Tema 8:
La capa de transporte y de aplicación.

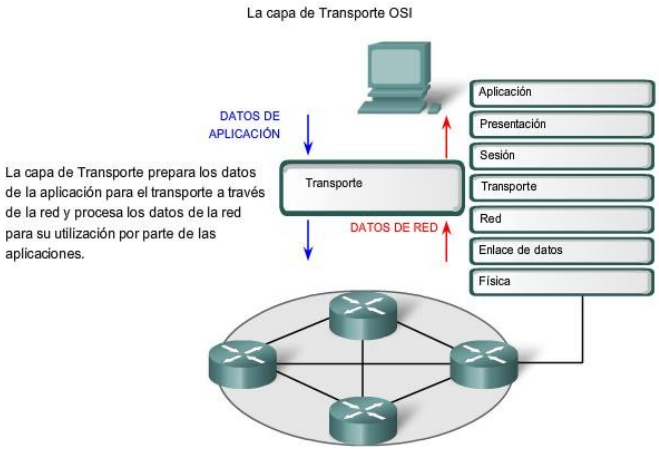
CONTENIDOS

1. Características de la capa de transporte.

1.1. Tareas de la capa de transporte.

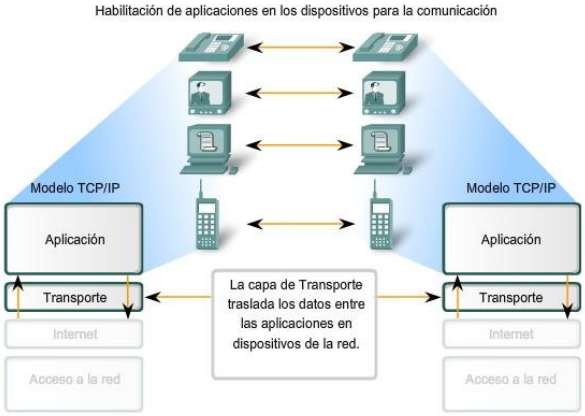
Transporte de información.

En las unidades anteriores hemos estudiado las capas orientadas a la red del modelo OSI que se encargaban principalmente de hacer llegar la información, de forma confiable, de un dispositivo a otro. La capa de transporte hace de intermediario entre las capas orientadas a la red y las orientadas a las aplicaciones. Se encarga de preparar los datos de la aplicación para el transporte a través de la red y de procesar los datos de la red para su utilización por parte de las aplicaciones.



Las tareas principales de la capa de transporte es transportar información entre aplicaciones que se ejecutan en diferentes máquinas. Para ello en la capa de transporte se sigue el siguiente procedimiento:

1) La capa de transporte **recoge** en el host de origen de la aplicación los datos a enviar.
2) **Fragmenta** los datos en segmento con la **longitud apropiada** para ser transferidos por la capa de red.
3) Los **etiqueta** de forma que se pueda identificar:
   La aplicación que los ha generado y la aplicación que ha de recibirlos.
4) Los entrega al nivel de red para que sea gestionado su transporte a través de la red.



1.2. Servicios principales de la capa de transporte.

Los protocolos de la capa de transporte se encargan de proveer algunos de estos servicios:

Control de flujo

Los hosts de la red cuentan con recursos limitados, como memoria o ancho de banda. Cuando la capa de Transporte advierte que estos recursos están sobrecargados, algunos protocolos pueden solicitar que la aplicación que envía reduzca la velocidad del flujo de datos. Esto se lleva a cabo en la capa de Transporte regulando la cantidad de datos que el origen transmite como grupo. El control del flujo puede prevenir la pérdida de datos en la red y evitar la necesidad de retransmisión.

Entrega confiable

Por varias razones, es posible que una sección de datos se corrompa o se pierda por completo a medida que se transmite a través de la red. La capa de Transporte puede asegurar que todas las secciones lleguen a destino al contar con el dispositivo de origen para volver a transmitir los datos que se hayan perdido.

Establecimiento de una sesión

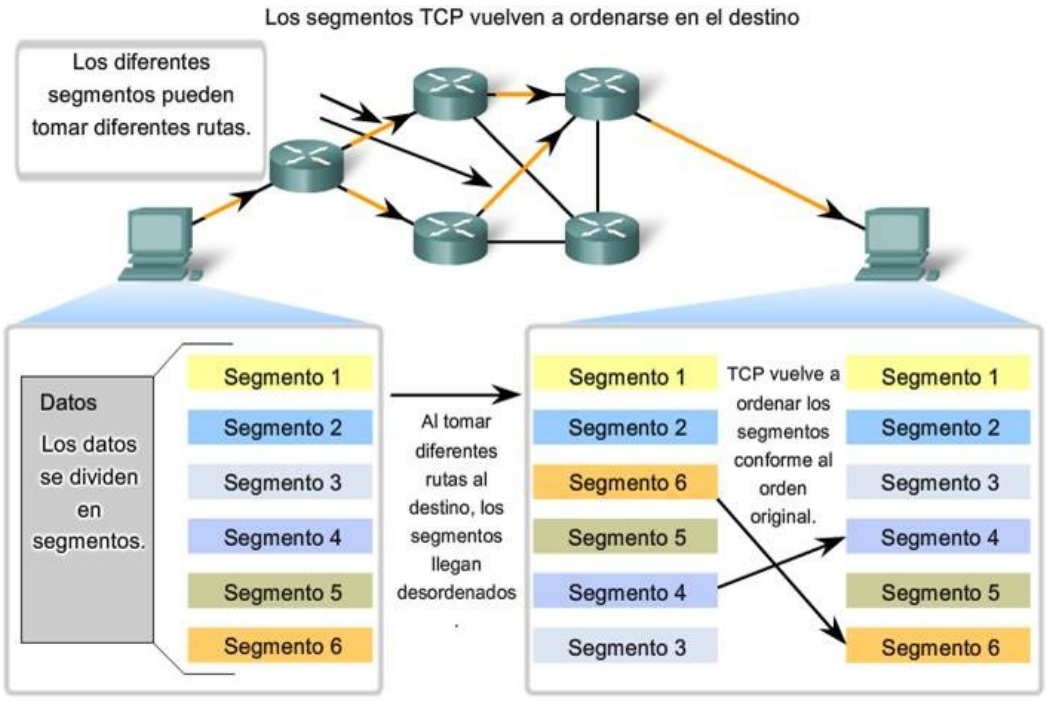
La capa de Transporte puede ofrecer conexión creando una sesión entre las aplicaciones. Estas conexiones preparan las aplicaciones para que se comuniquen entre sí antes de que se transmitan los datos. Dentro de estas sesiones, se pueden gestionar de cerca los datos para la comunicación entre dos aplicaciones.

Entrega en el mismo orden

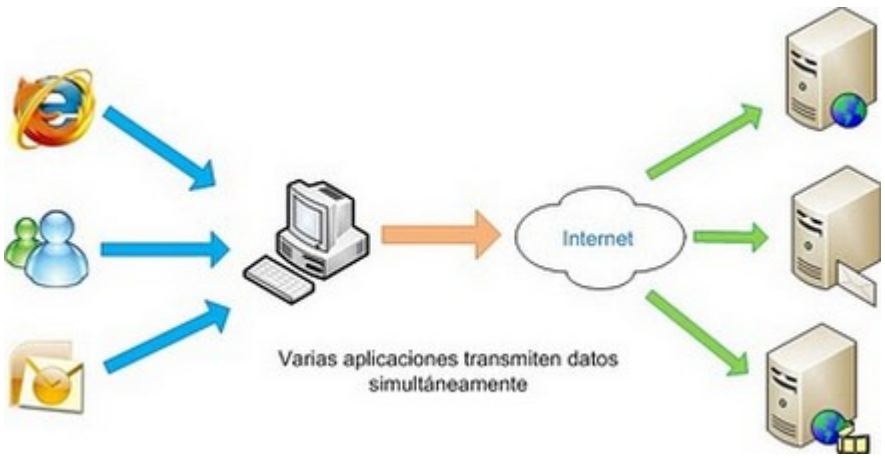
Ya que las redes proveen rutas múltiples que pueden poseer distintos tiempos de transmisión, los datos pueden llegar en el orden incorrecto. Al numerar y secuenciar los segmentos, la capa de Transporte puede asegurar que los mismos se reensamblen en el orden adecuado.

Segmentación y reensamblaje

La mayoría de las redes poseen una limitación en cuanto a la cantidad de datos que pueden incluirse en una única PDU (Unidad de datos del protocolo). La capa de Transporte divide los datos de aplicación en bloques de datos de un tamaño adecuado. En el destino, la capa de Transporte reensambla los datos antes de enviarlos a la aplicación o servicio de destino.



Multiplexación de conversaciones



Una misma máquina origen podrá establecer enlaces lógicos simultáneos con múltiples procesos de una o más máquinas destino. A esta característica se le llama **multiplexación de enlaces**.

1.3. Conceptos.

Aplicaciones y procesos

Una aplicación es un conjunto de instrucciones ejecutables almacenadas en un dispositivo. Cuando una aplicación se está ejecutando recibe el nombre de proceso. Algunas aplicaciones permiten que se ejecuten varias copias de sí misma y cada una de ellas es considerada un proceso distinto.

Puertos

En un equipo se ejecutan muchos procesos de forma simultánea, por lo que puede darse el caso de que dos procesos que están en diferentes equipos quieran comunicarse entre sí. En ese caso, la máquina origen no sólo ha de especificar cuál es la máquina destino de la comunicación, sino que también deberá especificar con que proceso de dicha máquina se quiere comunicar.

