del cliente Web que realiza el acceso.

63



63



Tema 2. Capa de aplicación

### 3. La navegación web

PROTOCOLO HTTP 1.1 (RFC 2616) ■

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

- **If-Modified-Since:** permite realizar operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la fecha proporcionada. Puede ser utilizada por los sistemas de almacenamiento temporal de páginas. Es equivalente a realizar un HEAD seguido de un GET normal.
- **Referer:** contiene la URL del documento desde donde se ha activado este enlace. De esta forma, un servidor puede informar al creador de ese documento de cambios o actualizaciones en los enlaces que contiene. No todos los clientes lo envían.
- **User-agent:** cadena que identifica el tipo y versión del cliente que realiza la petición. Por ejemplo, los *browsers* de Netscape envían cadenas del tipo User-Agent: Mozilla/3.0 (WinNT; I)

64



64



Tema 2. Capa de aplicación

### 3. La navegación web

PROTOCOLO HTTP 1.1 (RFC 2616)

✓ **Cabeceras sólo para respuestas del servidor HTTP**

- **Allow:** informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere esta respuesta. Por ejemplo, Allow: GET, POST.
- **Expires:** fecha de expiración del objeto enviado. Los sistemas de cache deben descartar las posibles copias del objeto pasada esta fecha. Por ejemplo, Expires: Thu, 12 Jan 97 00:00:00 GMT+1. No todos los sistemas lo envían.
- **Last-modified:** fecha local de modificación del objeto devuelto. Se puede corresponder con la fecha de modificación de un fichero en disco, o, para información generada dinámicamente desde una base de datos, con la fecha de modificación del registro de datos correspondiente.

65



65



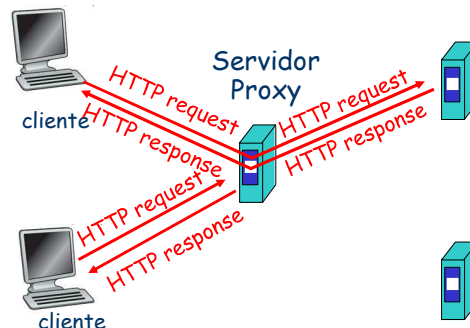
Tema 2. Capa de aplicación

### 3. La navegación web

WEB CACHE

**Objetivo:** satisfacer el requerimiento del cliente sin involucrar al servidor destino.

- Usuario configura el browser: Acceso Web vía cache
- browser envía todos los requerimientos HTTP al cache
  - Si objeto está en cache: cache retorna objeto
  - Sino cache requiere los objetos desde el servidor Web, y retorna el objeto al cliente



66



66



Tema 2. Capa de aplicación

### 3. La navegación web

WEB CACHE ■

- **Ejemplo de respuesta** típica de un servidor configurado para gestionar cachés:

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

- Las cabeceras se asocian al fichero en la caché local

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

67



67

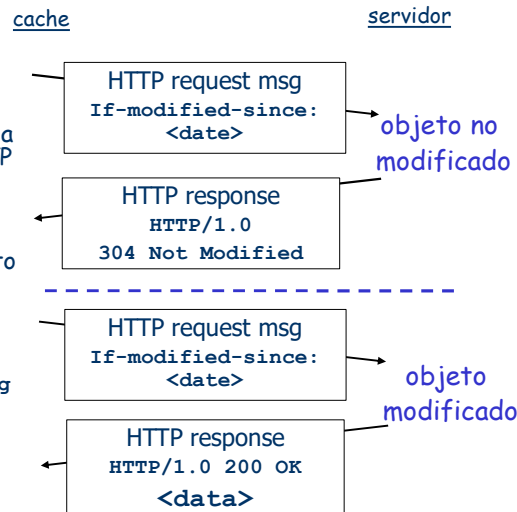


Tema 2. Capa de aplicación

### 3. La navegación web

WEB CACHE ■

- **Objetivo:** no enviar objetos si el cache tiene la versión actualizada
- **Cache:** especifica la fecha de la copia en el requerimiento HTTP  
If-modified-since: <date>  
If-None-Match: "686897696a7c876b7e"
- **servidor:** responde sin el objeto si la copia de la cache es la última. :  
HTTP/1.0 304 Not Modified  
Ver <https://www.keycdn.com/blog/http-cache-headers>



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

68



68



Tema 2. Capa de aplicación

### 3. La navegación web

COOKIES ■

- ✓ Las **cookies** son pequeños ficheros de texto que se intercambian los clientes y servidores HTTP, para solucionar una de las principales deficiencias del protocolo: la falta de información de estado entre dos transacciones. Fueron introducidas por Netscape, y han sido estandarizadas en el RFC 2109.
  - La primera vez que un usuario accede a un determinado documento de un servidor, éste proporciona una *cookie* que contiene datos que relacionarán posteriores operaciones.
  - El cliente almacena la *cookie* en su sistema para usarla después. En los futuros accesos a este servidor, el *navegador* podrá proporcionar la *cookie* original, que servirá de nexo entre este acceso y los anteriores.
  - Todo este proceso se realiza automáticamente, sin intervención del usuario.

