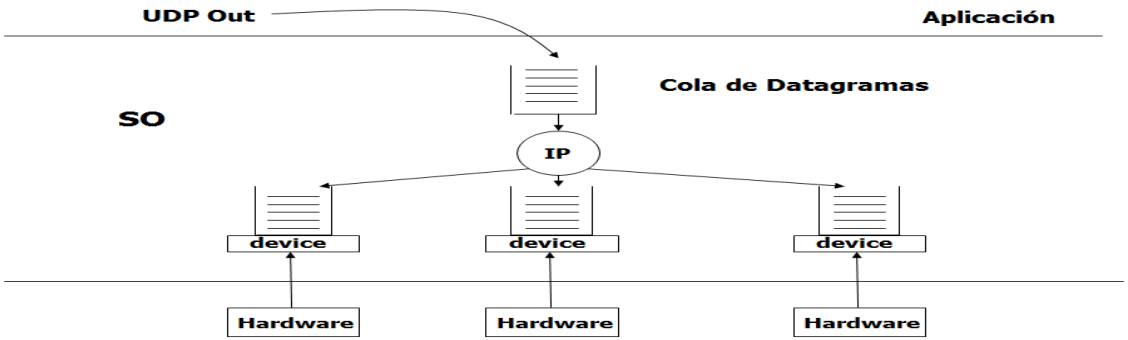
Redes
Nivel de Transporte – UDP

Luis Marrone

LINTI–UNLP

14 de abril de 2015

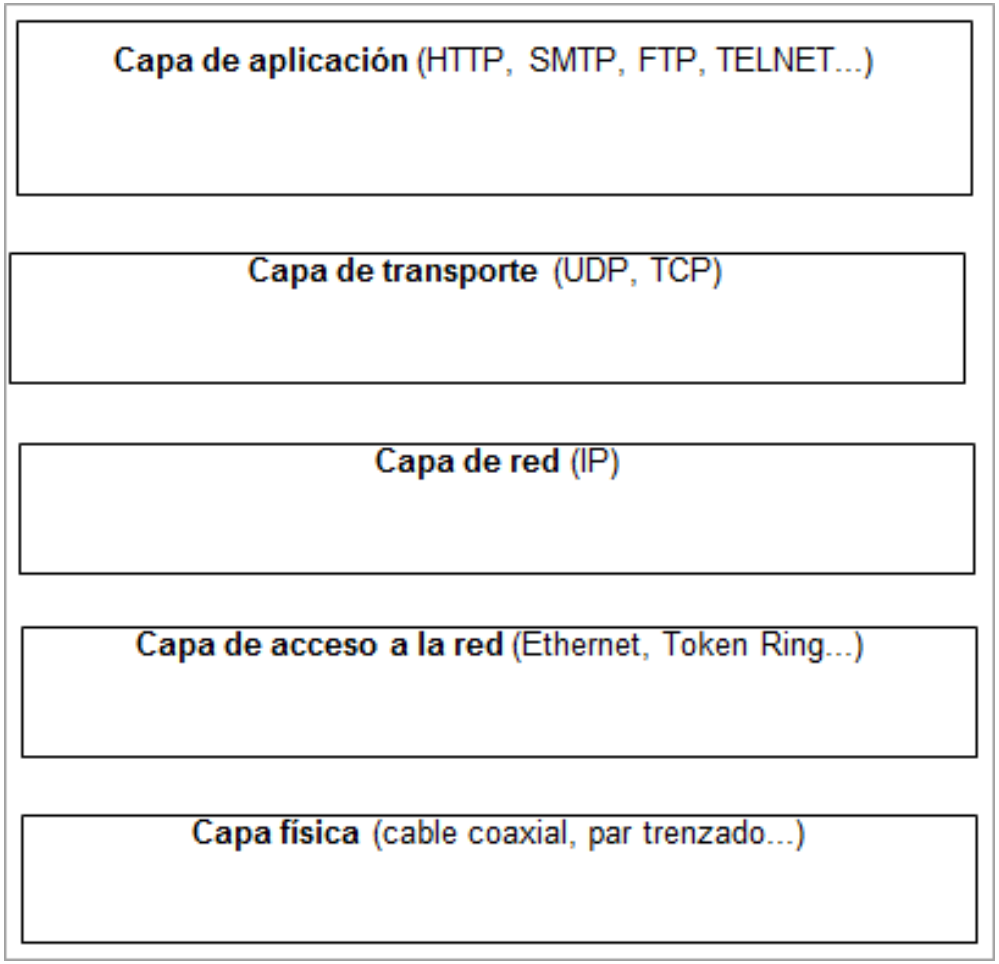


Contenidos

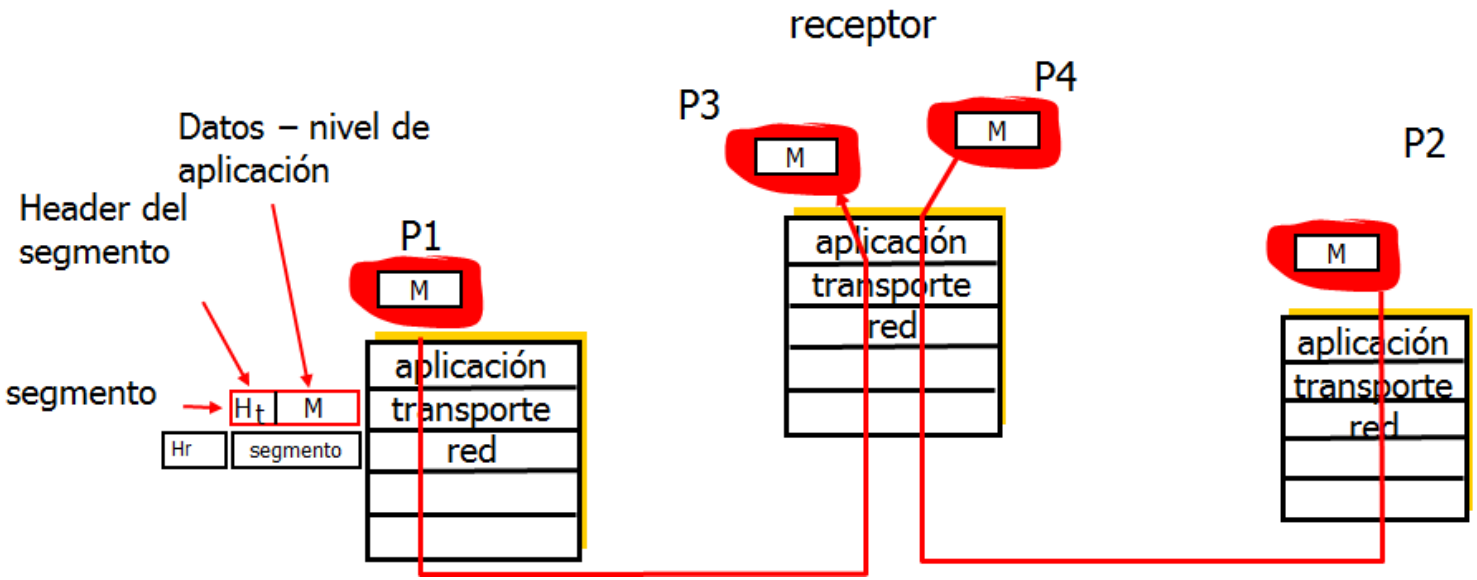
Características

- ✓ Implementados en los extremos
- ✓ Interfase con la aplicación
- ✓ Interfase con el sistema operativo
- ✓ Multiplexación
  - Colectar datos de múltiples procesos
- ✓ Demultiplexación
  - Entrega los segmentos a los procesos

Modelo TCP/IP



Funcionalidad



Implementación

- ✓ Identificación:
  - Imposibilidad de extender las direcciones IP
  - No quedan bits disponibles
  - No se puede utilizar parámetros OS-dependent
    - Process ID
    - Task number
    - Job name
  - Debe funcionar en todos los sistemas

Implementación...