Los protocolos TCP/IP denominan a este identificador de aplicaciones **número de puerto** y utiliza **16 bits** para identificarlos.

A todos los procesos de software que requieran acceder a la red se les **asigna** un número de puerto exclusivo en ese host. Este número de puerto se utiliza en el encabezado de la capa de Transporte para indicar con qué aplicación está asociada esa sección de datos.

Tipos

Con los 16 bits que se utilizan para identificar los números de puerto podemos tener valores que van desde 0 hasta 65535.

Existen distintos tipos de números de puerto:

- **Puertos bien conocidos** (Números del 0 al 1023): son usados normalmente por el **sistema** o por **procesos con privilegios**. Las aplicaciones que usan este tipo de puertos son ejecutadas como servidores y se quedan a la escucha de conexiones.

- **Puertos Registrados** (Números 1024 al 49151): estos números de puertos están asignados a procesos o aplicaciones del **usuario**. Estos procesos son principalmente aplicaciones individuales que el usuario elige instalar en lugar de aplicaciones comunes que recibiría un puerto bien conocido. Cuando no se utilizan para un recurso del servidor, estos puertos también pueden utilizarse si una aplicación los selecciona, de manera dinámica, como puerto de origen.

- **Puertos dinámicos o privados** (Números del 49152 al 65535): también conocidos como **puertos efímeros**, suelen asignarse de manera dinámica a aplicaciones de **cliente** cuando se **inicia una conexión**. No es muy común que un un servicio este a la escucha utilizando un puerto dinámico o privado (aunque algunos programas que comparten archivos punto a punto lo hacen).

Números de puerto

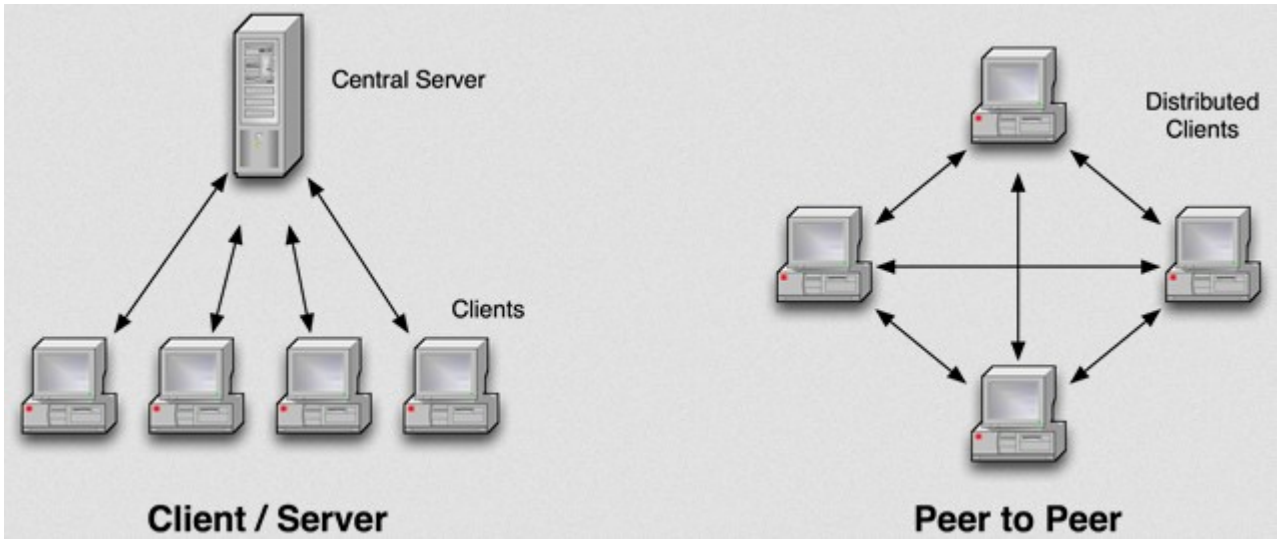
Rango de números de puerto	Grupo de puertos
De 0 a 1023	Puertos bien conocidos (Contacto)
De 1024 a 49151	Puertos registrados
De 49152 a 65535	Puertos privados y/o dinámicos

Puertos TCP registrados:
1863 MSN Messenger
8008 HTTP alternativo
8080 HTTP alternativo

Puertos TCP bien conocidos:
21 FTP
23 Telnet
25 SMTP
80 HTTP
110 POP3
194 Internet Relay Chat (IRC)
443 HTTP seguro (HTTPS)

Comunicación cliente/servidor y P2P

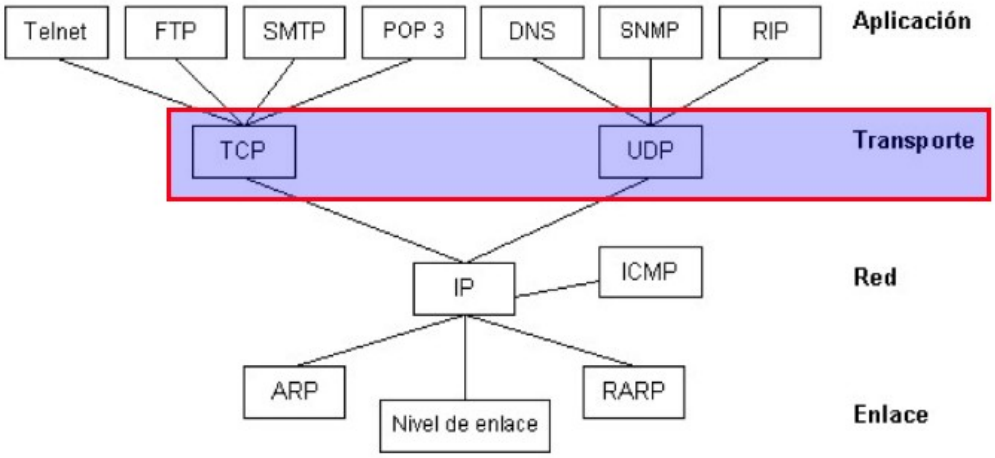
El modelo de comunicación **cliente/servidor** es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta.



En modelo de comunicación punto a punto (P2P, peer to peer), los dispositivos en red actúan como iguales, o pares entre sí. Cada dispositivo puede tomar el rol de cliente o la función de servidor. En un momento, el dispositivo A, por ejemplo, puede hacer una petición de un dato del dispositivo B, y este es el que le responde enviando el dato al dispositivo A. El dispositivo A funciona como cliente, mientras que B funciona como servidor. Un momento después los dispositivos A y B pueden revertir los roles: B, como cliente, hace una solicitud a A, y A, como servidor, responde a la solicitud de B. A y B permanecen en una relación reciproca o par entre ellos.

1.4. Principales protocolos de la capa de transporte.

Los principales protocolos de la capa de transporte en la pila de protocolos TCP/IP son **UDP** y **TCP**.



2. El protocolo UDP (User Datagram Protocol)

Es el protocolo de transporte más sencillo de la pila de protocolos TCP/IP.

2.1. Características.

Función principal del protocolo.

Permitir la comunicación entre dos sistemas en los que la velocidad es más importante que la fiabilidad.

En transferencias de video o audio en tiempo real, unos cuantos paquetes perdidos son tolerables. Recuperar paquetes crea una excesiva saturación que reduce el rendimiento.

Datagramas UDP

La **PDU** del protocolo UDP recibe el nombre de **datagrama**.

Características generales

Cada datagrama se envía de forma independiente del resto.

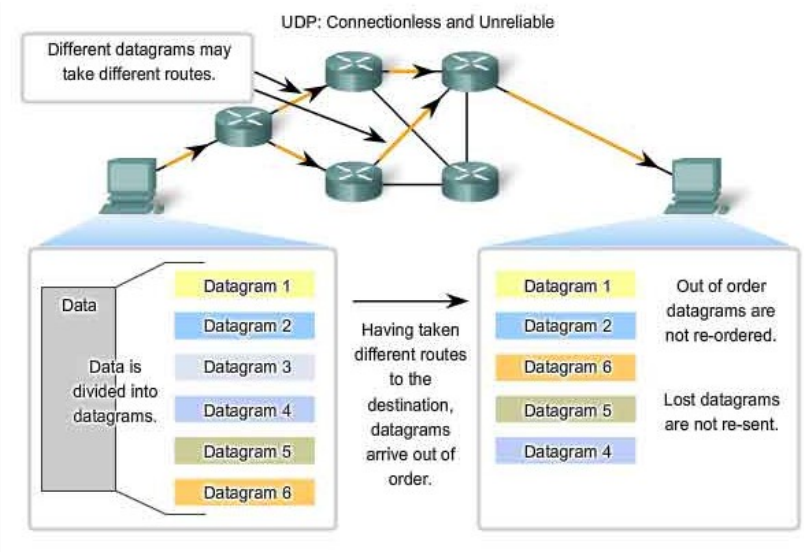
UDP no ofrece ningún mecanismo que permita garantizar al remitente que todos los datos hayan llegado al destino.

Los datagramas se envían sin que en el datagrama esté identificado su orden, por lo que no se puede identificar en el destino ni el orden de los datagramas ni si alguno ha llegado duplicado.

No hay mecanismos que eviten congestiones.

No dispone de ningún mecanismo de fragmentación incluido en el protocolo, sino que es la propia aplicación la que debe dividir su información en fragmentos de tamaño adecuado para que sean encapsulados mediante datagramas UDP.