69



69



Tema 2. Capa de aplicación

### 3. La navegación web

COOKIES ■

- ✓ Su aplicación más inmediata son los sistemas de **compra electrónica**. Estos supermercados virtuales necesitan relacionar el contenido de un pedido con el cliente que lo ha solicitado.
- ✓ Otro uso muy interesante son los **sistemas personalizados de recepción de información**, en los que es posible construir una página a medida, con información procedente de fuentes muy diversas. En accesos sucesivos, el cliente enviará la *cookie*, y el servidor podrá generar una página personalizada con las preferencias del usuario.
- ✓ Por último, algunas compañías emplean las *cookies* para realizar un seguimiento de los accesos a sus servidores WWW, identificando las páginas más visitadas, la manera en que se pasa de una a otra sección, etc.

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>

70



70



Tema 2. Capa de aplicación

### 3. La navegación web

#### COOKIES

- ✓ Un servidor HTTP envía los diferentes campos de una *cookie* con la nueva cabecera HTTP Set-Cookie:

*Set-Cookie: Domain=www.unican.es; Path=/; Nombre=Luis; Expires Fri, 15-Jul-97 12:00:00 GMT*

- Cuando se accede a una URL que verifica el par dominio/path registrado, el cliente enviará automáticamente la información de los diferentes campos de la cookie con la nueva cabecera HTTP Cookie:



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

71



71



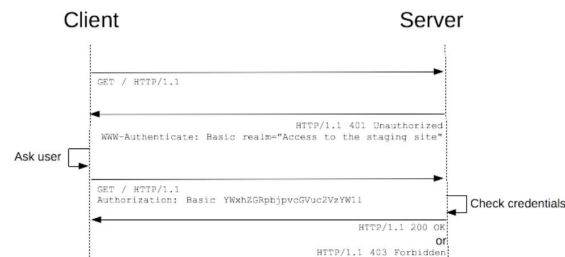
Tema 2. Capa de aplicación

### 3. La navegación web

#### ACCESO RESTRINGIDO

- HTTP no es seguro, pero incluye cabeceras (WWW-Authenticate y Authorization) para restringir el acceso a recursos.

- Es vulnerable a ataques por repetición.



- WWW-Authenticate: <type> realm=<realm>[, charset="UTF-8"]
- Authorization: <type> <credentials>  
<credentials> si <type> es BASIC incluye el username:password codificado en BASE64

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



72





## Tema 2. Capa de aplicación

## Esquema

1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
3. La navegación web
4. **El correo electrónico**
5. Aplicaciones multimedia
6. Cuestiones y ejercicios

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

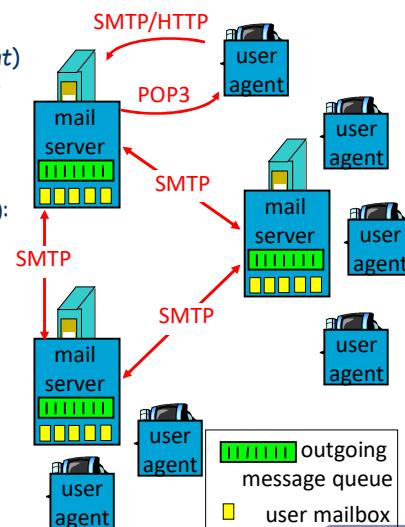
73



## Tema 2. Capa de aplicación

#### 4. El correo electrónico

- Elementos y protocolos principales:
  - Cliente de correo (*Mail User Agent*)
  - Servidor de correo (*Mail Server* ó *Mail Transfer Agent*)
  - Protocolo de envío:
    - Simple Mail Transfer Protocol (SMTP)
  - Protocolos de descarga (o lectura):
    - POP3, IMAP, HTTP
- Agente de usuario (*MUA*):
  - Compone, edita y lee mensajes de correo del buzón. Ej. Outlook, Thunderbird
- Servidor de correo (*MTA*)
  - Reenvía mensajes salientes y almacena en buzones los mensajes entrantes de cada usuario.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

74



Tema 2. Capa de aplicación

## 4. El correo electrónico

SMTP (RFC 2821)

- SMTP se implementa mediante dos programas (incluidos ambos en cada mail server):
  - Cliente SMTP: se ejecuta en el mail server (MTA) que está enviando correo
  - Servidor SMTP: se ejecuta en el mail server (MTA) que está recibiendo correo
  - "sendmail" <http://en.wikipedia.org/wiki/Sendmail>
- SMTP usa TCP en el puerto 25. Es un protocolo orientado a texto.
- SMTP es un protocolo orientado a conexión, es in-band y es state-full: implica tres fases
  - Handshaking ("saludo")
  - Transferencia de mensajes
  - Cierre
- La interacción entre cliente SMTP y servidor SMTP se realiza mediante comandos / respuesta
  - comandos: texto ASCII
  - respuestas: código de estado y frases explicativas
- Los mensajes deben estar codificados en ASCII de 7 bits!! → Con la definición posterior de las extensiones MIME se pueden enviar ASCII de 8 bits y formatos enriquecidos