Se crea una nueva abstracción

- ✓ “protocol port number”
- ✓ Identifica unívocamente la aplicación transmisora y/o receptora
- ✓ Independiente del SO
- ✓ Interactúa con el SO
- ✓ Utilizado en protocolos TCP/IP
- ✓ Reside en el SO
- ✓ Compartido con diferentes aplicaciones

Algunas Definiciones

Proceso

- ✓ Cómputo que se realiza en forma independiente de otros
- ✓ Creado por el system call “create”
  - procid=create(arg?.), procid = entero
- ✓ Finalizado con kill(procid)
- ✓ Independiente del código/programa que ejecuten
- ✓ Los procesos comparten código a través del SO

Procesos

- ✓ Si debe esperar la ocurrencia de un evento, se bloquea
- ✓ Ocurrido el evento $\Longrightarrow$ “resume”
- ✓ Se le asignan prioridades
- ✓ Intercambian información entre ellos
- ✓ Mecanismos de Comunicación
  - Semáforos
  - Mensajes
  - Ports

Ports

- ✓ Cola finita de mensajes con dos semáforos que controlan su acceso
- ✓ Se crea con “pcreate(size)” especificando el tamaño de la cola
  - ✓ Portid = pcreate(size)
  - ✓ Número de 16 bits
- ✓ Se envían con:
  - psend(portid,mensaje); /* es el productor */
- ✓ Se recibe con:
  - Message = preceive(portid); /* es el consumidor */

Ports...

- ✓ Indica su estado:
  - N = pcount(portid); /* da la cantidad de mensajes en el port */
- ✓ Port: identificador de proceso que desea comunicarse con otro
  - Bien conocidos (well known ports) asignados por el IANA (Internet Assigned Number Authority) 0 – 1023
  - Efímeros: proporcionados por el host a los procesos