En el encabezado del datagrama se envía una suma de comprobación (checksum) que permite averiguar en el destino si ha habido errores en la transmisión desde el origen.



Servicios que ofrece el protocolo

Multiplexación de envíos entre procesos por encima de un mismo enlace de red, identificando en cada máquina cada proceso con un número de puerto.

Detección de errores en la transmisión entre extremos para cada datagrama.

Consecuencias

Debido a su extrema sencillez, el protocolo UDP permite elevadas velocidades de transferencia, pero no ofrece garantías de la entrega.

Es un protocolo útil para aplicaciones que necesitan funcionar en tiempo real o con bajo tiempo de respuesta, siempre y cuando se puedan permitir eliminar los datagramas que no lleguen o que no lo hagan a tiempo.

Servicios adicionales

Cualquier otra funcionalidad que se precise deberá implementarla la aplicación que haga uso del protocolo UDP, ya que el protocolo no lo hace, como

por ejemplo la reordenación de datagramas, identificación de duplicados, solicitud de retransmisión de los datagramas que no hayan llegado, etc. Sin embargo, si es necesaria una elevada **fiabilidad** o **control de flujo** para evitar congestiones, es más conveniente usar TCP que UDP.

2.2. Transmisión de datos mediante UDP.

En el **origen** la aplicación divide la información en fragmentos lo suficientemente pequeños para ser enviados por UDP y los pasa a la capa de transporte, que los encapsula en datagramas UDP y los envía a través de la capa de red.

En el **destino**, cuando se recibe un datagrama, se comprueba el checksum. Si es incorrecto, el datagrama se descarta y, en caso contrario, se desencapsula la información y se pasa al proceso que corresponda según el número de puerto.

2.3. Aplicaciones que usan UDP.

- **Protocolo DNS**: permite averiguar la dirección IP de un recurso mediante su nombre DNS o a la inversa. Es importante obtener una respuesta rápida; en caso de que no se obtuviera, se vuelve a solicitar.

- **Protocolo DHCP**: permiete obtener de forma automática la configuración IP de un dispositivo desde un servidor de DHCP. Es importante que se haga de forma rápida y en caso de que no llegue se vuelve a solicitar.
- **Protocolos de streamming de voz y vídeo**: algunos protocolos de difusión de voz y vídeo en tiempo real requieren una gran velocidad de transmisión y además se diseñan para permitir que se pierdan algunos datos, por ello la mayoría utilizan UDP.

- **Juegos online** los juegos online en tiempo real entre varios jugadores requieren alta velocidad de respuesta en la interacción de los jugadores por eso suelen utilizar UDP.

- **Protocolos de difusión y multidifusión de datos**: permiten transmitir datos mediante tramas o paquetes broadcast o multicast y utilizan UDP. Un ejemplo de este tipo de aplicaciones es en el clonado por red de imágenes de disco duro.

2.4. Formato de encabezado de datagrama UDP.

puerto de origen (16 bits);	puerto de destino (16 bits);
longitud total (16 bits);	suma de comprobación del encabezado (16 bits);
datos (longitud variable).	

3. El protocolo TCP (Transmission Control Protocol)

TCP es el protocolo de la capa de transporte más popular de la pila de protocolos TCP/IP. Es mucho más complejo que el protocolo UDP.

3.1. Características.

Función principal del protocolo

Prestar un servicio que permita que la comunicación entre dos sistemas se efectúe libre de errores, sin pérdidas y con seguridad.

Segmentos TCP

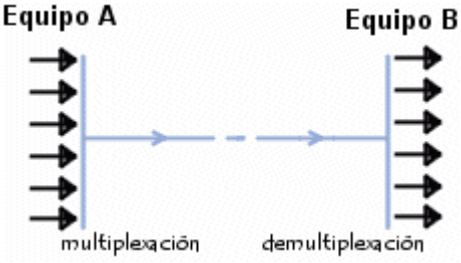
La **PDU** del protocolo TCP recibe el nombre de **segmento**

Orientado a conexión

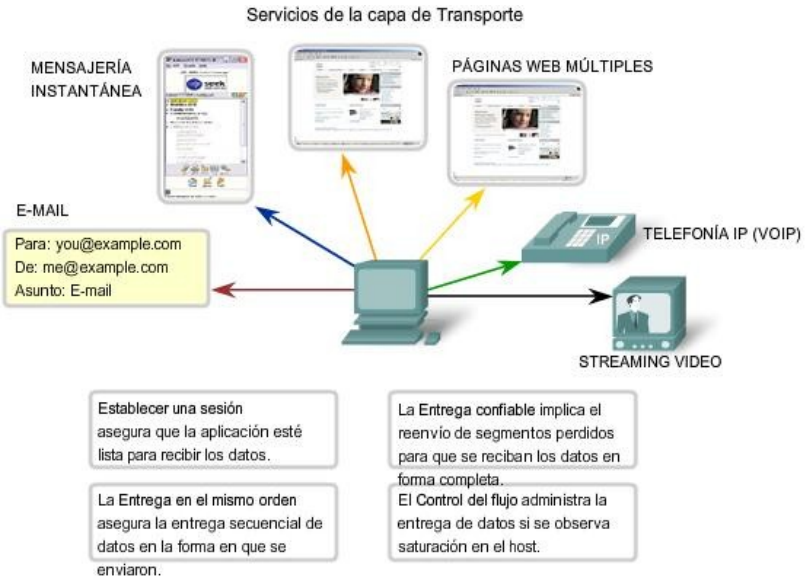
Cuando se quieren enviar datos de un proceso de un host origen a un proceso de un host de destino, TCP establece previamente una conexión lógica entre ambos a través de la cual enviará toda la información. Una vez finalizado el proceso de comunicación, si la conexión ya no es necesaria, se destruye.

Servicios que ofrece el protocolo

- Establecimiento y gestión de **conexiones lógicas** a través de las cuales se envían datos entre procesos.
- **Multiplexación de conexiones** entre procesos por encima de un mismo enlace de red, identificando en cada máquina cada proceso con un número de puerto. Podemos transmitir datos desde diversas aplicaciones en la misma línea.



- **Fragmentación** en el host origen de los datos provenientes del proceso en segmentos de tamaño adecuado para viajar a través del protocolo de red que se utilice.
- **Reordenación** en el host de destino de los segmentos procedentes de la fragmentación en el origen para reconstruir el bloque de datos originales a partir de estos segmentos.
- **Detección de errores** de transmisión en los segmentos.
- **Entrega fiable** a pesar de que usa el protocolo IP, que no incluye ninguna monitorización de la entrega de datagramas, el protocolo TCP tiene un sistema de acuse de recibo que permite al cliente y al servidor garantizar la recepción mutua de datos. **El host de destino informa al host origen de cada segmento que ha recibido correctamente**.
- **Retransmisión automática** de aquellos segmentos que no han llegado o han llegado con errores.
- **Eliminación** de los segmentos que han llegado duplicados.
- **Control de flujo**, permite adaptar la velocidad de envío para evitar las congestiones. Si el transmisor está desbordando el buffer del receptor por transmitir demasiado rápido, el receptor descarta paquetes. Los ACK fallidos que llegan al transmisor le alertan para bajar la velocidad de transferencia o dejar de transmitir.



Consecuencias

El protocolo TCP garantiza que el mensaje podrá ser reconstruido en el destino, ya que se reordenarán los segmentos, se eliminarán los duplicados y se retransmitirán aquellos segmentos que no hayan llegado o hayan llegado con errores. Además, si recibe confirmación por parte del receptor de que han llegado todos los segmentos, podemos asegurarnos de que el mensaje ha llegado bien.

Esta fiabilidad y la posibilidad de llevar a cabo un control de flujo tiene un alto coste en eficiencia, por lo que comparado con UDP, TCP es un protocolo mucho más lento. Por tanto es una buena opción cuando se requiera fiabilidad en la entrega, pero no para aplicaciones que requieran comunicarse en tiempo real.

3.2. Estados de una conexión.

Las conexiones TCP comunican entre sí a dos procesos en dos equipos. Cada proceso es identificado por un **número de puerto** y durante la vida del mismo, su puerto puede encontrarse en diferentes **estados**.

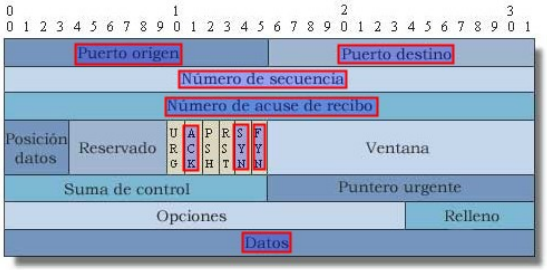
En el **modelo de comunicación** más utilizado por el protocolo TCP es **cliente/servidor**, en este modelo:

- Inicialmente los **servidores** suelen tener su puerto a la **escucha** y los **clientes** su puerto **cerrado** y sólo cambia de estado cuando quiere comunicarse con el servidor.

- Cuando se **establece una conexión TCP** entre dos dispositivos, después de pasar por una serie de estados intermedios, los puertos alcanzan el estado de **conexión establecida** (*Established*). Mientras se encuentran en ese estado ambos procesos podrán **intercambiar información** a través de una misma **conexión TCP**.

