Redes de Computadores

Nivel de Aplicación: Programación con sockets 3

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

En clases anteriores...

- Clientes y servidores TCP

En esta clase...

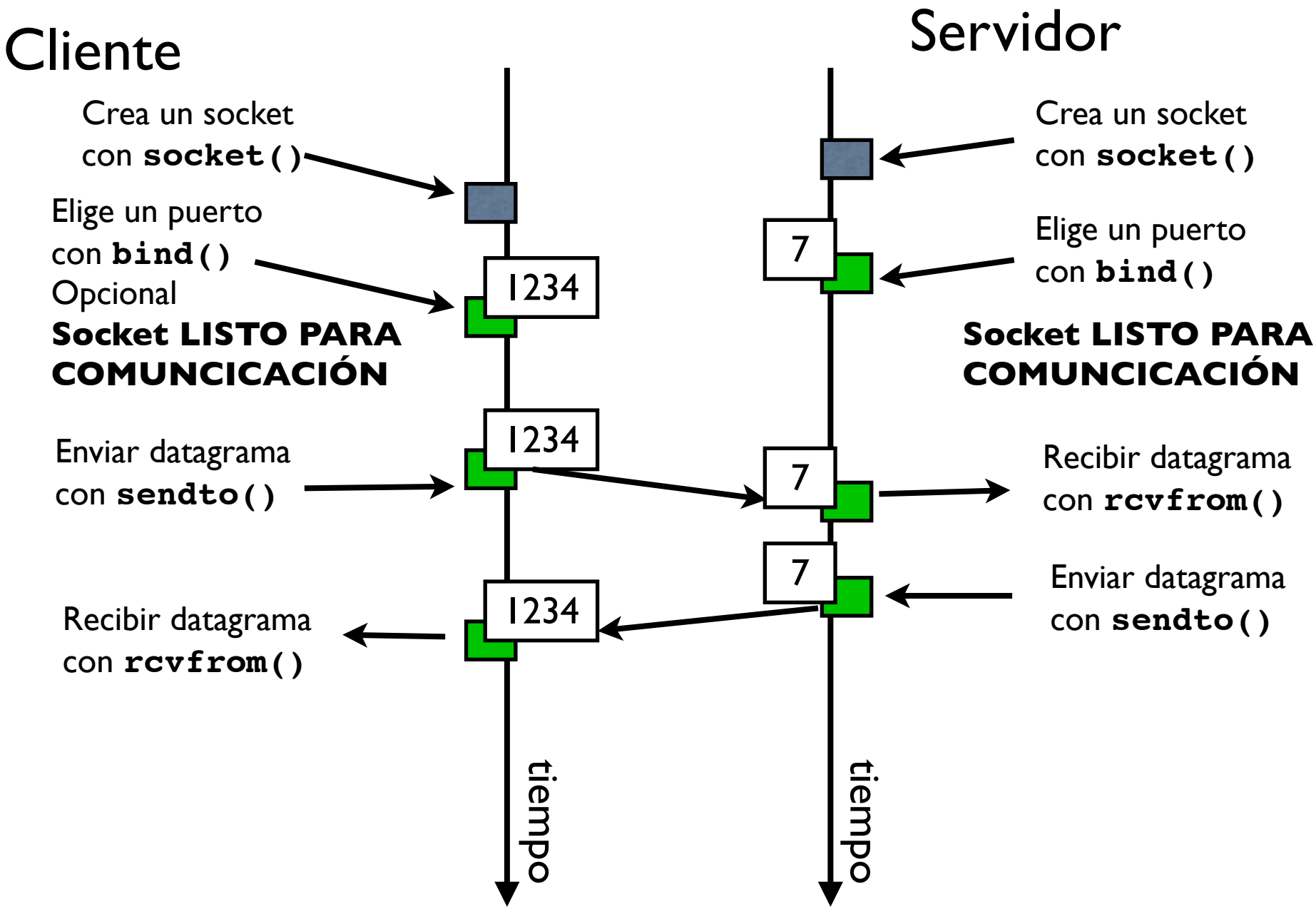
- Clientes UDP
- Servidores UDP iterativos
- Servidores UDP concurrentes
- Opciones avanzadas con sockets

Sockets UDP

- Servicio no orientado a Conexión.