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



75



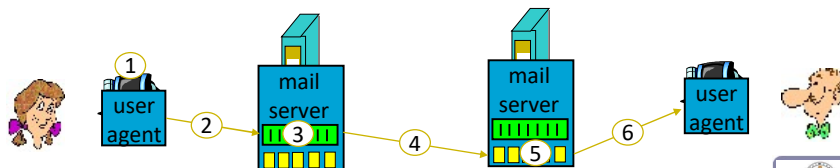
Tema 2. Capa de aplicación

## 4. El correo electrónico

SMTP (RFC 2821)

Pasos en el envío/recepción de correo

- 1) El usuario origen compone mediante su Agente de Usuario (MUA) un mensaje dirigido a la dirección de correo del usuario destino
- 2) Se envía con SMTP (ó HTTP) el mensaje al servidor de correo (MTA) del usuario origen que lo sitúa en la cola de mensajes salientes
- 3) El cliente SMTP abre una conexión TCP con el servidor de correo (MTA) (obtenido por DNS) del usuario destino
- 4) El cliente SMTP envía el mensaje sobre la conexión TCP
- 5) El servidor de correo del usuario destino ubica el mensaje en el mailbox del usuario destino
- 6) El usuario destino invoca su Agente de Usuario (MUA) para leer el mensaje utilizando POP3, IMAP ó HTTP



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



76



Tema 2. Capa de aplicación

## 4. El correo electrónico

SMTP (RFC 2821)

```
S: 220 smtp1.ugr.es
C: HELO ugr.es
S: 250 smtp1.ugr.es
C: MAIL FROM: uno@ugr.es
S: 250 Ok
C: RCPT TO: dos@ugr.es
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: Subject: Correo estúpido
C: Tengo ganas de enviarte un correo...
C: ¿Te importa si lo hago?
C: .
S: 250 Ok: queued as KJSADHFFWDF
C: QUIT
S: 221 Bye
```

*Propuesta de ejercicio:  
dibujar el diagrama de  
estados de SMTP*

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



77



Tema 2. Capa de aplicación

## 4. El correo electrónico

SMTP (RFC 2821)

### Comandos SMTP: cliente

Comando	Descripción
HELO (ahora EHLO)	Identifica el remitente al destinatario.
MAIL FROM	Identifica una transacción de correo e identifica al emisor.
RCPT TO	Se utiliza para <b>identificar un destinatario individual</b> . Si se necesita identificar múltiples destinatarios es necesario repetir el comando.
DATA	Permite enviar una serie de líneas de texto. El tamaño máximo de una línea es de 1.000 caracteres. Cada línea va seguida de un retorno de carro y avance de línea <CR><LF>. <b>La última línea debe llevar únicamente el carácter punto "."</b> seguido de <CR><LF>.
RSET	Aborta la transacción de correo actual.
NOOP	No operación. <b>Indica al extremo que envíe una respuesta positiva. Keepalives</b>
QUIT	Pide al otro extremo que envíe una respuesta positiva y cierre la conexión.
VRFY	Pide al receptor que confirme que un nombre identifica a un destinatario válido.
EXPN	Pide al receptor la <b>confirmación de una lista de correo</b> y que devuelva los nombres de los usuarios de dicha lista.
HELP	Pide al otro extremo información sobre los comandos disponibles.
TURN	El emisor pide que se <b>inviertan los papeles</b> , para poder actuar como receptor. El receptor puede negarse a dicha petición.
SOML	Si el destinatario está conectado, entrega el mensaje directamente al terminal, en caso contrario lo entrega como correo convencional.
SAML	Entrega del mensaje en el buzón del destinatario. En caso de estar conectado también lo hace al terminal.
SEND	Si el destinatario está conectado, entrega el mensaje directamente al terminal.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



78



Tema 2. Capa de aplicación

## 4. El correo electrónico

SMTP (RFC 2821)

### ➤ Códigos de respuesta SMTP: servidor

Código	Descripción
211	Estado del sistema.
214	Mensaje de ayuda.
220	Servicio preparado.
221	Servicio cerrando el canal de transmisión.
250	Solicitud completada con éxito.
251	Usuario no local, se enviará a <dirección de reenvío>
354	Introduzca el texto, finalice con <CR><LF>.<CR><LF>.
421	Servicio no disponible.
450	Solicitud de correo no ejecutada, servicio no disponible (buzón ocupado).
451	Acción no ejecutada, error local de procesamiento.
452	Acción no ejecutada, insuficiente espacio de almacenamiento en el sistema.
500	Error de sintaxis, comando no reconocido.
501	Error de sintaxis. P.ej contestación de SMTP a ESMTP
502	Comando no implementado.
503	Secuencia de comandos errónea.
504	Parámetro no implementado.
550	Solicitud no ejecutada, buzón no disponible.
551	Usuario no local, pruebe <dirección de reenvío>. Si no se tiene cuenta
552	Acción de correo solicitada abortada.
553	Solicitud no realizada (error de sintaxis).
554	Fallo en la transacción.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