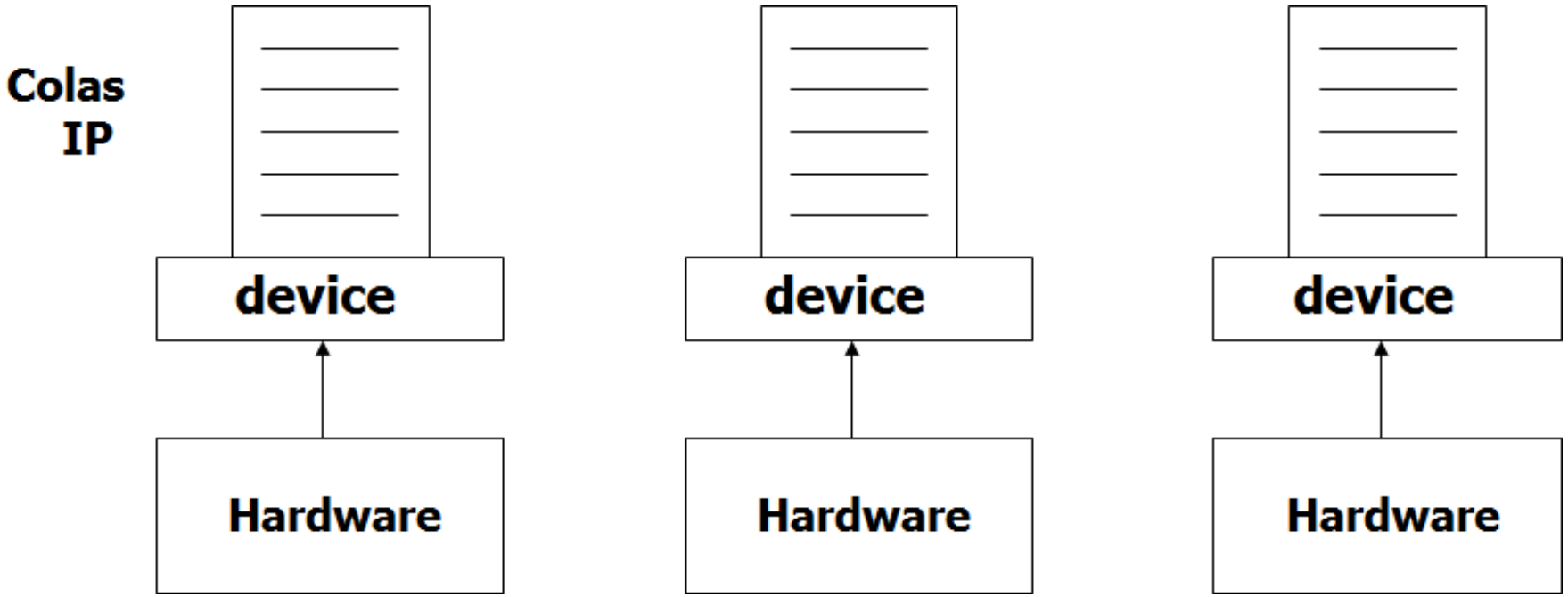
Mensajes

- ✓ La interfaz de red los recibe mediante mecanismos de interrupción
- ✓ Interrupción:
  - Ejecuta el código del “device driver”
  - Asociado a una operación de I/O
  - write(device,buff,len)
  - Read $->$es similar

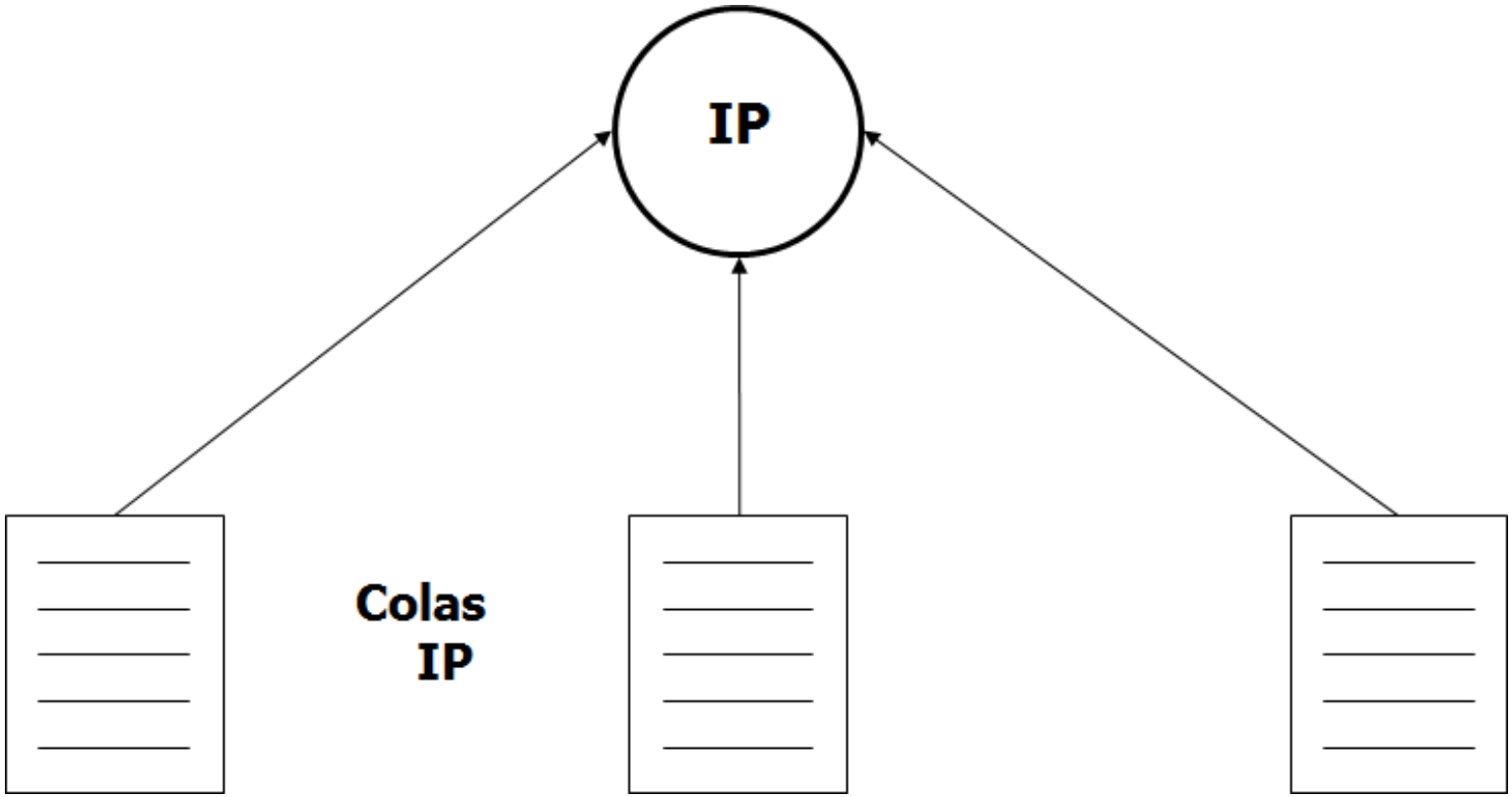
- ✓ La interfaz de red los recibe mediante mecanismos de interrupción
- ✓ Interrupción:
  - Ejecuta el código del “device driver”
  - Asociado a una operación de I/O
  - write(device,buff,len)
  - Read $->$es similar

