Redes de Computadores

Nivel de Transporte: Introducción + UDP

Área de Ingeniería Telemática
Dpto. Automática y Computación
http://www.tlm.unavarra.es/

Hasta ahora

- Introducción a grandes rasgos
  - Internet
  - Arquitecturas de protocolos
  - TCP/IP
- El nivel de aplicación en Internet
  - Protocolos de aplicacion, aplicaciones comunes

Siguiente paso:

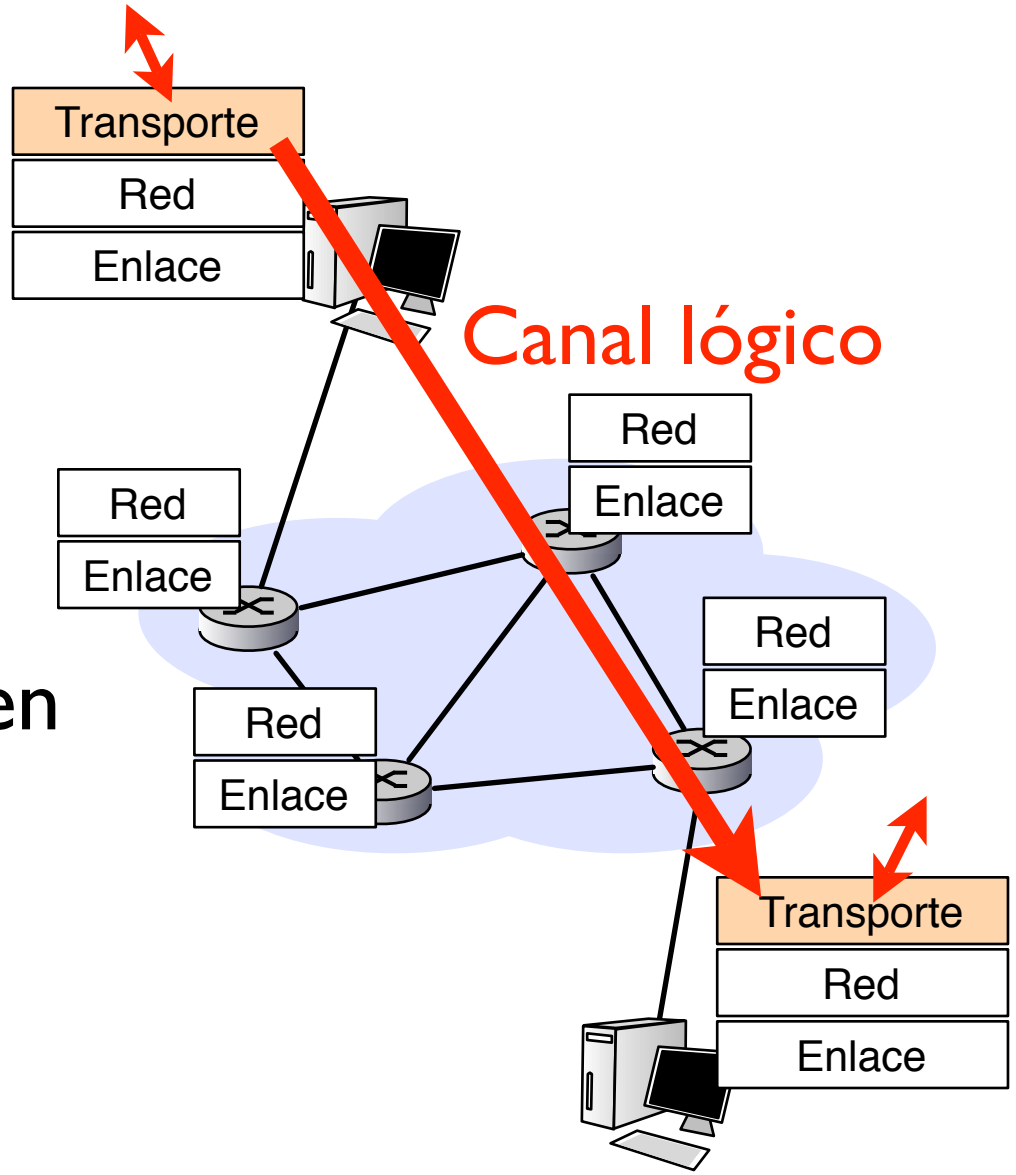
- El nivel de transporte en Internet

Objetivos

- Conceptos y principios del nivel de transporte
  - Multiplexación
  - Transporte fiable
  - Control de flujo
  - Control de congestion
- Estudio del nivel de transporte en Internet
  - Protocolos TCP y UDP

Funciones del nivel de transporte

- Comunicación lógica entre aplicaciones
- Protocolo en los extremos (end-to-end)
  - segmentación
  - reensamblado
- 2 niveles de transporte en Internet
  - TCP (SOCK_STREAM)
  - UDP (SOCK_DGRAM)

