Tema 6. Sistemas C/S Básicos con Sockets

Índice

Programación C/S Básica

Enrique Alba Torres

Lenguajes y Ciencias de la Computación

Universidad de Málaga (UMA)

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Introducción

- Los servicios Internet se ofertan como sistemas Cliente/Servidor
- Los protocolos de base suelen ser peer-to-peer (igual-a-igual)
- En cualquier caso se necesita una interfaz para programar
- BSD socket es una interfaz estándar orientada al paso de mensajes entre procesos para programar en red sobre TCP/IP
- Las primitivas (funciones en C) y tipos de datos (estructuras, ...) de los sockets conforman un API para programar servicios en red
- Existen sockets para UNIX, Linux, Windows y otros SS.OO.
- Existen API's para sockets en C, C++, Java y otros lenguajes

EJEMPLO

```
#include <sys/socket.h> ...
main()
{  int sservice; ...
   struct sockaddr_in sout; ...
   sservice = socket(…); ...
   sout.sin_family = AF_INET; ...
   sout.sin_port   = 4000; ...
   connect(sservice,sout,…); ...
}
```

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Sistemas C/S (I)

- Un servidor realiza un proceso repetitivo, normalmente sin final, que consiste en un bucle básico de atención a los clientes:

```
...
while(true)
 {
   esperar petición de un cliente;
   atender al cliente;
 }
...
```

- La espera de peticiones puede hacerse:
  - A. Sobre un protocolo o varios (servicio multiprotocolo)
  - B. Para proporcionar un servicio o varios (multiservicio)
- La atención al cliente puede hacerla:
  - A. Directamente el servidor (iterativo)
  - B. Un servidor esclavo generado (concurrente)
- Según el nivel de transporte usado el servicio puede ser:
  - A. Orientado a la conexión (si el servidor utiliza TCP)
  - B. Sin conexión (si el servidor utiliza UDP)

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Sistemas C/S (II)

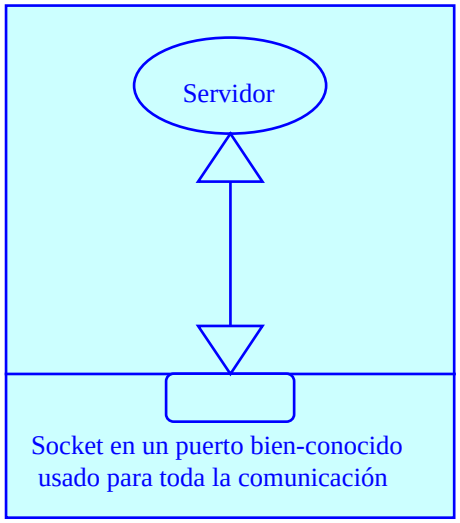
	Ventajas	Inconvenientes
Iterativo	simple de programar	la cola de clientes pendientes puede crecer
Concurrente	atención al cliente indep. del servicio	gestión de procesos hijos esclavos (fork+signal)
Or. Conexión	información fluye correcta y ordenada	puede ser costoso para algunos servicios
Sin Conexión	comunicación eficiente y “ligera”	posible pérdidas, desorden y + complejo
Multiprotocolo	servicio independ. de implementación	difícil de lograr para algunos servicios
Multiservicio	versatilidad y potencia	difícil de programar y mantener

