Sockets (UDP)

Tema 2.- Nivel de aplicación en Internet

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

Sockets y *UDP*

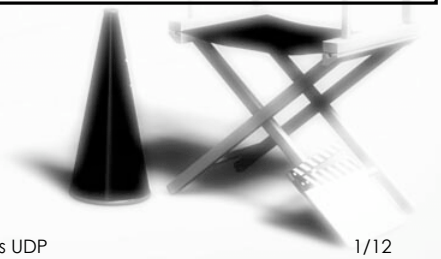
UDP: no hay "conexión" entre cliente y servidor

- » no hay *handshaking*
- » El emisor debe indicar explícitamente la dirección IP y el puerto del destino para cada paquete
- » El servidor debe extraer la dirección IP y el puerto del emisor del paquete

UDP: los datos pueden recibirse desordenados o incluso perderse

Para la aplicacación

UDP ofrece transferencia no fiable de grupos de bytes ("datagramas") entre el cliente y el servidor



Creación de un Socket

`int socket(int domain, int type, int protocol)`

» `int domain`
- Hay diferentes tipos de sockets para diferentes familias de protocolos

» `int type`
- `SOCK_STREAM`, **`SOCK_DGRAM`**, (otros)

» `int protocol`
- En caso de que haya varios protocolos en la misma categoría

Enviar datagrama

```
int sendto(int s, void* msg, int len, int flags,
  struct sockaddr *to, int tolen)
```

» `int s`
- Socket

» `void* msg`
- Puntero a la zona de memoria con los bytes a enviar

» `int len`
- Número de bytes de esa zona de memoria a enviar

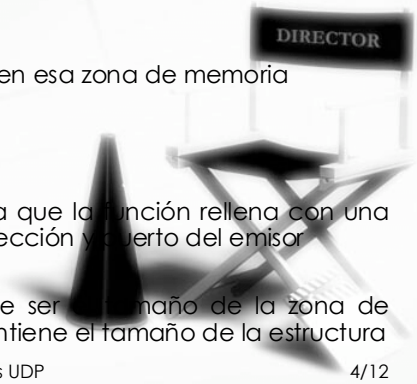
» `int flags`
- Opciones

» `struct sockaddr *to`
- Puntero a estructura con dirección IP y puerto del receptor

» `int tolen`
- Tamaño de la estructura anterior

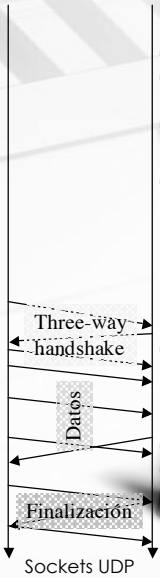
Recibir datagrama

- `int recvfrom(int s, void* buf, int len, int flags, struct sockaddr *from, int *fromlen)`
- int s
  - Socket
- void *buf
  - Zona de memoria donde guardar lo recibido (debe estar reservada!)
- int len
  - Espacio máximo a emplear en esa zona de memoria
- int flags
  - Opciones
- struct sockaddr *from
  - Puntero a zona de memoria que la función rellena con una estructura que incluye la dirección y puerto del emisor
- int *fromlen
  - Al llamar a la función debe ser el tamaño de la zona de memoria anterior. Al salir contiene el tamaño de la estructura

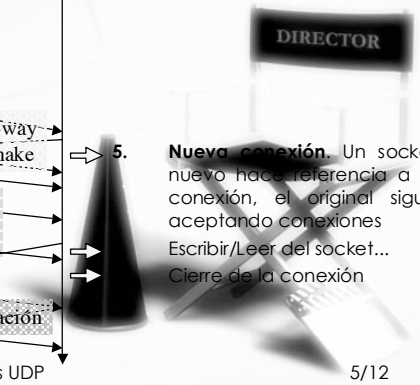


Ejemplo en pseudo-código

- Cliente
  1. Crear el socket TCP (Stream)
  2. Solicitar al S.O. que lo conecte con un destino (IP+puerto) concreto
  3. **Conexión establecida**
  4. Escribir/Leer del socket...
  5. Cerrar el socket/conexión



- Servidor
  1. Crear el socket TCP (Stream)
  2. Asignarle el puerto en el que esperar
  3. Solicitar al S.O. que escuche y acepte esas conexiones
  4. Esperar una conexión...
  5. **Nueva conexión.** Un socket nuevo hace referencia a la conexión, el original sigue aceptando conexiones
  6. Escribir/Leer del socket...
  7. Cierre de la conexión