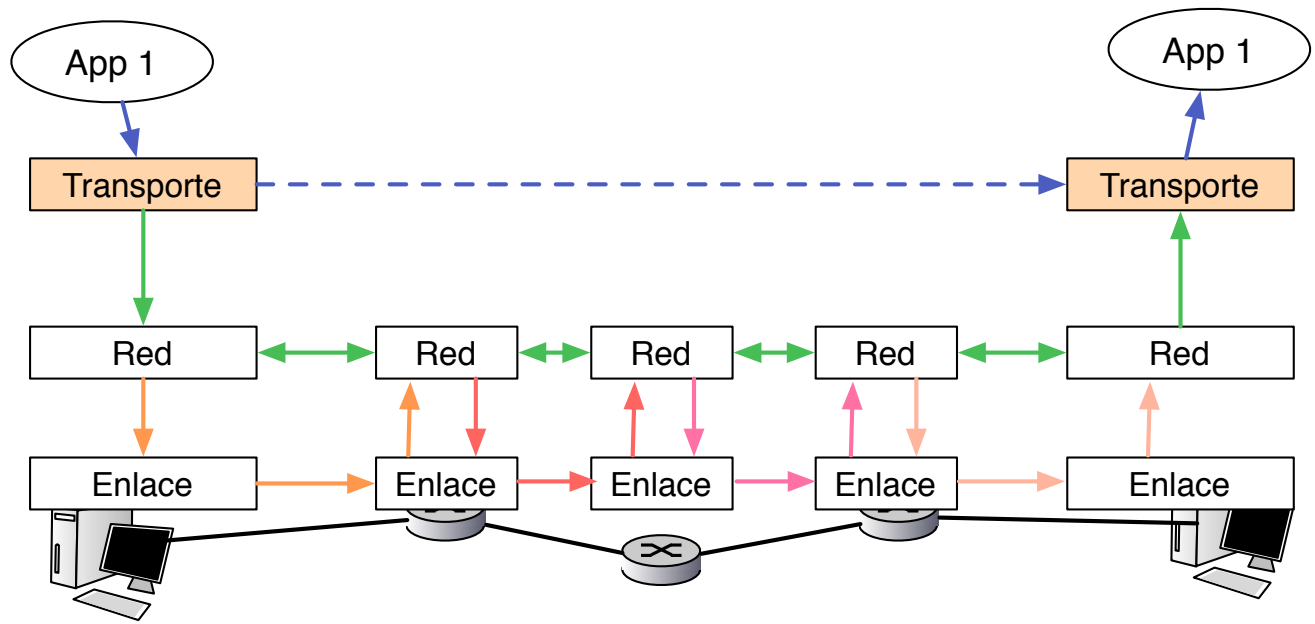


Red y transporte

- Nivel de red

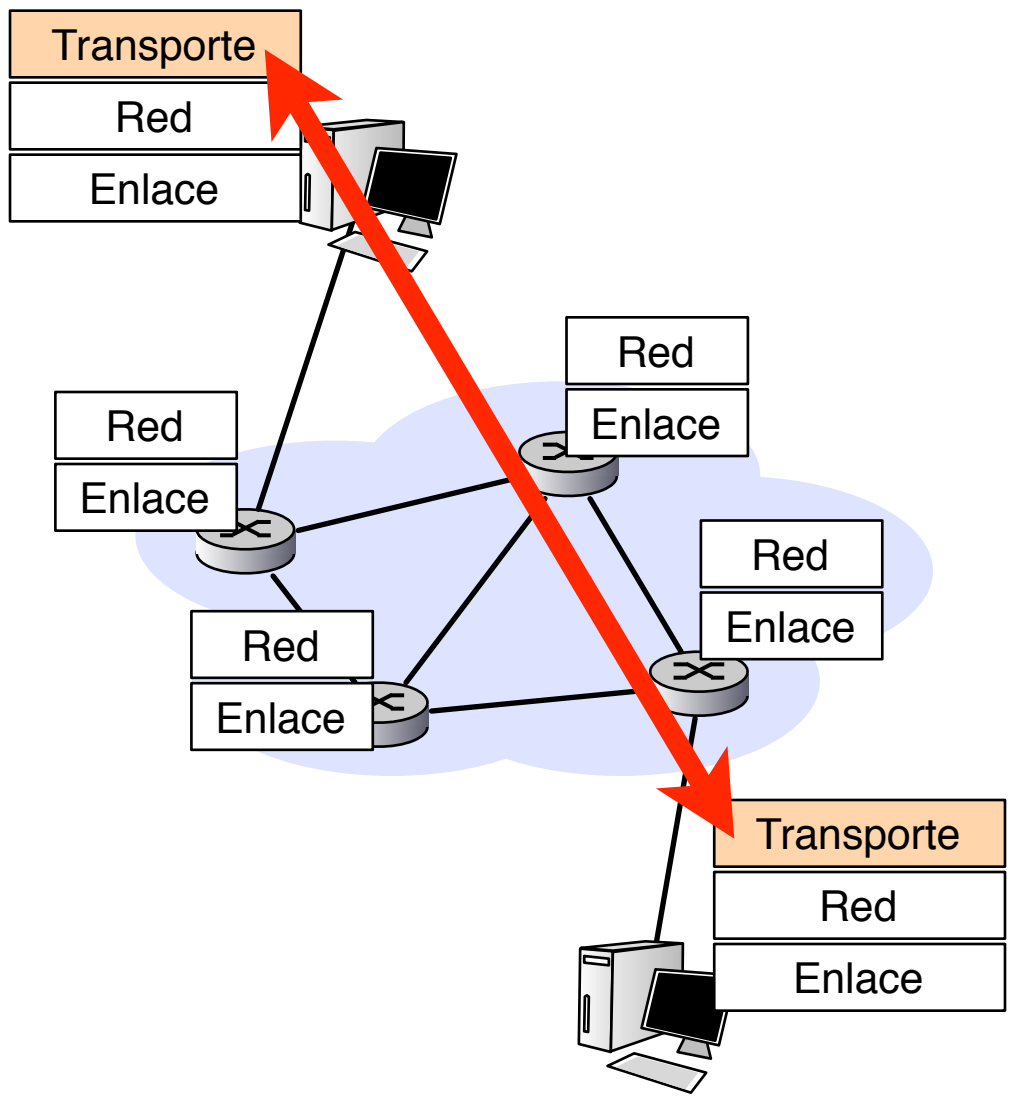
  Comunicación lógica entre **hosts**
- Nivel de transporte

