1

La capa de transporte

Servicios, multiplexación y demultiplexación.
Transporte sin conexión: UDP.

Tema 3: La capa de transporte.

2

La capa de transporte: introducción.

3

- Capa de transporte y sus servicios.
  - Ubicada entre las capas de aplicación y de red.
  - Proporciona servicios de comunicación "directa" entre procesos de aplicación → Protocolo de la capa de transporte.
    - Comunicación lógica entre procesos que se ejecutan en sistemas finales diferentes, mediante el intercambio de MENSAJES de una aplicación de red.
      - MENSAJES: denominación genérica de las PDU de la capa de aplicación.
  - Función básica:
    - Ampliar servicios de entrega de la capa de red a servicios de entrega entre procesos.
    - La capa de red sólo provee servicio de entrega entre sistemas finales.
  - Funciones complementarias.
    - Fiabilidad, control de flujo, control de congestión, etc.
    - Determinan el correspondiente protocolo: UDP, TCP, SCTP, etc.

La capa de transporte: introducción.

4

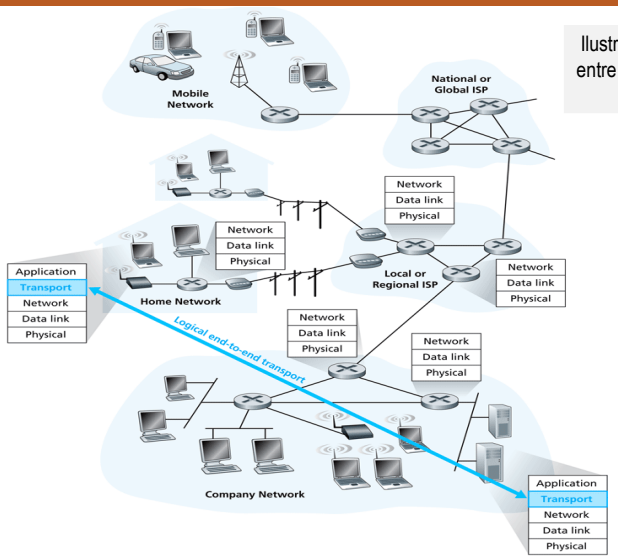


Ilustración de la comunicación lógica entre procesos mediante protocolo de la capa de transporte

Figure 3.1 ♦ The transport layer provides logical rather than physical communication between application processes.

La capa de transporte: introducción.

5

- Relación entre las capas de red y de transporte.
  - Mediante operaciones lógicas de MULTIPLEXADO y DEMULTIPLEXADO.
    - Varios procesos de la capa de transporte utilizan los servicios de la capa de red.
  - Protocolos de transporte.
    - Discurren SÓLO entre sistemas finales, son opacos a los sistemas intermedios de red.
    - Utilizan los servicios del protocolo básico de la capa de red (IP).
    - Son independientes de cómo se transfieren por la red los DATAGRAMAS* IP.
      - Datagramas IP = Paquetes IP = PDU de la capa de red en el contexto Internet.

*: Ver en el libro de texto comentario respecto a la denominación usual de las PDU en cada capa de la pila de protocolos de Internet (página 156, último párrafo del subapartado 3.1.2).

La capa de transporte: introducción.

6

- Relación entre las capas de red y de transporte (II).
  - Protocolos de transporte (II).
    - PDU denominadas genéricamente SEGMENTOS*.
      - Porque con frecuencia (no siempre) la capa de transporte tiene que segmentar/fraccionar los MENSAJES* de la capa de aplicación.
      - Emisor: trocea (si es necesario y posible) los MENSAJES en SEGMENTOS y los pasa a la capa de red.
      - Receptor: re-ensambla (si es necesario y posible) SEGMENTOS en MENSAJES y los pasa a la capa de aplicación.
    - Diferentes tipos, según modelo de servicios que provea cada uno.
      - En una red pueden funcionar/coexistir diferentes protocolos de transporte.
      - Una aplicación en ejecución SÓLO utiliza un protocolo de transporte.