Tema 6. Sistemas C/S Básicos con Sockets

Índice

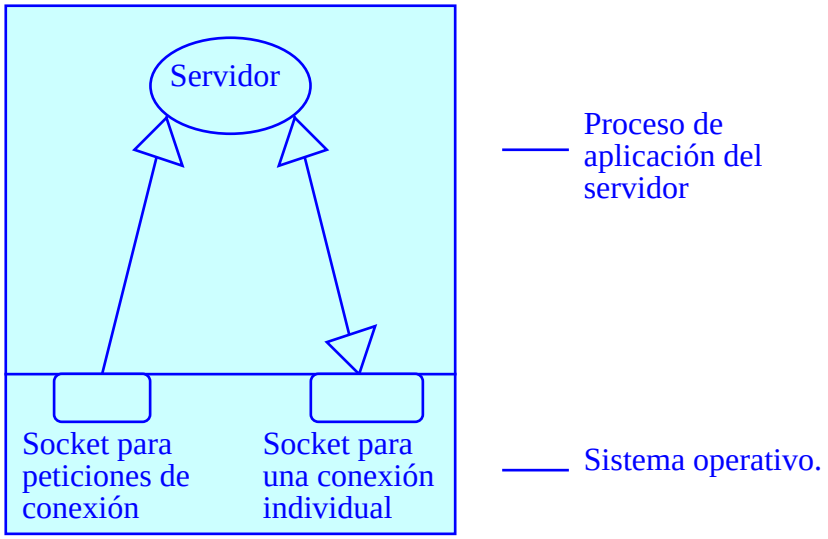
Esquemas de Implementación C/S usando Sockets

SERVIDORES ITERATIVOS



SIN CONEXIÓN

ORIENTADO A LA CONEXIÓN

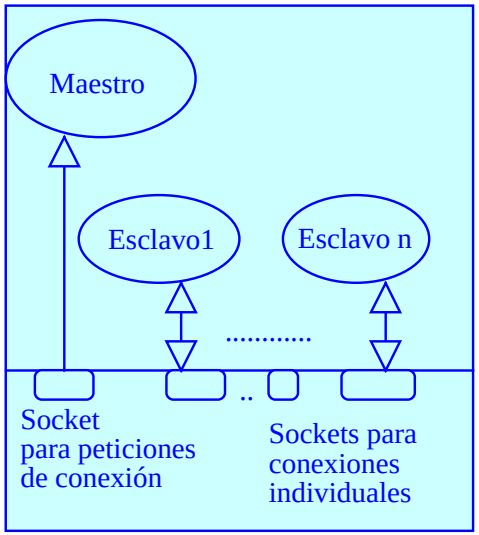


Tema 6. Sistemas C/S Básicos con Sockets

Índice

Esquemas de Implementación C/S usando Sockets

SERVIDORES CONCURRENTE (CON/SIN CONEXIÓN)

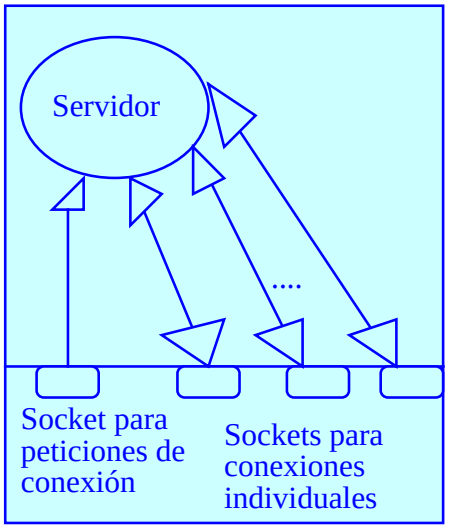


Procesos de aplicación del servidor

VARIOS PROCESOS

Sistema Operativo

MONOPROCESO



Proceso de aplicación de un servidor

Sistema Op.

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Primitivas

Primitiva	Función
socket	*Crea un TSAP del servicio especificado (TCP o UDP, p.e.)*
bind	*Registra en el sistema el servicio asociado a un socket*
listen	*Crea una cola para almacenar CR que lleguen (pasivo)*
accept	*Saca una CR de la cola del socket o espera a que llegue una*
connect	*Inicia una conexión con un socket remoto (local→remoto)*
Shutdown **close**	*Cierran la conexión de un socket (ambos extremos pueden usarlo independientemente)*
sendto **write**	*Envían un mensaje por un descriptor de socket dado normalmente de forma no bloqueante*
recvfrom **read**	*Reciben un mensaje por un descriptor de socket dado normalmente de forma bloqueante*
select	*Informa sobre los sockets listos para escribir o leer de ellos*

