UDP

Tema 3.- Nivel de transporte en Internet

Dr. Daniel Morató
Redes de Computadores
Ingeniero Técnico en Informática de Gestión, 2º curso

Material adaptado del libro *Computer Networking: A Top Down Approach Featuring the Internet*, 3rd edition. Jim Kurose, Keith Ross, Ed. Addison-Wesley, Julio 2004

UDP: User Datagram Protocol

- RFC 768
- Protocolo de transporte simple, sin gran inteligencia
- Servicio "best effort"
- Los datagramas UDP se pueden:
  - Perder
  - Llegar desordenados a la aplicación
- ¿Transferencia fiable sobre UDP?
  - Añadir fiabilidad en el nivel de aplicación
  - ¡Recuperación ante errores específica de cada aplicación!
- Sin conexión:
  - no hay handshaking entre emisor y receptor
  - cada datagrama UDP es procesado de forma independiente a los demás
- Empleado frecuentemente para aplicaciones de streaming multimedia
  - Soportan pérdidas
  - Sensibles a la tasa de envío
- Otros usos de UDP:
  - DNS
  - SNMP

UDP: User Datagram Protocol

- ¿Por qué existe UDP?
  - Es simple: no hay que mantener el estado de la conexión
  - Un establecimiento de conexión puede añadir retardo no deseado
  - Cabecera pequeña
  - No hay control de congestión: puede enviar tan rápido como desee
- Cada `sendto()` se convierte en un datagrama IP

Demultiplexación sin conexión

- Tras crear el socket UDP le asociamos un puerto:

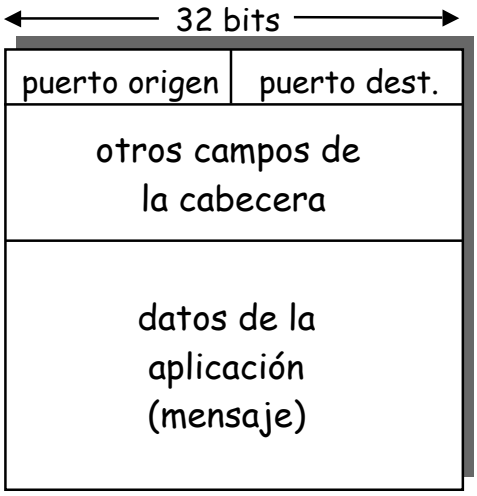
```
sockservidor= socket(PF_INET, SOCK_DGRAM, 0);
bind(sockservidor, (struct sockaddr*)&dirsock, sizeof(dirsock));
```

- Socket identificado por la pareja:

  (direc. IP destino, puerto IP destino)
- Cuando un host recibe un datagrama UDP :
  - Comprueba el puerto destino en el mismo
  - Dirige el segmento al socket UDP con ese puerto
- Datagramas IP con diferentes direcciones IP origen o diferentes puertos origen pueden ser dirigidos al mismo socket

Puerto origen

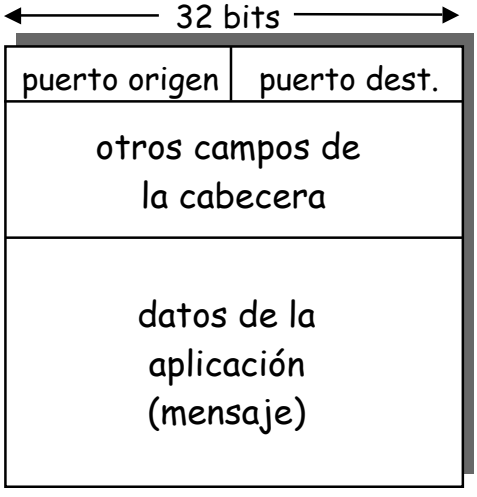
- Sirve para identificar a la aplicación que envía el segmento
- ¿Para qué?
  - ¡Para poder contestar!
  - recvfrom() rellena una estructura donde indica la IP y puerto origen
  - Ahora podemos emplear esa información en una llamada sendto()