*: Ver en el libro de texto comentario respecto a la denominación usual de las PDU en cada capa de la pila de protocolos de Internet (página 156, último párrafo del subapartado 3.1.2).

La capa de transporte: introducción.

7

- Relación entre las capas de red y de transporte (III).
  - Protocolos de transporte (III).
    - Los servicios que proveen pueden, o no, estar limitados por los servicios que provee la capa de red. P.e:
      - Si la capa de red no garantiza retardo ni ancho de banda mínimos → La capa de transporte tampoco los garantiza a la capa de aplicación.
    - Pueden proveer determinados servicios aún si la capa de red no los provee. P.e:
      - Capa de transporte puede proveer transferencia fiable aún si la capa de red no lo garantiza.
        - Fiabilidad: no pérdida, no alteración y/o no duplicación de los datos.
      - Capa de transporte puede proveer seguridad aún si la capa de red no los provee.
        - Seguridad: confidencialidad, autenticidad y/o integridad de los datos.

La capa de transporte: introducción.

8

- La capa de transporte en Internet.
  - Más de un protocolo especificado.
    - Fundamentales: UDP y TCP.
    - ¿Cuál utilizar? Lo determina el desarrollador de la aplicación usuaria.
  - UDP: *User Datagram Protocol*.
    - Sólo provee servicio de entrega de proceso a proceso, con comprobación de integridad de los datos que porta el segmento UDP.
    - Servicio no fiable, no garantiza: entrega, secuencia, integridad, no duplicados.
    - No fragmenta los datos para adaptarlos a las capas inferiores.
    - No orientado a conexión → Permite envío inmediato de los datos.
      - Pero… puede "inundar" la red con el envío NO controlado de datos.
    - Es la extensión, a la capa de transporte, del modelo de servicio *best effort* de la capa red de Internet (IP).

La capa de transporte: introducción.

9

- La capa de transporte en Internet (II).
  - TCP: *Transmision Control Protocol*.
    - Provee muchos más servicios que UDP.
      - Los mismos que UDP y otros más → Mayor complejidad y requiere más recursos.
    - Provee:
      - Fiabilidad de los datos (no pierde, no altera, no duplica).
      - Garantía de secuencia de los datos.
      - Fragmentación de los datos para adaptarlos a las capas inferiores.
      - Control de flujo (envío controlado de datos).
        - Servicio prestado a las aplicaciones (servicio de carácter individual).
      - Control de congestión (envío controlado de datos).
        - Servicio prestado a la red (servicio para "el bien común").
        - Es un "esfuerzo" para proporcionar equidad en el uso de los recursos de red.
      - Gestión de conexión (orientado a conexión) → Cierto retardo en el envío de los datos.
        - En el inicio y quizás durante el resto del tiempo.

La capa de transporte: introducción.

10

- La capa de transporte en Internet (III).
  - Servicios que NO proveen UDP ni TCP.
    - Garantía de mínimo retardo.
    - Garantía de mínima tasa de transferencia (ancho de banda).
    - Servicios de seguridad (confidencialidad, autenticidad y/o integridad).
  - Dadas las carencias de UDP y TCP (y de IP), otros protocolos de transporte y de red se han desarrollado. P.e:
    - Para proveer garantías de valores mínimos de retardos y tasas de transferencia.
      - Modelos IntServ y DiffServ, MPLS, etc.
    - Para proveer servicios de seguridad.
      - IPSec, SSL, TLS, DTLS, etc.

La capa de transporte: multiplexación/demultiplexación.