  Comunicación lógica entre **procesos**
  - mejora y utiliza la comunicación entre hosts



Transporte en Internet

- Entrega fiable y en orden (TCP)
  - control de congestión
  - control de flujo
  - establecimiento de conexión
- Entrega no fiable, sin garantias de orden (UDP)
  - best-effort igual que IP
- En los dos casos
  - retardo no garantizado
  - ancho de banda no garantizado



Funciones TCP/UDP

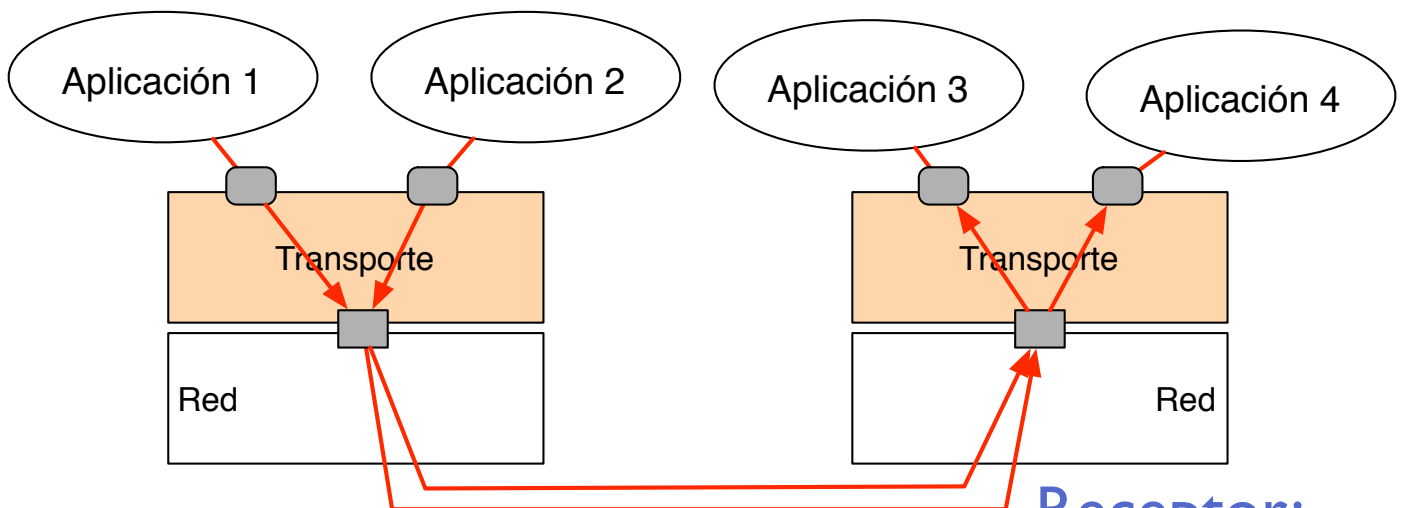
- Función común

  Multiplexación/demultiplexación de aplicaciones

- Funciones sólo UDP
  - Envio no orientado a conexión (Hoy)
- Funciones sólo TCP
  - Manejo de conexiones
  - Transporte fiable de datos
  - Control de flujo y de congestión

Multiplexación y demultiplexación

- Un host con varias aplicaciones/programas/ procesos corriendo
  - El nivel de red envia los paquetes al nivel de red del ordenador destino
  - El nivel de transporte arbitra la comunicación entre diferentes aplicaciones
  - **un** nivel de red, **un** nivel de transporte, **varias** aplicaciones



Emisor:

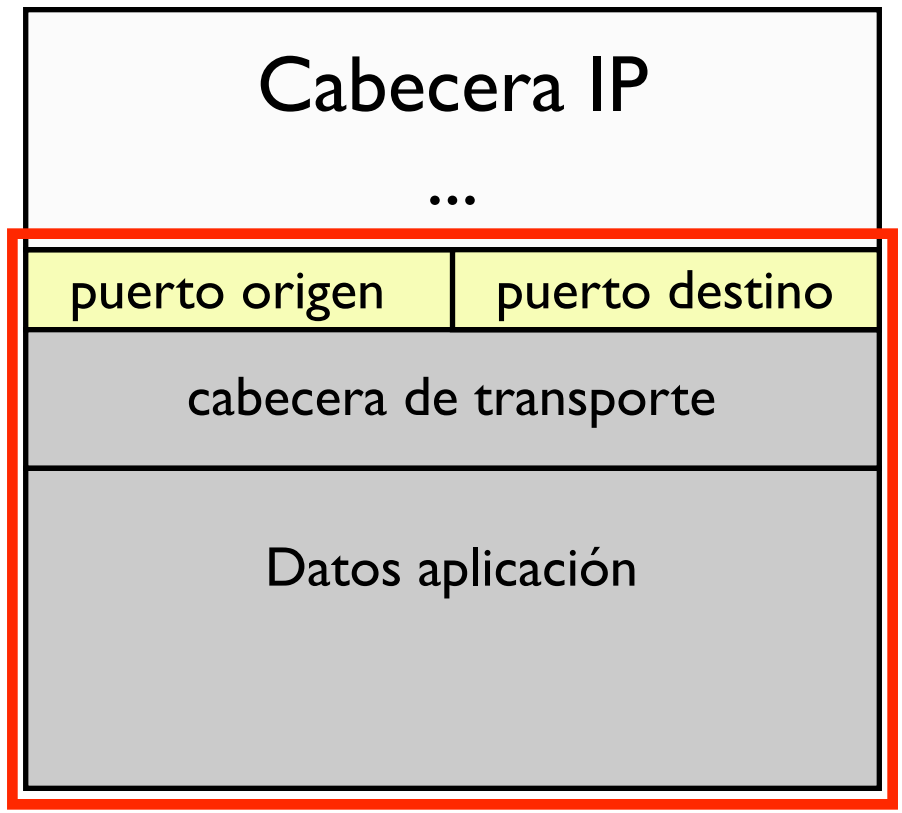
- recoger datos de distintas fuentes
- encapsular con información del origen