79



Tema 2. Capa de aplicación

## 4. El correo electrónico

EXTENSIONES MIME

### Multipurpose Internet Mail Protocol Extensions (MIME):

- Nada cambia respecto a la arquitectura de correo anterior.
- Las extensiones de MIME van encaminadas a soportar:
  - Texto en conjuntos de caracteres distintos de US-ASCII;
  - Adjuntos que no son de tipo texto;
  - Cuerpos de mensajes con múltiples partes (multi-part);
  - Información de encabezados con conjuntos de caracteres distintos de ASCII.
- MIME está especificado en seis RFCs: RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 y RFC 2077.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



80

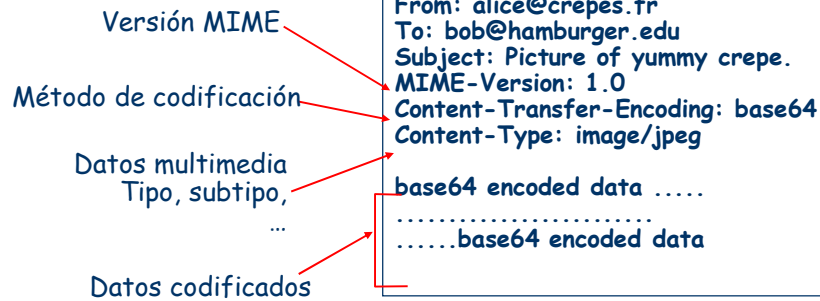


Tema 2. Capa de aplicación

## 4. El correo electrónico

### EXTENSIONES MIME

➤ No se debe confundir los mensajes del protocolo con el formato de almacenamiento.



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



81



Tema 2. Capa de aplicación

## 4. El correo electrónico

### EXTENSIONES MIME

#### ➤ Cabeceras de mensajes MIME

Cabecera	Descripción
MIME-Version:	Identifica la version de MIME. Si no existe se considera que el mensaje es texto normal en inglés.
Content-Description:	Cadena de texto que describe el contenido. Esta cadena es necesaria para que el destinatario sepa si desea decodificar y leer el mensaje o no.
Content-Id:	Identificador único, usa el mismo formato que la cabecera estándar Message-Id.
Content-Transfer-Encoding:	Indica la manera en que está envuelto el cuerpo del mensaje.
Content-Type:	Especifica la naturaleza del cuerpo del mensaje.

#### ➤ Content-Transfer-Encoding

- Indica la manera en que está envuelto el cuerpo para su transmisión, ya que podría haber problemas con la mayoría de los caracteres distintos de letras, números y signos de puntuación.
- Existen 5 tipos de codificación (RFC1521): *ASCII 7*, *ASCII 8*, *codificación binaria*, *base64* y *entrecodificada-imprimible.7.2*

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



82



Tema 2. Capa de aplicación

## 4. El correo electrónico

### EXTENSIONES MIME

#### MIME: Content-Type: tipos y subtipos

- La lista inicial de tipos y subtipos especificada por el RFC 1521 es:

Tipo	Subtipo	Descripción
Text	Plain	Texto sin formato.
	Richtext	Texto con comandos de formato sencillos.
Image	Gif	Imagen fija en formato GIF.
	Jpeg	Imagen fija en formato JPEG.
Audio	Basic	Sonido.
Video	Mpeg	Película en formato MPEG.
Application	Octet-stream	Secuencia de bytes no interpretada.
	Postscript	Documento imprimible PostScript.
Message	Rfc822	Mensaje MIME RFC 822.
	Partial	Mensaje dividido para su transmisión.
	External-body	El mensaje mismo debe obtenerse de la red.
Multipart	Mixed	Partes independientes en el orden especificado.
	Alternative	Mismo mensaje en diferentes formatos.
	Parallel	Las partes deben verse simultáneamente.
	Digest	Cada parte es un mensaje RFC 822 completo.



Tema 2. Capa de aplicación

## 4. El correo electrónico

### EXTENSIONES MIME

#### MIME: Content-Type: tipo application:

- El tipo **application** es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos.
- El subtipo **octet-stream** simplemente es una secuencia de bytes no interpretados, tal que a su recepción, un agente de usuario debería *presentarla en pantalla sugiriendo al usuario que se copie en un archivo y solicitando un nombre de archivo*.
- El subtipo **postscript**, se refiere al lenguaje PostScript de Adobe Systems. Aunque un agente de usuario puede llamar a un intérprete PostScript externo para visualizarlo, hacerlo no está exento de riesgos al ser PostScript un lenguaje de programación completo.