- Cuando la conexión deja de ser necesaria, se inicia un **proceso de finalización de la conexión** que, después de pasar también por una serie de estados intermedios, alcanza el de **espera** (*Time-Wait*), en el que espera a que pueda llegar algún mensaje que quedó rezagado. Pasado cierto tiempo, el puerto se cierra de nuevo y pasa al estado **cerrado** (*Closed*) en el cliente y **escucha** (*Listen*) en el servidor, ya que estos normalmente no cierran nunca el puerto, para poder así, recibir nuevas conexiones.

3.3. Campos segmento TCP.



3.4. Transmisión de datos mediante TCP.

Números de secuencia

Durante el establecimiento de conexión TCP, los **números iniciales de secuencia** son intercambiados entre las dos entidades TCP que intervienen en la comunicación. Estos números de secuencia son usados para identificar los datos dentro del flujo de bytes, y poder identificar (y contar) los bytes de los datos de la aplicación.

Los segmentos TCP pueden incluir dos tipos de números de secuencia, el **número de secuencia** y al número de reconocimiento o **número de ACK**. Un emisor TCP se refiere a su propio número de secuencia cuando envía información, mientras que con el número de ACK se refiere al número de secuencia del receptor.

A través del uso de números de secuencia y ACK, TCP puede **pasar** los **segmentos recibidos** en el **orden** correcto dentro del flujo de bytes a la **aplicación receptora**. Los números de secuencia son de **32 bits**, la secuencia **vuelve a cero** cuando llega al máximo valor posible con los 32 bits disponibles (2^{32}-1).

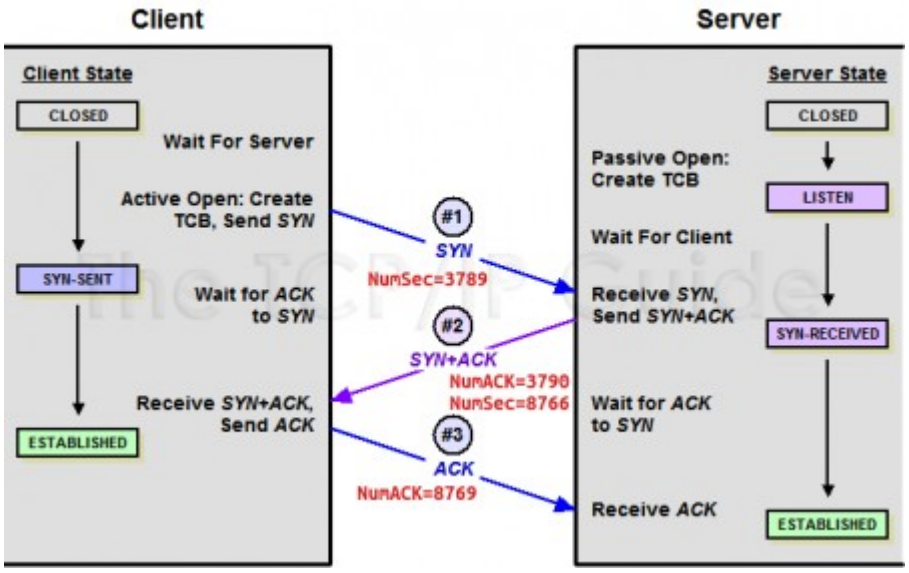
Fases en una comunicación TCP

Vamos a ver de forma breve lo que ocurre durante las diferentes etapas que se producen en la transmisión de datos del protocolo TCP.

1. Establecimiento de una conexión

La forma más frecuente de iniciar una conexión es la que se establece entre un cliente que se encuentra en estado *Closed* y un servidor en estado *Listen*.

El procedimiento requiere tres mensajes para el establecimiento de la conexión:



1. El cliente que desea comunicarse con el servidor envía un segmento TCP con un número de secuencia inicial (32 bits) y el **bit SYN** activado. El cliente pasa al estado *SYN-sent*.
2. El servidor recibe el segmento y envía a su vez otro de confirmación al cliente con los **bits ACK y SYN** activados, con número de secuencia inicial propio definido por el servidor y con número de reconocimiento (ACK) con valor el número de secuencia recibido del cliente + 1. El servidor pasa al estado *SYN-Received*
3. Cuando el cliente recibe la confirmación del servidor, le envía a su vez su propia confirmación mediante un segmento TCP con el bit ACK activado, con número de secuencia igual al del anterior segmento enviado + 1 y con el número de reconocimiento con valor el número de secuencia recibido del servidor + 1. El cliente pasa así al estado *Established*.
4. Cuando el servidor recibe la confirmación del cliente pasa también al estado *Established*

2. Fragmentación y entrega fiable de los segmentos

Los datos que se reciben de la aplicación suelen **fragmentarse** (se dividen en trozos) para que puedan caber en las PDU del nivel de red, que suelen ser paquetes IP. En el protocolo TCP la **fragmentación** de los datos en origen y su **reensamblado** en el destino la realiza en propio TCP.

Después de fragmentar los datos se **identifica** cada fragmento dándole un **número** para que pueda ser identificada y reordenado en el destino. El número que se le asigna se corresponde con el **número del byte inicial** de los datos incluidos en el PDU.

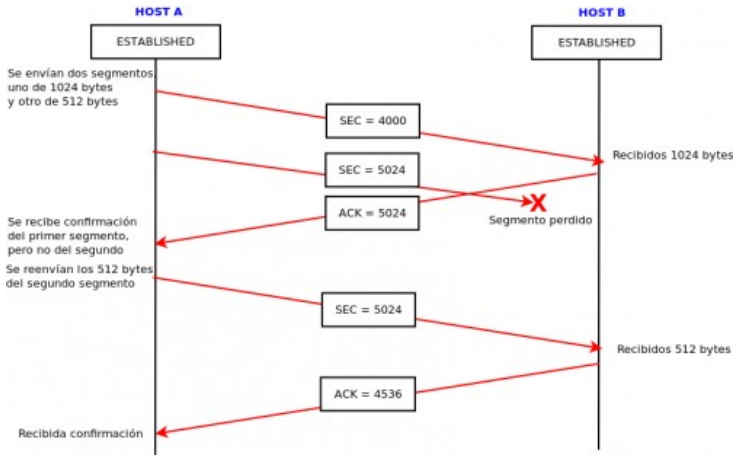
Cada fragmento viaja en un **segmento TCP diferente**. Para **identificar el segmento** se suma al número de secuencia actual el **número de byte que ocupa el dato**.

Cuando el **receptor recibe un segmento** correcto, almacena el fragmento de datos que contiene en su memoria junto con su **número de secuencia** para poder reordenar los fragmentos a medida que llegan. Si llega algún segmento **duplicado** lo identifica por el número de secuencia y lo **elimina**.

Recibido el segmento y **comprobado** que es correcto el **receptor confirma al emisor** que ha llegado enviando un **segmento con el bit de ACK activado** y con un **número de reconocimiento** (número ACK) cuyo valor es el **número de secuencia recibido + el número de bytes leídos**.

Si el emisor de un segmento **no recibe** la confirmación de su recepción en un **tiempo determinado**, vuelve a reenviar el segmento.

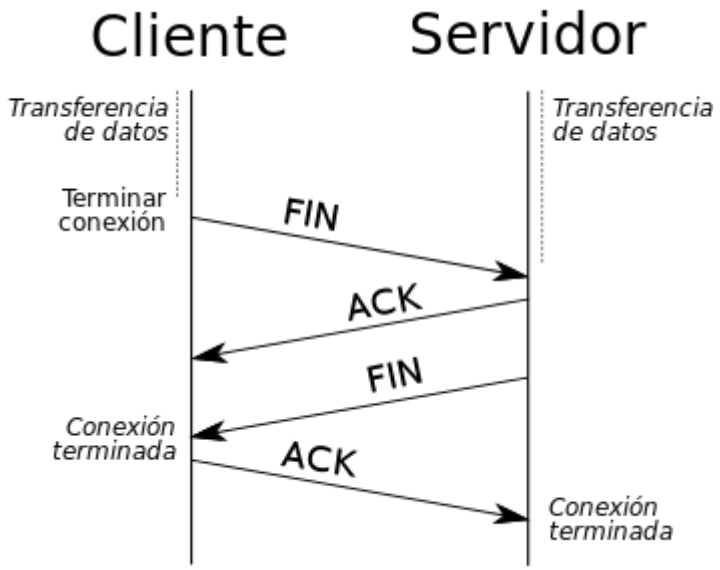
Hay mecanismos más complejos que mejoran la eficiencia de este procedimiento, pero van más allá de los contenidos de este curso.



En la imagen anterior hay un error. ¿Eres capaz de detectarlo?

