

---

# Capítulo 4. Transporte UDP y TCP

---

Redes de Ordenadores  
2º Grado en Ingeniería en Tecnologías de Telecomunicación



# Índice

## *Hora 1*

- 1 Paradigmas de comunicaciones
  - 1.1 Paradigma cliente/servidor
  - 1.2 Paradigma Peer-to-Peer (P2P)
- 2 Multiplexación por puerto
- 3 UDP
  - 3.1 Cabecera UDP
  - 3.2 Ejemplo de servicio UDP
  - 3.3 Cuándo usar UDP

## *Hora 2*

- 4 TCP
  - 4.1 Cabecera TCP
  - 4.2 Opciones cabecera TCP

## *Hora 3*

- 4.3 Conexiones TCP
- 4.4 Diagrama de transición de estados de TCP
- 4.5 Transferencia interactiva

## *Hora 4*

- 4.6 Fiabilidad en TCP
- 4.7 Transferencia masiva
  - 4.7.1 Transferencia normal
  - 4.7.2 Control de flujo
  - 4.7.3 Control de congestión
- 4.8 Producto RTTxBW
- 4.9 Ejemplo de traza TCP

# Índice hora 1

## *Hora 1*

- 1 Paradigmas de comunicaciones
  - 1.1 Paradigma cliente/servidor
  - 1.2 Paradigma Peer-to-Peer (P2P)
- 2 Multiplexación por puerto
- 3 UDP
  - 3.1 Cabecera UDP
  - 3.2 Ejemplo de servicio UDP
  - 3.3 Cuándo usar UDP

## *Hora 2*

- 4 TCP
  - 4.1 Cabecera TCP
  - 4.2 Opciones cabecera TCP

## *Hora 3*

- 4.3 Conexiones TCP
- 4.4 Diagrama de transición de estados de TCP
- 4.5 Transferencia interactiva

## *Hora 4*

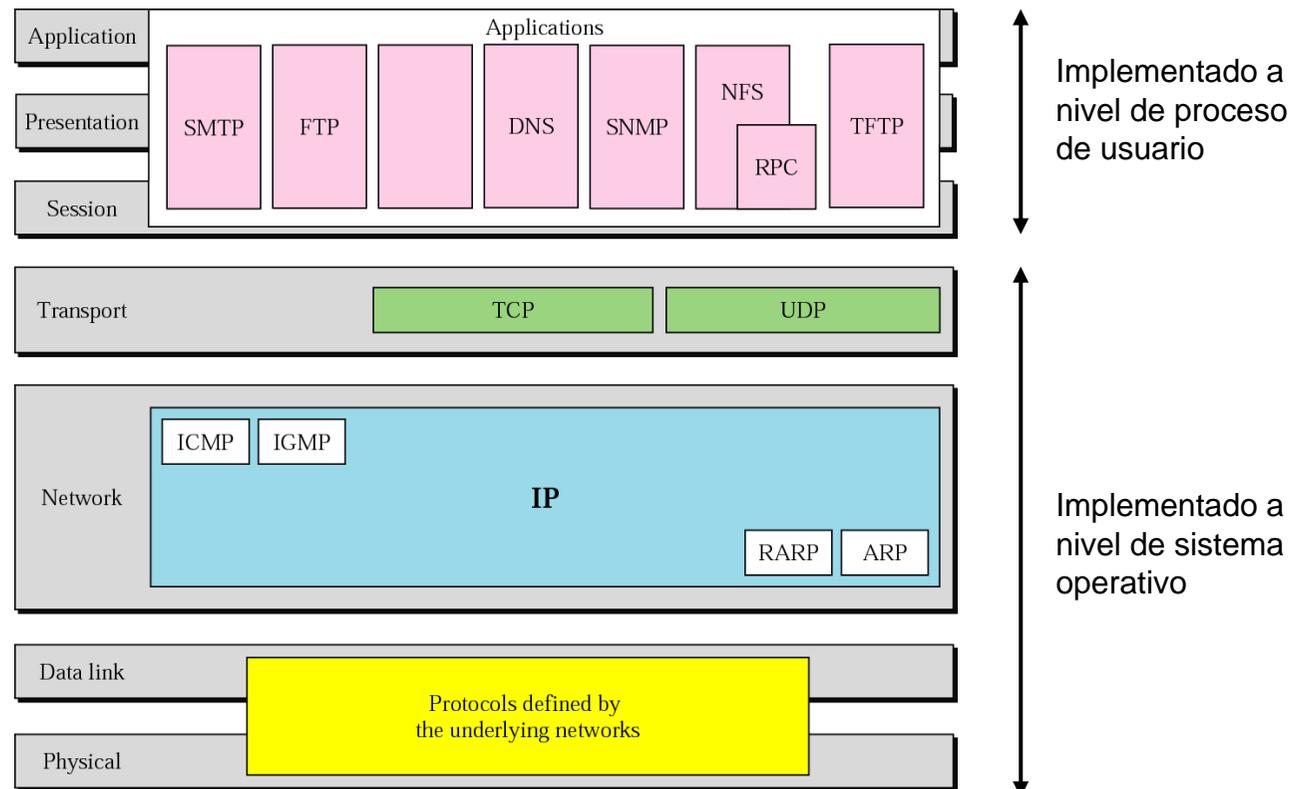
- 4.6 Fiabilidad en TCP
- 4.7 Transferencia masiva
  - 4.7.1 Transferencia normal
  - 4.7.2 Control de flujo
  - 4.7.3 Control de congestión
- 4.8 Producto RTTxBW
- 4.9 Ejemplo de traza TCP