Tema 2. Capa de aplicación

## 4. El correo electrónico

### EXTENSIONES MIME



#### MIME: Content-Type: tipo *message*:

- El tipo **message** permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar correo electrónico.
- El subtipo **rfc822** se utiliza cuando se encapsula un mensaje RFC 822 completo en un mensaje exterior.
- El subtipo **partial** hace posible dividir un mensaje encapsulado en pedazos y enviarlos por separado. Los parámetros hacen posible ensamblar correctamente todas las partes en el destino. E.g. 1/3, 2/3, 3/3.
- El subtipo **external-body** puede usarse para mensajes muy grandes, por ejemplo, películas de vídeo. En lugar de incluir el archivo MPEG en el mensaje, se da una dirección de FTP y el agente del receptor puede obtenerlo a través de la red cuando se requiera.



Tema 2. Capa de aplicación

## 4. El correo electrónico

### EXTENSIONES MIME



#### MIME: Content-Type: tipo *multipart*

- El tipo es **multipart**, que permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte claramente delimitados.
- El subtipo **mixed** permite que cada parte sea diferente.
- El subtipo **alternative** indica que cada parte contiene el mismo mensaje, pero expresado en un medio o codificación diferente.
- El subtipo **parallel** se usa cuando todas las partes deben "verse" simultáneamente, por ejemplo, en los canales de audio y vídeo de las películas.
- El subtipo **digest** se usa cuando se juntan muchos mensajes en un mensaje compuesto.







Tema 2. Capa de aplicación

## 4. El correo electrónico

### EJEMPLO MULTIPART/MIXED ■

```
From: Nathaniel Borenstein <nbsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"

This is the preamble. It is to be ignored, though it
is a handy place for mail composers to include an
explanatory note to non-MIME compliant readers.
--simple boundary

This is implicitly typed plain ASCII text.
It does NOT end with a linebreak.
--simple boundary
Content-type: text/plain; charset=us-ascii

This is explicitly typed plain ASCII text.
It DOES end with a linebreak.

--simple boundary--
This is the epilogue. It is also to be ignored.
```

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



87



Tema 2. Capa de aplicación

## 4. El correo electrónico

### PROTOCOLOS DE ACCESO: POP3 ■

Ej: POP3 PROTOCOL TCP PORT = 110

#### Fase de autorización

Comandos del cliente:

user: nombre de usuario

pass: contraseña

Respuestas del servidor

+OK

-ERR

#### Fase de transacción, cliente:

list: lista mensajes por número

retr: obtiene mensajes por num.

dele: borra

quit

#### Fase de actualización del servidor (tras desconexión)

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



88



Tema 2. Capa de aplicación

## 4. El correo electrónico

### PROTOCOLOS DE ACCESO: POP3

#### Comandos POP3

Comando	Descripción
USER identification	Este comando permite la autenticación. Debe estar seguido del nombre de usuario, es decir, una cadena de caracteres que identifique al usuario en el servidor. El comando USER debe preceder al comando PASS.
PASS password	El comando PASS permite especificar la contraseña del usuario cuyo nombre ha sido especificado por un comando USER previo.
STAT	Información acerca de los mensajes del servidor
RETR	Número del mensaje que se va a recoger
DELE	Número del mensaje que se va a eliminar
LIST [msg]	Número del mensaje que se va a mostrar
NOOP	Permite mantener la conexión abierta en caso de inactividad
TOP <messageID> <n>	Comando que muestra <i>n</i> líneas del mensaje, cuyo número se da en el argumento. En el caso de una respuesta positiva del servidor, éste enviará de vuelta los encabezados del mensaje, después una línea en blanco y finalmente las primeras <i>n</i> líneas del mensaje.
UIDL [msg]	Solicitud al servidor para que envíe una línea que contenga información sobre el mensaje que eventualmente se dará en el argumento. Esta línea contiene una cadena de caracteres denominada <i>unique identifier listing</i> (lista de identificadores únicos) que permite identificar de manera única el mensaje en el servidor, independientemente de la sesión. El argumento opcional es un número relacionado con un mensaje existente en el servidor POP, es decir, un mensaje que no se ha borrado.
QUIT	El comando QUIT solicita la salida del servidor POP3. Lleva a la eliminación de todos los mensajes marcados como eliminados y envía el estado de esta acción.

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



89



Tema 2. Capa de aplicación

## 4. El correo electrónico

### PROTOCOLOS DE ACCESO: IMAP

#### IMAP4:

- Permite trabajar con el correo como si fuese local.

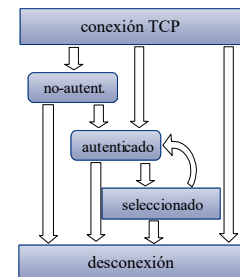
#### Estados:

- No autenticado (NA)
- Autenticado (A)
- Seleccionado (S)
- Desconexión (D)

#### Comandos (puerto 143):

- CAPABILITY, NOOP, LOGOUT
- NA → LOGIN, AUTHENTICATE
- A → SELECT, CREATE, DELETE, LIST, APPEND, UN/SUBSCRIBE, ...
- S → CHECK, CLOSE, SEARCH, FETCH, STORE, COPY, ...