  No hace falta establecimiento
  - Preparación del socket más simple
- El mismo socket se puede usar para recibir y enviar a varios sockets remotos
  - La sincronización de cuándo tengo que enviar o recibir dependerá del protocolo de aplicación que construyamos
- Aparte UDP se puede usar para otros usos eminentemente no conection oriented
  - Uno a varios
  - Multicast y broadcast

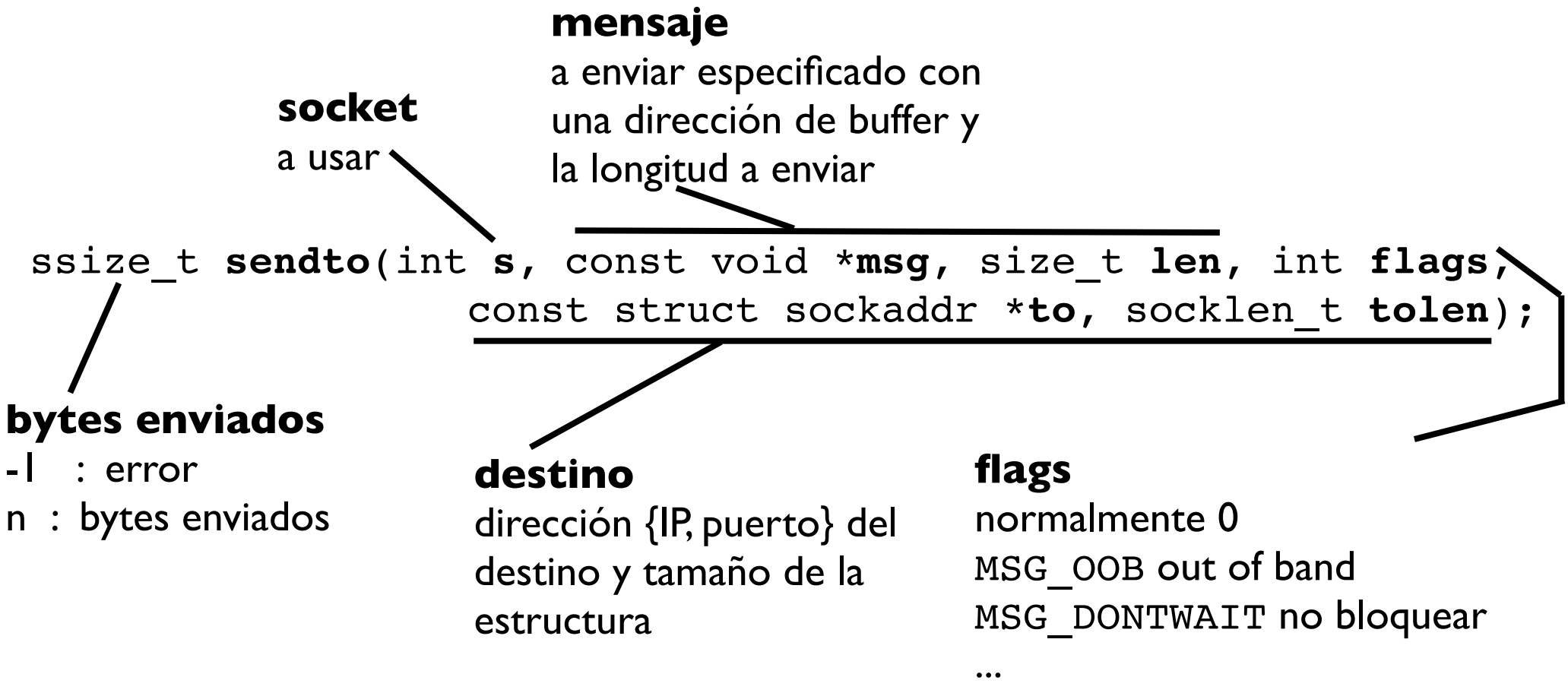
Sockets UDP: operaciones



Sockets UDP: envío

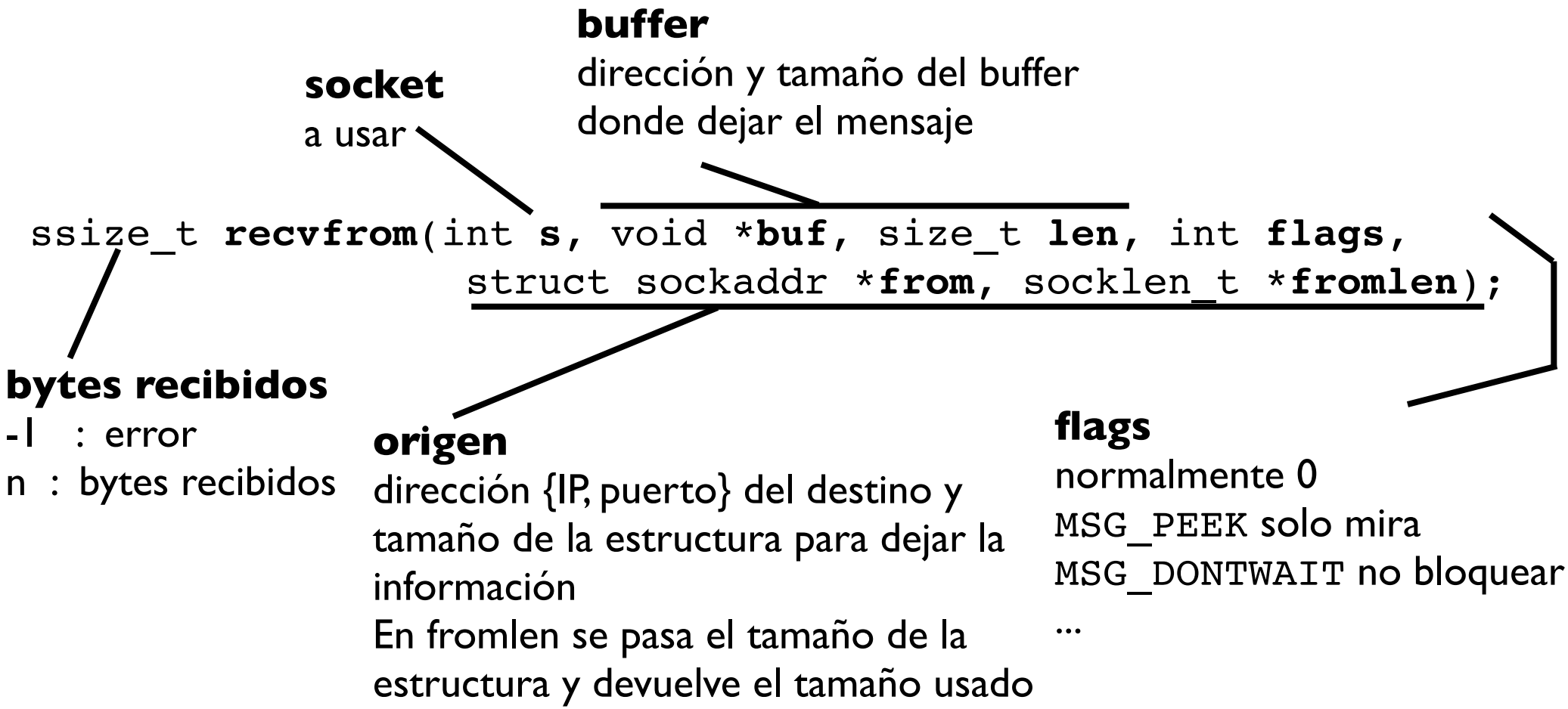
- Función `sendto()`

  Envía este mensaje a este socket {IP, puerto}



Sockets UDP: recepción

- Función `recvfrom()`
  Recibe el siguiente datagrama



Servidores UDP

- Servidores sin estado
  - No hay accept() al llegar una petición de un cliente la atendemos y nos olvidamos
- El mismo socket para atender a varios clientes

Servidor UDP

- Recibimos datagramas y enviamos respuestas

```
int socket;
struct sockaddr_in direccion;
void *buffer;
int direccionlen, buflen, recibidos;
while (1) {
    direccionlen = sizeof(direccion);
    recibidos = recvfrom(socket,buffer,buflen,0,
                (struct sockaddr *)&direccion, &direccionlen);

    aenviar =
       respuesta_al_cliente( buffer, recibidos, direccion );
    /* la respuesta se construye en el buffer */
    sendto(socket,buffer,aenviar,0,
                (struct sockaddr *)&direccion, &direccionlen);
}
```

- Y esto es iterativo o concurrente ???