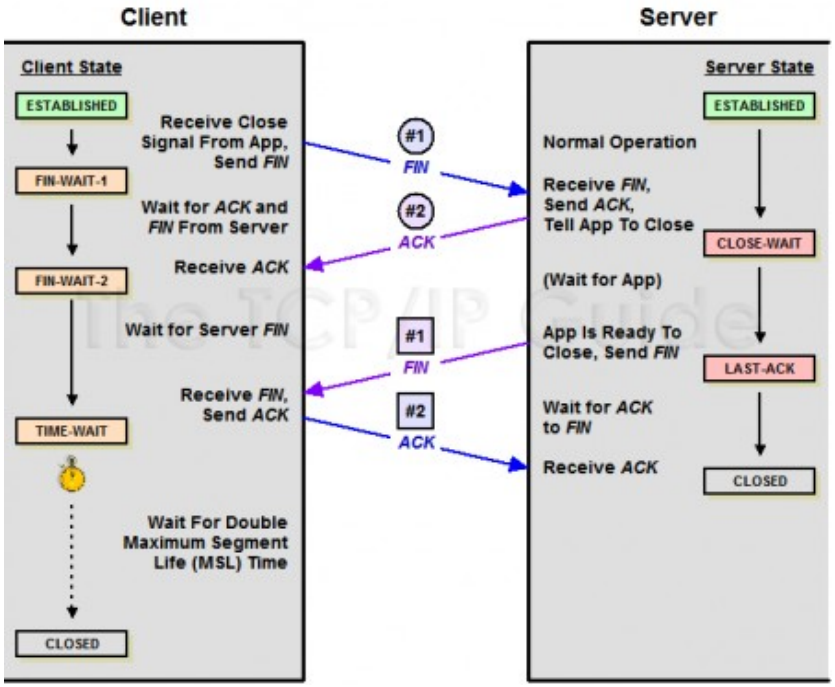
3. Finalización de la conexión

La fase de finalización de la conexión, normalmente utiliza una negociación en cuatro pasos, terminando la conexión desde cada lado independientemente.



Una conexión puede estar “medio abierta” en el caso de que uno de los lados la finalice pero el otro no. El lado que ha dado por finalizada la conexión no puede enviar más datos pero la otra parte si podrá.

El modo más habitual de finalizar una conexión consiste en que el cliente se lo indique al servidor mediante el envío de un segmento TCP con el **bit F IN** activado.



1. La aplicación cliente envía al protocolo la señal de que quiere cerrar la conexión. El cliente envía al servidor un segmento TCP con el **bit FIN** activado y número de secuencia el inicial + 1. El cliente pasa al estado *FIN-Wait-1* (a la espera de recibir confirmación de cierre.
2. El receptor, al recibir la petición responde con un segmento TCP con el **bit ACK** activado y número de reconocimiento igual al número de secuencia recibido + 1 y pasa al estado *Closed-Wait*.
3. El cliente recibe el ACK del servidor y queda a la espera de que el servidor le envíe un segmento de FIN. Pasa al estado *FIN-Wait-2*.
4. El servidor se queda a la espera de que la aplicación esté lista para cerrarse. Cuando lo esté envía la señal correspondiente al protocolo y este genera un segmento TCP con el **bit FIN** activado y número de secuencia actual del servidor + 1 y queda a la espera de recibir confirmación de su recepción en estado *Last-ACK*.
5. Cuando el cliente recibe la petición de cierre por parte del servidor responde con un segmento TCP con el **bit ACK** activado y número de reconocimiento igual al número de secuencia del segmento recibido +1. Pasa al estado *Time-Wait*, pasado un tiempo (para evitar que se pueda crear una nueva conexión con el mismo puerto) el cliente pasa a estado *Closed*
6. Cuando el servidor recibe el segmento TCP de ACK pasa a estado *Closed.*

3.5. Video – Comparación UDP y TCP.

https://www.youtube.com/watch?time_continue=18&v=Vdc8TCESIg8

4. La utilidad *netstat*

El comando netstat nos permite obtener estadísticas de la red. Está disponible tanto en máquinas Windows como en Unix/Linux. En función de los parámetros utilizados cambiará la información que se muestre como resultado.

Con netstat podremos ver, para el equipo en el que lo ejecutemos, tanto conexiones entrantes como salientes (TCP y UDP), información de las tablas de enrutamiento del host y estadísticas de la interfaz de red.

4.1. Parámetros comunes.

Los parámetros más comunes que podemos pasar a netstat son:

Opción	Descripción
-a	Muestra todas las conexiones y puertos a la escucha
-r	Muestra la tabla de enrutamieento del host
-i <iface>	Muestra estadísticas de red para la interfaz de red especificada
-n	No resuelve los puertos y direcciones, mostrándolo en formato numérico
-c	Visualiza el resultado de forma continua actualizándolo cada segundo
-p	Muestra el pid y el nombre del proceso de la aplicación local involucrada en la conexión
-l	Muestra los datos de los procesos que están a la escucha
-t	Muestra conexiones que utilizan el protocolo TCP
-u	Muestra conexiones que utilizan el protocolo UDP

Podemos combinar varias opciones en una ejecución del comando netstat. Así, por ejemplo, podríamos ejecutar:

```
$ netstat –cnlput
```

4.2. Columnas de información.

Al ejecutar *netstat* se muestran, entre otras, las siguientes columnas:

Columna	Descripción
Proto	Protocolo utilizado en la conexión
Dirección local	Dirección IP y puerto de origen de la conexión
Dirección remota	Dirección IP y puerto de origen de la conexión
Estado	Si la conexión es TCP muestra el estado en que se encuentra
PID/Program name	Número de identificación del proceso y aplicación que ejecuta el proceso origen de la comunicación

Ejemplo:

```
$ sudo netstat -antup
Conexiones activas de Internet (servidores y establecidos)
Proto  Recib Enviad Dirección local        Dirección remota       Estado      PID/Program name
tcp        0      0 127.0.0.1:631          0.0.0.0:*              ESCUCHAR    1236/cupsd
tcp        0      0 0.0.0.0:25             0.0.0.0:*              ESCUCHAR    1562/master
tcp        0      0 0.0.0.0:17500          0.0.0.0:*              ESCUCHAR    2787/dropbox
tcp        0      0 0.0.0.0:445            0.0.0.0:*              ESCUCHAR    1150/smbd
tcp        0      0 0.0.0.0:139            0.0.0.0:*              ESCUCHAR    1150/smbd
tcp        0      0 0.0.0.0:51413          0.0.0.0:*              ESCUCHAR    6445/transmission-g
tcp        0      0 192.168.10.2:51413     106.215.89.132:56284   SYN_RECV    -
tcp        0      0 192.168.10.2:51413     151.42.21.193:61288    SYN_RECV    -
tcp        0      0 192.168.10.2:51413     90.149.219.185:60511   SYN_RECV    -
tcp        0      0 192.168.10.2:51413     187.78.118.52:16643    SYN_RECV    -
tcp        0      0 127.0.1.1:53           0.0.0.0:*              ESCUCHAR    2320/dnsmasq
tcp        0      0 0.0.0.0:22             0.0.0.0:*              ESCUCHAR    1157/sshd
tcp        0      0 192.168.10.2:51413     85.65.254.49:3829      TIME_WAIT   -
tcp        0      0 192.168.10.2:51413     190.101.114.33:65334   TIME_WAIT   -
tcp        0      0 192.168.10.2:51413     95.234.148.50:53826    TIME_WAIT   -
tcp        1      0 192.168.10.2:57606     173.194.41.212:443     CLOSE_WAIT  3126/evolution-addr
```

4.3. Direcciones IP mostradas por netstat.

Las direcciones IP mostradas por netstat suelen ser de las siguientes categorías.

Dirección IP	Descripción
127.0.0.1	Esta dirección se refiere al host local o a el propio equipo
0.0.0.0	Una dirección global, lo que significa cualquiera Dirección
Dirección IP	La dirección del dispositivo local o remota que participa en la conexión

Para el caso de procesos servidores (Utilizan el protocolo TCP y están en estado escucha (Listen). En función de la dirección local:

127.0.0.1: indica que el proceso sólo puede recibir peticiones de conexión desde el propio equipo.

0.0.0.0: El proceso puede recibir peticiones de cualquier dirección IP que tenga configurada el equipo.

dirección IP: El proceso sólo puede recibir peticiones a través de la interfaz de red que tenga configurada dicha IP.

5. Redireccionamiento de puertos.

5.1. Características.

El protocolo NAT nos permite dar salida a Internet a diferentes equipos de nuestra red a partir de una o varias direcciones IP públicas.

El redireccionamiento de puertos o **Port Address Translation** (PAT) es una **ampliación de NAT** que nos permite dar acceso a los equipos de la red local (con direcciones IP privadas) desde Internet a través de una única IP pública.