- Más información en: <http://www.imap.org>



Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



90



Tema 2. Capa de aplicación

## 4. El correo electrónico

### PROTOCOLOS DE ACCESO: IMAP

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

```
#> telnet sal.ugr.es 143
* OK sal.ugr.es IMAP4rev1 v12.264 server ready
a001 LOGIN usuario clave
a001 OK LOGIN completed
a002 SELECT inbox
* 18 EXISTS
* 2 RECENT
* OK [UIDVALIDITY 3857529045] UID validity status
* OK [UIDNEXT 17] Predicted next UID
* FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
* OK [PERMANENTFLAGS (\* \Answered \Flagged \Deleted \Draft \Seen)] Permanent flags
a002 OK [READ-WRITE] SELECT completed
a003 FETCH 12 full
* 12 FETCH (FLAGS (\Seen) INTERNALDATE "14-Jul-1993 02:44:25 -0700"
RFC822.SIZE 4282 ENVELOPE ("Wed, 14 Jul 1993 02:23:25 -0700 (PDT)"
"IMAP4 WG mtg summary and minutes"
(("Terry Gray" NIL "gray" "cac.washington.edu"))
((NIL NIL "imap" "cac.washington.edu"))
((NIL NIL "minutes" "CNRI.Reston.VA.US")
("John Klensin" NIL "KLENSIN" "INFOODS.MIT.EDU")) NIL NIL
"<B27397-01000000@cac.washington.edu>")
BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028 92))
a003 OK FETCH completed
a004 FETCH 12 RFC822.HEADER
* 12 FETCH (RFC822.HEADER {346}
Date: Wed, 14 Jul 1993 02:23:25 -0700 (PDT)
From: Terry Gray <gray@cac.washington.edu>
Subject: IMAP4 WG mtg summary and minutes
To: imap@cac.washington.edu
cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@INFOODS.MIT.EDU>
Message-Id: <B27397-01000000@cac.washington.edu>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
)
a004 OK FETCH completed
a005 STORE 12 +flags \deleted
* 12 FETCH (FLAGS (\Seen \Deleted))
a005 OK STORE completed
a006 LOGOUT
* BYE sal.ugr.es IMAP4rev1 server terminating connection
a006 OK LOGOUT completed
#> _
```



91



Tema 2. Capa de aplicación

## 4. El correo electrónico

### PROTOCOLOS DE ACCESO: IMAP

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz

#### telnet capone.rutgers.edu 143

Trying 192.168.5.240...

Connected to 192.168.5.240.

Escape character is '^['.

```
* OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS AUTH=LOGIN] capone.rutgers.edu
IMAP4rev1 2003.339 at Wed, 13 Apr 2005 01:38:58 -0400 (EDT)
```

**Client A1** LOGIN mailtest Password

```
Server A1 OK [CAPABILITY IMAP4REV1 IDLE NAMESPACE MAILBOX-REFERRALS BINARY
UNSELECT SCAN SORT THREAD=REFERENCES THREAD=ORDEREDSUBJECT MULTIAPPEND]
User mailtest authenticated
```

**Client A2** SELECT Inbox

```
* 2 EXISTS
* 2 RECENT
* OK [UIDVALIDITY 1113370837] UID validity status
* OK [UIDNEXT 3] Predicted next UID
* FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
* OK [PERMANENTFLAGS (\* \Answered \Flagged \Deleted \Draft \Seen)]
Permanent flags
* OK [UNSEEN 1] first unseen message in /var/mail/mailtest
```

**Server A2** OK [READ-WRITE] SELECT completed

**Client A3** FETCH 2 BODY[HEADER]

```
* 2 FETCH (BODY[HEADER] {670}
Return-Path:
X-Original-To: mailtest@capone.rutgers.edu
Delivered-To: mailtest@capone.rutgers.edu
```



92



Tema 2. Capa de aplicación

## 4. El correo electrónico

### PROTOCOLOS DE ACCESO: IMAP

```
Received: from node18.rutgers.edu (node18 [192.168.5.38])
  by capone.rutgers.edu (Postfix) with ESMTTP id A291B2B15C
  for ; Tue, 12 Apr 2005 22:23:53 -0400 (EDT)
Received: from me?here.com (unknown [192.168.5.250])
  by node18.rutgers.edu (Postfix) with SMTP id 4653B14112
  for ; Tue, 12 Apr 2005 22:24:03 -0400 (EDT)
To: some_guru@somewhere.com
From: pp@pp.com
Subject: Forged e-mail
Message-Id: <20050413022403.4653B14112@node18.rutgers.edu>
Date: Tue, 12 Apr 2005 22:24:03 -0400 (EDT)

)
* 2 FETCH (FLAGS (\Recent \Seen))
Server A3 OK FETCH completed
Client A4 FETCH 2 BODY[TEXT]
* 2 FETCH (BODY[TEXT] {88})
Hey,
The "To:" and "From:" are non-existent, but you still get the e-mail.
bye, bye
)
Server A4 OK FETCH completed
Client A5 LOGOUT
* BYE capone.rutgers.edu IMAP4rev1 server terminating connection
Server A5 OK LOGOUT completed
Connection closed by foreign host.
```