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Familias de Direcciones

- Cada familia representa un dominio con sockets de similares características
- La familia determina el formato y los protocolos disponibles
- Las direcciones determinan el host y la aplicación de manera inequívoca
- La estructura genérica de una dirección es:

```
struct sockaddr
{  u_short  sa_family;     /* Familia: ctes. de la forma AF_xxx  */
   char     sa_data[14];   /* 14 bytes dependientes de la familia */
};
```

- Algunos de los donimios disponibles son:
  - **AF_INET**: protocolos de Internet
  - **AF_UNIX**: sockets locales que corresponden a ficheros
  - **AF_CCITT**: protocolos CCITT como X.25
  - **AF_SNA**: protocolos de redes IBM SNA
  - **AF_DECnet**: protocolos de redes DEC
- La estructura para Internet-TCP/IP (AF_INET) es como sigue:

```
struct sockaddr_in
{   short     sin_family;     /* Valor AF_INET*/
    u_short   sin_port;       /* 16 bits con el número del puerto */
    u_long    sin_addr;       /* 32 bits (red+ host)*/
    char      sin _zero[8];   /* 8 bytes no usados */
};
```

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Tipos de Sockets

- Cada tipo de socket tiene ciertas propiedades para realizar las transmisiones
- Los tipos más utilizados son:

SOCK_STREAM

Orientado a la conexión (TCP si AF_INET)

SOCK_DGRAM

Servicio sin conexión (UDP si AF_INET)

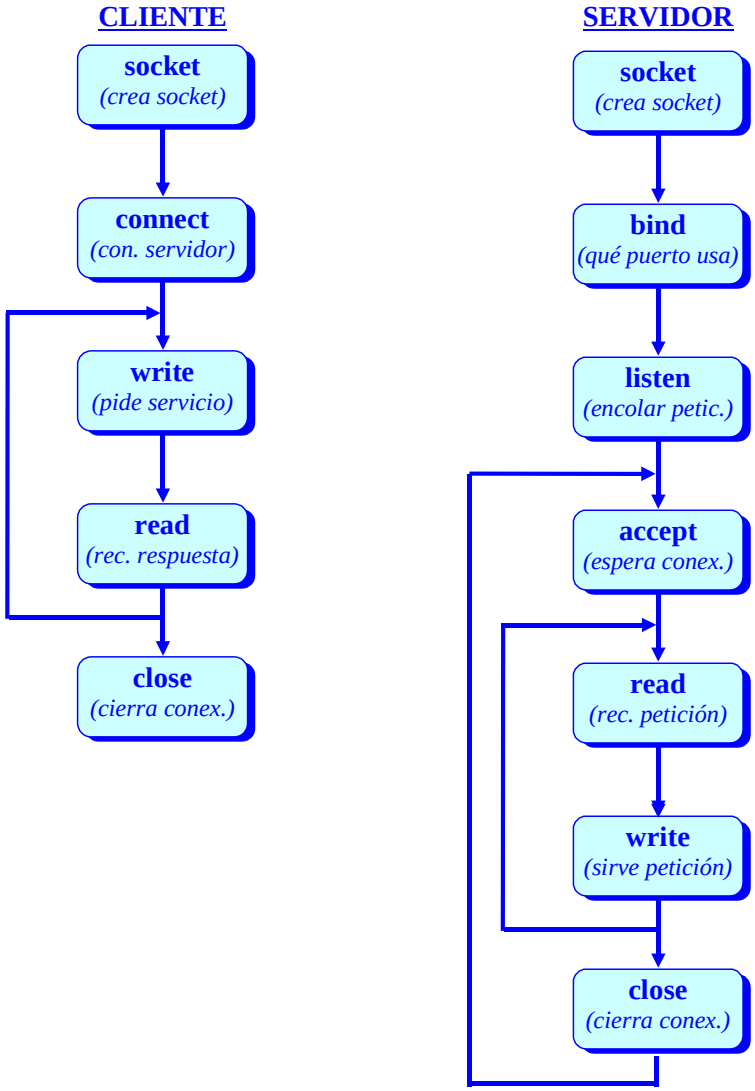
SOCK_RAW

Acceso a bajo nivel (IP si AF_INET) –¡sólo puede usarlos el root! –

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Implementación de un C/S sobre TCP