Receptor:

- entregar a la aplicación (socket) correcta

Multiplexación y demultiplexación

- El host recibe datagramas IP
  - Cada datagrama IP lleva una dirección IP de origen y de destino
  - Cada datagrama IP lleva un segmento del nivel de transporte
  - Cada segmento del niel de transporte lleva un puerto origen y destino (puerto: identificador de proceso/aplicación)
- El nivel de tranporte usa las direcciónes IP y los puertos para decidir a quien entrega los datos



Segmento del nivel de transporte

Uso de los puertos

- Servidores usan puertos conocidos por el cliente
  - El cliente conoce la dirección del servidor y el puerto según el servicio

    ej: para cliente web: *http://www.unavarra.es* significa puerto 80

    para cliente ssh: *ssh 10.1.1.21* significa puerto 22
  - O bien el cliente puede permitir especificarlo

    ej: http://miweb.com:8000 ssh -p 30 10.1.1.21
  - El IANA reserva el rango de puertos 0-1023 como well-known ports
  - Normalmente solo root tiene acceso a usar ese rango
- Rango de puertos registrados
  - El IANA lleva el registro de puertos asignados a servicios concretos

    Se reserva el rango 1024-49151 para aplicaciones que registran puertos
  - Cualquier usuario puede pedir con *bind()* un puerto en este rango
  - El registro no prohibe a otros usar el puerto

    Vea las asignaciones en */etc/services*

Uso de los puertos

- Los clientes usan puertos efímeros
  - Si el cliente no hace *bind()* el sistema operativo le elige un puerto
  - Normalmente en el rango no reservado del IANA 49152-65535
  - Depende del sistema operativo (antiguamente en 1024-5000)
- Normalmente el sistema operativo...
  - Requiere permiso de administrador para hacer *bind()* en 0-1023
  - Elige puertos efimeros para los sockets que empiezan a enviar sin hacer *bind()*
  - No deja hacer *bind()* a un puerto si ya hay otro proceso que tiene un socket en ese puerto

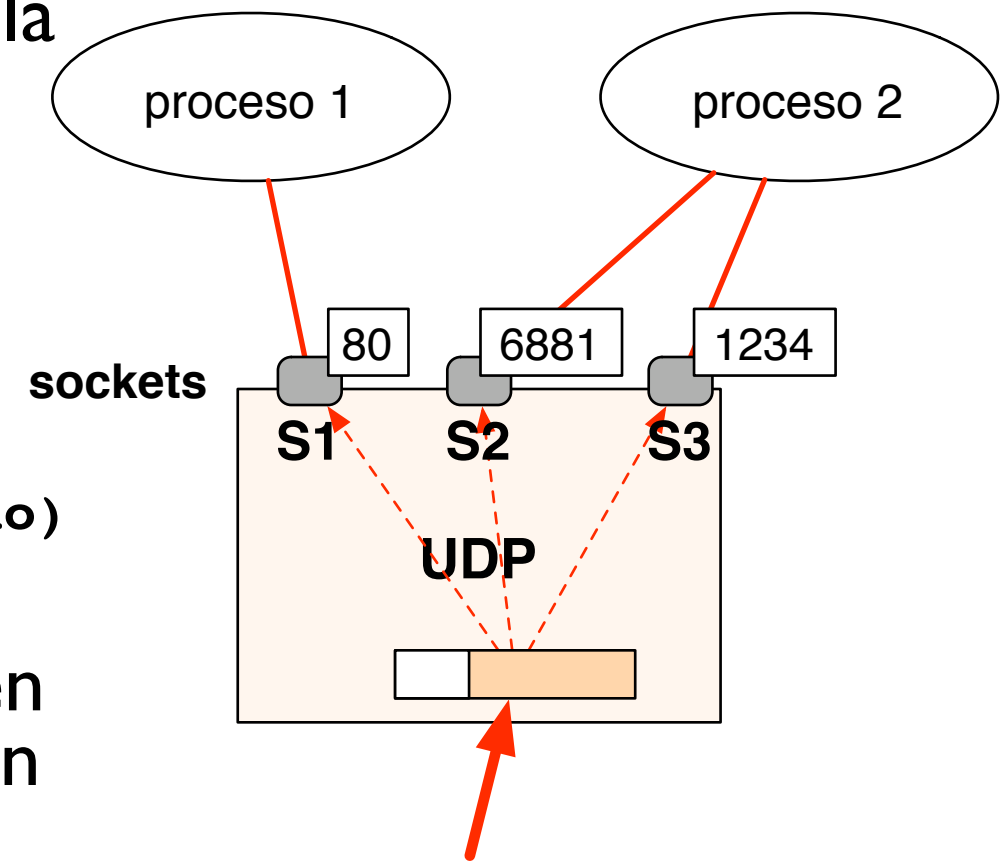
    ¿Esto quiere decir que no puede haber dos sockets en el mismo puerto?