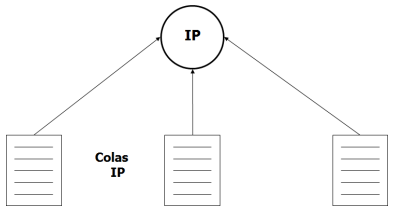
Device: descriptor de la interfaz. Buff: dirección de almacenamiento de la trama. Len: longitud de la trama en bytes

Interrupciones



Interrupciones. . .





La interrupción ethernet utiliza el “protocol type” y determina que llegó un datagrama IP.
El código del device no procesa el paquete, debe regresar de la interrupción rápido.
Es decir, la interrupción no llama a IP directamente, el sistema implementa colas junto con el pasaje de mensajes.

Cuando llega un datagrama la interrupción lo encola y ejecuta el send para avisarle a IP.

Sockets

- ✓ Descriptor de archivo que un proceso utiliza para solicitar servicios de red al sistema operativo.
- ✓ Dirección de socket
  - (protocolo, proceso local, dirección local)
  - (tcp, 2456, 193.44.234.3).
- ✓ Sesión
  - Quíntupla
  - (protocolo, proc. local, direcc. local, proc. remoto, direcc. remota)
  - (tcp, 1500, 193.44.234.3, 21, 193.44.234.5)

Estamos en:

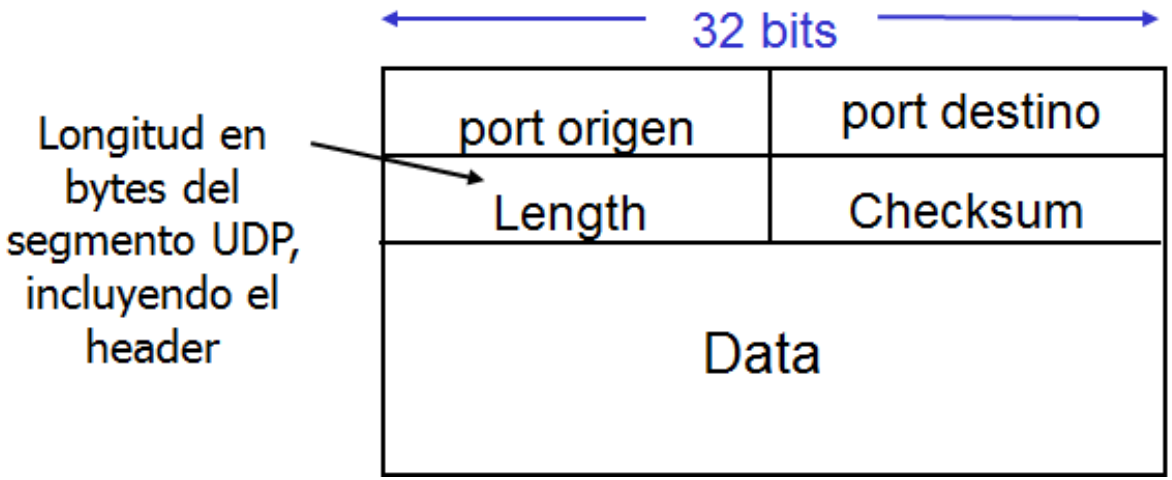
Estructura

- ✓ User Datagram Protocol (RFC 768)
- ✓ Best effort:
  - Los segmentos pueden perderse
  - Los segmentos pueden llegar desordenados
- ✓ Sin conexión
  - No hay “handshake”
  - Los segmentos son independientes
- ✓ A cada mensaje de la aplicación le corresponde un segmento

UDP . . .

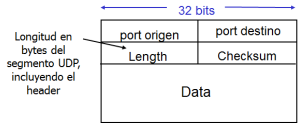
- ✓ Utilizado en aplicaciones multimediales
- ✓ DNS
- ✓ SNMP
- ✓ TFTP
- ✓ RPC
- ✓ VoIP

