



## UF1288: Desarrollo de componentes software para servicios de comunicaciones

**Certificado de Profesionalidad**  
IFCT0609 - Programación de sistemas informáticos



IFCT0609 > MF0964\_3 > UF1288

**ic editorial**

Federico Huércano Ruiz  
José Villar Cueli

**Desarrollo de  
componentes *software*  
para servicios de  
comunicaciones  
IFCT0609**

Federico Huércano Ruiz

José Villar Cueli

**ic editorial**

**Desarrollo de componentes software para servicios de comunicaciones.**

**IFCT0609**

© Federico Huércano Ruiz

© José Villar Cueli

1<sup>a</sup> Edición

© IC Editorial, 2024

Editado por: IC Editorial

c/ Cueva de Viera, 2, Local 3

Centro Negocios CADI

29200 Antequera (Málaga)

Teléfono: 952 70 60 04

Fax: 952 84 55 03

Correo electrónico: [iceditorial@iceditorial.com](mailto:iceditorial@iceditorial.com)

Internet: [www.iceditorial.com](http://www.iceditorial.com)

**IC Editorial** ha puesto el máximo esfuerzo en ofrecer una información completa y precisa. Sin embargo, no asume ninguna responsabilidad derivada de su uso, ni tampoco la violación de patentes ni otros derechos de terceras partes que pudieran ocurrir. Mediante esta publicación se pretende proporcionar unos conocimientos precisos y acreditados sobre el tema tratado. Su venta no supone para **IC Editorial** ninguna forma de asistencia legal, administrativa ni de ningún otro tipo.

Reservados todos los derechos de publicación en cualquier idioma.

Según el Código Penal vigente ninguna parte de este o cualquier otro libro puede ser reproducida, grabada en alguno de los sistemas de almacenamiento existentes o transmitida por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro, sin autorización previa y por escrito de IC EDITORIAL; su contenido está protegido por la Ley vigente que establece penas de prisión y/o multas a quienes intencionadamente reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica.

ISBN: 978-84-1184-305-8

# Índice

**Portada**

**Título**

**Copyright**

**Presentación del manual**

**Índice**

Capítulo 1

**Programación concurrente**

- 1. Introducción**
  - 2. Programación de procesos e hilos de ejecución**
  - 3. Programación de eventos asíncronos**
  - 4. Mecanismos de comunicación entre procesos**
  - 5. Sincronización**
  - 6. Acceso a dispositivos**
  - 7. Resumen**
- Ejercicios de repaso y autoevaluación**

Capítulo 2

**Fundamentos de comunicaciones**

- 1. Introducción**
  - 2. Modelos de programación en red**
  - 3. El nivel físico**
  - 4. El nivel de enlace**
  - 5. El nivel de transporte**
  - 6. Resumen**
- Ejercicios de repaso y autoevaluación**

Capítulo 3

**Programación de servicios de comunicaciones**

- 1. Introducción**
- 2. Aplicaciones y utilidades de comunicaciones. Estándares de comunicaciones**
- 3. Librerías de comunicaciones de uso común**
- 4. Programación de componentes de comunicaciones**
- 5. Técnicas de depuración de servicios de comunicaciones**
- 6. Rendimiento en las comunicaciones**

**7. Resumen**  
**Ejercicios de repaso y autoevaluación**

Capítulo 4  
**Seguridad en las comunicaciones**

- 1. Introducción**
- 2. Principios de seguridad en las comunicaciones**
- 3. Herramientas para la gestión de la seguridad en red. Scanners**
- 4. Seguridad IP**
- 5. Seguridad en el nivel de aplicación. El protocolo SSL**
- 6. Seguridad en redes inalámbricas**
- 7. Resumen**

**Ejercicios de repaso y autoevaluación**

Bibliografía

# Capítulo 2

## Fundamentos de comunicaciones

### Contenido

1. Introducción
2. Modelos de programación en red
3. El nivel físico
4. El nivel de enlace
5. El nivel de transporte
6. Resumen

### 1. Introducción

La programación en red es una de las áreas de la informática que más está evolucionado en los últimos años, debido principalmente a redes como Internet. Hay una evolución constante hacia modelos que optimicen la capacidad de ejecutar programas de manera distribuida, aprovechando la capacidad de procesamiento y reutilización de componentes ya creados.

Varios son los paradigmas que se presentan en este capítulo, mostrando desde mecanismos menos abstractos y simples hasta los más abstractos y complejos. La utilización de un modelo u otro se determinará por el tipo de problema a resolver.

La comunicación a través de redes y su evolución hacia modelos normalizados en protocolos permiten minimizar los problemas de seguridad y velocidad que presenta este tipo de programación frente a una programación local más segura pero menos efectiva.

### 2. Modelos de programación en red

La programación en red se basa en el desarrollo de aplicaciones que a diferencia de la programación en local hacen uso de una red de ordenadores para su ejecución. Los procesos que componen el programa van a separarse en módulos que se distribuyen por la red.

Las ventajas que proporciona la programación en red son numerosas, aunque también presentan una serie de inconvenientes. Entre las ventajas se pueden encontrar la compartición de recursos, la escalabilidad y la tolerancia a fallos; y entre los inconvenientes ciertos puntos críticos en cuanto a seguridad y posibles fallos.

Los modelos de programación en red han ido evolucionando, adoptando diferentes paradigmas basados en niveles de abstracción. De esta forma, y enumerando desde una nivel bajo de abstracción a un nivel alto, se tendría: paso de mensajes, cliente-servidor, procedimientos remotos, métodos remotos o aplicaciones colaborativas.



### Recuerde

Un sistema en red es aquel en el cual componentes de *hardware* y *software*, localizados en computadores en red, se comunican y coordinan por medio de mecanismos de sincronización; y la programación en red son paradigmas de programación que permiten desarrollar una aplicación para que se ejecute en una red de computadoras.

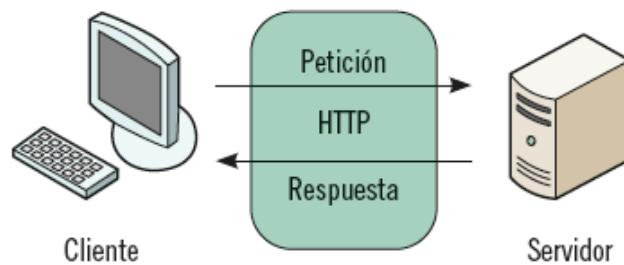
## 2.1. El modelo cliente/servidor

El paradigma más conocido para aplicaciones en red es el modelo cliente-servidor. Este modelo está adaptado a una arquitectura de computadores cliente-servidor, que es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realice se efectúe con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema.

En este modelo se asignan roles diferentes a los procesos que intervienen en la aplicación:

- **Un proceso:** el servidor se encarga de proveer de un servicio esperando de forma pasiva la llegada de peticiones.
- **El otro proceso:** el cliente realiza peticiones al servidor y aguarda la respuesta. En la siguiente imagen, se ilustra este paradigma.

### Petición web como ejemplo de modelo cliente-servidor



En este caso, se muestra un único cliente que interactúa con un servidor. Sin embargo, la actuación más común es que muchos clientes acceden al servidor para obtener el servicio solicitado. Otros ejemplos de este modelo pueden ser FTP (*File Transfer Protocol*), DNS (*Domain Name System*), etc.

En este modelo, al igual que se pueden tener muchos clientes también se pueden tener varios servidores en el sistema, lo que permite una escalabilidad alta. Un servidor puede ser sustituido por otro que ofrezca el mismo servicio y esto es totalmente transparente para el cliente. Los clientes no conocen la ubicación del servidor, lo cual no afecta a la prestación de servicios. Por último, indicar que el servidor puede regular el acceso a recursos compartidos para conseguir una mayor optimización en la ejecución de las aplicaciones.

Los servidores se pueden clasificar en función de los servicios que ofrecen. De esta forma se pueden tener:

▪ **Servidores de archivos:**

- Msg.: peticiones de archivos.
- NFS, SAMBA...

▪ **Servidores de bases de datos:**

- Msg.: peticiones SQL.
- Oracle, Sybase, SQL Server...

▪ **Servidores de transacciones:**

- Msg.: transacción (conjunto de peticiones SQL).
- OLP...

▪ **Servidores de objetos:**

- Msg.: invocación a procedimientos remotos.

- Servidores CORBA, OLE/DCOM...

- **Servidores web:**

- Msg.: peticiones HTTP.
- servidores HTTP...

- **Servidores de *groupware*:**

- Msg.: mensajes de *groupware*, e-mails...
- Lotus Notes, Exchange...

Otro tipo de clasificación es en función de cómo se distribuye la carga de la aplicación que se está desarrollando bajo el paradigma cliente-servidor. Así se tiene la siguiente clasificación:

Concepto de carga balanceada

- **Cliente pesado/servidor ligero:**

- La mayor parte de la aplicación corre en el lado cliente.
- El servidor exporta datos en bruto.
- Los clientes conocen la organización de los datos en el servidor.

- **Cliente ligero/servidor pesado:**

- La mayor parte de la aplicación corre en el lado servidor.
- El servidor exporta métodos que operan sobre los datos.
- El cliente no es mucho más que el interfaz de la aplicación.

Se tiene que definir muy bien si se quiere que la mayor carga de trabajo la tenga el cliente o el servidor. Eso va a depender de aspectos tales como disponibilidad del servidor, seguridad, etc.



### Actividades

1. Indique un servicio donde se identifique un modelo de cliente pesado/servidor ligero y otro de cliente ligero/servidor pesado.
2. Comente si cree que un ordenador puede actuar como cliente y servidor al mismo tiempo.
3. Razone su respuesta y, si cree que sí, identifique situaciones donde sea interesante usar esta característica.

---

Cuando se diseña una aplicación en red se necesita dividir el proceso en niveles, para diferenciar los aspectos funcionales del programa. Así se tiene el nivel de acceso a datos, el nivel de presentación y la lógica de negocio.

El nivel de acceso de datos se encarga de acceder a la base de datos y proporcionar la información solicitada. En el nivel de presentación se muestran los resultados obtenidos de la aplicación. Se puede decir que es el GUI (Graphical User Interface) del programa. Por último, la lógica de negocio o de aplicación es la encargada de hacer el procesamiento de los datos obtenidos y pasarlo a la capa de presentación.

Esta división funcional va íntimamente relacionada con diferentes arquitecturas C/S, basadas en el número de niveles adoptados. Así se tiene:

- **Arquitectura de dos niveles.** En este caso la lógica de negocio va integrada con el nivel de presentación o de datos.
- **Arquitectura de tres niveles.** Los tres niveles funcionales están separados en máquinas diferentes.
- **Arquitectura de multinivel.** La lógica de negocio se divide en varias máquinas.

La programación cliente-servidor se puede realizar a través de *sockets*, llamada a procedimientos remotos (RPC) o por invocación de métodos remotos (RMI).

## Sockets

Los *sockets* constituyen un mecanismo por el cual se pueden comunicar procesos locales con procesos remotos. Existen varios tipos de *sockets* y la comunicación puede ser unidireccional o bidireccional.