Ejemplo: servidor de ECHO UDP

- Socket de tipo SOCK_DGRAM
- bind en el puerto indicado

```
int main (int argc, char * argv[]) {
    int sock;
    int puerto;
    struct sockaddr_in servidor,cliente;
    int dirlen;
    char buf[2000];
    int leidos;


    if (argc<=1) puerto=1234;
    else sscanf(argv[1],"%d",&puerto);
    printf("puerto %d\n",puerto);

    servidor.sin_family=AF_INET;
    servidor.sin_port=htons(puerto);
    servidor.sin_addr.s_addr=INADDR_ANY;

    sock=socket(PF_INET,SOCK_DGRAM,0);
    if (bind(sock,(struct sockaddr *)&servidor,sizeof(servidor))==-1) {
        printf("Error: no puedo coger el puerto\n");
        exit(-1);
```

el buffer para recibir y construir respuestas

Socket + bind

Ejemplo: servidor de ECHO UDP

- esperamos recibir y enviamos respuesta
- para hacer ECHO la respuesta es lo recibido

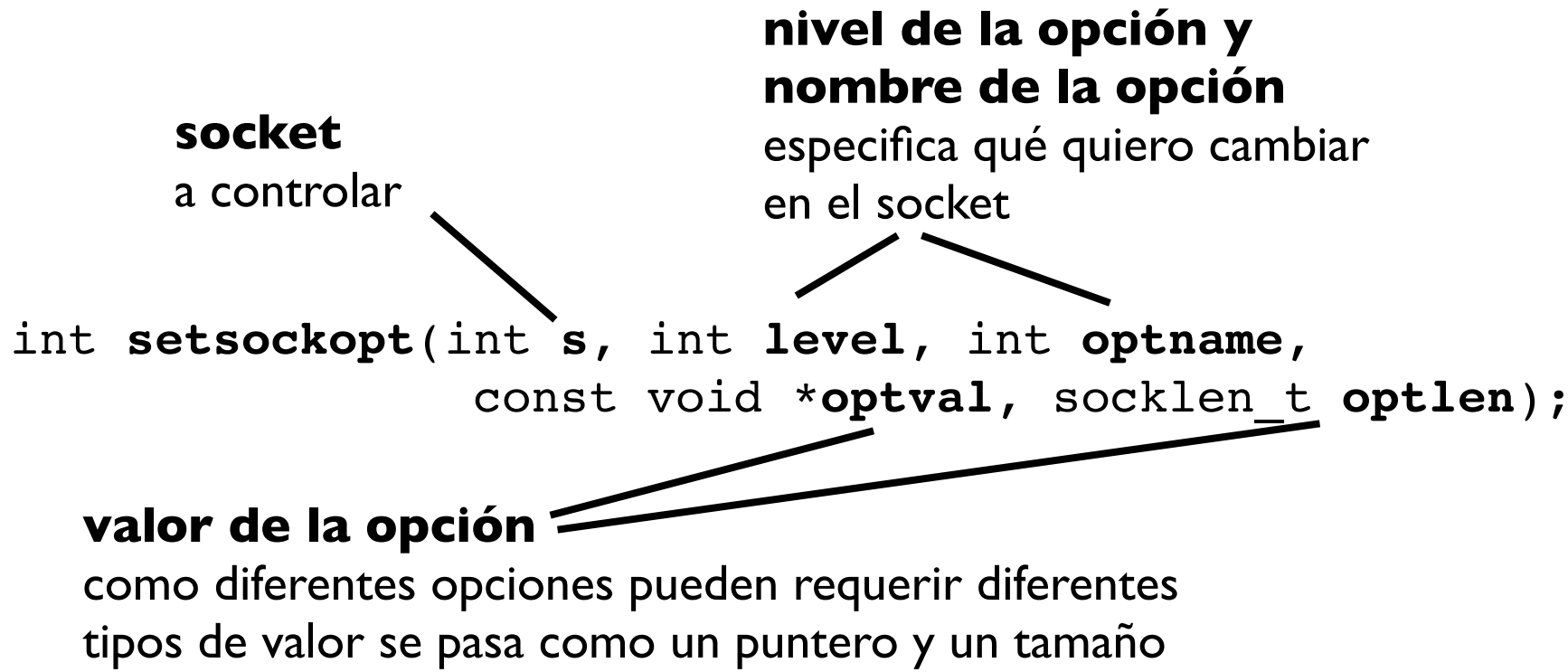
```
int puerto;

sock=socket(PF_INET,SOCK_DGRAM,0);
if (bind(sock,(struct sockaddr *)&servidor,sizeof(servidor))==-1) {
    printf("Error: no puedo coger el puerto\n");
    exit(-1);
}

while(1) {
    dirlen=sizeof(cliente);
    /* Recibir 1 paquete */
    leidos=recvfrom(sock,buf,2000,0,(struct sockaddr *)&cliente,&dirlen);
    printf("Contestado 1 paquete de %x\n",ntohl(cliente.sin_addr.s_addr));
    /* Devolver el paquete */
    if (leidos>0)
        sendto(sock,buf,leidos,0,(struct sockaddr *)&cliente,dirlen);
}
}
```