## Objetivos

- Revisar los paradigmas de comunicaciones sobre el que sustentan los servicios de Internet
- Entender la necesidad de multiplexar múltiples comunicaciones sobre el mismo nivel de red IP
  - ¿Qué ocurre si se quiere mantener varias comunicaciones simultáneas con diferentes aplicaciones en la misma máquina destino?
  - Las direcciones IP no son suficiente
- Presentar el protocolo de transporte UDP
  - Características
  - Cabecera
  - Casos de uso

# 1 Paradigmas de comunicaciones

- Las aplicaciones se implementan tradicionalmente sobre el nivel de transporte (UDP o TCP).
  - Las aplicaciones hacen uso de los servicios proporcionados por el nivel de transporte.

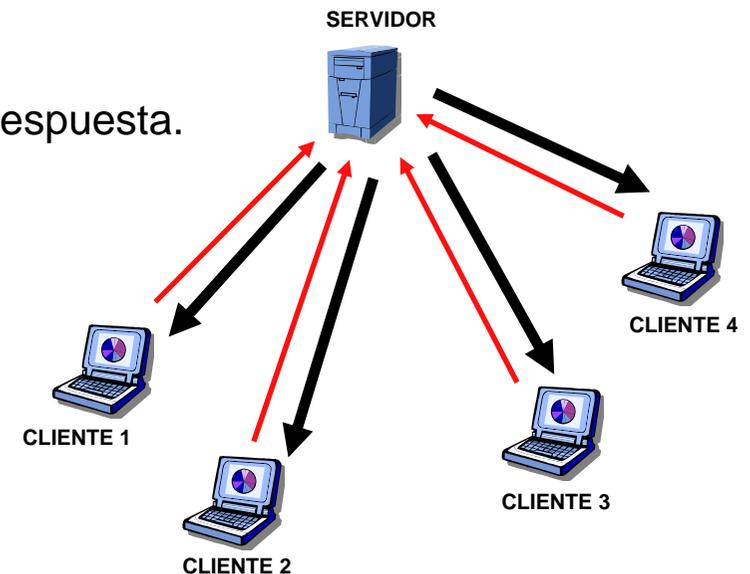
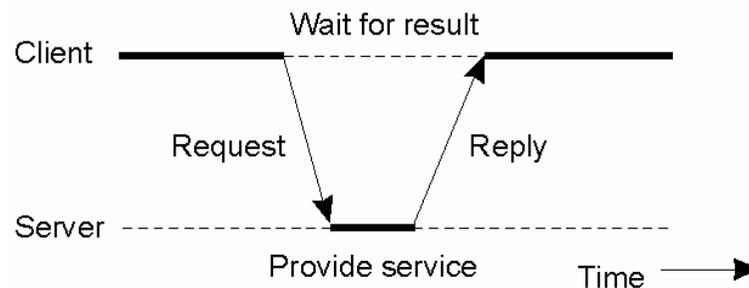


## Paradigmas de comunicaciones

- El sistema operativo (kernel) implementa la pila de protocolos TCP/IP
  - Es un servicio del sistema operativo
- La aplicación corre como proceso de usuario
  - Más fácil su desarrollo y depuración
- Existen APIs (Application Programming Interface) que proporciona el sistema operativo para el acceso a los servicios del nivel de transporte
  - Sockets BSD
- Existen diferentes estrategias o paradigmas de comunicación a la hora de distribuir la responsabilidad de la comunicación:
  - Cliente/Servidor
  - P2P
  - Híbridos: combinan los dos anteriores