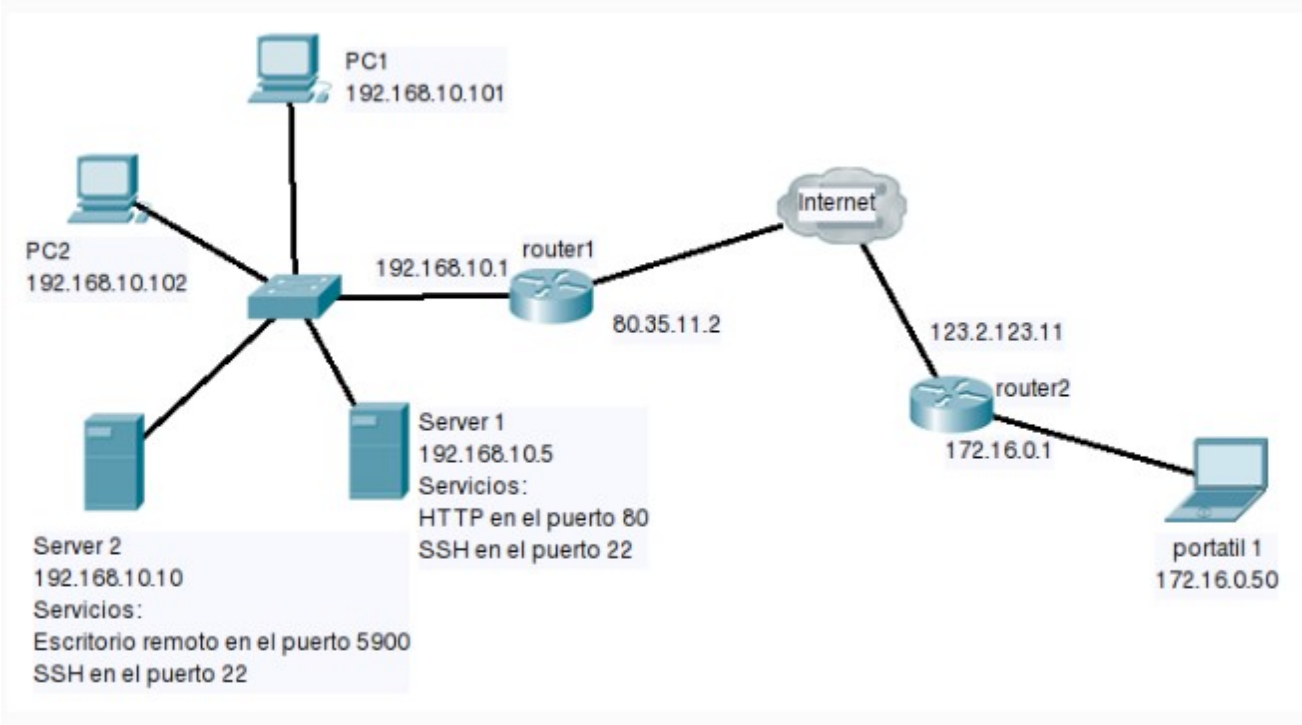
PAT no funciona dando acceso al equipo interno completo, sino que se lo da sólo a los puertos que lo necesitan. Así, con una dirección IP pública que dispone de 65536 puertos, podemos dar acceso a Internet a 65536 procesos distintos que pueden estar ubicados en diferentes máquinas de la red local.

Al igual que NAT, el PAT se suele ubicar en el router que da acceso a Internet y puede funcionar en los modos estático y dinámico:

Modo estático: en este modo, el administrador de la red asigna a vun servicio concreto de un equipo de la red local un número de puerto concreto de la dirección IP pública para que se pueda acceder al servicio desde Internet.

Modo dinámico: en este modo, los puertos de la dirección IP pública se asignan automáticamente a medida que se necesitan. Cuando ya no son necesarios se liberan, también de forma automática. Para configurar PAT dinámico se debe indicar que rango de hosts de la red privada se quiere dotar de acceso a Internet.

Ejemplo:



Queremos publicar en Internet los servicios HTTP y SSH del servidor1 (192.168.10.5) y el servicios de escritorio remoto VNC y SSH en el servidor2 (192.168.10.10). En este caso, en el router1, que es el router que da acceso a Internet a la red 192.168.10.0/24, podríamos configurar el PAT estático de la siguiente manera:

IP red local	Puerto interno	Protocolo	IP pública	Puerto externo
192.168.10.5	80	TCP	80.35.11.2	8080
192.168.10.5	22	TCP	80.35.11.2	2022
192.168.10.10	5900	TCP	80.35.11.2	59000
192.168.10.10	22	TCP	80.35.11.2	4022

Al hacerlo de esta forma:

Cualquier dispositivo con acceso a Internet podría acceder al servidor web (HTTP) del servidor 1 mediante la URL http://80.35.11.2:8080

Para acceder al servicio de escritorio remoto del servidor 2, desde Internet, abrimos un **programa cliente de escritorio remoto** y escribimos la dirección `80.35.11.2:59000`

De igual forma, para acceder por ssh al servidor1 lo podemos hacer desde Internet abriendo un terminal y ejecutando `ssh usuario@80.35.11.2 -p 2022` y

Para acceder al servidor 2 por ssh, lo haríamos ejecutando `ssh usuario@80.35.11.2 -p 4022.`

5.2. Conclusiones.

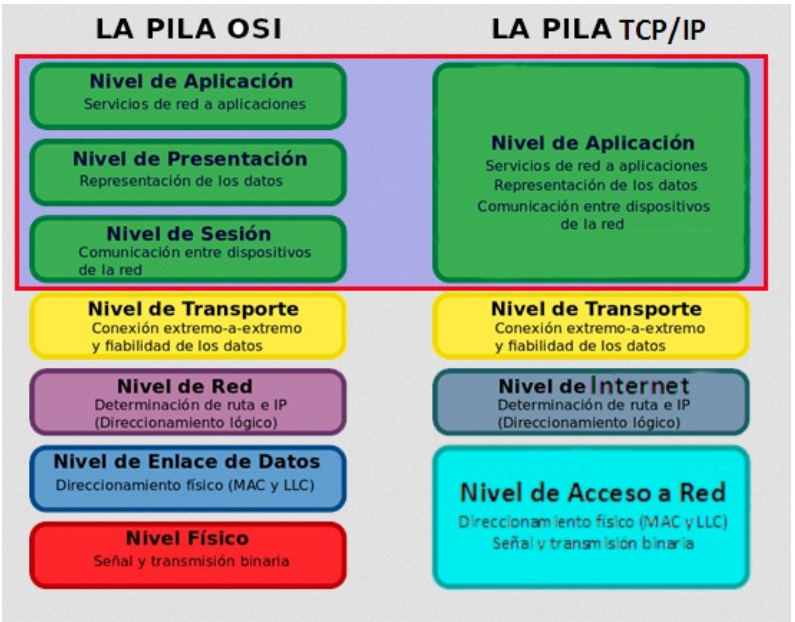
PAT permite ahorrar direcciones IP públicas, ya que con una única IP pública, además de dar acceso a Internet a múltiples equipos de una red local con NAT, podemos acceder a servicios múltiples servicios de la red privada alojados en diferentes equipos también con una única IP privada. Este ahorro es importante porque ha evitado el agotamiento de direcciones IPv4 y ha permitido retrasar la implantación de IPv6.

6. Características de la capa de aplicación.

La capa de aplicación define las **aplicaciones de red** y los **servicios de Internet** estándar que puede utilizar un usuario. Ofrece a las aplicaciones la posibilidad de acceder a los servicios de las demás capas y define los **protocolos** que utilizan las aplicaciones para intercambiar datos.

El usuario normalmente no interactúa directamente con el nivel de aplicación. Suele interactuar con **programas** que a su vez interactúan con el nivel de aplicación pero ocultando la complejidad subyacente. Así por ejemplo un usuario no manda una petición «GET /index.html HTTP/1.0» para conseguir una página en html sino que interactúa con un navegador web que es el que solicita al nivel de aplicación los datos de la web que queremos visualizar.

Cuando se diseñó el modelo TCP/IP, las capas de sesión, presentación y aplicación del modelo OSI se agruparon en la capa de aplicación del modelo TCP/IP.

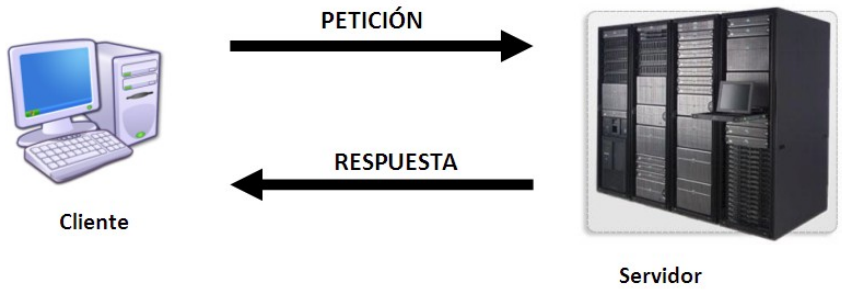


Esto significa que en la arquitectura TCP/IP los aspectos de representación, codificación y control de diálogo se administran en la capa de aplicación en lugar de hacerlo en las capas inferiores individuales, como sucede en el modelo OSI.

6.1. Modelos de intercambio de información.

Modelo cliente-servidor

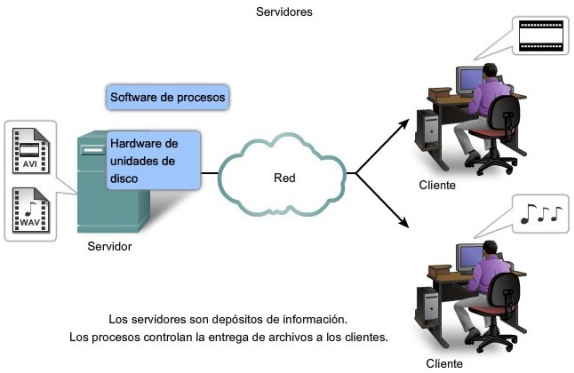
Como hemos visto, en el modelo cliente/servidor el equipo que solicita la información se llama cliente y el equipo que **responde a la solicitud** se denomina servidor. Los procesos de cliente y servidor se consideran una parte de la capa de aplicación.



Un servidor generalmente es un equipo que contiene información que se desea compartir con muchos clientes. Por ejemplo, páginas Web, documentos, bases de datos, imágenes, archivos de audio y vídeo pueden ser almacenados en un servidor y enviarse a los clientes que lo solicitan. En otros casos, como por ejemplo una impresora de red, el cliente de impresión envía al servidor documentos para que sean impresos.