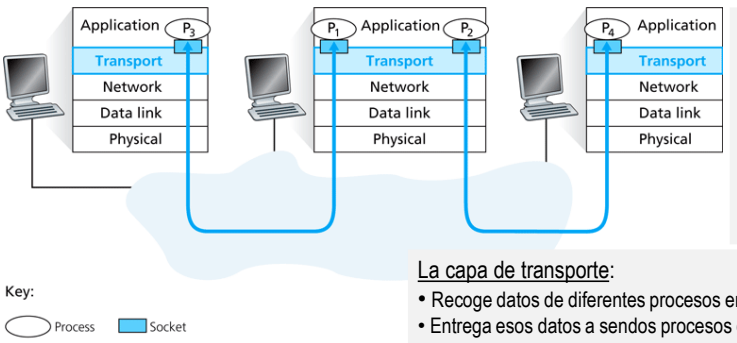
11

- Multiplexación y demultiplexación*.
  - En principio servicios válidos para cualquier capa de la pila de protocolos respecto a la capa inmediata superior.
    - Servicio que presta un proceso (protocolo) de capa "N" a varios procesos (protocolos) de capa "N+1".
    - A través de los SAP (*Services Access Point*), puertas lógicas entre capas adyacentes.
      - Se le suele llamar "PUERTO" en la capa de transporte de la arquitectura TCP/IP.
  - Multiplexación (Mx).
    - Varios procesos de capa "N+1" pasan datos a un proceso de capa "N".
  - Demultiplexación (DMx).
    - Un proceso de capa "N " pasa datos a varios procesos de capa "N+1".

*: Multiplexación/demultiplexación **lógica**, diferente al concepto homónimo de la capa física.

La capa de transporte: multiplexación/demultiplexación.

12



Socket: abstracción con la que se denomina **cada extremo de la comunicación lógica entre procesos de capa de aplicación**, ubicados normalmente en sistemas finales (hosts) diferentes.
A su través las capas de aplicación y de transporte se "pasan" datos. Actúa como "intermediario lógico" entre procesos de ambas capas.

La capa de transporte:
- Recoge datos de diferentes procesos en un sistema origen → Mx.
- Entrega esos datos a sendos procesos en el sistema destino → DMx.

Figure 3.2 ♦ Transport-layer multiplexing and demultiplexing

Distintos procesos (aplicaciones) en un sistema pueden requerir los mismos servicios de la capa de transporte (mismo protocolo de transporte).
- P.e: DNS y TFTP utilizan UDP; la Web, FTP y email utilizan TCP.
- Para distinguir cada proceso la capa de transporte utiliza nºs de "puertos".

La capa de transporte: multiplexación/demultiplexación.

13

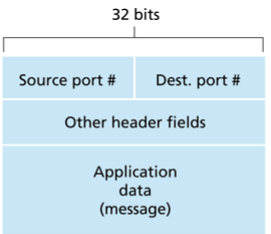
- Identificación de sockets.
  - Cada *socket* tiene un identificador único, según el correspondiente protocolo de Transporte.
    - *Socket* UDP, identificado por dos elementos.
      - Dirección IP destino - puerto destino.
      - Identificado sólo por el destino.
    - *Socket* TCP, identificado por cuatro elementos.
      - Dirección IP origen - puerto origen y dirección IP destino - puerto destino.
      - Identificado por el origen y por el destino.

La capa de transporte: multiplexación/demultiplexación.

14

- ¿Cómo funciona la demultiplexación?
  - Cada datagrama IP que se recibe en un host porta:
    - Las direcciones IP origen y destino.
    - Un segmento (TCP o UDP).
    - Cada segmento porta los nºs de puerto origen y destino.

Con esta información los sistemas finales son capaces de entregar la información contenida en un segmento TCP o UDP al *socket* apropiado.



Nº de bits por puerto = 16
Rango de valores posibles: 0 - 65535
Nºs de puertos restringidos (reservados): 0 - 1023
Nºs de puertos disponibles (no reservados): 1024 - 65 535

Figure 3.3 ♦ Source and destination port-number fields in a transport-layer segment

La capa de transporte: multiplexación/demultiplexación.