## 1.1 Paradigma cliente/servidor

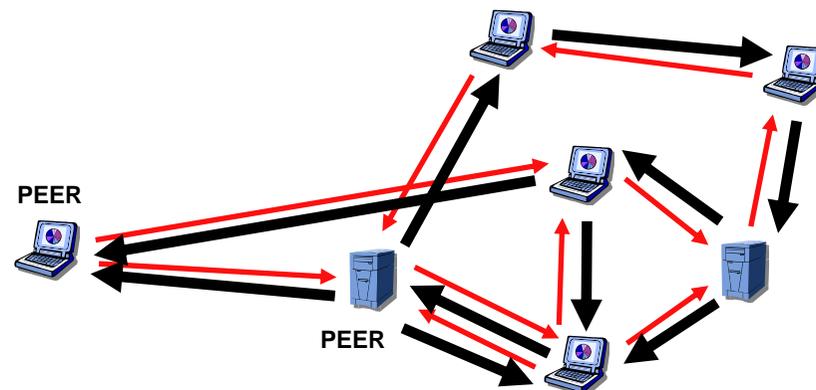
- Cliente: requieren un servicio.
  - Normalmente inicia la comunicación y la finaliza.
- Servidor: prestan un servicio.
  - Normalmente atiende la comunicación iniciada por el cliente.
- Patrón típico de petición-respuesta
  - Cliente: envía solicitud y recibe respuesta.
  - Servidor: recibe solicitud, la procesa y envía respuesta.



- Comunicación asimétrica:
  - Alta carga del servidor
  - Necesidad de elevado ancho de banda en la red de acceso al servidor

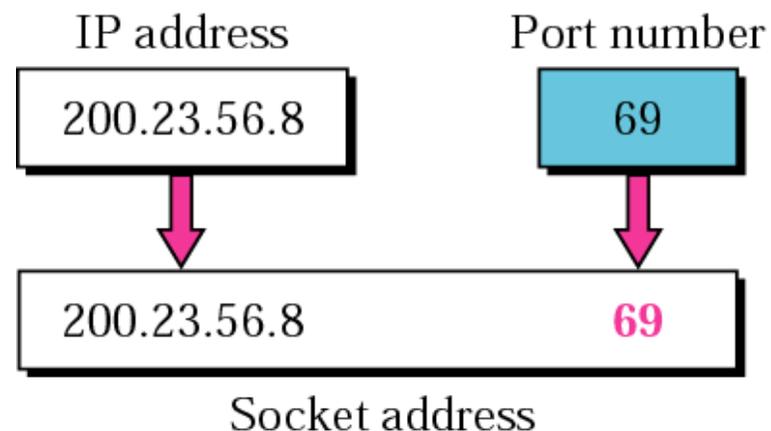
## 1.2 Paradigma Peer-to-Peer (P2P)

- Igual-a-igual, par-a-par, todos los nodos son clientes y servidores simultáneamente.
  - Todos los nodos son funcionalmente iguales.
- Popularidad de aplicaciones de intercambio de ficheros basadas en P2P.
- Ventajas de P2P:
  - Alta escalabilidad: la información se intercambia directamente entre los usuarios finales sin pasar por un servidor intermedio. Se dispone de los recursos (ancho de banda, almacenamiento y CPU) de miles de nodos.