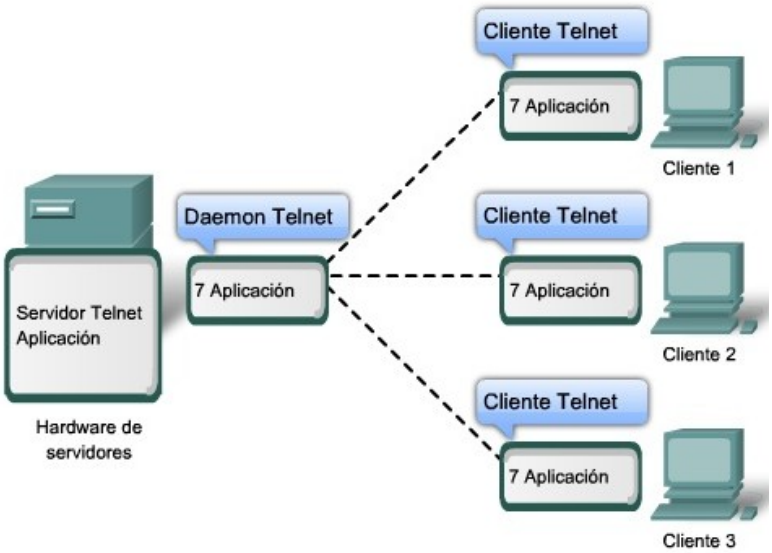
El servidor ejecuta uno o varios procesos, normalmente en segundo plano y sin control directo del usuario. Estos procesos se suelen describir como servidores que “escuchan” solicitudes de los clientes y están programados para responder cada vez que recibe una solicitud para el servicio proporcionado. A los procesos de servidores se les suele denominar también “daemons” o “demonios”.

Los diferentes tipos de procesos de servidor pueden tener diferentes requisitos para el acceso del cliente. Algunos servidores pueden requerir de autenticación del usuario para verificar si el usuario tiene permiso para acceder a los datos solicitados o para que realice una operación en particular. Dichos servidores deben poder acceder a los datos de las cuentas de usuarios y sus permisos. Por ejemplo, cuando se utiliza un cliente FTP si se solicita cargar datos al servidor FTP, podemos tener permiso para escribir en la carpeta personal, pero no para escribir en otras carpetas del sitio.



Los servidores generalmente reciben solicitudes de información al mismo tiempo de múltiples clientes. Por ejemplo, un servidor Telnet puede tener varios clientes que requieren conectarse a él. Estas solicitudes individuales del cliente pueden manejarse en forma simultánea y separada.

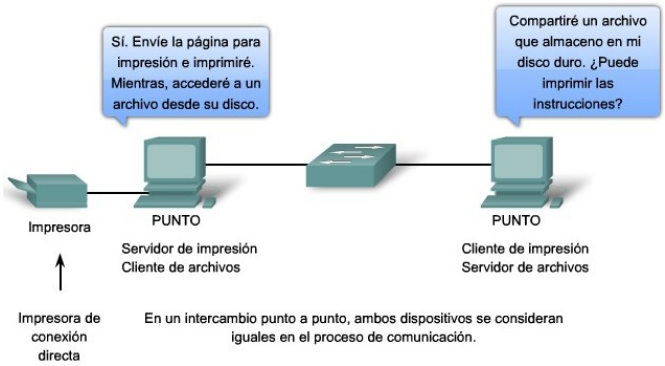
Los protocolos de la capa de aplicación describen el **formato de las solicitudes y respuestas** entre clientes y servidores. Además de la transferencia de datos, el intercambio entre servidor y cliente puede requerir información adicional como por ejemplo, la autenticación del usuario o identificar un archivo que se quiere transferir.



Modelo punto a punto

Además del modelo cliente-servidor existe también un modelo punto a punto. En una red punto a punto, dos o más equipos están conectadas por medio de una red y pueden compartir recursos (como impresoras y archivos) sin tener un servidor central. Cada equipo final conectado (conocido como punto) puede funcionar como un servidor o como un cliente. Una computadora puede asumir la función de servidor para una transacción mientras funciona en forma simultánea como cliente para otra transacción. Las funciones de cliente y servidor se establecen en función de quién realiza la solicitud y quién la responde.

Una red doméstica sencilla con dos computadoras conectadas compartiendo una impresora es un ejemplo de una red punto a punto. Se puede configurar cada equipo para compartir archivos, habilitar juegos en red o compartir una conexión de Internet.

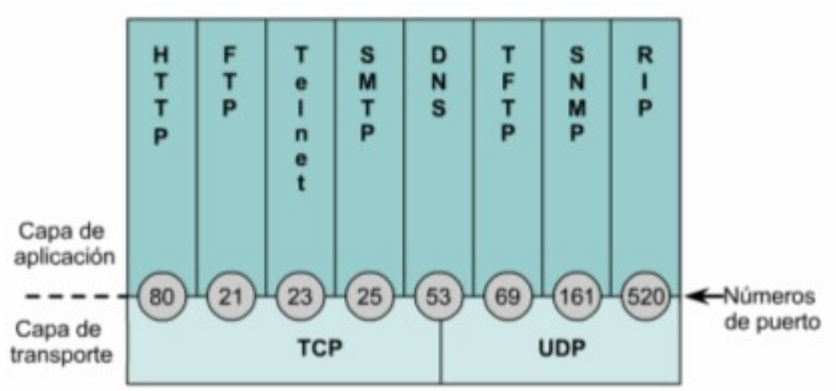


Otro ejemplo sobre la funcionalidad de la red punto a punto son dos computadoras conectadas a una gran red que utilizan aplicaciones de software para compartir recursos entre ellas a través de la red.

6.2. Relación con la capa de transporte.

Los protocolos de la capa de aplicación se apoyan en la capa de transporte para enviar y recibir datos. En la siguiente tabla se muestran algunos de los protocolos TCP/IP de la capa de aplicación más conocidos y el protocolo y puerto por defecto de la capa de transporte que suelen utilizar.

Protocolo aplicación	**Descripción**	**Protocolo transporte**	**Puerto por defecto**
TelNet	terminal virtual	TCP	23
FTP	transferencia de ficheros	TCP	21,20
SSH	Acceso remoto de terminal seguro	TCP	22
DNS	Servidor de resolución de nombres	TCP o UDP	53
HTTP	Web	TCP	80
DHCP	Transferencia de configuración de red	UDP	67
SMTP	Transferencia de correo electrónico	TCP	25
POP /IMAP	Acceso a las cuentas de correo en servidor	TCP	110 y 143



6.3. Protocolo y servicio DHCP.

El servicio del Protocolo de configuración dinámica de host (DHCP) permite a los dispositivos de una red obtener direcciones IP, máscaras de subred, puerta de enlace predeterminada, servidores de DNS y otros parámetros de red.

DHCP funciona de forma dinámica. Cuando un equipo se conecta a la red se realiza el contacto con el servidor de DHCP y se solicita una dirección. El servidor DHCP elige una dirección del rango configurado llamado **pool** y la asigna (“alquila”) al host por un tiempo prestablecido.

En redes locales de gran tamaño, o donde los usuarios cambien con frecuencia, es más eficaz que las direcciones IP se asignen automáticamente mediante el DHCP. Sin DHCP los usuarios tiene que configurar manualmente los parámetros de red para poder unirse a la red.

Las direcciones distribuidas por DHCP no se asignan de forma permanente a los hosts, sino que sólo se alquilan por un periodo de tiempo. Si el host se apaga o se desconecta de la red, la dirección regresa al pool y puede ser utilizada por otro equipo. Esto es especialmente útil para los usuarios móviles que entran y salen de la red. Los usuarios pueden moverse libremente desde una ubicación a otra y volver a establecer las conexiones de red.

6.4. Protocolo y servicio DNS.

En las redes de datos, los equipos, a nivel de red, se direccionan con una dirección IP numérica. A las personas les resulta difícil recordar estos números, por eso se crearon los nombres de dominio, ya que permiten convertir las direcciones numéricas en un nombre sencillo y reconocible. Por ejemplo, el nombre de dominio www.iesharia.org es mucho más fácil de recordar que 31.22.4.109. Además, si el IES Haría cambia la dirección IP, este hecho le será indiferente al usuario, ya que el nombre de dominio seguirá siendo www.iesharia.org. La nueva dirección simplemente estará enlazada con el nombre de dominio existente y la conectividad se mantendrá.

El Sistema de nombres de dominios (DNS) utiliza un conjunto **distribuido** de servidores para resolver los nombres asociados con estas direcciones numéricas.

DNS es un servicio cliente-servidor; sin embargo, es diferente de otros servicios cliente-servidor que estamos examinando. Mientras otros servicios utilizan un cliente que es una aplicación (como un explorador Web o un cliente de correo electrónico), el cliente de DNS ejecuta un servicio por sí mismo. El cliente DNS, a veces denominado “resolver” DNS, realiza la resolución de nombres para otras aplicaciones o servicios del mismo equipo que lo necesiten.

Cuando se configura un dispositivo de red, generalmente proporcionamos una o más direcciones del servidor DNS que el cliente DNS puede utilizar para la resolución de nombres. En general, el proveedor de servicios de Internet (ISP) provee las direcciones para utilizar con los servidores DNS y también hay en Internet servidores de DNS que son accesibles públicamente (8.8.8.8 y 8.8.4.4 son direcciones de servidores de DNS proporcionados por Google). Cuando la aplicación del usuario pide conectarse a un equipo remoto por nombre, la aplicación le solicita al cliente de DNS que consulte uno de estos servidores de nombres para resolver el nombre y obtener su dirección IP numérica.