15

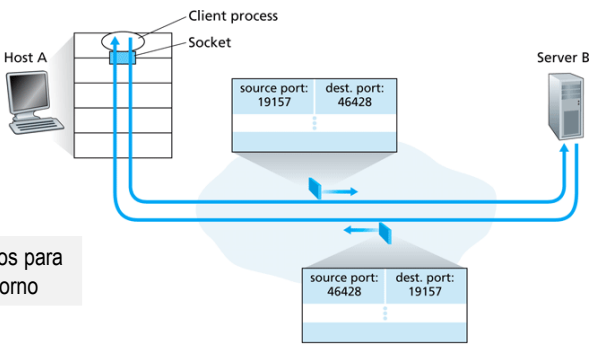
- Demultiplexación en modo NO conectivo (UDP).
  - *Socket* UDP se identifica por su destino: dirección IP y puerto destino.
  - Cuando un host recibe un segmento UDP:
    - Comprueba el nº de puerto destino.
    - Entrega los datos que porta el segmento al *socket* con nº de puerto indicado.
  - Datagramas IP con:
    - Diferente orígen (dirección IP y/o nº de puerto) e
    - Igual destino (dirección IP y nº de puerto).

→ Los datos que portan los segmentos UDP se entregan al mismo *socket*.

La capa de transporte: multiplexación/demultiplexación.

16

- Demultiplexación en modo NO conectivo (UDP) (II).
  - ¿Para qué enviar en cada datagrama IP la dirección y puerto origen?
    - Para utilizarlo, si es necesario, como dirección de retorno.



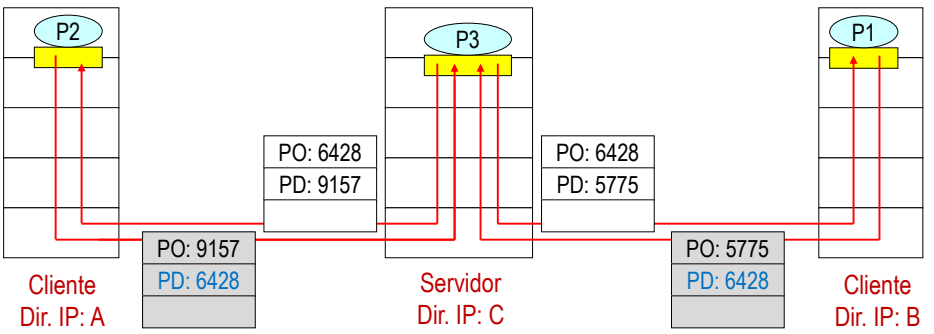
Inversión de puertos para dirección de retorno

Figure 3.4 ♦ The inversion of source and destination port numbers

La capa de transporte: multiplexación/demultiplexación.

17

- Demultiplexación en modo NO conectivo (UDP) (III).



PO: Puerto Origen.
PD: Puerto Destino.

El puerto origen proporciona la dirección de retorno.

Datagramas IP con diferentes orígenes e igual destino → Datos que portan los segmentos UDP se entregan al mismo *socket*

La capa de transporte: multiplexación/demultiplexación.

18

- Demultiplexación en modo conectivo (TCP).
  - *Socket* TCP se identifica por cuatro elementos:
    - Dirección IP destino - puerto destino / Dirección IP origen - puerto origen.
  - Sistema final receptor utiliza los cuatro elementos para entregar los datos que porta el segmento TCP al *socket* apropiado.
    - Segmentos TCP con orígenes diferentes (IP y/o puerto) al mismo destino (IP y puerto) → Sockets diferentes en el receptor.

La capa de transporte: multiplexación/demultiplexación.

19

- Demultiplexación en modo conectivo (TCP) (II).
  - Un host servidor puede soportar varios *sockets* abiertos simultáneamente.
    - Cada uno identificado por cuatro elementos.
    - Un *socket* de "bienvenida".
    - Tantos *sockets* "de conexión" como conexiones TCP simultáneas haya.

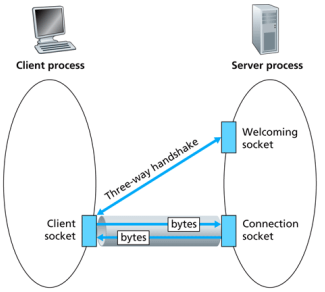


Figure 2.28 ♦ Client-socket, welcoming socket, and connection socket

La capa de transporte: multiplexación/demultiplexación.