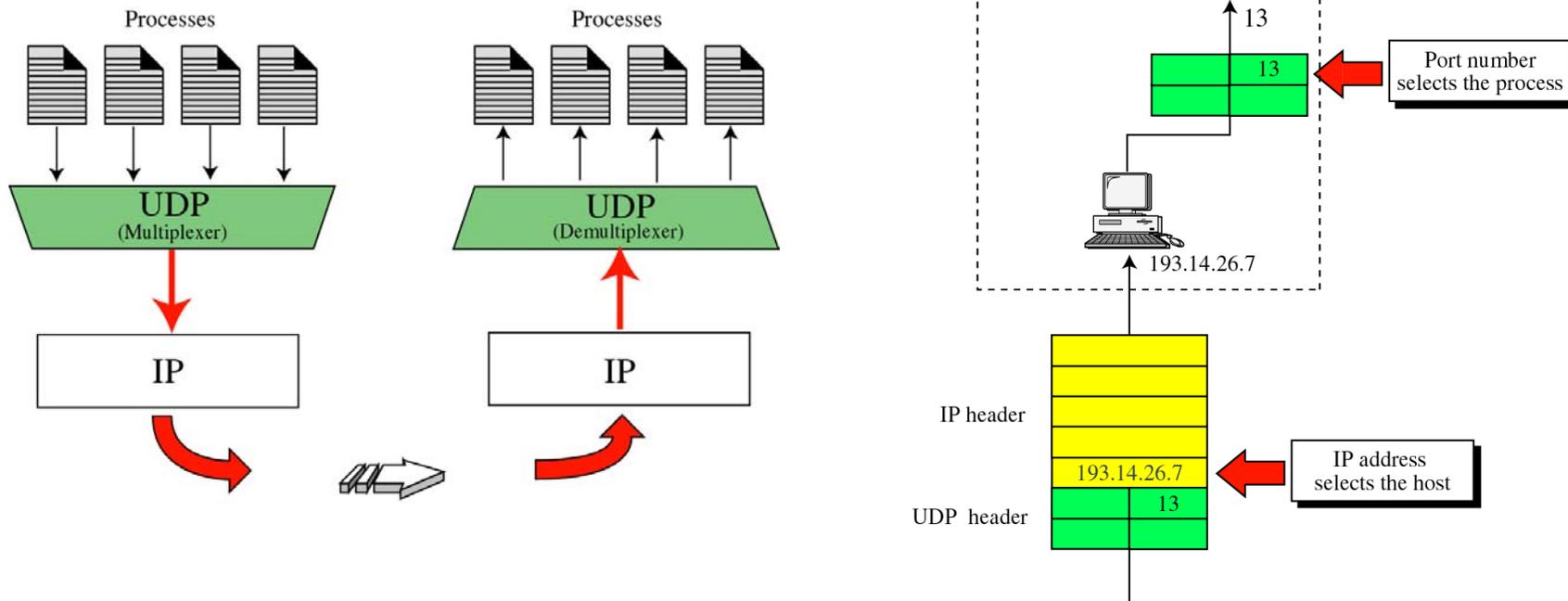


## 2 Multiplexación por puerto

- El puerto identifica a la aplicación o proceso corriendo en determinada máquina.
- El par (IP, puerto) se denomina *socket* e identifica unívocamente a un proceso de aplicación en una máquina que puede enviar/recibir datos.
- Una comunicación estará caracterizada por dos sockets, uno correspondiente a cada extremo de la comunicación.



# Multiplexación por puerto

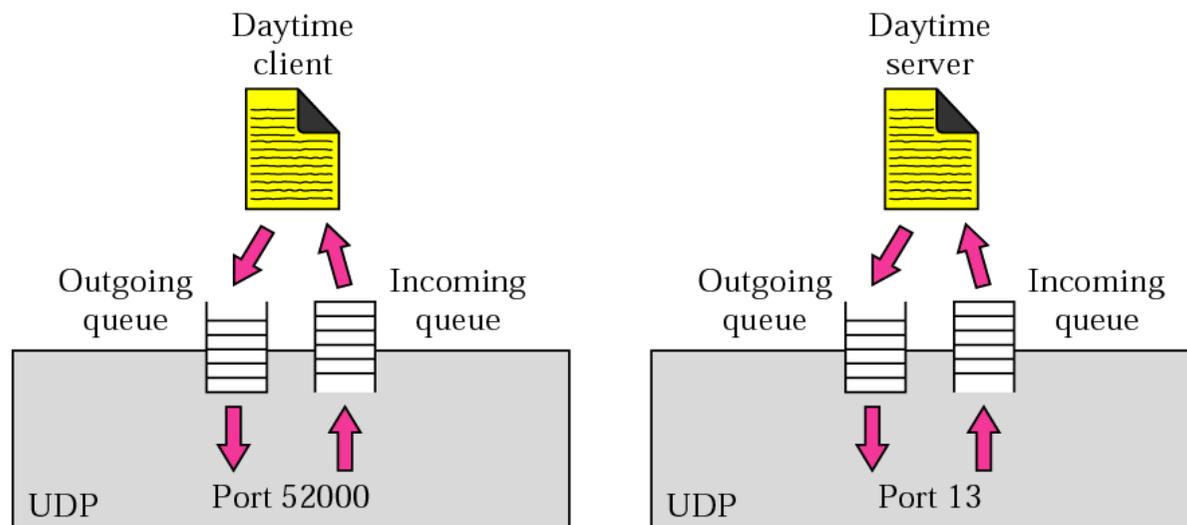


## Multiplexación por puerto

- La aplicación escuchando en un puerto recibirá todos los paquetes dirigidos a esa (IP, puerto, protocolo de transporte).
- Puerto, identificador de 16bits: 1 a 65535, el 0 no se utiliza.
  - Puertos bien conocidos: <1024, registrados por la ICANN identifican el tipo de servicio normalmente.
    - Puertos UDP típicos: 7 Echo, 13 Daytime, 53 DNS, 69 TFTP, 123 NTP, 161/162 SNMP, etc.
    - Puertos TCP típicos: 20/21 FTP, 23 Telnet, 25 SMTP, 53 DNS, 80 HTTP, 110 POP3, etc.
    - Lista completa de puertos en el fichero /etc/services en Linux.
  - Puertos efímeros: se escogen por encima del 1024 sin que colisionen con uno ya existente.
- Normalmente en toda comunicación cada extremo realiza una función:
  - Servidor: está escuchando en un puerto bien conocido peticiones de los clientes.
  - Cliente: elige un puerto efímero al azar para comunicarse con el servidor.

## Multiplexación por puerto

- Envío de un datagrama a un puerto sin aplicación escuchando devuelve:
  - En el caso de UDP: ICMP de error de puerto inalcanzable.
  - En el caso de TCP: mensaje de RESET.
- Puerto bidireccional: transmisión/recepción.
- Buffers de entrada/salida de tamaño configurable por el sistema operativo.