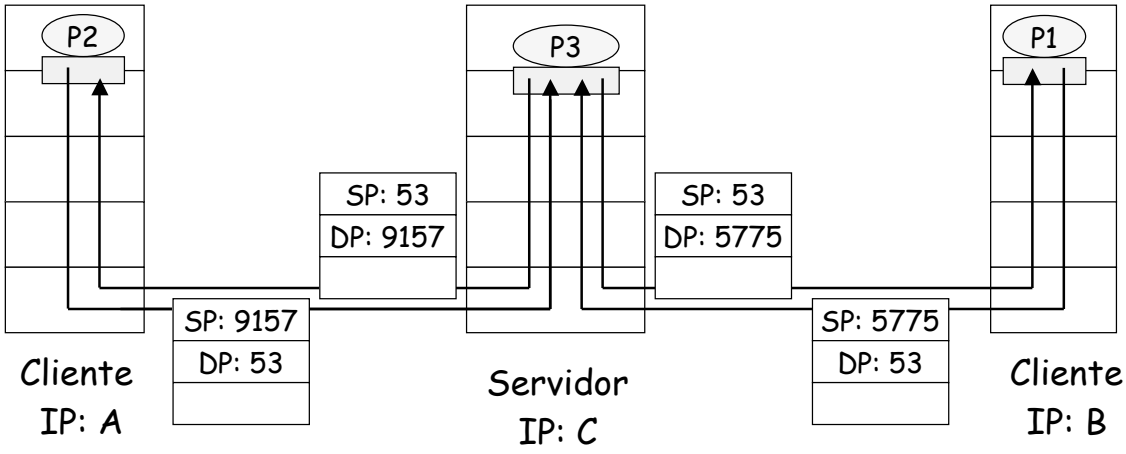

```
int recvfrom(int s, void* buf, int len, int flags, struct
sockaddr *from, int *fromlen)
```


Puerto origen

- Hasta ahora no hemos visto forma de especificarlo
- El sistema operativo escoge un valor la primera vez que se envía un datagrama empleando el socket UDP
- Valor mayor que 1024 (fuera del rango de *well known port numbers)*



Demultiplexación

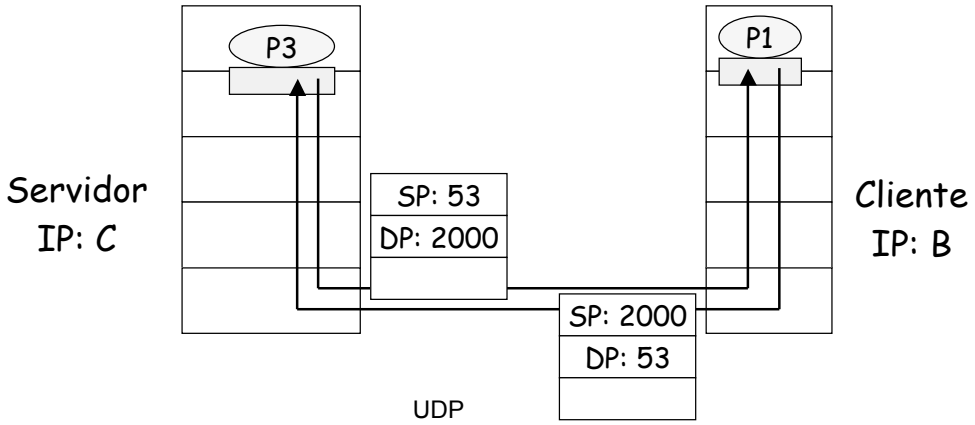


Puerto origen ofrece la "dirección de retorno"

Puerto origen

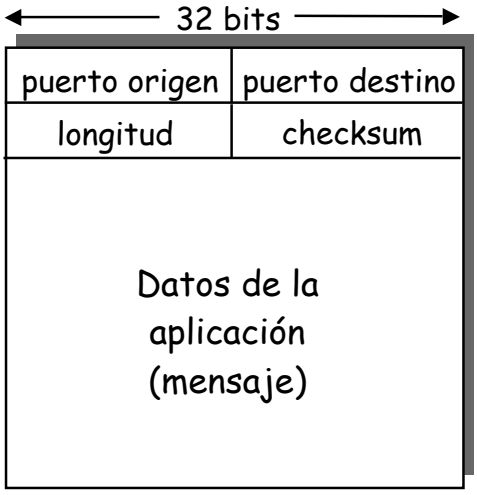
- ¿Se puede escoger?
- Claro.... ¡ bind() !

```
dirsock.sin_port=htons(2000);
bind(sockservidor, (struct sockaddr*)&dirsock, sizeof(dirsock));
```



Cabecera UDP

- Puertos
  - 16 bits
  - Origen y destino, identifican a las aplicaciones
- Longitud
  - En bytes del datagrama UDP (incluida la cabecera)
- Checksum...



formato del datagrama UDP

Checksum UDP

Objetivo: detectar "errores" (ej., bits cambiados) en un datagrama
Cubre a la cabecera y los datos (y parte de la cabecera IP)

Emisor:

- Trata el datagrama como una secuencia de enteros de 16 bits
- checksum: complemento a 1 de la suma (en complemento a 1) del datagrama
- Emisor coloca el checksum en el campo

Receptor:

- Hace la suma en complemento a 1 de todo el datagrama
- ¿Da 0?
  - NO - error detectado
  - Sí - no hay errores detectados. *Pero aún así puede haberlos...*

Temario

0.- Presentación de la asignatura
1.- Introducción
2.- Nivel de aplicación en Internet
3.- Nivel de transporte en Internet
- Principios
- Protocolo UDP
- Protocolo TCP

4.- Nivel de red en Internet
5.- Nivel de enlace

Próxima clase

TCP

- Lecturas recomendadas:
  - [1] 3.4-3.5