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



93



Tema 2. Capa de aplicación

## 4. El correo electrónico

### PROTOCOLOS DE ACCESO

#### ➤Ventajas de IMAP4:

- Permite organización en carpetas en el lado del servidor (MTA)
- Para ello, mantiene información entre sesiones (asociando *flags* a los mensajes).
- Permite la descarga de partes de los mensajes.
- Posible acceder con varios clientes (POP también, pero en modo descargar y guardar)

#### ➤Ventajas de Web MAIL:

- Organización total en el servidor, accesible desde cualquier cliente con HTTP.
- Seguridad: Uso extendido de HTTPS

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



94



## 4. El correo electrónico

### ➤ Listado de puertos relacionados con e-mail:

POP3 - port 110  
IMAP - port 143  
SMTP - port 25  
HTTP - port 80  
Secure SMTP (SSMTP) - port 465  
Secure IMAP (IMAP4-SSL) - port 585  
IMAP4 over SSL (IMAPS) - port 993  
Secure POP3 (SSL-POP) - port 995



## Esquema

1. Introducción a las aplicaciones de red
2. Servicio de Nombres de Dominio (DNS)
3. La navegación web
4. El correo electrónico
5. **Aplicaciones multimedia**
6. Cuestiones y ejercicios



Tema 2. Capa de aplicación

## 5. Aplicaciones multimedia

- Conceptos: IP = "tecnología de convergencia" →



- IP ofrece Mejor esfuerzo (*best effort*): sin garantías de QoS

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



97



Tema 2. Capa de aplicación

## 5. Aplicaciones multimedia

- Tipos de aplicaciones

- Flujo de audio y vídeo (*streaming*) almacenado → Ej. YouTube
- Flujo de audio y vídeo en vivo → Ej. emisoras de radio o IPTV
- Audio y vídeo interactivo → Ej. Skype

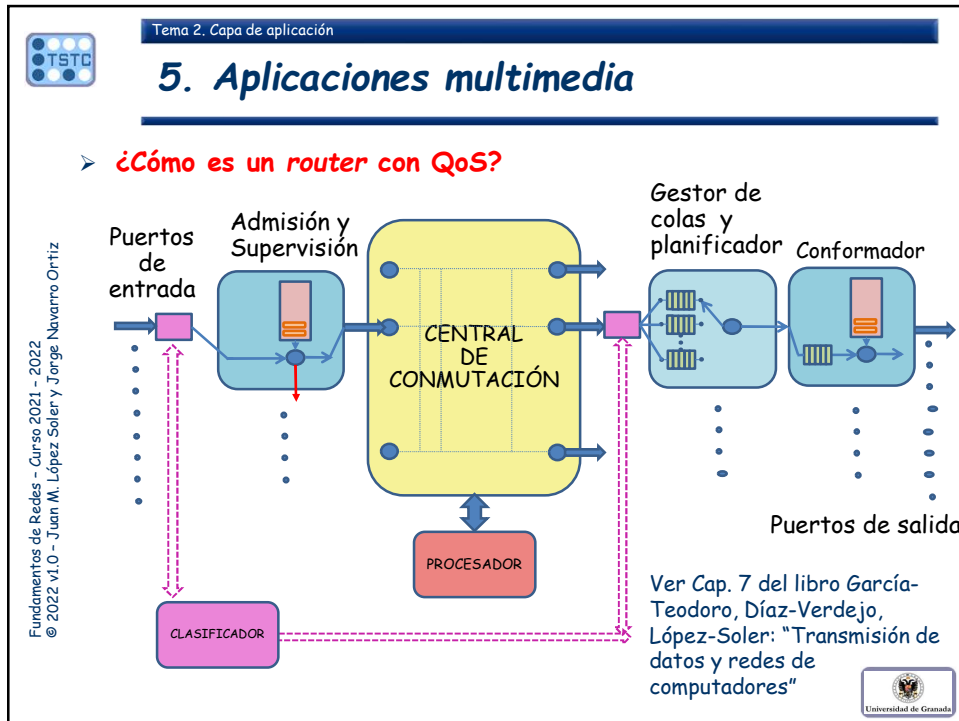
- Características fundamentales

- Elevado ancho de banda
- Tolerantes relativamente a la pérdida de datos
- Exigen Delay (retardo) acotado
- Exigen Jitter (fluctuación del retardo) acotado
- Se pueden beneficiar de usar de *multicast* (direcciones destino de grupo)