## 3 UDP

RFC768(STD6)

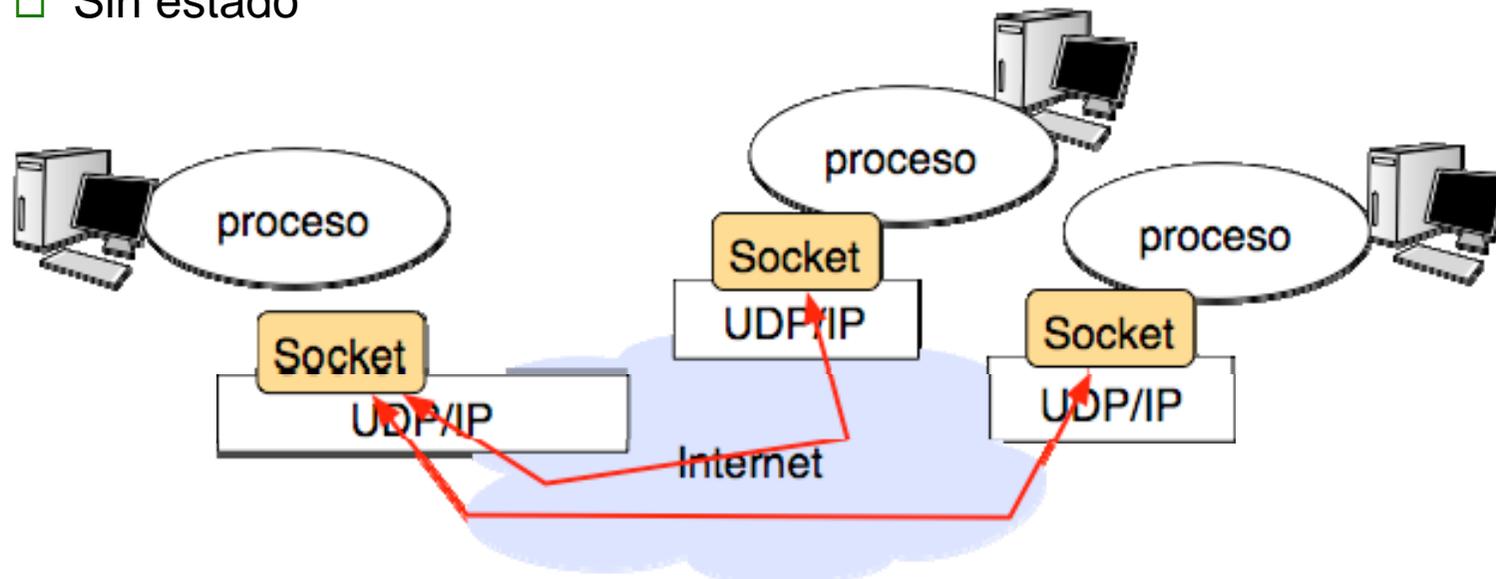
- User Datagram Protocol
- UDP es un protocolo de nivel de transporte: se encapsula por encima de IP (cabecera IP con campo protocol=17).
- Ofrece un servicio de datagramas (no orientado a conexión). Cada datagrama enviado es independiente.
  - 1 write() de la aplicación  $\Rightarrow$  1 datagrama UDP  $\Rightarrow$  1 datagrama IP (o varios si hay fragmentación).
  - Control de tamaño. La aplicación determina el tamaño de datos de cada datagrama UDP (payload).
    - Si es mayor que la MTU de la red sufrirá fragmentación.
  - Control de tiempos. No buffered, UDP acepta datos y los transmite inmediatamente (siempre que los niveles inferiores se lo permitan).
- Habitualmente menos del 5-10% del tráfico total de redes de área local o troncales es UDP. El resto del tráfico es mayoritariamente TCP.

## Características UDP

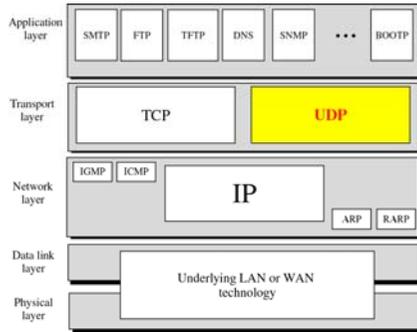
- Deficiencias cubiertas por otros niveles de transporte:
  - UDP no es fiable
    - Los paquetes se pueden perder.
    - Los paquetes se pueden entregar fuera de orden.
  - UDP no incorpora mecanismos de control de flujo y congestión
    - La aplicación ha de implementarlos.
    - Si todas las aplicaciones de una red fueran UDP y empezasen a mandar a elevadas tasas, se producirían desbordamientos en las colas de los routers sin ningún tipo de control.
  