6.5. Servicio de correo electrónico.

El servicio de correo es uno de los servicios de Internet más utilizado debido a su facilidad de uso y las utilidades que nos ofrece. Con el crecimiento de Internet, el correo electrónico ha reemplazado en muchos casos a servicios tradicionales como el **fax** o el **correo postal**.

Su funcionamiento es muy similar al del correo postal tradicional. Para poder enviar un correo electrónico lo único que necesitamos es la dirección electrónica o **e-mail** del destinatario.

Es un servicio **cliente-servidor**, el **cliente** nos permite enviar correo electrónico a través de un **servidor de correo saliente** y un **servidor de correo entrante** que hace las labores del cartero en el correo tradicional, depositará el correo electrónico en el buzón del destinatario.

Inicialmente, en los correos electrónicos sólo se podía enviar **texto**, pero actualmente, gracias a los tipos **MIME**, podemos **adjuntar** a nuestros correos

ficheros en cualquier formato como imágenes, documentos en formato binario, audio, vídeo, etc.

Por la forma en la que se diseñó inicialmente y por las características propias del servicio, el correo electrónico presenta algunas **desventajas**:

- El remitente tal vez no es quien dice ser
- El contenido ha podido ser cambiado o interceptado
- Puede ser una fuente de entrada a nuestro sistema de virus o malware
- Existencia del spam o correo no deseado, etc.

Lo único cierto de un correo electrónico es que ha llegado a un destinatario.

De todas formas, existen herramientas que permiten reducir estos problemas:

- Las firmas digitales
- Cifrado del correo
- Filtros antispam
- Programas antivirus
- Extensiones al servicio como SPF que permiten detectar la falsificación de direcciones en el envío de correo electrónico.

Práctica

Investiga qué elementos básicos están involucrados en un sistema de correo electrónico y los agentes más importantes que intervienen en un sistema de correo electrónico.

6.6. Protocolo y servicio FTP.

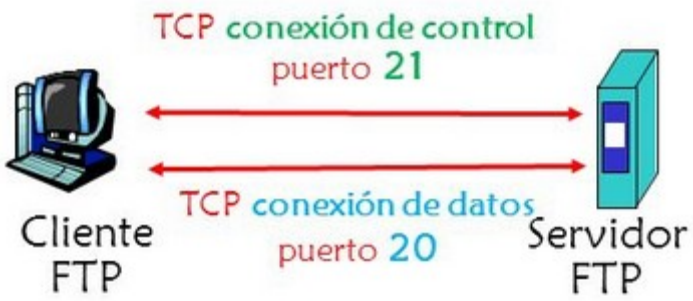
FTP se desarrolló para permitir las transferencias de archivos entre un cliente y un servidor. Un cliente FTP es una aplicación que se ejecuta en un equipo y que carga y descarga archivos de un servidor que ejecuta el demonio/servidor FTP.

El servicio de FTP necesita dos conexiones entre el cliente y el servidor para transferir archivos:

- Una para comandos y respuestas
- Otra para la transferencia real de archivos.

El cliente establece la primera conexión con el servidor utilizando el puerto TCP 21. Esta conexión se utiliza para **controlar** el tráfico, que consiste en comandos del cliente y respuestas del servidor.

A continuación el cliente establece una segunda conexión con el servidor por el puerto TCP 20. Esta conexión es para la **transferencia** de datos y se crea cada vez que se transfiere un archivo.



La transferencia de archivos puede producirse en **ambas direcciones**. El cliente puede descargar (bajar) un archivo desde el servidor o el cliente puede cargar (subir) un archivo en el servidor.

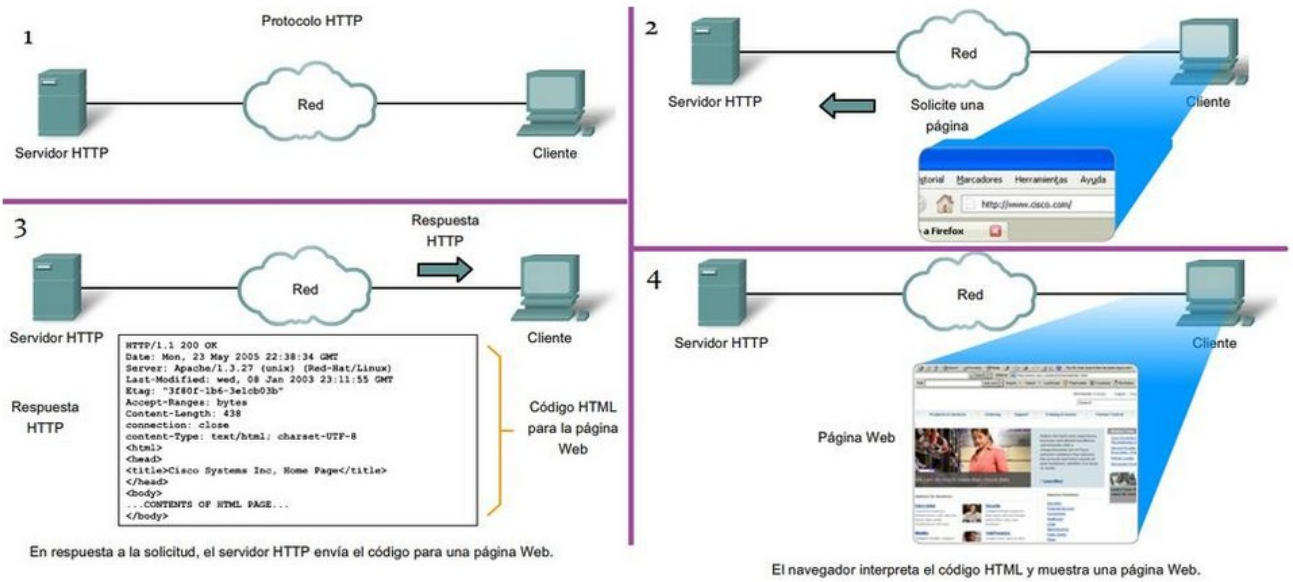
6.7. Servicio WWW y HTTP.

Cuando se escribe una dirección Web (o URL) en un explorador de Internet, el explorador establece una conexión con el **servicio Web** del servidor que utiliza el **protocolo HTTP**.

URL (o Localizador uniforme de recursos) y URI (Identificador uniforme de recursos) son los nombres que la mayoría de las personas asocian con las direcciones Web. Por ejemplo, la URL http://www.iesharia.org/index.html es un ejemplo que se refiere a un recurso específico, una página Web llamada index.html en un servidor identificado como www.iesharia.org

Los exploradores Web son las aplicaciones cliente que utilizan los equipos para conectarse a la **World Wide Web** y acceder a recursos almacenados en un **servidor Web**.

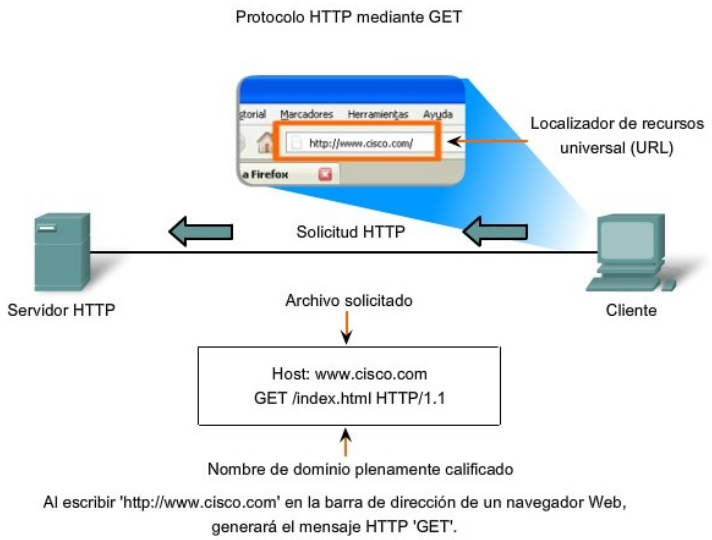
Para acceder al contenido, los clientes Web realizan conexiones al servidor y solicitan los recursos deseados. El servidor responde con el recurso y, al recibirlo, el explorador **interpreta** los datos y los **presenta** al usuario.



Los navegadores pueden interpretar y presentar muchos tipos de datos, como páginas escritas en lenguaje de marcas de hipertexto (**HTML**, el lenguaje en el que se crean las páginas Web). Pero también otros tipos de datos que requieren de un programa para su presentación. A estos programas generalmente se les conoce como plug-ins o complementos. Para ayudar al explorador a determinar qué tipo de archivo está recibiendo, el servidor especifica qué clase de datos contiene el archivo.

El protocolo de transferencia de hipertexto (HTTP) se utiliza a través de la World Wide Web para transferencia de datos y es uno de los protocolos de aplicación más utilizados.

HTTP es un protocolo de solicitud/respuesta. En el protocolo HTTP se definen los tipos de mensajes que el cliente utiliza para solicitar la página Web y los tipos de mensaje que el servidor utiliza para responder. Los tres tipos de mensajes comunes son GET, POST y PUT.



HTTPS

Aunque es muy flexible, HTTP no es un protocolo seguro. Los mensajes que se intercambian utilizando este protocolo se puede interceptar y leer al no viajar cifrados. Para una comunicación segura a través de la WWW, se utiliza el protocolo HTTP seguro (HTTPS). HTTPS puede utilizar autenticación y encriptación para asegurar los datos cuando viajan entre el cliente y el servidor.