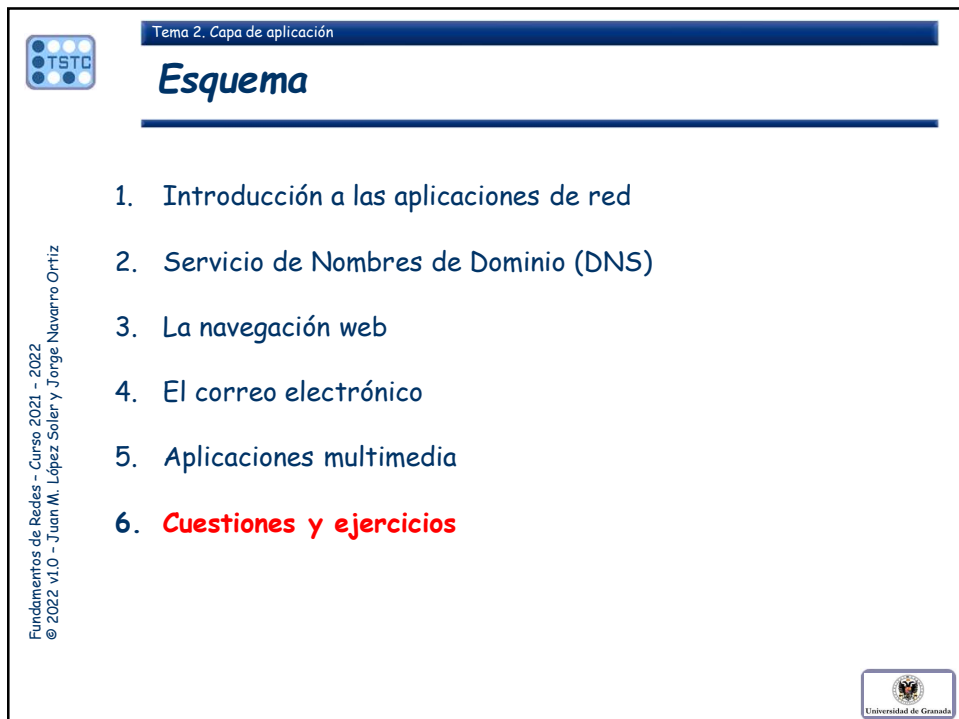
Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



98



99



100





Tema 2. Capa de aplicación

## 7. Cuestiones y ejercicios

2. Discuta las características de las siguientes aplicaciones en términos de su tolerancia a la pérdida de datos, los requisitos temporales, la necesidad de rendimiento mínimo y la seguridad.

La telefonía móvil  
WhatsApp  
YouTube  
Spotify  
Comercio electrónico

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



101



Tema 2. Capa de aplicación

## 7. Cuestiones y ejercicios

8. Una sucursal con 50 empleados en Granada tiene una red interna basada en FastEthernet (100Mbps) que se conecta a Internet con una red de acceso ADSL de 0,5 Mbps de subida y 1,5 Mbps de bajada. Cada empleado, en el desempeño de su trabajo, realiza un promedio de 2000 solicitudes de información a la hora a un servidor de Base de Datos ubicado en la central del banco, en Madrid, donde cada solicitud supone el envío por parte del servidor de 10 registros de 1KB cada uno. Adicionalmente, la modificación de datos tras algunas de estas solicitudes supone el envío de 100 actualizaciones, de 10 registros, a la hora desde la sucursal al servidor. El resto de los servicios telemáticos se restringe.

- Calcule la velocidad de transmisión requerida. ¿Es la velocidad del enlace de acceso suficiente?
- ¿y si se dobla la velocidad del enlace? ¿cuál sería el tiempo de cola que esperaría en promedio cada solicitud en el enlace descendente antes de ser enviada? Considere que cada registro se envía por separado, con una cabecera de tamaño despreciable
- Si, alternativamente, se diseña una caché que permite evitar un 70% de los accesos a la BD ¿cuál sería el tiempo de cola que esperaría en promedio cada solicitud en el enlace descendente? ¿qué solución es mejor, la b. o esta?

Fundamentos de Redes - Curso 2021 - 2022  
© 2022 v1.0 - Juan M. López Soler y Jorge Navarro Ortiz



102



Tema 2. Capa de aplicación

## 7. Cuestiones y ejercicios

1. Explicar por qué cuando solicitamos <http://www.google.com> desde nuestro navegador, se muestra la URL servida desde ([www.google.es](http://www.google.es))

¿qué relación tienen esos 2 nombres de dominio?

¿guarda google información sobre nuestra localización? ¿cómo se obtiene?

¿qué herramientas e información se necesita?

¿qué ocurre y cómo influye si configuro en mi navegador como lenguaje preferido "francés"?

¿pueden servirse páginas dependiendo de nuestra localización? ¿en su caso, con qué precisión?

Sugerencia: Usar el analizador <http://www.wireshark.org> para mostrar trazas

Fundamentos de Redes – Curso 2021 - 2022  
© 2022 v1.0 – Juan M. López Soler y Jorge Navarro Ortiz