Los datos que identificarán únicamente a un *socket* son los **siguientes**:

- **La dirección IP** del ordenador en donde reside el programa que está usando este punto de conexión.
- **Un número de puerto** El puerto por el cual el socket se va a comunicar.
- **El tipo de socket.** Pueden ser de dos tipos según la comunicación sea orientada a conexión o no. De esta forma se tendrán *sockets UDP* y *sockets TCP*. La comunicación solo es posible entre *sockets* del mismo tipo.

## Procedimientos remotos

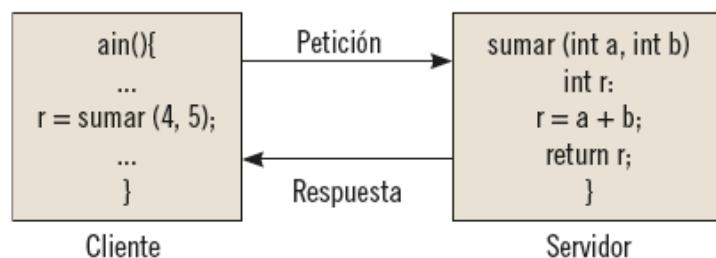
El modelo de paso de mensajes a través de *sockets* funciona bien para protocolos básicos de red y para aplicaciones de red sencillas. Sin embargo, cuando las aplicaciones crecen y la lógica de negocio se complica resulta necesario un nivel de abstracción mayor.

Resultaría muy interesante un paradigma que permitiera que un programa en red se pudiera programar de manera parecida a las aplicaciones de sobremesa que se

ejecuten en local sobre un único procesador. Pues bien, esto se consigue a través de llamadas a procedimientos remotos (RPC, *Remote Procedure Call*), proporcionando la abstracción indicada.

RPC permite a los programadores realizar aplicaciones de red usando una construcción de programación similar a una llamada a procedimiento local, proporcionado un nivel de abstracción conveniente tanto para la comunicación entre procesos como para la sincronización de eventos. En el siguiente gráfico se muestra el funcionamiento.

#### Ejemplo de llamada RPC



En el ejemplo, el cliente invoca a la función **suma ()** que no está en local, sino que está en el servidor. Junto a la petición se envía también la lista de valores de los argumentos de la suma. Al finalizar la función, el servidor devuelve el resultado como respuesta y se lo envía al cliente.



#### Actividades

4. Comente si cree que se podrían pasar los argumentos de una función en RPC por referencia. Razone la respuesta.

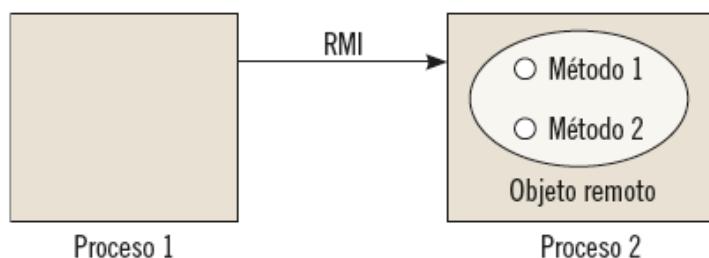
## 2.2. El modelo de objetos distribuidos

En este modelo se busca aplicar la filosofía de la orientación a objetos a las aplicaciones en red. Por lo tanto, se puede decir que **es una extensión del desarrollo de software orientado a objetos**. En este caso, los objetos están distribuidos sobre una red de ordenadores, y proporcionan métodos y propiedades a través de los cuales una aplicación ubicada en otro ordenador obtiene acceso a los servicios.

## Invocación de Métodos Remotos

La Invocación de Métodos Remotos (*Remote Method Invocation-RMI*) es el equivalente en orientación a objetos de las Llamadas a Procedimientos Remotos ya vistas. En este modelo, un proceso invoca métodos de un objeto que reside en un ordenador remoto.

Ejemplo de llamada RMI



Como en el caso del RPC, los argumentos y valores necesarios para solicitar el servicio se pasan con la llamada, y se devolverá el resultado cuando el método invocado haya finalizado. Para el programador es totalmente transparente y el diseño del programa, funcionalmente hablando, es idéntico a su ejecución en local.



### Aplicación práctica

Como programador de la empresa le encargan la labor de desarrollar un programa en red que permita conectar diferentes procesos, y debe decidir entre usar sockets o RMI. El responsable del departamento le va detallando las características de las cuatro redes donde se quiere implantar el programa. ¿Cómo razonaría el uso de un método u otro, para los siguientes escenarios planteados?

1. **Enlace físico lento.** se utilizan sockets tcp/udp
2. **Envío de pocos mensajes y transmisión de muchos datos.** Sockets
3. **Muchos servicios ofrecidos por el servidor.** RMI
4. **Modificación frecuente de la aplicación servidor.** RMI

### SOLUCIÓN

1. **Sockets.** Se envían solo los mensajes que usted construye. La cantidad de información enviada es mínima, mucho menor que en RMI, que se debe enviar todo el protocolo interno.

2. *Sockets*. Este puede ser el caso de envío de ficheros, por ejemplo. Es una funcionalidad simple de conexión y envío de información. No se necesita mayor sincronización.
3. *RMI*. Se pueden programar fácilmente los métodos necesarios para los nuevos servicios que se quieran ofrecer.
4. *RMI*. Es mucho más fácil escalar nuevos objetos de programación que en el caso de los *sockets*.

### Paradigma basado en Object Request Broker

En el paradigma basado en *Object Request Broker*, un proceso solicita una petición a un **ORB** (*Object Request Broker*), el cual redirige la petición al objeto apropiado que proporciona dicho servicio. El paradigma se parece bastante al modelo de Invocación de Métodos Remotos en su soporte para acceso remoto a objetos. La diferencia es que el **ORB** funciona como *middleware*, permitiendo a una aplicación, como solicitante de un objeto, acceder potencialmente a varios objetos remotos (o locales).



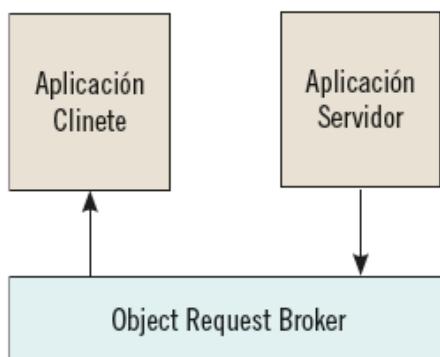
#### Definición

##### **Middleware**

Software necesario para el soporte de interacciones entre clientes y servidores a través de una plataforma heterogénea.

El ORB permite también la conexión entre objetos de diferentes tecnologías. Para ello se hace uso de APIs o interfaces que permitan que ambas tecnologías puedan interactuar.

#### ORB actuando de planificador

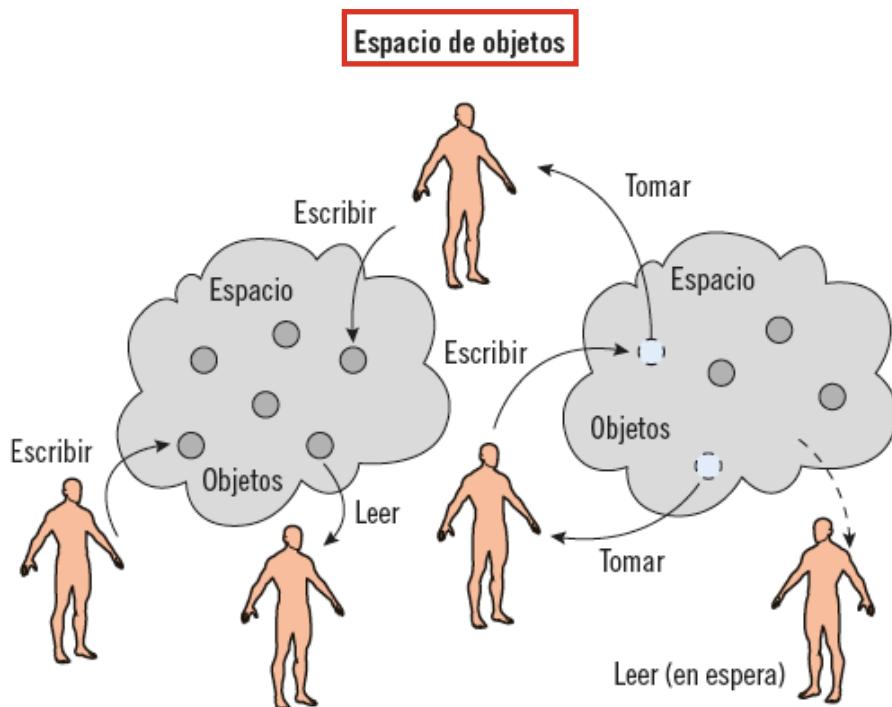


Este modelo se usa en la arquitectura **CORBA** (*Common Object Request Architecture*), permitiendo el desarrollo de programas en entornos distribuidos.

Otras tecnologías basadas en este modelo pueden ser Java Beans o Microsoft COM.

### Espacio de objetos

Es el más abstracto de los paradigmas orientados a objetos. El paradigma de espacio de objetos asume la existencia de entidades lógicas conocidas como **espacios de objetos**. Los participantes en una aplicación convergen en un espacio de objetos común. Un suministrador coloca objetos como entidades dentro de un espacio de objetos y los solicitantes que se suscriben al espacio pueden acceder a dichas entidades. El gráfico siguiente ilustra este paradigma.



Habrá que entender este paradigma como puntos de encuentro entre productores y consumidores de objetos, donde la exclusión mutua forma parte de esa relación, ya que solo un objeto puede ser usado por un participante a la vez.

*JavaSpaces* se basa en el uso de espacio de objetos en Java.

## 2.3. Modelos basados en mensajes. Introducción a los servicios web

La forma más básica de comunicación entre procesos distribuidos en una red de ordenadores es el paso de mensajes. En este modelo, los datos se envían a base de mensajes entre los procesos emisor y receptor.

Un proceso envía un mensaje como petición de un determinado servicio a un servidor. El mensaje llega al receptor que interpreta la petición y envía un mensaje como respuesta. Es una comunicación de ida y vuelta, que puede provocar nuevas peticiones y nuevas respuestas.

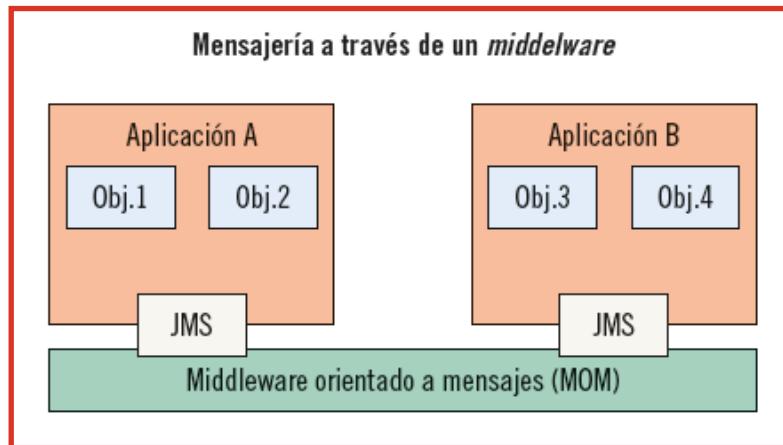
En este tipo de comunicaciones las operaciones básicas son enviar y recibir. El cliente es el proceso activo que solicita y el servidor actúa como un proceso pasivo a la espera. Cuando la comunicación está orientada a la conexión (el cliente y el servidor tienen que estar previamente conectados para enviar el mensaje), son necesarias las operaciones de conexión y desconexión.