UDP – Estructura



- ✓ Checksum opcional
- ✓ Abarca todo el segmento

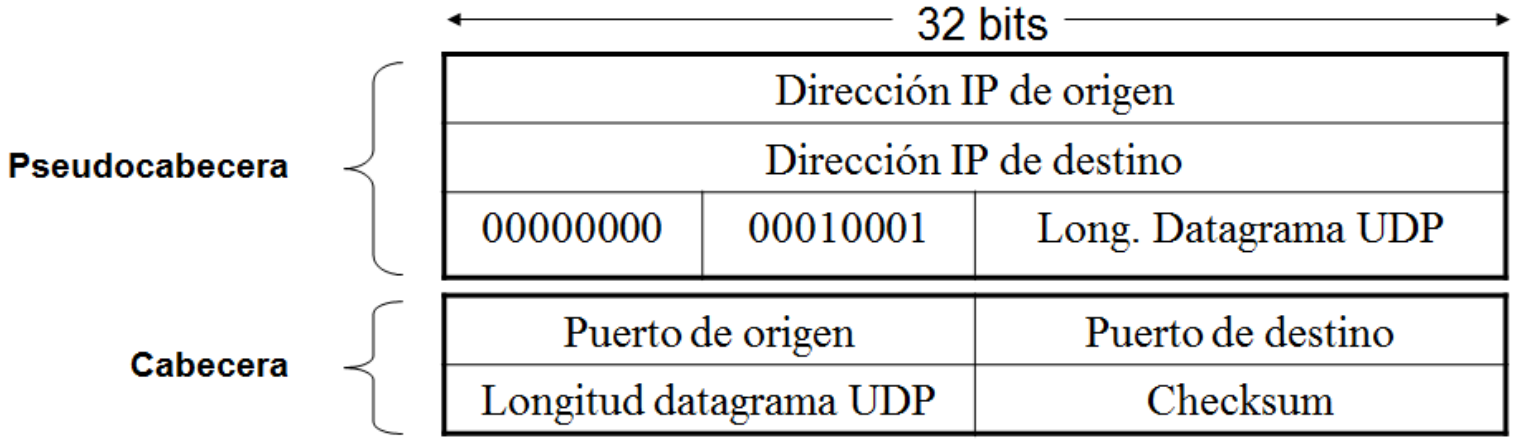
UDP – Estructura



- ✓ Checksum opcional
- ✓ Abarca todo el segmento

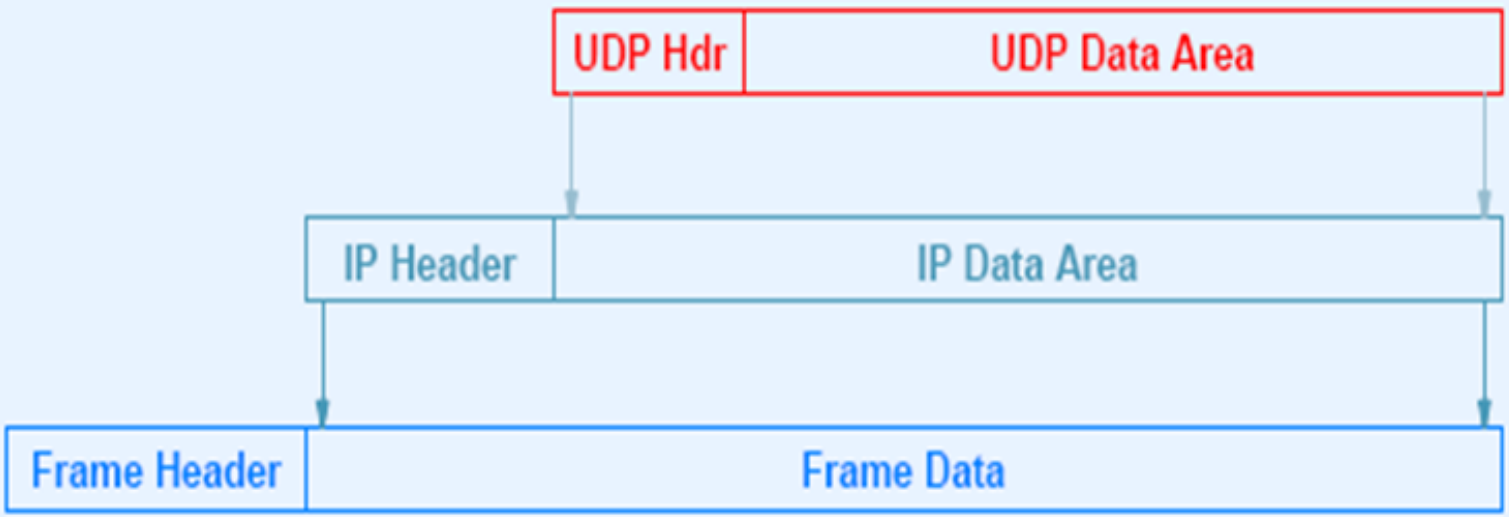
El checksum se realiza sumando en complemento a 1 el pseudo header y el segmento UDP considerado en palabras de 16 bits.
Luego se complementa el resultado.
Para llegar a palabras enteras de 16 bits en el segmento, se completa con ceros.
Si el resultado diera todos ceros, se lo reemplaza por todos unos.
Como no es mandatorio, cuando no se lo calcula, este campo se pone en cero.

Cabecera UDP



- La pseudocabecera se añade al principio del datagrama para el cálculo del checksum, pero no se envía.
  Permite a UDP comprobar que IP no se ha equivocado (ni le ha engañado) en la entrega del datagrama
  El valor $10001_2 = 17_{10}$ indica que el protocolo de transporte es UDP

Encapsulamiento UDP



UDP ports – Ejemplo

- ✓ Al servidor de DNS se le asignó el port 53
- ✓ La aplicación que requiere el servicio DNS(cliente) obitene el port 28900
- ✓ El datagrama UDP enviado desde la aplicación al servidor DNS tiene:
  - Source port number 28900
  - Destination port number 53
  - Cuando el servidor DNS contesta, el datagrama UDP tiene:
    - Source port number 53
    - Destination port number 28900

Conexión UDP Cliente–Servidor