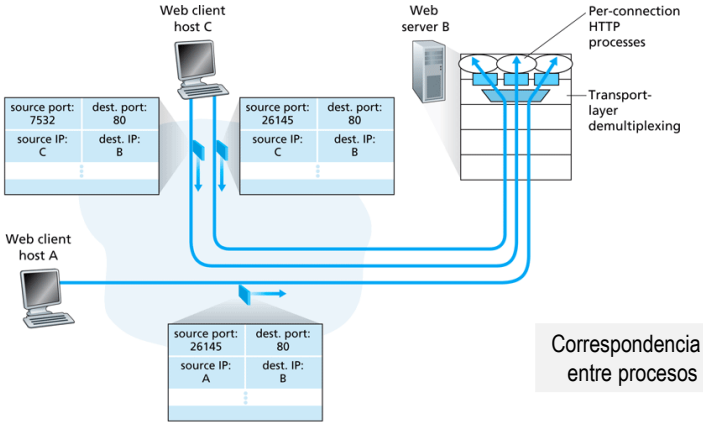
20

- Demultiplexación en modo conectivo (TCP) (III).
  - P.e, servidor Web.
    - Todas las peticiones para la conexión TCP inicial y para cada objeto al puerto 80 TCP.
      - Servidor diferencia las peticiones según su origen (dirección IP y nº de puerto origen).
    - Dispone de diferentes *sockets* para cada cliente con el que mantiene sesiones HTTP.
      - HTTP con TCP no persistente → Diferentes *sockets* para cada petición.
      - HTTP con TCP persistente → Mismo *socket* para cada petición.
    - Por cada conexión TCP se genera un proceso de aplicación en el servidor.
      - Cada proceso accesible a través del correspondiente *socket* de conexión.
      - No siempre la correspondencia es "uno a uno" entre *sockets* y procesos.
      - En servidores de altas prestaciones la correspondencia es "uno a varios".
        - Proceso principal genera varios subprocesos ("hilos").
        - Para cada subproceso ("hilo") un *socket*.

La capa de transporte: multiplexación/demultiplexación.

21

- Demultiplexación en modo conectivo (TCP) (IV).



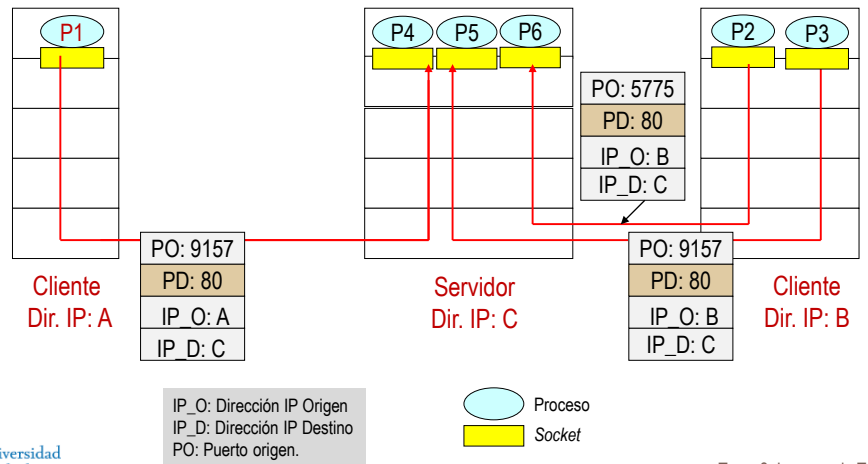
Correspondencia "uno a uno" entre procesos y *sockets*.

Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

La capa de transporte: multiplexación/demultiplexación.