- UDP ofrece:
  - Multiplexación de aplicaciones gracias al uso de puertos.
  - Checksum del mensaje.

## Características UDP

- Orientado a datagramas
  - Un socket ligado a un puerto
  - Puede mandar a cualquier otro socket UDP
  - Puede recibir de cualquier otro socket UDP
  - Un socket UDP puede hacer de cliente o servidor indistintamente, y además simultáneamente con diferentes máquinas remotas.
  - Sin estado

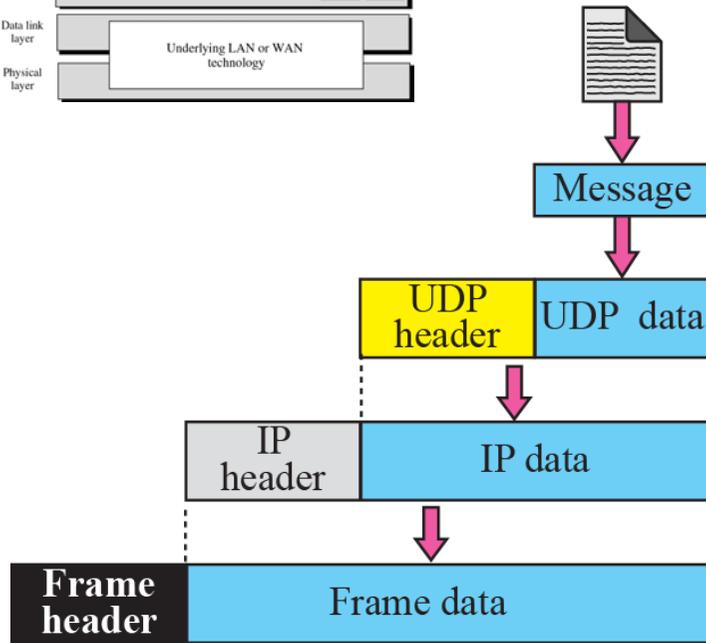


# UDP encapsulación

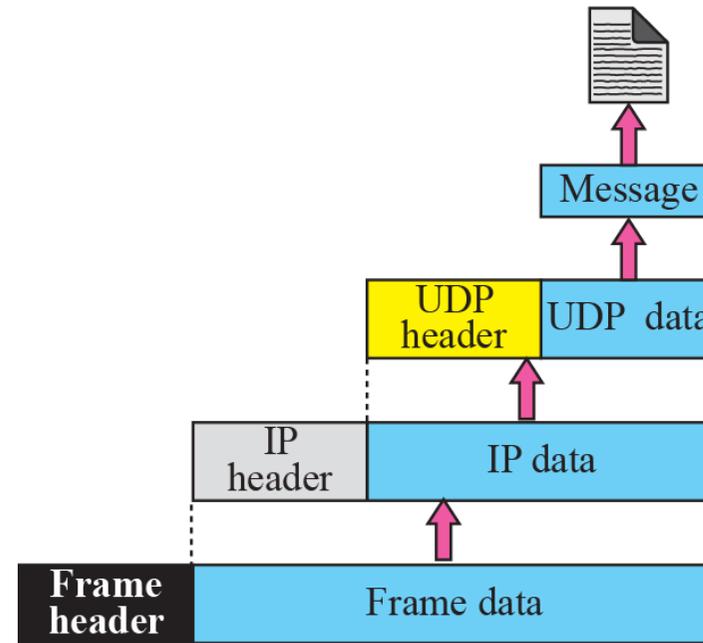


Sender Process

Receiver Process

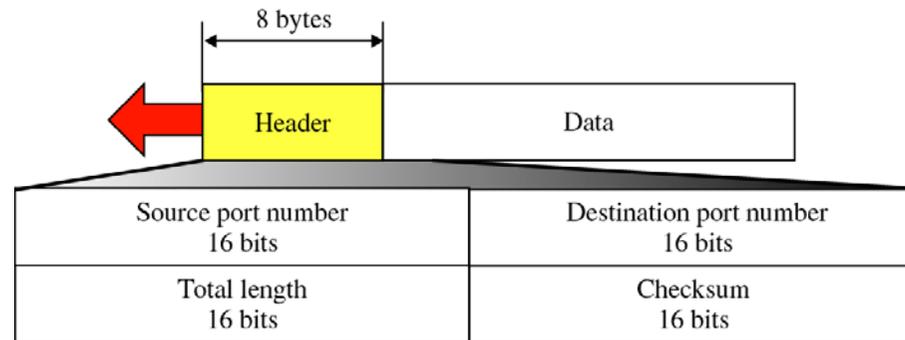


a. Encapsulation



b. Decapsulation

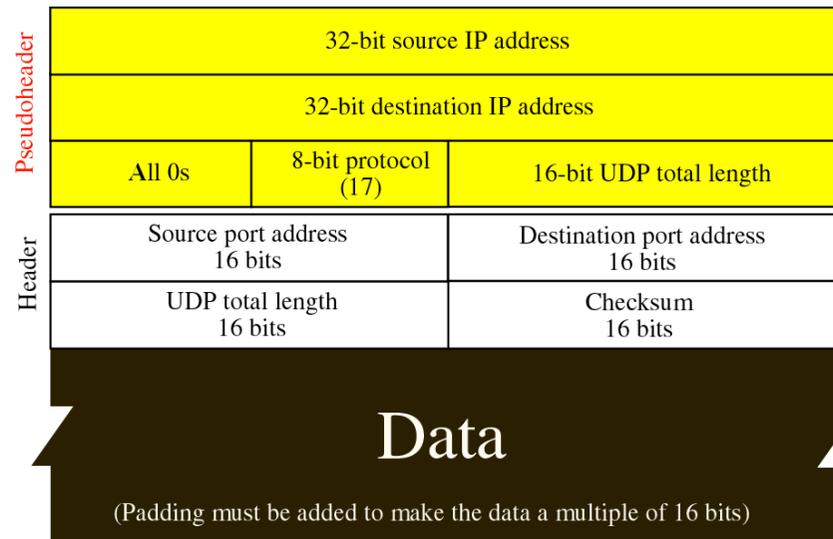
## 3.1 Cabecera UDP



- Source/Destination port (16bits)