    Salvo que sean varios sockets generados al hacer *accept()* sobre un socket de servidor, los sockets que devuelve *accept()* tienen el mismo puerto que el socket que los generó

Demultiplexación no orientada a conexión

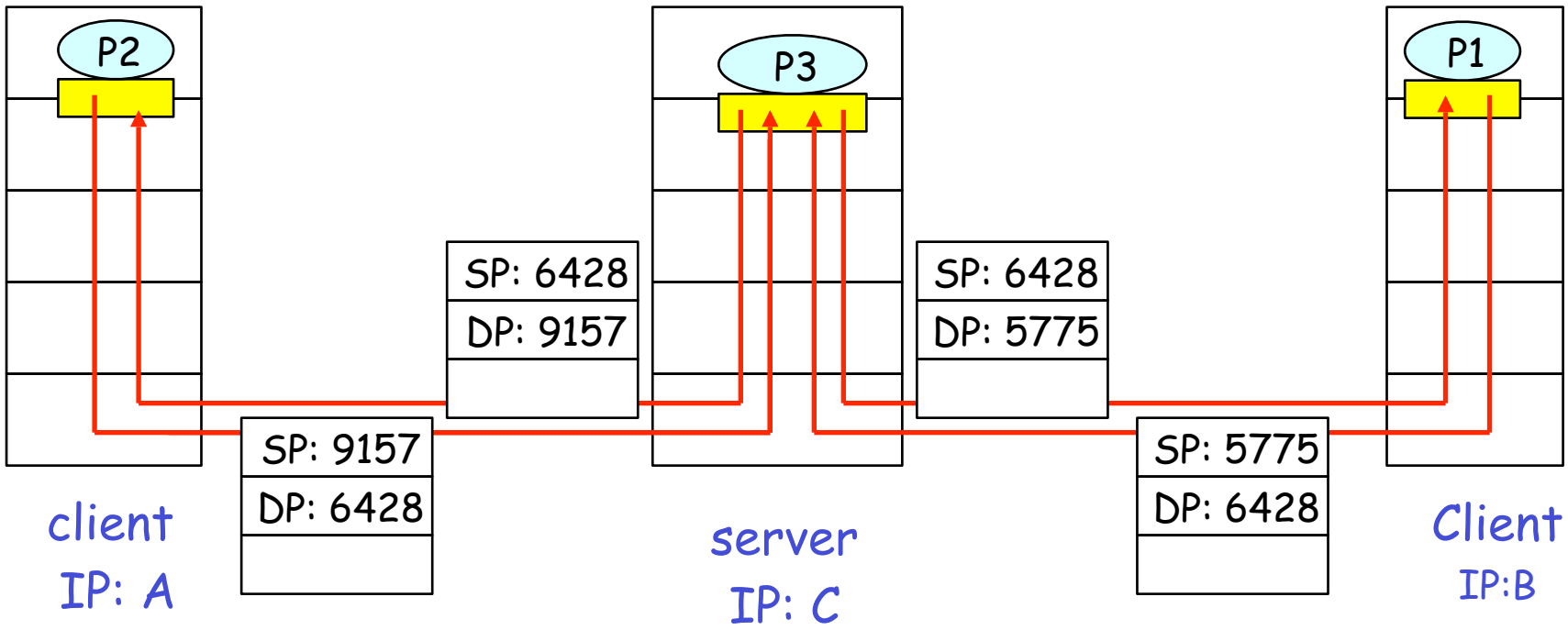
- Se extraen de la cabecera IP la dirección IP destino y de la cabecera UDP el puerto destino
- Se entregan los datos al socket identificado por la tupla

  `(dirIP destino, puerto destino)`
- Paquetes provenientes de diferentes direcciónes origen y puertos origen se entregan al mismo socket