El nivel de abstracción se asemeja mucho a las operaciones E/S de un fichero, pues las operaciones básicas son las de lectura y escritura de datos.

La programación de *sockets* se basa en este modelo. Un proceso escribe o inserta un mensaje en el *socket*, y en el otro extremo un receptor lee o extrae un mensaje del *socket*.

### Sistema de mensajes

El **Sistema de Mensajes** o **Middleware Orientado a Mensajes** (*Message-Oriented Middleware-MOM*) es una extensión del modelo básico de paso de mensajes.



JAVA MESSAGE SERVER

En este modelo el *middleware* es el encargado de mediar entre las aplicaciones de manera independiente. El sistema de mensajes actúa como un gestor de mensajes, a través del cual los procesos envían peticiones y reciben respuestas. La comunicación es asíncrona (no hay bloqueo) y totalmente desacoplada. Un emisor deposita un mensaje en el sistema de mensajes, el cual redirige el mismo a la cola de mensajes asociada a dicho receptor. Una vez que se ha enviado, el emisor queda liberado para que realice cualquier otra tarea.

Hay dos subtipos de modelos de sistema de mensajes:

- **Modelo de mensajes punto a punto.** En este modelo se envía la información desde el emisor hasta la cola de mensajes asociada con el receptor. Parece muy similar al modelo básico de paso de mensajes, sin embargo, hay una diferencia muy importante, y es que el *middleware* proporciona un buffer que permite que el envío y la recepción de mensajes sea asíncrono.  
El proceso receptor extrae los mensajes de su cola de mensajes y procesa cada mensaje de forma independiente.
- **Modelo de mensajes publicación/suscripción.** Como indica su nombre, en este modelo los actores publican o se suscriben a temas o eventos que son de su interés. Las aplicaciones interesadas en el suceso de un evento específico se pueden suscribir a los mensajes de dicho evento. Este tipo de comunicación es muy útil cuando es necesario un envío de mensajes a grupos de procesos, o sea, multidifusión. La operación **publicar** permite al proceso difundir a un grupo de procesos, y la operación **suscribir** permite a un proceso escuchar dicha difusión de mensajes.



#### Nota

MOM ha tenido una larga historia en las aplicaciones distribuidas. Los *Message Queue Services* (MQS) se llevan utilizando desde los años 80. El MQ\*Series de IBM es un ejemplo de dicho servicio. Otros soportes existentes para este paradigma son *Microsoft's Message Queue* (MSMQ) y el *Java's Message Service* (JMS).

### Introducción a los servicios web

Los servicios web son una colección de funciones que se empaquetan como entidad y son publicadas en la red para que sean accesibles por otros programas. Permiten de esta manera la reusabilidad de componentes que se pueden utilizar en un programa en red. El cliente diseña el programa y busca servicios en la red que le permitan realizar la funcionalidad que persigue.

La base fundamental de los servicios web es la mensajería XML (*eXtensible Markup Language*) sobre el protocolo http (*Hypertext Transfer Protocol*).

Este mecanismo es muy ligero, liberando de la carga de trabajo al cliente que solicita el servicio.

**El XML** es un lenguaje de marcas desarrollado por W3C y que sigue una estructura que permite almacenar datos de forma jerárquica.

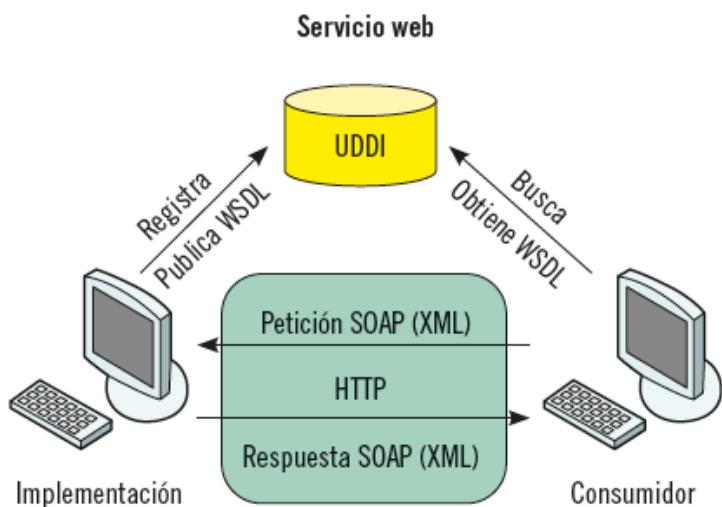
Por ejemplo, para identificar los datos de un pedido se podría identificar con el siguiente fichero xml:

```
<pedido>
    <id>PE-34</id>
    <cliente>7657</cliente>
    <producto>PE-456</producto>
    <cantidad>15</cantidad>
    <precio>14,98 €</precio>
</pedido>
```

Los **pasos** a seguir para la publicación y uso de un servicio web serían los siguientes:

1. Un proveedor crea, ensambla y despliega un servicio web mediante el lenguaje de programación y la plataforma que considere oportuna. Esto es independiente, puesto que la comunicación va a ser a través de un protocolo estándar de mensajería.
2. Una vez creado el proveedor define a base de interfaces públicas el servicio que se ofrece usando el WSDL (*Web Services Description Language*).
3. Posteriormente, el servicio web debe publicarse para que sea conocido por la comunidad en red. Para ello, se utiliza el servicio en registros de UDDI (*Universal Description, Discovery and Integration*). Además de permitir publicar, también admite búsquedas de servicios por parte de clientes que quieran usar el servicio.
4. Por último, y una vez que se localiza el servicio web que se quiere usar, se invoca dicho servicio a través de las operaciones definidas en su interfaz.

público. Para ello, se utiliza el protocolo SOAP (*Simple Object Access Protocol*). Se utiliza XML tanto para el envío de parámetros como para la recepción de resultados sobre HTTP.



Un servicio web consta de los siguientes tres elementos:

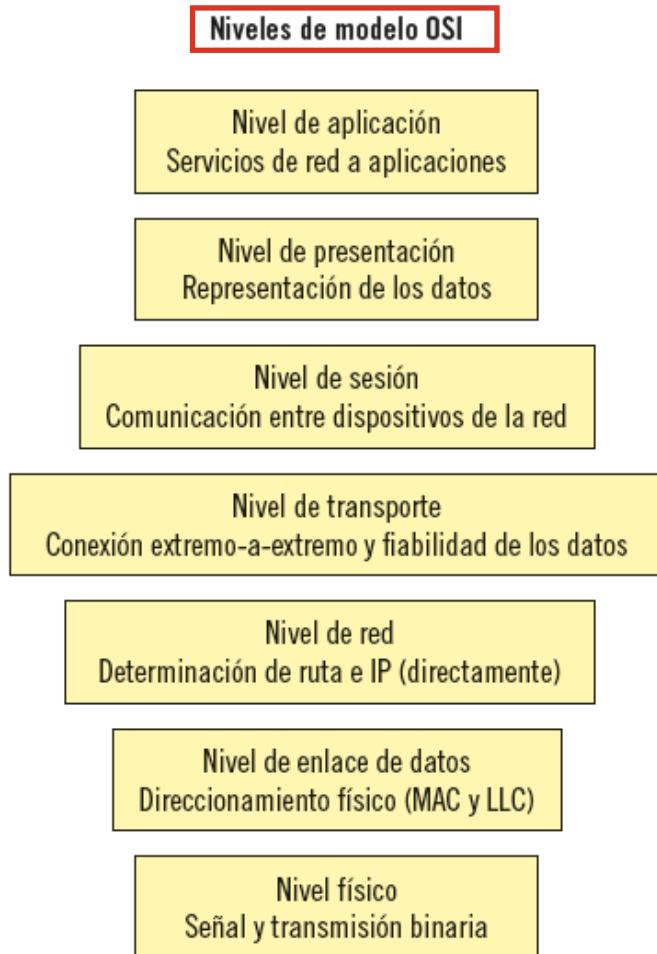
- **El servicio:** el programa en sí. Recibe parámetros, los procesa y devuelve resultados.
- **El documento XML:** a través de este lenguaje de marcas se envían los parámetros y resultados del servicio web.
- **La dirección:** es la dirección de red donde se encuentra ubicado el servicio.

### 3. El nivel físico

En 1980, la Organización Internacional de Estándares (ISO) desarrolló un modelo de interconexión entre sistemas que pretendía diseñar un estandar en las comunicaciones por red. El modelo desarrollado se denominó OSI (*Open Systems Interconnection*), y se basó en niveles-capas, por donde tienen que ir pasando los datos que viajan de un ordenador a otro de la red.

En cada una de esas capas se van añadiendo las cabeceras de datos necesarios para que la información sepa a dónde tiene que viajar, o cómo se debe mostrar en el ordenador destino, entre otras cosas.

En la siguiente imagen, se muestran los siete niveles que componen el modelo, y una breve descripción de la funcionalidad de cada uno de ellos:



- **Nivel de aplicación:** en este nivel se encuentran las aplicaciones de red, que permiten interactuar con el equipo remoto. Aplicaciones de este tipo son la transferencia de archivos (FTP), emulación de terminal (Telnet), etc.
- **Nivel de presentación:** este nivel se encarga del formato y conversión de los datos para que puedan ser interpretados por los programas utilizados en el nivel de aplicación. Por ejemplo: el formato ASCII, EBCDIC, etc.
- **Nivel de sesión:** este nivel se encarga de establecer y mantener la comunicación entre las aplicaciones de los ordenadores que están conectados. Entre sus funciones están la retransmisión de paquetes que se hayan podido perder y el control del flujo de mensajes.
- **Nivel de transporte:** busca la confiabilidad en las comunicaciones, es decir, fiabilidad en la transmisión de los paquetes de datos.
- **Nivel de red:** aquí se define las rutas que debe seguir el paquete de datos para llegar a su destino.
- **Nivel de enlace:** se controla el acceso al canal de comunicación.
- **Nivel físico:** este es el nivel más cercano al *hardware* del ordenador, y se definen aspectos físicos, mecánicos e incluso eléctricos.

Cada capa tiene un conjunto definido de funciones que da servicios a la capa que tiene por encima, usando a su vez los servicios que le proporciona la capa que tiene por debajo. Una determina capa "n" llega a establecer una comunicación virtual con la capa "n" del ordenador destino.

De este modo, por ejemplo, el nivel de red tiene que ser capaz de generar la información necesaria, para que cuando el paquete de datos llegue al nivel de red del otro ordenador este sea capaz de interpretar dicha información. Este modelo de división en niveles consigue que el problema de envío de información se divida en siete subproblemas a resolver.

Existe un modelo llamado TCP/IP que sigue la misma filosofía en capas que el modelo OSI, pero aplicado a la red de Internet. En este caso solo se cuenta con los niveles de aplicación, transporte, Internet y red.