recibir

enviar

Controlar opciones de los sockets

- Funcion `setsockopt()`
  Especifica valores de opciones en un socket



- Hay `getsockopt()` para leer los valores

Opciones

- Permiten cambiar el comportamiento de los niveles de protocolos inferiores usados por el socket
- Algunos ejemplos:

Nivel	Opción	Efecto
SOL_SOCKETS	**SO_SNDBUF**	Elige el tamaño del buffer de envío del nivel de transporte del socket
	SO_RCVBUF	Elige el tamaño del buffer de recepción del nivel de transporte del socket
	SO_BROADCAST	Permite el uso de envio Broadcast en un socket. Solo se puede usar con sockets UDP

Usando el DNS

- Función gethostbyname()
  - Pregunta al resolver cual es la dirección IP asociada al nombre **name**

    `struct hostent *`**`gethostbyname`**`(const char *`**`name`**`);`
  - El resultado es una estrcutura `struct hostent`

    Contiene todos las IPs asociadas a ese nombre

```
struct  hostent {
        char    *h_name;        /* official name of host */
        char    **h_aliases;    /* alias list */
        int     h_addrtype;     /* host address type */
        int     h_length;       /* length of address */
        char    **h_addr_list;  /* list of addresses from name server */
};
#define h_addr  h_addr_list[0]  /* address, for backward compatibility */
```

Ejemplo: gethostbyname

- Programa para buscar un nombre

  `$ busca www.tlm.unavarra.es`

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include <netdb.h>


int main(int argc,char *argv[]) {
    char *nombreabuscar;
    struct hostent *resultado;

   if (argc<=1) {
      nombreabuscar="www.tlm.unavarra.es";
   } else {
      nombreabuscar=argv[1];
    }
```

include necesario para gethostbyname()

Leemos el nombre de la linea de comandos

Ejemplo: gethostbyname

- Llamada a gethostbyname()
  + interpretar el resultado

```
    if (argc<=1) {
        nombreabuscar="www.tlm.unavarra.es";
    } else {
        nombreabuscar=argv[1];
      }

    /* Pedimos la resolucion del nombre */
     resultado = gethostbyname( nombreabuscar );
     if (resultado == NULL) {
         printf("No he podido resolver el nombre\n");
         exit(-1);
     }

    /* Imprimimos el resultado */
     printf(" Nombre: %s \n Dir IP: 0x%X \n -------------- \n",
                        nombreabuscar,  *(int*) (resultado->h_addr_list[0])   );

     printf(" Nombre: %s \n Dir IP: %s \n",
                        nombreabuscar,  inet_ntoa( *(struct in_addr *)(resultado->h_addr) )   );

}
```

gethostbyname()
NULL = no encontrado

convirtiendo el resultado a entero

inet_ntoa() para convertir el resultado a cadena

Conclusiones

- Sockets UDP controlados
- Cambiar opciones de sockets
- Usar el DNS para obtener una IP

- +*Sockets TCP en días anteriores*
- =*Ya sabemos como usar los servicios de la red*
- ¿Como proporcionan los servicios TCP y UDP?

- Próxima clase: **Nivel de Transporte**