22

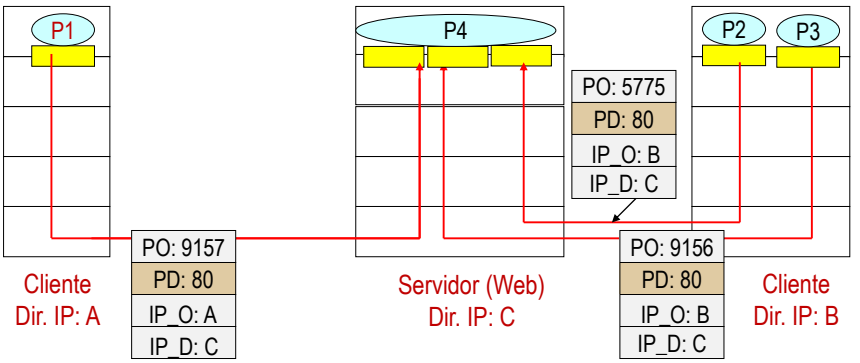
- Demultiplexación en modo conectivo (TCP) (V).
  - Correspondencia "uno a uno" entre procesos y *sockets*.



La capa de transporte: multiplexación/demultiplexación.

23

- Demultiplexación en modo conectivo (TCP) (VI).
  - Correspondencia "uno a varios" entre procesos y *sockets*.



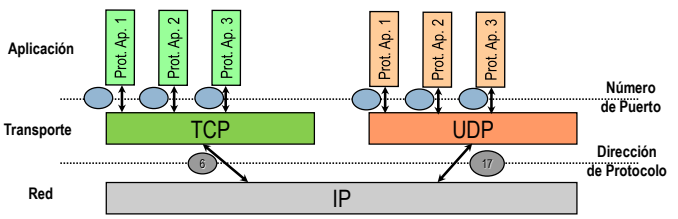
IP_O: Dirección IP Origen
IP_D: Dirección IP Destino
PO: Puerto origen.
PD: Puerto Destino.

Proceso
Socket

La capa de transporte.

24

- Niveles de direccionamiento en host.
  - Se requiere tres niveles de direccionamiento en cada host para:
    - Identificación de host (dirección de host).
      - Contenida en la cabecera del datagrama/paquete IP.
    - Identificación de protocolo de transporte (o protocolo encapsulado en IP).
      - Contenida en la cabecera del datagrama/paquete IP.
    - Identificación de proceso.
      - Contenida en la cabecera del segmento UDP/TCP (puerto).



La capa de transporte: UDP.

25

- UDP, *User Datagram Protocol* (RFC 768).
  - Muy "ligero", hace casi lo mínimo que debe hacer un prot. de transporte.
    - Multiplexado y demultiplexado.
    - Comprobación de errores.
    - Prácticamente la aplicación se comunica "directamente" con la capa IP.
  - No conectivo y sin mecanismos de:
    - Segmentación, fiablilidad, control de flujo, control de congestión, control de secuencia.
  - No tiene "estados" → No requiere gestionar parámetros de control, p.e:
    - Nºs de secuencia, temporizadores, asentimientos, buffers temporales, etc.
    - Facilita que un servidor puede soportar muchos clientes activos al mismo tiempo.
  - Consecuencia: UDP introduce poca sobrecarga de encapsulado.

La capa de transporte: UDP.

26

- ¿Qué aplicaciones requieren servicios como los que provee UDP?
  - Aplicaciones que por sí mismas requieren controlar qué datos se envían y en qué momento.
    - UDP pasa a la capa IP los datos tan pronto se los entrega la capa de aplicación.
  - Aplicaciones de tiempo real.
    - Requieren mínima velocidad de transmisión.
    - No admiten retardo excesivo.
    - Toleran algunas pérdidas.
  - Aplicaciones que por su naturaleza operan mejor con la rapidez de UDP.
    - P.e: gestión de red (SNMP), DNS, encaminamiento IP, etc.
  - Tales aplicaciones, si lo requieren, tendrán que implementar los servicios que UDP no provee y "pagar" por ello (p.e, TFTP).

La capa de transporte: UDP.

27

- Aplicaciones populares de Internet y protocolos de transporte que utilizan.

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Figure 3.6 ♦ Popular Internet applications and their underlying transport protocols

La capa de transporte: UDP.