### Importante

Algunas de las capas en el modelo TCP/IP poseen el mismo nombre que las capas del modelo OSI. Sin embargo, tienen funciones diferentes.

Dentro de los niveles del modelo OSI, el **nivel físico** es el encargado de las conexiones físicas del ordenador en la red. Sus **principales funciones** son:

- Establecer sobre qué medios se va a realizar la comunicación: cable de pares trenzados, coaxial, fibra óptica, etc.
- Definir aspectos tan materiales como los componentes o conectores necesarios para la conexión o eléctricos como niveles de tensión utilizados en el envío de datos.
- Concretar la funcionalidad de la interfaz, es decir, protocolizar el establecimiento, mantenimiento y liberación del enlace físico.
- Enviar el flujo de datos (bits) a través del medio que se haya decidido utilizar.
- Interpretar las señales físicas utilizadas (eléctricas/electromagnéticas/ ópticas).
- Detallar el tipo de cables o conectores utilizados en el medio de transmisión.
- Asegurar que la conexión sea estable.

Un aspecto fundamental en la comunicación física entre ordenadores es la sincronización, mediante la cual el equipo receptor debe conocer los momentos en los que tiene que interpretar las señales emitidas, para de esta forma recibir exactamente la información que quiso emitir el ordenador origen.

Existen **tres niveles de sincronización: de bit, de octeto y de trama;** dependiendo de si la unidad de información enviada se corresponde con un **bit, un byte o una trama** con un tamaño especificado.

La transmisión se puede realizar de manera asíncrona (paquetes independientes identificados con bits de inicio y fin) o síncrona (continuamente se envían paquetes que son interpretados por el equipo receptor, sincronizados por un reloj).

### 3.1. Dispositivos físicos

Para la conexión de los ordenadores que componen la red son necesarios una serie de dispositivos, que en algunos casos actúan solo en el nivel físico y en otros realizan también funciones de niveles superiores.

Entre los principales dispositivos utilizados en redes caben destacar los siguientes:

#### Tarjeta de red

Tarjeta que se instala en el ordenador bien a través de un slot o integrada en la placa madre. Permite la comunicación entre los ordenadores que componen la red.

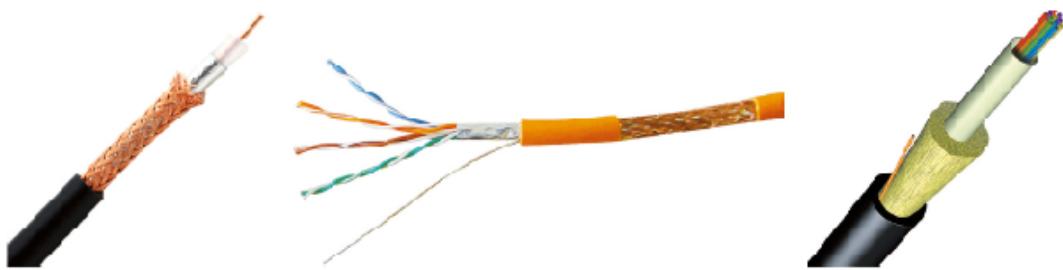
#### Cableado o canal de comunicación

Es el conjunto de cables, conectores o medios que son necesarios instalar para conseguir una infraestructura de telecomunicaciones. Este tipo de cableado debe cumplir una serie de estándares. Se puede hacer una división atendiendo a si el canal de comunicación es por cable o inalámbrico:

- **Por cable:**

- **Cable coaxial:** es un conductor de cobre rodeado de una capa aislante flexible.
- **Par trenzado:** pares de cables envueltos en una malla de cobre o una hoja metálica. Se entrelazan para evitar la aparición de interferencias.
- **Cable de fibra óptica:** son fibras de plástico o vidrio que permiten conducir los impulsos de luz desde el emisor hasta el receptor. En este tipo de cables, a diferencia de los dos anteriores, los bits se codifican como impulsos de luz.

### Cable coaxial, par trenzado y cable de fibra óptica



#### • Inalámbricos:

■ **Wi-Fi:** se basa en el envío de tramas en el espectro radioeléctrico.

■ **Bluetooth:** se basa en radiofrecuencia de corto alcance.

**Nota:** el nombre de Bluetooth se puso en honor a un rey noruego llamado Harald Bluetooth, por sus cualidades como comunicador y unificador entre tribus noruegas.

■ **Wimax:** utiliza las ondas de radio en las frecuencias de 2,3 a 3,5 GHz y puede tener una cobertura de hasta 60 km.

#### Repetidores

Cuando la distancia entre los ordenadores que se quieren comunicar es alta, la señal que se emite puede debilitarse y llegar con errores al destino. Para ello existen los repetidores, que su única función es recibir una señal de entrada y amplificarla, para que pueda llegar en óptimas condiciones a su destino.

#### Concentradores (hubs)

Este dispositivo, como su propio nombre indica, lo que hace es concentrar la señal que recibe por una de sus entradas y la repite por los diferentes puertos que tiene. De esta manera, se pueden conectar varios ordenadores a este dispositivo, recibiendo la misma señal en todos ellos.

#### Puente (bridge)

Este dispositivo opera en la capa de enlace de datos y permite conectar dos segmentos de red, o dividir la red en segmentos. Estudia la cabecera del paquete de datos que llega y lo pasa al segmento de red adecuado.

## Comutadores (switch)

Un comutador actúa igual que un puente, pero puede conectar y filtrar información entre más de dos redes, fusionándolas en una sola.

## Router

Este dispositivo es utilizado para la conexión de una red de ordenadores, operando en la **capa tres del modelo OSI**, o sea, a nivel de **red**. Permite el enruteamiento de paquetes entre redes, o decide la ruta que tiene que seguir el paquete de datos para llegar a su destino.



## Actividades

5. Identifique cuáles de los dispositivos físicos enumerados anteriormente utiliza en el equipamiento informático de su lugar de trabajo o estudio.
6. El par trenzado es el más utilizado en redes de área local. Señale si sabría identificar el conector al que va unido este tipo de cables, y qué nombre recibe.

## 3.2. Protocolos de nivel físico

Un protocolo de red es un conjunto de **reglas que especifica el intercambio de datos**, órdenes o cualquier otra información que defina cómo es la comunicación entre los ordenadores que forman parte de una red. Para cada nivel del modelo OSI se tendrán diferentes protocolos.

Para el caso del nivel físico, los protocolos están especificados por **comités de estándares** tales como la **EIA** (*Electronic Industries Alliance*), **ITU-T** (*International Telecommunication Union*), **SO** (**Organización Internacional de Normalización**), **IEEE** (*Institute of Electrical and Electronics Engineers*), etc.; describiendo cómo se hace físicamente la comunicación entre equipos.

Los protocolos del nivel físico definen las características propias del medio físico a través del cual se produce la comunicación entre ordenadores. Estas características son:

- **Eléctricas:** relación entre voltajes y datos binarios. De esta forma se puede interpretar que un voltaje de -3 voltios represente un 1 binario y un voltaje superior a +3 voltios, el 0 binario.

- **Mecánicas:** aquí intervienen principalmente los tipos de conectores utilizados, cableado e incluso el número de contactos de los conectores.
- **Funcionales:** verificar aspectos como el control de los datos o la temporalización.
- **Procedimientos:** secuenciación de los eventos necesarios para la comunicación física.

Una de las interfaces de comunicación más conocida es la comunicación serie (**Protocolo RS-232**), que conecta el ordenador con algunos dispositivos que no requieren de mucha velocidad. Las especificaciones de esta interfaz vienen definidas por los siguientes protocolos.

- Especificaciones mecánicas: ISO 2110.
- Especificaciones eléctricas: V.28.
- Especificaciones funcionales: V.24.
- Especificaciones de procedimiento: V.24.

Indicando los siguientes aspectos sobre RS-232:

- **Aspectos eléctricos:** el 1 binario está representado por un valor más negativo que -3 voltios, y el 0 binario, un voltaje superior a +4 voltios. Está diseñado para distancias cortas (15 metros) y velocidades bajas (20kb/seg).
- **Aspectos mecánicos:** el rs-232 es un conector tipo db-25 (25 pines), aunque también existe el de 9 pines (de-9).
- **Aspectos funcionales:** los datos se transmiten por el pin2 y se reciben por el pin 3.
- **Aspectos de procedimiento:** se basa en eventos de acción-reacción. El equipo receptor emite una señal de preparado para recibir, y cuando le llega al equipo emisor comienza el envío de datos.

Otros ejemplos de **protocolos**:

- **V.92:** define la comunicación a través de módem.
- **xDSL:** comunicación a través de la línea telefónica.
- **IrDA capa física:** comunicación a través de infrarrojos.
- **USB capa física:** comunicación de numerosos dispositivos *hardware* con el PC.
- **Firewire:** comunicación serie a gran velocidad.
- **EIA-422, EIA-423, RS-449 y RS-485:** Comunicación serie.
- **DSL:** comunicación a través de la línea telefónica.
- **ISDN:** conexiones digitales extremo a extremo.
- **10BASE-T, 10BASE2, 10BASE5, 100BASE-TX, 100BASE-FX, 100BA-SE-T, 1000BASE-T, 1000BASE-SX y otras variedades de la capa física de Ethernet:** conexiones Ethernet.
- **SONET/SDH:** protocolo de comunicación a través de fibra óptica.
- **GSM interfaz radio:** comunicación móvil.

- **Bluetooth capa física:** comunicación por radiofrecuencia.
- **IEEE 802.11x Wi-Fi capas físicas:** conexión entre dispositivos de manera inalámbrica.



### Aplicación práctica

**Últimamente se está detectando que las comunicaciones son lentas en la red de la Empresa X, a pesar de tener montada una red con velocidad de 100 Mbps. Sin embargo, utiliza hubs para conectar a todos los ordenadores. ¿Qué solución se podría plantear para aumentar dicha velocidad?**

#### SOLUCIÓN

Los hubs son los dispositivos más baratos, pero poco eficientes en cuanto a gestionar tráfico dentro de una red. Si son muchos los ordenadores conectados por hubs, las colisiones se multiplican, pues todos los mensajes que llegan a un puerto del hub son reenviados a todos los demás puertos, sin importar si el mensaje es para ellos o no. Al mismo tiempo, cuando se produce una respuesta, también esta se multiplica por todos los puertos.

Una solución sería el uso de varios *switch*, que son capaces de aprender a través de la información suministrada por el mensaje a qué único ordenador se han de enviar los datos.

## 4. El nivel de enlace

El nivel de enlace se encuentra situado en la pila de niveles OSI, sobre el nivel físico, y debajo del nivel de red. Este nivel tiene **dos funciones** básicas:

- Adaptar la información recibida por el nivel de red en paquetes, para que pueda ser enviada por el medio físico elegido.
- Interpretar la información enviada por el nivel físico, cuando se está recibiendo información de la red, y crear tramas que son enviadas al nivel superior.

El objetivo principal del nivel de enlace es la transmisión y recepción de manera segura de tramas de datos entre equipos transmisores de datos. En este nivel se identifican las tramas que van a ser enviadas como agrupaciones de bits. El control de errores de esas tramas es fundamental para que la información no llegue con ningún tipo de ruido.