Demultiplexación no orientada a conexión

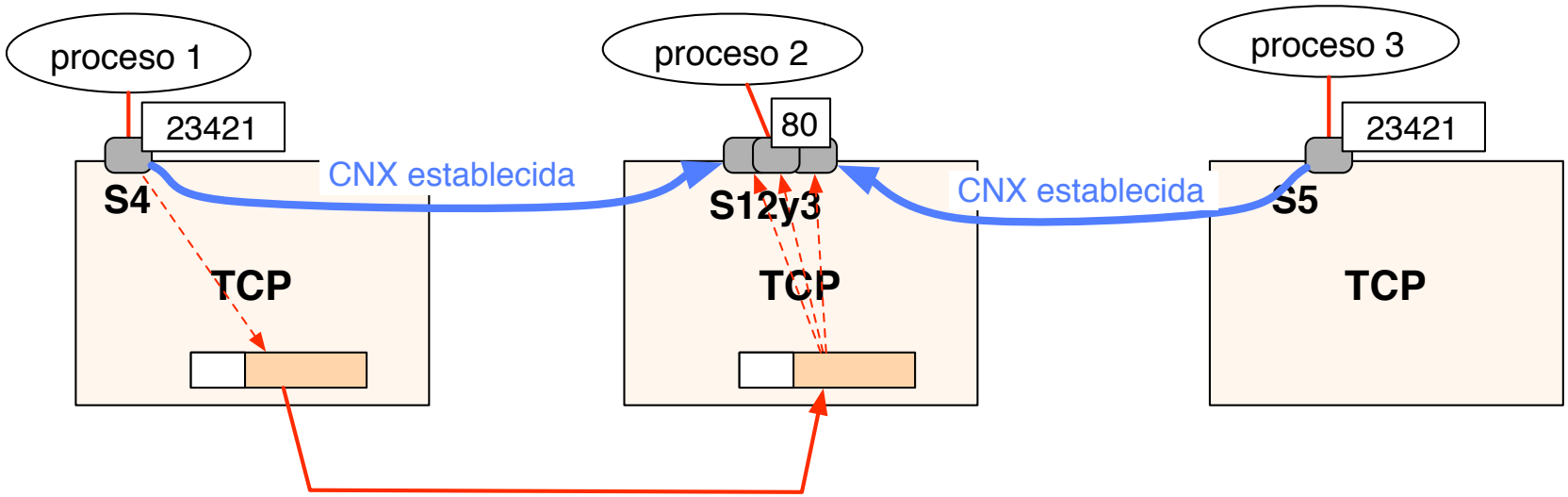
- Ejemplo



- La dirección origen y puerto origen permiten responder al cliente

Demultiplexación orientada a conexión

- Cuando TCP recibe un paquete puede pertenecer a varias conexiones establecidas
- Varios sockets con el mismo puerto...