- Length (16bits): longitud total de la cabecera UDP+datos en bytes. Valor máximo:  $2^{16}-1=65535$ , pero el campo longitud de la cabecera IP también tiene este máximo, por lo que el mayor tamaño posible de datos UDP será:  $65535-20(IP)-8(UDP) = 65507$  bytes.
- Checksum (16bits): calculado como el CRC de IP en palabras de 16 bits aplicado sobre
  - cabecera UDP
  - datos UDP: se hace padding con 0's hasta múltiplo de 16bits.
  - seudocabecera UDP

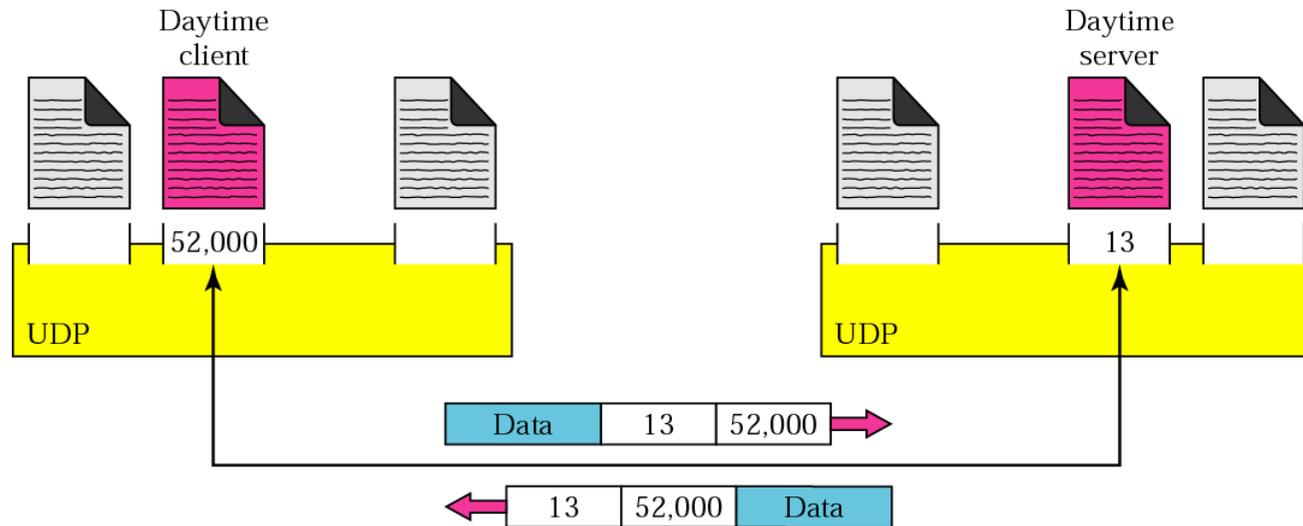
# Seudocabecera UDP para el checksum

- Incluye ciertos campos de la cabecera IP:
  - IP origen
  - IP destino
  - Protocolo
  - Longitud UDP
- Permite una doble comprobación de que los datos han llegado al destino adecuado.



## 3.2 Ejemplo de servicio UDP

- Servicio Daytime.