Es frecuente que la capa de enlace de datos se incluya como una parte física en el sistema, como en una NIC Ethernet.

Esta capa se divide en dos subcapas:

- **Control de enlace lógico (LLC):** en este control se añade a la trama la información del nivel de red, incluyendo qué protocolo se usa. Por ejemplo: IP (*Internet Protocol*), IPX (*Internetwork Packet Exchange*), etc.
- **Control de acceso al medio (MAC):** en este control se añade el direccionamiento y los campos de delimitación de la trama, según las señales físicas del medio utilizado y el protocolo del enlace de datos.

Los estándares utilizados en la capa de enlace de datos son:

- **ISO:** HDLC (High-Level Data Link Control). ISO 9314 (Control de acceso al medio (MAC) de la DFFI).
- **IEEE:**

802.2 Control de enlace lógico (LLC).  
802.3 Ethernet.  
802.4 Token Bus.  
802.5 Token Ring.  
802.11 LAN Inalámbrica (WLAN) y malla (certificación Wi-Fi).  
802.15 Bluetooth.  
802.16 WiMax.

- **ITU:**

G.992 ADSL  
G.8100 – G8199 Aspectos relativos al protocolo MPLS sobre capa de transporte.  
Q.921 ISDN.  
Q.922 Frame Relay.

- **ANSI:**

ADCCP (Procedimientos de Control para Comunicación Avanzada de Datos). Equivalente al estándar ISO HDLC.  
X3T9.5 Interface de datos distribuida por fibras (FDDI) Doble anillo (Token Ring).  
X3T12 Actualización X3T9.5.  
Las tramas pueden venir identificadas por diferentes métodos:

- **Contador de caracteres:** se conoce el tamaño de la trama y se van leyendo datos hasta que se llegue al límite. Este método puede presentar problemas de sincronismo.

- **Caracteres delimitadores:** ciertos caracteres identifican cuando comienza y finaliza una trama. Normalmente es STX para el comienzo y ETX para el final.
- **Secuencia de bits:** igual funcionalidad que los caracteres delimitadores, solo que aquí se realiza con grupos de bits.

Para el control de errores se utilizan dos tipos de mecanismos. Un primer mecanismo de **detección de errores**, que consiste en codificar la trama y añadir información adicional, que procesará el equipo receptor. A raíz de los datos recibidos y de esa información adicional, se sabrá si la información enviada es correcta. Un ejemplo de este tipo de mecanismo es el llamado **CRC (Cyclic Redundancy Check)**. En el caso de que el código CRC no sea correcto, se debe solicitar al emisor que envíe de nuevo la información pues la que llegó es errónea.



## Definición

### Error de CRC

Indica que la información original no concuerda con la información obtenida.

El segundo mecanismo consiste en la **detección y corrección de errores**. En este caso no solo se detecta que la información recibida no es errónea, sino que es capaz de reconstruir la trama correcta. En este caso no es necesaria la transmisión de nuevo del mensaje. ¿Y cómo se puede hacer esto? Pues bien, hay diferentes métodos y uno de ellos es la **distancia de Hamming**.

El método Hamming se basa en un algoritmo donde los bits pares se utilizan como bits de paridad y los impares como bits de información. A partir de esa información y con una serie de cálculos, es posible averiguar qué bit fallo y volver a poner el correcto.

La tasa de errores se mide por la BER (*Bit Error Rate*), y se define así:

$$\text{BER} = \text{bits erróneos} / \text{bits transmitidos}$$

La siguiente tabla presenta la tasa de error de los principales medios físicos utilizados:

Medio físico	BER Típico
Fibras ópticas	< 10^-12
LANs de cobre y radioenlaces fijos (microondas)	<10^-8
Enlaces telefónicos, satélite, ADSL y CATV	<10^-5
GSM	>10^-5



### Actividades

7. Comente si cree que la técnica de detección de errores CRC solo se utiliza en la transmisión de datos. Si cree que no, identifique otro proceso donde se aplique.
8. Si en GSM la transmisión es a 7600 bps y BER = 10^-5, señale cuánto tiempo pasará hasta que falle un bit.

El elemento principal con el que se trabaja en el nivel de enlace es la trama que se envía del emisor al receptor. La estructura tipo de una trama de red es la que se indica en la tabla.

Inicio	Dirección	Tipo	Secuencia	Confirmación	DATOS	Redundancia	Fin

- **Inicio:** marca que indica que comienza una nueva trama.
- **Dirección:** dirección del receptor del mensaje.
- **Tipo:** indica de qué tipo es la trama que se envía. Existen tres tipos de tramas: de datos (cuando se envían datos), de confirmación negativa o positiva (cuando es información de control).
- **Secuencia:** es un contador que va identificando el número de la trama. De esta forma el receptor es capaz de ordenarlas para poder agruparlas y crear el mensaje en su conjunto.
- **Confirmación:** en este campo se indica si la recepción ha sido positiva o no.
- **DATOS:** este campo contiene la información en sí de lo que se quiere enviar.
- **Redundancia:** información necesaria para el control de errores. Cuando el receptor recibe la trama, lee este campo y a través de un algoritmo se detecta si el mensaje ha sufrido algún cambio en la transmisión.
- **Fin:** marca que identifica el final de la trama.

Una vez que está definida la trama con todos sus campos, es hora de ponerla en la red, para su envío al dispositivo destino. A la colocación de tramas en los medios se conoce como **control de acceso al medio**.

Existen varias técnicas de control de acceso al medio, diferenciando si se comparte el medio o no. Hay dos métodos básicos:

- **Controlado.** En este caso cada nodo (ordenador) de la red tiene un tiempo de uso de ella. Aquí no se pueden producir conflictos. Ejemplo: *Token Ring*, FDDI (Determinista).
- **Basado en la contención.** Aquí todos los nodos quieren hacer uso de la red para enviar sus tramas. Ejemplo: Ethernet e inalámbricas.

Existen varios mecanismos que se pueden usar en el método basado en la contención. Uno de los más utilizados es el acceso múltiple por detección de portadora (CSMA).

Con el mecanismo CSMA/CD se envían los datos a través de la red y si hay colisión se para y se intenta enviar la trama más tarde. Ese tiempo de espera se calcula de manera aleatoria.



#### Importante

---

El envío de mensajes se intenta un número determinado de veces según el tipo de red. Si no se pudiera realizar la transmisión en esos intentos, el envío se descarta.

---

En el caso de las redes inalámbricas se usa CSMA/CA, y previamente al envío de la trama se envía un mensaje indicándolo. De esta manera, los nodos reciben el mensaje de que hay otro nodo que quiere emitir y esperan un corto intervalo de tiempo, estimando que la comunicación ya se ha realizado.

### 4.1. Redes Ethernet

La red Ethernet es una tecnología de redes de ordenadores de área local (LAN) basada en tramas de datos. El nombre viene definido por Ether (características físicas del cableado IEEE 802.2) y los formatos de trama del nivel de enlace de datos (Net). Está definida bajo el protocolo IEEE 802.3, utilizando el método de acceso al medio CSMA/CD.

Inicialmente se diseñó para enviar datos a 10Mbps, pero ha sido perfeccionada y ahora permite velocidades de 100Mbps, 1Gbps o 10 Gbps.

Ethernet se ha extendido de manera rápida porque tiene numerosas ventajas frente a otras redes. Entre esas ventajas se cuenta con su velocidad, bajo costo y que es fácil de instalar. Además, es capaz de aceptar los protocolos de red más utilizados.

Una trama Ethernet 802.3 tiene los siguientes campos:

Preámbulo (7 bytes)	Inicio (1)	Direc. Destino (6)	Direc. Origen (6)	Long. Datos (2)	Datos (0-1500)	Relleno (0-46)	CRC (4)
------------------------	---------------	-----------------------	----------------------	--------------------	-------------------	-------------------	------------

- **Preámbulo:** al ser transmitido este conjunto de bits se produce la sincronización a nivel de reloj entre receptor y emisor de la trama.
- **Inicio:** indica el comienzo de la trama.
- **Dirección de destino:** contiene la dirección MAC (dirección física) a quien se envía la información. El bit más a la izquierda indica si el envío es individual (0) o es a un grupo de direcciones (1).
- **Dirección origen:** contiene la dirección MAC de quien envía el mensaje. El bit más a la izquierda es siempre 0. El receptor cuando recibe la trama identifica a través de este campo quién se lo envía por si tiene que enviarle alguna respuesta.
- **Longitud de datos:** longitud de los datos que se envían.
- **Datos:** contiene los datos a transmitir.
- **Relleno:** relleno para completar el tamaño estándar de una trama.
- **CRC:** campo de control de errores.

Las redes Ethernet pueden adoptar diferentes topologías físicas y lógicas. La topología indica la disposición de los ordenadores dentro de una red, cómo están conectados y cómo fluyen las señales a través de dicha red. Dentro de la topología hay que diferenciar entre topología física, que indica cómo están físicamente ubicadas las estaciones, y topología lógica, que indica cuál es el trayecto seguido por la información a través de una topología física.

Las conexiones entre las estaciones de una red pueden ser

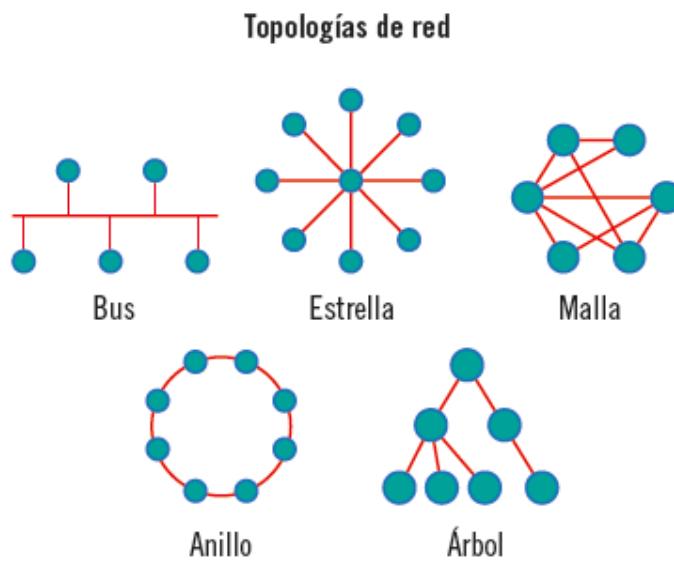
- **Conexiones punto a punto:** la comunicación entre estaciones es directa, sin ordenadores por medio.
- **Conexiones multipunto:** existe un único canal de conexión, y la información enviada la ven todas las demás estaciones.

Entre las topologías más utilizadas destacan:

- **Topología en malla:** cada dispositivo tiene un enlace punto a punto con todos los demás dispositivos de la red. No puede existir ninguna interrupción en la transmisión; si no puede ir por un camino puede elegir otro.