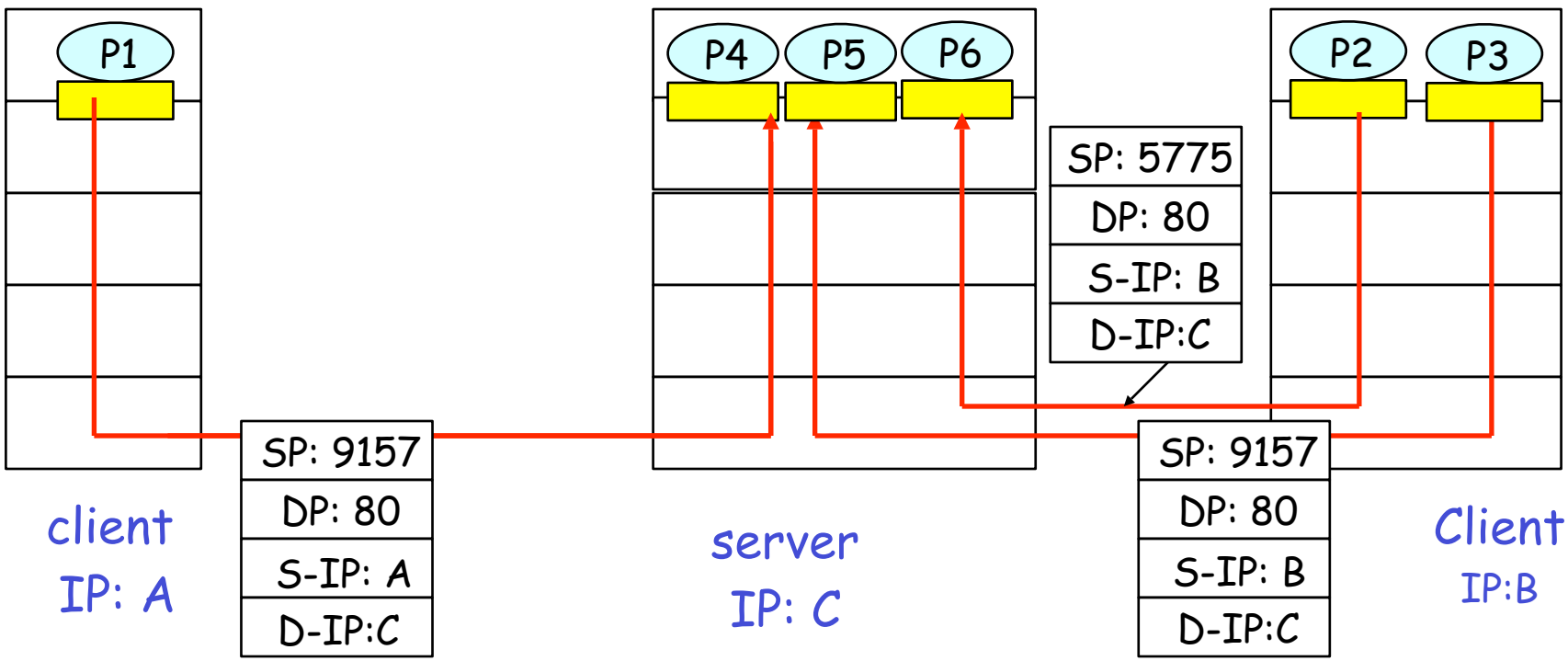


- Se entrega al socket identificado por la 4-tupla

`(dirIP origen, puerto origen, dirIP destino, puerto destino)`

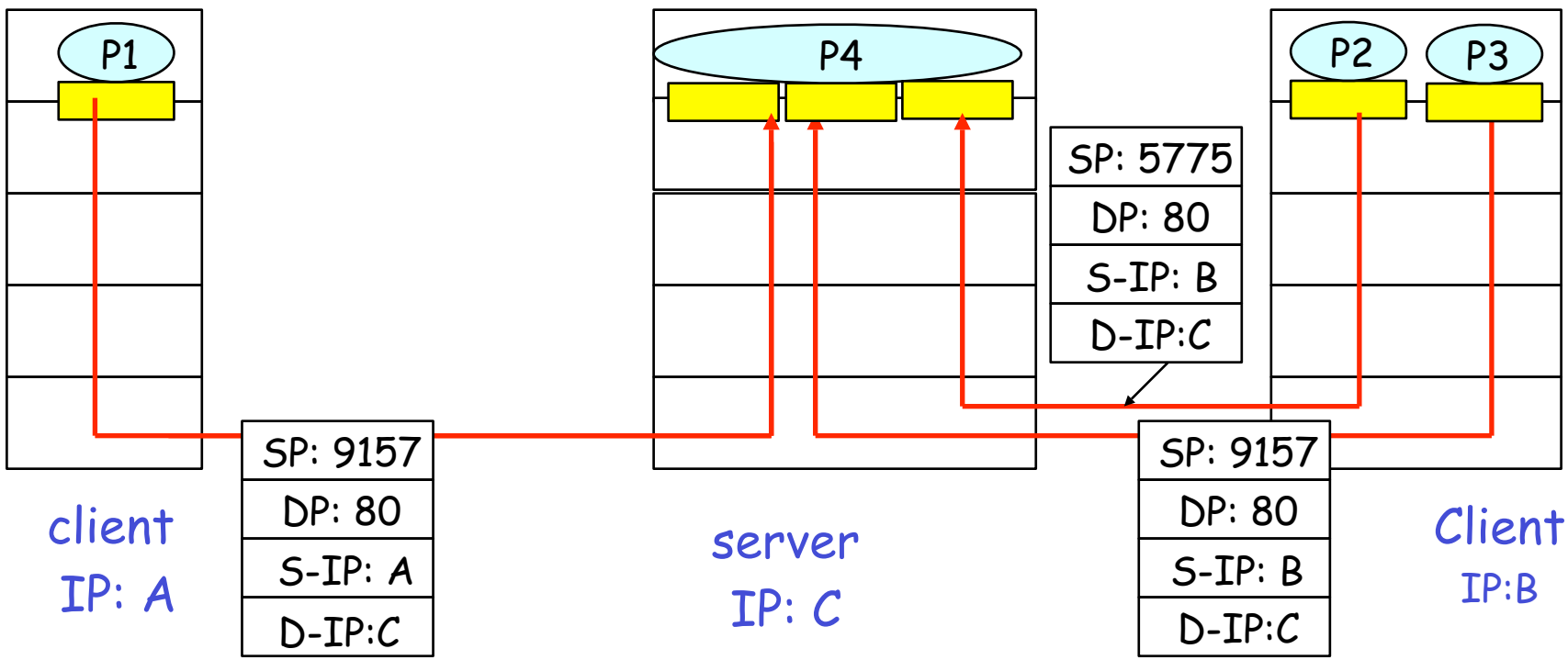
Demultiplexación orientada a conexión

- Ejemplo



Demultiplexación orientada a conexión

- Otro ejemplo



Nivel de transporte en Internet

- **UDP (Hoy)**
- TCP (Próxima clase)

UDP: User Datagram Protocol

- UDP: User Datagram Protocol (RFC-768)
- Proporciona un servicio de transporte para aplicaciones sobre IP de tipo **Best-Effort**
  - Sin garantizar la entrega
  - Sin garantizar el orden de entrega
  - Mucho menos con tiempo o con ancho de banda garantizado
  - Lo unico que añade a IP es la multiplexación de aplicaciones y deteccion de errores ??
- No orientado a conexión
  - No hay establecimiento
  - Cada datagrama se trata independientemente (protocolo sin estado)

UDP

¿Por qué un protocolo como UDP?

- Es rapido: no hay establecimiento

  aunque no garantize el retardo no añade retardos innecesarios
- Es simple: no hay estado de conexión

  ni en el emisor ni en el receptor
- Poco overhead

  la cabecera UDP ocupa lo minimo posible
- Es eficiente: no hay control de congestión