## Ejemplo de servicio UDP

■ `tcpdump -n`

```
11:27:20.948616 IP 130.206.158.132.46612 > 130.206.158.141.daytime: UDP, length 2
11:27:20.948830 IP 130.206.158.141.daytime > 130.206.158.132.46612: UDP, length 26
```

■ `tcpdump -xX`

```
11:27:20.948616 IP 130.206.158.132.46612 > 130.206.158.141.daytime: UDP, length 2
  0x0000:  4500 001e 098d 4000 4011 ee93 82ce 9e84  E.....@. ....
  0x0010:  82ce 9e8d b614 000d 000a c7ff 3f0a      .....?.
```

|       |       |        |     |
|-------|-------|--------|-----|
| ↑     | ↑     | ↑      | ↑   |
| Sport | Dport | TotLen | CRC |
| 46612 | 13    | 10     |     |

```
11:27:20.948830 IP 130.206.158.141.daytime > 130.206.158.132.46612: UDP, length 26
  0x0000:  4500 0036 0000 4000 4011 f808 82ce 9e8d  E..6..@. ....
  0x0010:  82ce 9e84 000d b614 0022 aa12 3135 204d  .....".15.M
  0x0020:  4152 2032 3031 3220 3131 3a32 373a 3230  AR.2012.11:27:20
  0x0030:  2043 4554 0d0a      .CET..
```

Respuesta: 15 MAR 2012 11:27:20 CET



## Ejemplo de servicio UDP

| <i>Port</i> | <i>Protocol</i> | <i>Description</i>                            |
|-------------|-----------------|---|
| 7           | Echo            | Echoes a received datagram back to the sender |
| 9           | Discard         | Discards any datagram that is received        |
| 11          | Users           | Active users                                  |
| 13          | Daytime         | Returns the date and the time                 |
| 17          | Quote           | Returns a quote of the day                    |
| 19          | Chargen         | Returns a string of characters                |
| 53          | Nameserver      | Domain Name Service                           |
| 67          | Bootps          | Server port to download bootstrap information |
| 68          | Bootpc          | Client port to download bootstrap information |
| 69          | TFTP            | Trivial File Transfer Protocol                |
| 111         | RPC             | Remote Procedure Call                         |
| 123         | NTP             | Network Time Protocol                         |
| 161         | SNMP            | Simple Network Management Protocol            |
| 162         | SNMP            | Simple Network Management Protocol (trap)     |

## 3.3 Cuándo usar UDP

- La falta de fiabilidad y segmentación supone más tareas para la aplicación.
- Sin embargo, UDP es:
  - Rápido, no hay fase de establecimiento de conexión.
  - Ligero, supone poca sobrecarga de protocolo (8 bytes).
- Será por tanto útil para:
  - Aplicaciones de control y gestión.
  - Aplicaciones de difusión.
  - Aplicaciones de tiempo real.

## Cuándo usar UDP

- Aplicaciones de control y gestión
  - Requieren normalmente poco intercambio de información, del tipo petición-respuesta.
  - Los paquetes son pequeños. Dependiendo de la aplicación pueden generarse en gran número.
  - UDP evita el coste de apertura y cierre de conexiones TCP.
  
- Aplicaciones de difusión
  - Necesiten usar direcciones destino Multicast o Broadcast.
  - Con TCP no es posible.

## Cuándo usar UDP

- Aplicaciones de tiempo real
  - Necesitan un control absoluto de los paquetes generados en la red, por ejemplo, del espaciado entre paquetes.
  - El retardo extremo a extremo y el jitter son importantes.
    - El buffer en recepción permite ocultar parte del retardo y jitter.
  - Pequeñas pérdidas de paquetes son tolerables.
    - Algunas de estas pérdidas pueden ser ocultadas por los codecs.
  - Ejemplos:
    - Aplicaciones de voz sobre IP (VoIP).
    - Transmisión de audio/video en tiempo real (la aplicación es la encargada del control de la comunicación casi en su totalidad). Streaming.

## Resumen

- Paradigmas de comunicaciones:
  - Cliente/Servidor
  - P2P
- Multiplexación de aplicaciones gracias al uso de puertos.
- UDP
  - Ofrece un servicio de datagramas (no orientado a conexión)
  - Permite utilizar puertos para identificar aplicaciones origen/destino de los mensajes
  - No aporta fiabilidad
  - No incorpora mecanismos de control de flujo y congestión
  - Es un protocolo sin estado
  - La aplicación tiene control sobre el tamaño de los paquetes y cuando se envían
  - Útil en aplicaciones que requieran un protocolo con poca carga de cabeceras, sin señalización extra, con múltiples destinatarios y/o con control absoluto por parte de la aplicación

## Referencias

- [Forouzan]
  - Capítulo 13, “Introduction to the Transporte Layer”, sección 13.1
  - Capítulo 14, “User Datagram Protocol (UDP)”, secciones 14.1-14.4
- [Stevens]
  - Capítulo 11 “UDP: User Datagram Protocol”