- **Topología en estrella:** cada dispositivo tiene un enlace punto a punto con un concentrador central, que es el que se encarga de derivar tramas. A diferencia de la malla, los ordenadores no están conectados directamente. Si un ordenador se queda aislado porque falla su conexión solo queda fuera de la red dicho equipo, siguiendo el resto de equipos operativos.
- **Topología en árbol:** es una topología de tipo estrella, es decir, existen varios concentradores que van reuniendo estaciones. Esto va formando una red en forma de árbol. Si el segmento principal falla, todo sus hijos quedarán aislados con respecto a las otras ramas del árbol.
- **Topología en bus:** aquí la conexión es multipunto. Un único canal troncal es el que se encarga de conectar todas las estaciones. Un fallo en el canal central provocaría que la red se partiera en dos, impidiendo la conexión de un ordenador de un segmento con otro ordenador del otro segmento.
- **Topología en anillo:** esta es una conexión punto a punto donde una estación solo se conecta con las dos que tiene a su lado. Si una conexión falla se puede acceder al ordenador por el camino contrario, aunque se encuentre más alejado.

En el siguiente gráfico se muestran los diferentes tipos de topologías.



### Aplicación práctica

En el departamento de informática se está estudiando qué topología de red sería la más conveniente en una red con cinco ordenadores si fallara una sola conexión, o sea, que una de las líneas mostradas en los gráficos anteriores se cortara.

**No se plantea que deje de funcionar un ordenador sino que la línea que une un nodo de la red con otro dejara de funcionar.**

**¿Qué topología cree que sería la más conveniente si lo que se persigue es que sigan comunicándose el mayor número de ordenadores posibles entre sí?**

## SOLUCIÓN

A continuación, se muestran los efectos en cada una de las redes:

- | Malla: un fallo provocaría no poder acceder al dispositivo directamente, pero se podría hacer con dos saltos.
- | Estrella: provocaría no poder acceder al concentrador central, y por lo tanto no se podría comunicar con ningún otro dispositivo.
- | Bus: esto provocaría que la red se dividiera en dos. Los equipos de la misma subred podrían comunicarse, pero no entre subredes.
- | Anillo: en este caso uno de los sentidos de la comunicación quedaría inutilizado. Se llegaría al dispositivo dando la vuelta al anillo.

La topología anillo y malla son las únicas que consiguen que la comunicación siga siendo completa. Por lo tanto, cualquiera de ellas sería óptima para solucionar el problema.

---

En la tabla siguiente se muestran familias de redes Ethernet, indicando aspectos físicos y lógicos que intervienen en el nivel de enlace:

Tipo	Medio	Ancho de banda máximo	Longitud máxima de segmento	Topología Física	Topología Lógica
10Base5	Coaxial	10 Mbps	500 m	Bus	Bus
10Base-T	Partrenzado	10 Mbps	100 m	Estrella	Bus
10Base-FL	Fibra óptica	10 Mbps	2000 m	Estrella	Bus
100Base-TX	Partrenzado	100 Mbps	100 m	Estrella	Bus
100Base-FX	Fibra óptica	100 Mbps	2000 m	Estrella	Bus
1000Base-T	Partrenzado	1000 Mbps	100 m	Estrella	Bus

Fast Ethernet, Gigabit Ethernet y 10 Gigabit Ethernet son los nombres de las redes con ancho de banda 100 Mbps, 1000 Mbps y 10000 Mbps, respectivamente.

Como se observa, la topología física más común es la estrella, donde todos los dispositivos de la red se conectan a un conmutador central, y a través de él se derivan las comunicaciones. Este método es óptimo porque los datos se pueden enviar entre dispositivos sin que afecten al resto.

El conmutador tiene registradas todas las direcciones físicas de los dispositivos de la red. Cuando llega una trama, identifica la dirección destino y le envía los datos al puerto del dispositivo que tiene esa dirección.

Los dispositivos conectados pueden funcionar en modo dúplex, donde se puede enviar y recibir datos al mismo tiempo.

En las redes Ethernet se pueden hacer diferentes tipos de envíos, según sean los destinatarios:

- **Unicast:** envío de tramas uno a uno. Solo se envía la trama a un dispositivo concreto dentro de la red.
- **Multicast:** envío de tramas de uno a muchos. Se envía la trama a varios dispositivos dentro de la red.
- **Broadcast:** envío de tramas de uno a todos los dispositivos de la red.

## 4.2. Direcciones físicas

La dirección física o MAC (*Media Access Control*) identifica dentro de una trama la estación origen y destino de los ordenadores que intervienen en la comunicación. La dirección MAC se encuentra grabada en el dispositivo de red, y es única dentro de dicha red. No puede haber dos dispositivos con la misma dirección MAC.

Está formada por 48 bits representados por dígitos hexadecimales, agrupándose por parejas separadas por el símbolo ":" o mediante "-". Un ejemplo de dirección MAC podría ser: F6:B5:D3:E1:A4:B5.

Dentro de esos bits, 24 identifican al fabricante de la tarjeta y los otros 24 se utilizan para diferenciar las tarjetas creadas por ese fabricante.

En una comunicación entre ordenadores los paquetes de datos se envían a la dirección física indicada en la trama, y no a su dirección lógica.

La red de Internet utiliza el protocolo IP para identificar estaciones dentro de la red a través de su dirección IP. Sin embargo, este sistema de direccionamiento no está relacionado con el *hardware*, y por lo tanto con su dirección física. Para hacer llegar físicamente la información a la estación destino es necesario un protocolo de conversión. Ese protocolo es ARP (Protocolo de Resolución de Direcciones), y permite averiguar la dirección MAC correspondiente a una dirección IP determinada.



## Actividades

9. Según el sistema operativo utilizado, averigüe el comando que le muestra la dirección MAC de su tarjeta de red.

Cuando se quiere hacer un envío **multicast** el primer bit del primer byte tiene que ser 1. Si por el contrario es 0, se trata de un envío **unicast**.

Para el caso del broadcast la dirección MAC es: FF:FF:FF:FF:FF:FF

## 5. El nivel de transporte

El nivel de transporte es el encargado de asegurar que los paquetes de información enviados por las aplicaciones llegan a su destino, libre de errores, bien secuenciados y sin ningún tipo de ruido que pudieran ocasionar pérdidas o duplicaciones. Este nivel libera a las capas superiores de preocuparse de si la información enviada llega o no.

Entre las **funciones** que proporciona esta capa destacan:

- **División del mensaje en segmentos.** Acepta la información que le envía el nivel de sesión que se encuentra por encima de ella, y la divide en paquetes más pequeños que son los que se van a enviar. Posteriormente, el dispositivo de destino se encargará de volver a ensamblar para generar el mensaje origen.
- **Confirmación.** Proporciona mecanismos de acuse de recibo indicando que los paquetes enviados han llegado a su destino.
- **Control de flujo.** Proporciona el mecanismo de control para evitar desbordamiento cuando el emisor quiere enviar más información de la que puede asumir el receptor.
- **Multiplexación de sesión.** Permite mezclar diferentes paquetes de distintos mensajes sobre el mismo canal.

## 5.1. El protocolo TCP/IP

TCP/IP es un conjunto de protocolos diseñado para redes de gran tamaño que pueden ser subdivididas en segmentos de red. La comunicación de estos segmentos de red se realiza a través de enrutadores. Internet utiliza este conjunto de protocolos. En su nombre se pueden identificar dos protocolos:

- **TCP (Transmision Control Protocol):** este protocolo proporciona los mecanismos necesarios para poder enviar información de un extremo a otro de la conexión, pudiéndose detectar posibles errores y corregirse.
- **IP (Internet Protocol):** protocolo del nivel de red que permite el desarrollo y transporte de datagramas de paquetes de datos. Uno de los campos principales que se trata en este protocolo es la dirección IP, que identifica equipos dentro de la red.

También se contemplan más protocolos. A modo de ejemplo, se citan los más comunes, pues existen más de 100:

- **HTTP:** acceso a páginas web.
- **ARP:** protocolo de resolución de direcciones.
- **FTP:** transferencia de ficheros.
- **SMTP y POP:** protocolos para correo electrónico.
- **TELNET:** acceso a equipos remotos.

El éxito y la difusión del protocolo TCP/IP se debe principalmente a su uso por la red Internet, y que responde perfectamente a las necesidades que se pide a una red de ordenadores. Entre estas características caben destacar:

- Son protocolos abiertos y soportados por cualquier sistema independientemente del *hardware* o sistema operativo utilizado.
- Funciona sobre cualquier medio físico, bien sea ADSL, fibra óptica, coaxial, etc.
- Proporciona mecanismos de identificación únicos dentro de las redes, aunque esa red sea tan inmensamente grande como es la de Internet.

El protocolo TCP/IP fue creado antes que el modelo capas OSI, por lo tanto no existe una relación exacta entre los niveles de un modelo y de otro.

En la siguiente figura se muestra la relación entre una y otra.

---

### **Relación entre el modelo TCP/IP y el modelo OSI**

---

<b>TCP/IP</b>	<b>Modelo OSI</b>
	Capa de aplicación
Capa de aplicación	Capa de presentación
	Capa de sesión
Capa de transporte	Capa de transporte
Capa de internet	Capa de red
	Capa de enlace de datos
Capa de acceso a la red (NAL)	Capa física

---

Como se observa, algunas capas del modelo TCP/IP agrupan varias del modelo OSI. En ambos modelos los datos van viajando hacia niveles inferiores, donde cada capa va añadiendo sus cabeceras. Cuando las tramas llegan al receptor, los datos van viajando desde niveles inferiores a superiores, interpretando cada nivel la cabecera que le corresponda.



#### **Importante**

---

Aunque algunas capas de los dos modelos tengan nombres iguales su funcionalidad y objetivos varían, aunque comparten los aspectos fundamentales.

---

## **5.2. Esquemas de direccionamiento**

Los esquemas de direccionamiento son la manera de poder localizar un dispositivo dentro de una red. En este apartado se definen los diferentes mecanismos para llevar a cabo ese direccionamiento.

Anteriormente, se ha hecho referencia a las direcciones IP como la manera de identificar dispositivos de manera única dentro de una red. Estas direcciones IP están formados por 32 bits.

Para facilitar la forma de identificar las direcciones IP, los bits se dividen en grupos de 8 bits (octetos), creando 4 grupos separados por puntos. Cada byte se convierte a su equivalente decimal obteniendo un número entre 0 y 255.



### Ejemplo

---

192.168.2.1 representa en binario: 11000000. 10101000. 00000010. 00000001

---

Para **pasar un número de decimal a binario** se divide el número sucesivamente por 2 y los residuos junto con el último cociente componen el número en binario.



### Ejemplo

---

Pasar el número 100 a binario.

$$100 : 2 = 50; \text{resto} = 0$$

$$50 : 2 = 25; \text{resto} = 0$$

$$25 : 2 = 12; \text{resto} = 1$$

$$12 : 2 = 6; \text{resto} = 0$$

$$6 : 2 = 3; \text{resto} = 0$$

$$3 : 2 = 1; \text{resto} = 1$$

---

Por lo tanto, el número en binario sería 1100100.

Para **pasar de binario a decimal**, hay que multiplicar cada cifra por 2 elevado a su potencia comenzando por  $2^0$ , y sumar los resultados.