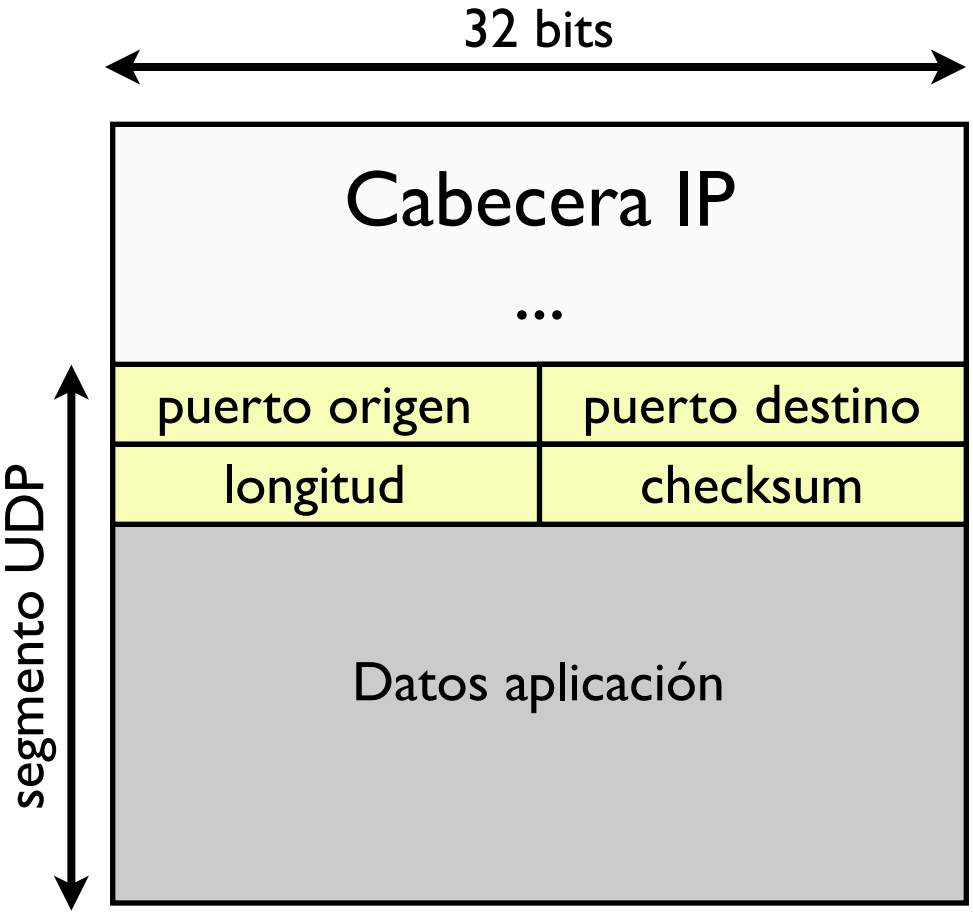
  puede usar todo el ancho de banda que consigas

UDP: detalles

Formato del paquete

- puerto origen
- puerto destino
- longitud del segmento UDP
  - 8 - 65535 bytes
- checksum

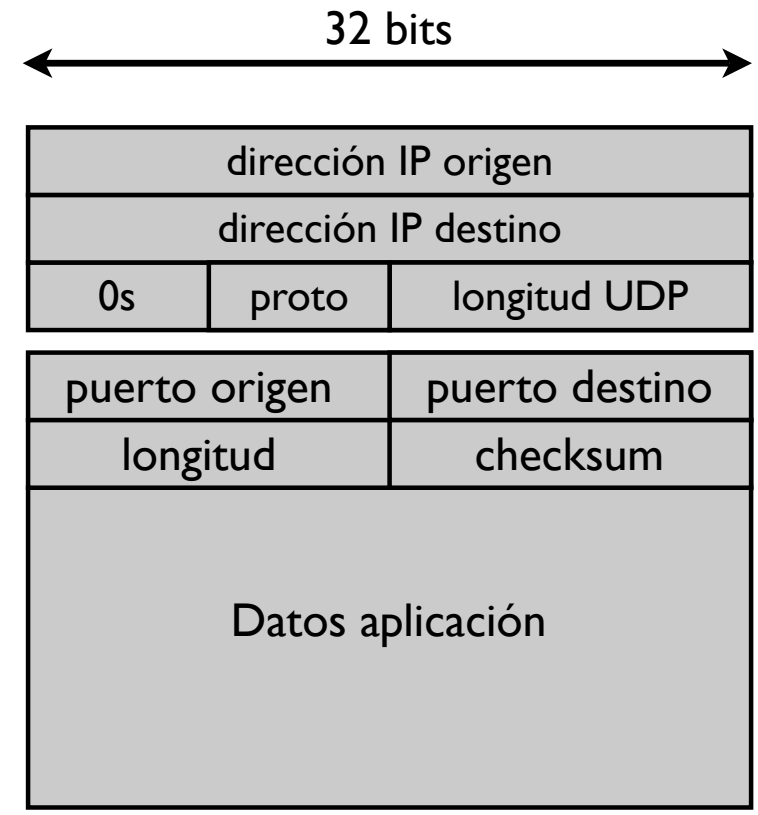
sólo 8 bytes de cabecera !



UDP: checksum

- Cálculo del checksum
  - Se trata el segmento como serie de valores de 16 bits
  - Suma binaria

    en complemento a 1
    - **pseudo-cabecera** con algunos campos de la cabecera IP (para proteger errores en las direcciones y el protocolo)
    - **segmento UDP**



- Es el complemento a uno de la suma binaria en complemento a 1 en 16 bits del bloque de arriba
- De esta forma al hacer la suma binaria en complemento a 1 de un paquete correcto debe salir 0xFFFF
- Detecta errores en todo el segmento UDP (aunque no todos)
- Si UDP detecta errores en un paquete recibido no lo entrega

UDP

En que se usa UDP ?

- Aplicaciones de streaming multimedia (audio/ video en tiempo real o audio/video-conferencia)
  - tolerantes a las pédidas y sensibles al ancho de banda
- Mensajes de DNS
  - bajo retardo y poco overhead
- SNMP (monitorización de red), RIP, DHCP...
  - poco overhead y protocolo sencillo
- Juegos en red
  - mínimo retardo posible

Conclusiones

- El nivel de transporte proporciona comunicación lógica entre aplicaciones mejorando los servicios de IP
- Dos protocolos de transporte
  - TCP orientado a conexión y comunicación fiable con muchas prestaciones
  - UDP orientado a datagramas y con no demasiadas prestaciones pero gracias a ello
    - Simple de implementar, entrega rápida y más eficiente
- Próxima clase:

  TCP