28

- Problemas potenciales de UDP.
  - No dispone de mecanismos de control de congestión, por lo que:
    - Muchas aplicaciones utilizando UDP pueden congestionar la red y producir elevadas pérdidas de paquetes por colas llenas en los *routers*.
    - Resultado:
      - Aplicaciones sobre UDP experimentan tasas de error muy altas.
      - "Estrangulamiento" de las sesiones sobre TCP.
  - Consecuencia.
    - Algunos ISP bloquean el tráfico UDP.
    - Cada vez más se utiliza TCP para el transporte de flujos multimedia.

La capa de transporte: UDP.

29

- Estructura del segmento UDP.

Pseudo cabecera para el cálculo del ***checksum***.

- Tres campos de la cabecera del paquete IP que porta al segmento UDP:
  - IP origen.
  - IP destino.
  - Protocolo.
- Campo "longitud" de la cabecera UDP.*

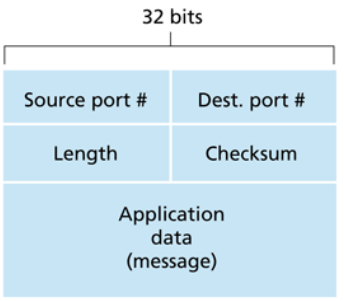


Figure 3.7 ♦ UDP segment structure

34

La capa de transporte: UDP.

30

- Campos de cabecera UDP.
  - Puerto Origen (opcional, si no se utiliza se pone a cero).
    - Nº del puerto emisor de los datos en el host origen.
    - Si no se indica lo contrario es también el puerto receptor para los Ms de respuesta.
  - Puerto Destino.
    - Nº del puerto receptor de los datos en el host destino.
  - Longitud.
    - Tamaño en octetos del segmento UDP completo (cabeceras y cuerpo).
  - *Checksum* (suma de verificación, opcional en IPv4 y obligatorio en IPv6).
    - Mecanismo para detección de errores. Si no se utiliza se pone a cero.
    - Contiene el complemento a 1 de la suma de todas las palabras de 16 bits del segmento UDP más la **pseudo cabecera**, con acarreo del bit de desbordamiento de la suma sobre el bit de menor peso.
    - Procedimiento resumido en el subapartado 3.3.2, página 167 del libro de texto.

La capa de transporte: UDP.

31

- ¿Cómo procede UDP en el destino para verificar el *checksum*?
  1. Obtiene la **pseudo cabecera** del paquete IP que porta el segmento UDP.
  2. Almacena el valor del *checksum* contenido en el segmento UDP recibido.
  3. Pone a cero el campo *checksum* del segmento UDP recibido y calcula el *checksum* correspondiente (del segmento UDP + la pseudo cabecera*).
     - *Checksum* calculado = *Checksum* recibido → Segmento UDP sin error.
     - *Checksum* calculado ≠ *Checksum* recibido → Segmento UDP con error.

*: el valor de la DIRECCIÓN IP DESTINO para el cálculo del *checksum* NO SE OBTIENE DE LA CABECERA DEL PAQUETE IP RECIBIDO, sino **se obtiene de la dirección IP del host receptor**. Esto permite detectar errores de encaminamiento.

La capa de transporte: UDP.

32

- Utilidad del campo *checksum*.
  - Sólo para detección de errores extremo a extremo (no para recuperarse de ellos), pues no hay garantía respecto a que:
    - Todos los enlaces entre origen y destino dispongan de mecanismos para detectar errores.
    - No se produzcan errores internamente en los nodos intermedios.
- ¿Cómo actúa la lógica de UDP cuando el *checksum* detecta error?
  - Descarta el segmento corrupto o
  - Pasa los datos a la capa de aplicación con una advertencia.

La capa de transporte: UDP.

33

- Tabla resumen UDP.

Nombre de la PDU	Segmento/Datagrama UDP
Tamaño de cabecera (PCI)	8 octetos
Orientado a conexión	NO
Segmentación de datos de aplicación	NO
Detección de errores	Opcional
Fiabilidad	NO
Control de secuencia	NO
Control de flujo	NO
Control de congestión	NO