### Ejemplo

---

Pasar el número 1100100 a decimal:

$$0 * 2^0 + 0 * 2^1 + 1 * 2^2 + 0 * 2^3 + 0 * 2^4 + 1 * 2^5 + 1 * 2^6 = 100.$$

---



### Actividades

---

10. Averigüe la dirección IP de su ordenador y pásela a binario.

---

Las redes de gran tamaño se dividen en otras más pequeñas para facilitar su gestión. La dirección IP dedica parte de sus bits para definir a qué red se quiere alguien dirigir,

y otra parte para indicar a qué dispositivo (host) dentro de esa red va a enviar la información. El número de bits dedicado a la red y al host varía de unos casos a otros.

Para identificar redes y hosts se utilizan máscaras. Estas se usan para diferenciar los bits que identifican ambos elementos de la dirección IP. Para ello se hace una operación AND entre los bits de la dirección y los de la máscara. Cuando en la máscara se tiene un 1, el bit correspondiente de la dirección IP identifica a la red, y por el contrario si es un 0 se identifica al host.



### Ejemplo

---

Si se tiene la siguiente dirección IP: 192.168.234.123; y la máscara 255.255.255.0. ¿Cuál sería la red y cuál el host al referenciado?

En este caso 255 representado en binario es 11111111. Al aplicarle la máscara a la dirección IP, se tiene como dirección de red: 192.168.234.0 y como dirección del host dentro de esa red 123.

---

El número de ceros de la máscara aplicados a la dirección IP representan el número de ordenadores que se puede tener dentro de la red. En el ejemplo mostrado anteriormente hay 8 bits para representar direcciones de host. Por lo tanto, se tienen  $2^8$  posibilidades, o sea = 256 hosts. Sin embargo, hay que quitar una posibilidad que es el representado por el 0 porque identifica a la red y el 255 que representa a la multidifusión. Quedarían 254 direcciones IP a utilizar.

La máscara se puede modificar para lograr subredes más pequeñas dentro de la red inicial. Esto puede ser interesante para dividirla por equipos que tienen mayor comunicación, evitando conflictos innecesarios.



### Ejemplo

---

Se podría definir la máscara 255.255.255.192 (FF.FF.FF.C0 o 11111111.11111111.11111111.11000000), que aplicada a las direcciones IP dividiría la red en 4 subredes.

Los 26 primeros bits identifican a la red y los 6 últimos bits, a los dispositivos dentro de esa red. Por lo tanto, podría tener  $2^2 = 4$  subredes y  $2^6 = 64$  hosts por red.

La dirección IP 192.168.12.75 AND 255.255.255.192 indicaría que pertenece a la subred 1 (01).

La dirección IP 192.168.12.204 AND 255.255.255.192 indicaría que pertenece a la subred 3 (11).

---

Existen cinco clases diferentes de direcciones IP, dependiendo de los valores de sus bits de mayor peso:

- **Clase A:** si el bit de mayor peso es 0, la máscara aplicada tendrá un prefijo de 8 bits a 0. Por lo tanto se tendría 8 bits para redes y 24 para hosts dentro de las redes. Traducido a decimal: direcciones cuyo primer octeto es menor de 128 su máscara sería 255.0.0.0.
- **Clase B:** si los dos primeros bits son 10, entonces la máscara por defecto tendrá un prefijo de 16 bits -16 primeros bits para redes y 16 últimos bits para hosts-. Traducido a decimal: direcciones cuyo primer octeto está entre 128 y 191 su máscara sería 255.255.0.0.
- **Clase C:** si los tres primeros son 110, la máscara por defecto tendrá un prefijo de 24 bits -24 bits para redes y 8 para hosts-. Esta dirección es la representada por el ejemplo anterior. Traducido a decimal: direcciones cuyo primer octeto está entre 192 y 223 su máscara sería 255.255.255.0.
- **Clase D:** si tenemos los primeros bits como 1110, entonces se trata de una dirección multicast. No representa la dirección de un equipo sino de un grupo de equipos a los que se quiere enviar la información. Traducido a decimal: direcciones cuyo primer octeto está entre 224 y 239 su máscara sería 255.255.255.255.
- **Clase E:** si los 4 primeros bits son 1111 se está ante direcciones que se utilizan para experimentación. Traducido a decimal sería desde 240 a 255.



### Aplicación práctica

---

**Se hace un estudio del tráfico de red en los equipos de una empresa y se detectan que algunos ordenadores usan frecuentemente ciertos servidores e impresoras y raramente otros dispositivos. Las IPs de los dispositivos y los grupos que se han decidido formar son:**

#### Grupo 1:

- | Impresora 1: 192.168.15.25
  - | Servidor 1: 192.168.15.80
  - | Ordenador 1: 192.168.1.10
  - | Ordenador 2: 192.168.1.11
  - | Ordenador 3: 192.168.1.12
  - | Ordenador 4: 192.168.1.14
-

## Grupo 2

- I Portátil 1: 192.168.15.140
- I Portátil 2: 192.168.15.141
- I Servidor 2: 192.168.15.150

**¿Qué máscara sería necesaria para crear esas dos subredes?**

### SOLUCIÓN

Se observa que la red es común a todos los hosts, 192.168.15.0. Por lo tanto, ya se tiene la primera parte de la máscara, que sería 255.255.255.

Ahora se tendría que dividir la parte de la máscara que corresponde a los hosts en 2, pues se observa que los hosts del grupo 1 están por debajo de 128 y los del grupo 2, por encima de 128. Por lo tanto, se pondría un 1 como el primer bit del último octeto.

De esta forma la máscara quedaría 255.255.255.128.

---

## 5.3. El nivel de transporte. Protocolos TCP y UDP. Otros protocolos de uso común

Dos de los protocolos más utilizados en la capa de transporte son **TCP** (Protocolo de Control de Transmisión) y **UDP** (Protocolo de Datagramas de Usuario). Son protocolos que permiten el envío de paquetes entre aplicaciones de ordenadores conectados en red. Sin embargo, hay diferencias importantes entre ellos.

### TCP

Este protocolo es orientado a conexión, lo que indica que antes de enviar cualquier paquete a través de la red se tiene que haber realizado la conexión con el equipo remoto. Una vez establecida la ruta por la cual se envían los datos, permanece fija hasta que haya finalizado la transmisión. Si en algún momento alguno de los enruteadores cae por cualquier motivo, la comunicación se interrumpe y se tiene que volver a iniciar. Por lo tanto, el procedimiento en una comunicación TCP sería la siguiente:

- Conexión con el equipo remoto.
- Transmisión de la información.
- Desconexión.

Este protocolo dispone de mecanismos que aseguran la confiabilidad en la entrega de mensajes, división de los mensajes en datagramas, seguimiento (secuencia) de los paquetes enviados, multiplexación de datos y detección de errores por medio de checksums.

Servicios como correo electrónico, envío de ficheros o el acceso remoto a otros equipos utilizan este protocolo. Y lo usan porque se necesita asegurar que los bloques de caracteres enviados no se pierdan por el camino, o que la conexión con un equipo remoto que controla un servicio crítico no reciba ruidos que provoquen un mal funcionamiento.

## UDP

Este otro protocolo no está orientado a conexión y se utiliza principalmente en aquellos servicios donde no es tan importante que lleguen todos los paquetes enviados como la velocidad de transmisión de estos. Un ejemplo claro de esto puede ser la transmisión de voz o vídeo en tiempo real.

Como es un protocolo que no está orientado a conexión, en el mismo paquete va incluida toda la información necesaria para que llegue a su destino. Una ventaja con respecto a TCP es que si una de las rutas elegidas cae porque se rompe algún router, el paquete es capaz de buscar alternativas para llegar a su destino.

Sin embargo, este protocolo presenta el inconveniente de que no asegura que los paquetes lleguen o que si llegan lo hagan en orden, con lo cual los procesos que reciban esta información tendrán una carga de computación adicional para recomponer los datagramas recibidos.

## Otros protocolos de uso común

**Otros protocolos** del servicio de transporte son:

- **RTP (Real Time Transport Protocol)**: usado para la transmisión de gran cantidad de información en tiempo real.
- **SCTP (Stream Control Transmission Protocol)**: protocolo orientado al mensaje.
- **SPX (Sequence Packet Protocol)**: similar al TCP. Se utiliza principalmente para aplicaciones cliente/servidor.
- **SCTP (Stream Control Transmission Protocol)**: Protocolo orientado a la conexión con los mismos servicios que TCP, añadiéndose que admite conexiones entre sistemas de host múltiple (con más de una dirección).



## Recuerde

Tanto el protocolo TCP como el UDP utilizan un esquema de direccionamiento idéntico que se compone de una dirección IP que sirve para identificar el dispositivo de red con el que se quiere comunicar y un número de puerto.

### 5.4. Puertos

Los puertos son la puerta de entrada/salida de un ordenador cuando se envían/reciben segmentos TCP o datagramas UDP.

El número de puerto tiene un tamaño de 16 bits, y hay que estudiarlos por pares: el primer número identifica el puerto origen del ordenador que envía la información y el segundo número, el puerto destino del ordenador al que va dirigido. Esta información va incluida en la cabecera de los paquetes enviados.

Como ejemplo, para acceder a una página web desde el navegador se podría abrir una conexión desde el puerto 1048 del cliente y accedería al puerto 80 del servidor, que es el encargado de suministrar la información de las páginas web.

La **organización IANA** es la encargada de tener actualizados los números asignados a los servicios de red suministrados, y se puede consultar en la página <<http://www.iana.org/protocols/>>.

Se puede tener un mismo número de puerto para TCP y para UDP, pues las comunicaciones entre procesos se producen con las combinaciones protocolonúmero de puerto.

En la siguiente tabla se muestra la lista de puertos más conocidos:

Aplicación	Puerto TCP	Puerto UDP
FTP	<b>21</b>	21
Telnet	<b>23</b>	23
SMTP	25	25
DNS	53	53
TFTP	69	<b>69</b>
http	<b>80</b>	80
POP3	<b>110</b>	100
NetBios	<b>139</b>	139
SNMP	161	161

En negrita se indica el protocolo usado con mayor frecuencia.

Al disponer de 16 bits para la definición de los puertos, esto permite tener un rango que va desde 0 a 65535. Sin embargo, no todos ellos se pueden utilizar de manera libre.

Existen algunas normas:

- Los puertos del 0 al 1023 son los **puertos conocidos** o reservados. Estos están preasignados.
- Los puertos del 1024 al 49151 son los **puertos registrados**, y son de libre uso. Estos puertos son los que se pueden programar.
- Los puertos del 49152 al 65535 son los **puertos dinámicos y/o privados**. Son temporales y los suelen utilizar los clientes para conectar con el servidor.

Es muy importante tener ciertos puertos abiertos para algunas aplicaciones. Si no fuera así la comunicación no sería posible y, por lo tanto, la aplicación no funcionaría. Por ejemplo, para poder utilizar el sistema de mensajería *Messenger* hay que tener abierto el puerto 1863.

### Asignación dinámica de puertos

Los puertos conocidos facilitan la labor de conectar con aplicaciones, pues se conoce al puerto al que se tiene que conectar para obtener su servicio. Sin embargo,

como se indicó antes, también se necesita un número de puerto origen para la conexión por pares. Este número se elige de manera aleatoria y no puede ser inferior a 1024.

Esta asignación permite que un mismo servicio pueda ser dado por puertos diferentes. De esta manera se puede tener dos navegadores abiertos, y que uno de ellos utilice el puerto 1051 y otro el puerto 1052.