Ejemplo en pseudo-código

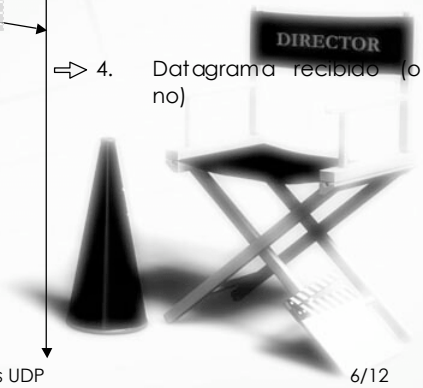
» Cliente

1. Crear el socket UDP (Dgram)
2. Solicitar al S.O. que se envíen ciertos datos a un destino (IP+puerto) concreto

Datos

» Servidor

1. Crear el socket UDP (Dgram)
2. Asignarle el puerto en el que esperar
3. Esperar un datagrama...
4. Datagrama recibido (o no)



Ejemplo en C (1)

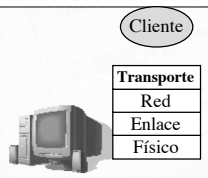
» Cliente

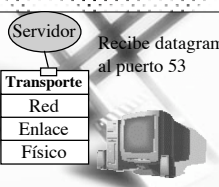
» Servidor

```
struct sockaddr_in dirsock, emisor;
int sockservidor, ret, frlen=sizeof(emisor);
char      *buf[2000];

sockservidor=socket(PF_INET,SOCK_DGRAM,0);
if (sockservidor==-1) ERROR();
dirsock.sin_family=AF_INET;


dirsock.sin_addr.s_addr=INADDR_ANY;
dirsock.sin_port=htons(53);
ret= bind(sockservidor, (struct
    sockaddr*)&dirsock, sizeof(dirsock));
if (ret==-1) ERROR();
```

Crear el socket UDP

Asignar puerto...





Recibe datagramas al puerto 53

Ejemplo en C (y 2)

» Cliente

» Servidor

Crear el socket UDP

```
int sockcliente, ret;
struct sockaddr_in dirsock;
struct hostent *resolvhost;

sockcliente=socket(PF_INET,SOCK_DGRAM,0);
if (sockcliente==-1) ERROR();
```

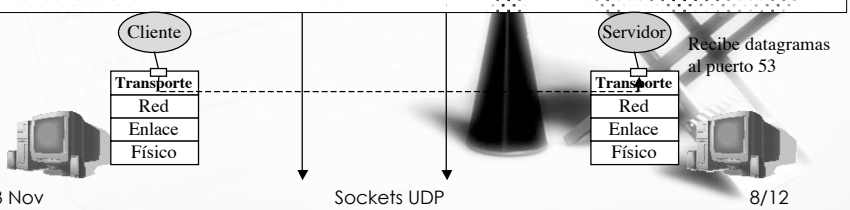
Enviar Datagrama...

```
dirsock.sin_family=AF_INET;
resolvhost=gethostbyname("servidor.tlm.unavarra.es");
if (resolvhost==NULL) ERROR();
dirsock.sin_addr.s_addr=*(u_long*)resolvhost-
    >h_addr_list[0];
dirsock.sin_port=htons(53);

ret=sendto(sockcliente, buf, max, 0, (struct
    sockaddr*)dirsock,  sizeof(dirsock));
```

Esperar a recibir

```
ret=recvfrom(sockservidor, buf,
    2000, 0, (struct
    sockaddr*)&emisor, &frlen);
```

```
if (ret==-1) ERROR();
```



Conectar

¡Completamente prescindible!

```
int connect(int s, struct sockaddr *name, int len)
```

» La misma función que con sockets TCP pero:

- No conlleva una conexión ni el envío de ningún paquete
- Lo único que hace es que el socket memorice un destinatario (IP y puerto)
- De esa forma podemos emplear `write()`

Resumen del tema

- El mundo de las aplicaciones de Internet se divide en tres grupos: clientes y servidores :-)
- El protocolo de nivel de aplicación determina los mensajes que intercambian
- API de sockets para programarlas
- Encarnaciones similares en distintos S.O. y lenguajes
- En UNIX lo sockets son descriptores de fichero

Temario

0.- Presentación de la asignatura

1.- Introducción y revisión de conceptos

2.- Nivel de aplicación en Internet

- Principios
- Funcionamiento de servicios
- Diseño y programación de servicios

3.- Nivel de transporte en Internet

4.- Nivel de red en Internet

5.- Nivel de enlace

Próximo Tema

» Nivel de Transporte en Internet
- Principios
- Protocolos UDP y TCP

Próxima clase

Principios

Protocolo UDP

» Lecturas recomendadas:
- [1] 3.1-3.3