Durante el proceso de conexión entre equipo origen y destino se intercambia la información de los puertos que se van a usar.

La combinación de dirección IP junto al puerto conforman lo que se llama *socket*. El *socket* identifica un servicio de red único. Para el ejemplo anterior, si el equipo cliente tiene la dirección IP 192.168.1.124 y se comunica a través del puerto 1051, el *socket* sería 192.168.1.124:1051.



### Actividades

11. Averigüe qué comando del sistema operativo muestra los puertos que se tienen abiertos en el ordenador. Ejecútelo e interprete la información que suministra.
12. Señale si cree usted que la asignación dinámica de puertos podría hacerse del lado del servidor. Razone la respuesta.

## 5.5. Servicios de red básicos

Los servicios de red básicos identifican el conjunto de servicios mínimos necesarios para trabajar en red. La mayoría de ellos actúan en el nivel de aplicación. Entre ellos cabe destacar los descritos a continuación.

### DHCP (Dynamic Host Configuration Protocol)

Este protocolo permite a un cliente de red obtener de manera automática su configuración dentro de la red. Es un protocolo cliente-servidor, de ahí que se oiga hablar de servidor DHCP.

El servidor DHCP dispone de una lista de direcciones IPs dinámicas, que las va asignando a los clientes que se van conectando a la red. El servidor DHCP conoce en todo momento las estaciones que se han conectado, tiempo de conexión, las que se han desconectado, etc.

Una vez que una estación se desconecta de la red, esa dirección IP se libera para que pueda ser utilizada por otra estación de trabajo.

Este protocolo es muy útil pues libera al cliente de tener que estudiar qué direcciones IP están libres dentro de la red, para de manera manual asignársela, para poder operar en la red.

### **SNMP (Simple Network Management Protocol)**

Este protocolo facilita la administración de los elementos que componen una red, ayudando a resolver incidencias que surjan gracias a la información suministrada. Tres son los elementos que intervienen en el protocolo:

- **Los dispositivos administrados:** son los elementos que se quieren estudiar dentro de la red. Así se tendrán servidores, hubs, routers, etc. De estos elementos se quiere obtener información tal como configuración, tráfico, estadísticas, rendimiento, etc.
- **Los agentes:** software que se encuentra ubicado en los dispositivos administrados, y son los encargados de enviar la información solicitada en formato SNMP.
- **El sistema de administración de red:** sistema desde el cual se accede a toda la información suministrada por los agentes.

El puerto normalmente utilizado para este servicio es el 161 y se utiliza un protocolo UDP para el intercambio de información, para de esta manera molestar lo menos posible en el funcionamiento del dispositivo, pues se evitan mecanismos de conexión que sí se necesitarían en una comunicación TCP.

### **SMTP (Simple Mail Transfer Protocol)**

Este protocolo permite el envío de correos electrónicos. Se encarga de la transmisión del *e-mail* que ha redactado el cliente, hasta su destino. Para el caso de la recepción de correos se utilizan otros protocolos.

La función principal de este protocolo es la transmisión de correos entre servidores. El cliente lo envía a su servidor SMTP, y este es el encargado de conectar con el servidor del correo destino para hacerle la entrega.

Una característica singular de este protocolo es que no requiere autentificación, con lo cual cualquier cliente puede enviar correos, lo que en algunos casos provoca lo que se denomina **spam**. Algunos servidores SMTP empiezan a implementar mecanismos que minimicen este problema admitiendo solo a clientes conocidos. Los servidores que no ponen ninguna limitación son llamados *open relay*.

## **POP (Post Office Protocol)**

Este protocolo permite recuperar los correos que se reciben en el servidor SMTP. El cliente se conecta al servidor por TCP en un puerto determinado y a base de mensajes entre cliente-servidor se leen los correos recibidos. Este protocolo permite dejar o eliminar los correos traídos en local.

Entre las ventajas que muestra este protocolo destacan:

- Los mensajes se almacenan en local, por lo que siempre se tendrán accesibles, sin necesidad de conexión a Internet.
- Los adjuntos se abren más rápido pues se descargan junto con el mensaje.
- Debido a que los mensajes se descargan en local, el espacio solo se encuentra limitado al espacio del disco duro, a diferencia de otros protocolos que tienen limitado el espacio al hosting contratado.

Entre sus inconvenientes se pueden citar:

- Posibilidad de infección por virus debido a que los mensajes se descargan en su totalidad al PC.
- Si hay algún problema con el PC, y se estropea el disco duro, se perderán todos los mensajes descargados.

## **IMAP (Internet Message Access Protocol)**

Este protocolo también es para leer los correos electrónicos del servidor. A diferencia de POP, permite acceder a los correos desde el lado del servidor, sin tener que descargarlos. Trabaja *online* y permite un número más elevado de transacciones que POP.

Entre sus ventajas destacan:

- Es más flexible, pues se puede acceder desde cualquier navegador a la cuenta de correo.
- Los correos se encuentran respaldados en el servidor.

Y sus inconvenientes son:

- En cuanto a seguridad puede interesar que los mensajes no estén en un servidor ajeno, y se quiera tener en local.
- Si no se dispone de conexión no se podrán leer los correos.

## DNS (Domain Name System)

Este protocolo es el encargado de asociar nombres de dominio con direcciones IP. De esta manera, no se tiene que conocer la dirección IP de un servidor al que se quiere conectar sino su nombre de dominio. Esto se hace a través de bases de datos que relacionan dominio con dirección IP. También es el encargado de localizar los servidores de correo.

El acceso al servidor DNS es transparente para el cliente. Cuando se pone una URL en el navegador web se accede al servidor local DNS, por si se tiene en local. Si no es así, se conectaría con el servidor DNS remoto, que normalmente lo suministra el proveedor de servicios de Internet. Es posible modificar por medio de DHCP el servidor DNS que se quiera usar.

## FTP (File Transfer Protocol)

Este protocolo permite la transmisión de ficheros a través de la red. Utiliza el modelo cliente-servidor, con intercambio de mensajes para la sincronización. Normalmente los puertos utilizados son TCP 20-21.

El cliente FTP es el encargado de conectar con el servidor para bien subir o bajar ficheros. Este tipo de funcionamiento es muy común para el caso de estar creando una web, y subir documentación necesaria para dicha web.

Por el otro lado, está el servidor FTP, que es el encargado de enviar o recibir los ficheros solicitados.

Uno de los inconvenientes del FTP es que, como se busca la máxima velocidad, no tiene mecanismos de encriptación, con lo cual la información viaja abierta y puede ser fruto de ataques.

La transmisión en FTP puede ser tipo ASCII o tipo binario, según si la información viaja codificada en caracteres o en binario.

Los mensajes enviados por el cliente e interpretados por el servidor permiten la creación de directorios, ficheros, cambios de nombres, listado de ficheros, etc.

## HTTP (Hypertext Transfer Protocol)

Este protocolo permite acceder a las páginas web, y visualizarlas en el navegador del cliente. Es de tipo transaccional, donde el cliente solicita una página y el servidor se la suministra, enviándole la información en formato web.

## NFS (Network File System)

Este protocolo permite un sistema de archivos distribuido por la red. De esta forma se podrá acceder a ficheros a través de unidades como si se encontraran en local.

Utiliza también un modelo cliente-servidor, y libera al cliente de tener que asumir el almacenamiento de datos que por volumen y seguridad es mejor ubicarlo en un servidor.

A través de este protocolo también se pueden compartir dispositivos como impresoras, lectores DVD, unidades de cinta, etc.

Las operaciones que se pueden realizar son muy similares a las que se pueden hacer con ficheros en local. Así se tienen comandos como: *create, open, read, remove, rename, restorefh, savefh, secinfo, setattr, setclientid, verify* y *write*.

## RDP (Remote Desktop Connection)

Este servicio permite conectar a usuarios de redes externas en redes internas de manera segura, a través del puerto 3389. Para ello, este servicio utiliza una conexión entre una VPN y la aplicación Remote Desktop Connection (nativa en *Microsoft Windows* aunque deshabilitada por defecto, *rdesktop* en *Linux*, de forma general).

Este servicio se implementa a partir de una arquitectura cliente-servidor.

- **Servidor RDP.** Es el lísstenner del puerto TCP 3389 que atenderá las peticiones de los clientes.
- **Cliente RDP.** El que solicita las conexiones a la máquina remota en la que corre el servidor RDP.

Para habilitar este servicio desde *Windows*, se debe activar que el sistema permita conexiones de asistencia remota y habilitar el permiso en el *firewall*.

Para *Linux*, existe una implementación libre de un servidor RDP llamada XRD, basada en *rdesktop*. Respecto al cliente RDP, además del estándar ya mencionado *rdesktop*, existen varias implementaciones como *tsclient*, que es un terminal gráfico, *KRDC* (para *KDE*, en el paquete *KRDC*), *grdesktop* (para *GNOME*, en el paquete *grdesktop*), etc.



### Aplicación práctica

**Usted es el encargado de decidir si el acceso al correo electrónico de la empresa va a ser usando protocolo IMAP o POP.**

**¿Qué protocolo propondría si se busca seguridad, flexibilidad en el acceso e incluso compartir buzón de correo?**

## SOLUCIÓN

Sería más conveniente utilizar el protocolo IMAP por los siguientes motivos:

- | Seguridad. Los correos se quedan en el servidor, por lo que si se tiene algún problema en local estos aún se mantienen en este lugar.
  - | Flexibilidad en el acceso. El cliente de correo no está limitado a configurar cada vez que se conecte en diferentes dispositivos, pues el acceso es a través de un cliente web, y se podría acceder con una tablet o un smartphone.
  - | Compartir buzón de correo. Si se quiere disponer de una cuenta corporativa de empresa, con conocer el usuario y *password* que se solicita en web podrá acceder el personal autorizado a esa cuenta.
- 

## 6. Resumen

Los modelos cliente-servidor, de objetos distribuidos o basado en mensajes, permiten crear programas distribuidos haciendo uso de recursos *software* que se encuentran disponibles en la red. La mensajería es el método más simple y directo, y se usan pequeñas programaciones que no requieren de complejos algoritmos de funcionamiento. Por el contrario, otros sistemas como RPC o RMI permiten crear modelos más complejos, utilizados cuando el problema a resolver requiere de mayor nivel de servicios o de continuos cambios de especificaciones.

Las redes sobre las que se diseñan los programas distribuidos siguen una normalización en niveles, permitiendo dividir un problema en siete subproblemas. Cada uno de esos niveles o capas tienen funciones u objetivos bien definidos, siguiendo protocolos de funcionamiento. Cada nivel se encarga de enviar y recibir datos a su nivel inferior y superior, respectivamente.

El nivel físico especifica los aspectos más cercanos al *hardware* de la red, identificando detalles eléctricos, mecánicos y funcionales.

El nivel de enlace es responsable de la fiabilidad en la transferencia de los datos a través de circuitos eléctricos. Para ello divide la información en tramas que son enviadas a través del canal de comunicación.

El nivel de transporte es el encargado de transportar los paquetes de información y su principal objetivo es que lleguen a su destino libres de errores.