Tema 6. Sistemas C/S Básicos con Sockets

Índice

Implementación del Cliente sobre TCP (I)

```
/*************************************************************\
 * (client.c)  Client for reading input messages from keybd. <CdD>*
\*************************************************************/

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

#define TRUE  1
#define FALSE 0

main()
{
        int     sservice, end;
        char    buf[128];
        struct protoent *ppe;
        struct sockaddr_in sout;
        ppe = getprotobyname("tcp");      /*Get the tcp-entity data*/
        /*Open the socket*/
        sservice=socket(PF_INET,SOCK_STREAM,ppe->p_proto);
```

❶ Incluimos el API

❷ Pedimos información sobre TCP

❸ Abrimos un TSAP

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Implementación del Cliente sobre TCP

```
sout.sin_family      = AF_INET;          /*ARPANET */
sout.sin_addr.s_addr = getservbyport(4000,"tcp"); /*Which server?*/
sout.sin_port        = 4000;             /*Output port*/
```

❹Dirección del Servidor

```
/*Connect to the server*/
connect(sservice,(struct sockaddr *)&sout, sizeof(sout));
```

❺ Conectar con servidor

```
end = FALSE;
while(!end)  /*Client Service Loop*/
{           /*Ask for the service*/
            printf("\nCLIENT> Send a message...: ");     fflush(stdout);
            scanf("%s",buf);
            write(sservice,buf,strlen(buf));
            printf("\nCLIENT>Client sent: %s", buf);     fflush(stdout);
            if (!strcmp(buf,"end"))
            {
             printf("\nCLIENT>Bye...");        fflush(stdout);
             end=TRUE;
            }
 }
            close(sservice);
} /*main()*/
```

❻Msj. para servidor

❼Cerrar TSAP

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Implementación del Servidor sobre TCP

```
/*****************************************************************\
 * (server.c) The server shows the received messages in the screen. <CdD>    *
\*****************************************************************/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#define TRUE  1
#define FALSE 0

main()
{       int    sservice, sclient,l,nbytes_read,end;
        char   buf[128];
        struct protoent *ppe;
        struct sockaddr_in sin,clientfsin;
        ppe = getprotobyname("tcp");
         /*Open the socket*/
        sservice=socket(PF_INET,SOCK_STREAM,ppe->p_proto);
        /*pse = getservbyname("messages", "tcp");  <---If service is reg.*/
```

❶ Incluimos el API

❷ Pedimos información sobre TCP

❸ Abrimos un TSAP

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Implementación del Servidor sobre TCP

```
sin.sin_family      = AF_INET;
sin.sin_addr.s_addr = INADDR_ANY;
sin.sin_port        = 4000;
```

❹Dirección IP+TCP

❺ Registrar, conf., aceptar

```
sclient=bind(sservice, (struct sockaddr*)&sin, sizeof(sin)); /*Register serv*/
listen(sservice,5);          /*Up to 5 pending connections*/
sclient=accept(sservice,(struct sockaddr*)&clientfsin,&l); /*Accept a client*/
 /*If concurrent, a children server should be spawned here by using fork()*/
end = FALSE;
while(!end)          /*Give the service*/
{       nbytes_read=read(sclient,buf,sizeof(buf));
        if (nbytes_read>=0)
                buf[nbytes_read]='\0';
        else    {strcpy(buf,"\nSERVER>ERROR IN SERVER!");
                 end=TRUE; }
        printf("\nSERVER>Server received: %s", buf);    fflush(stdout);
        if(!strcmp(buf,"end"))    end=TRUE;
}
printf("\nSERVER>Bye...");    fflush(stdout);
close(sclient);      /*Never forget to close a socket!*/
close(sservice);
} /*main()*/
```

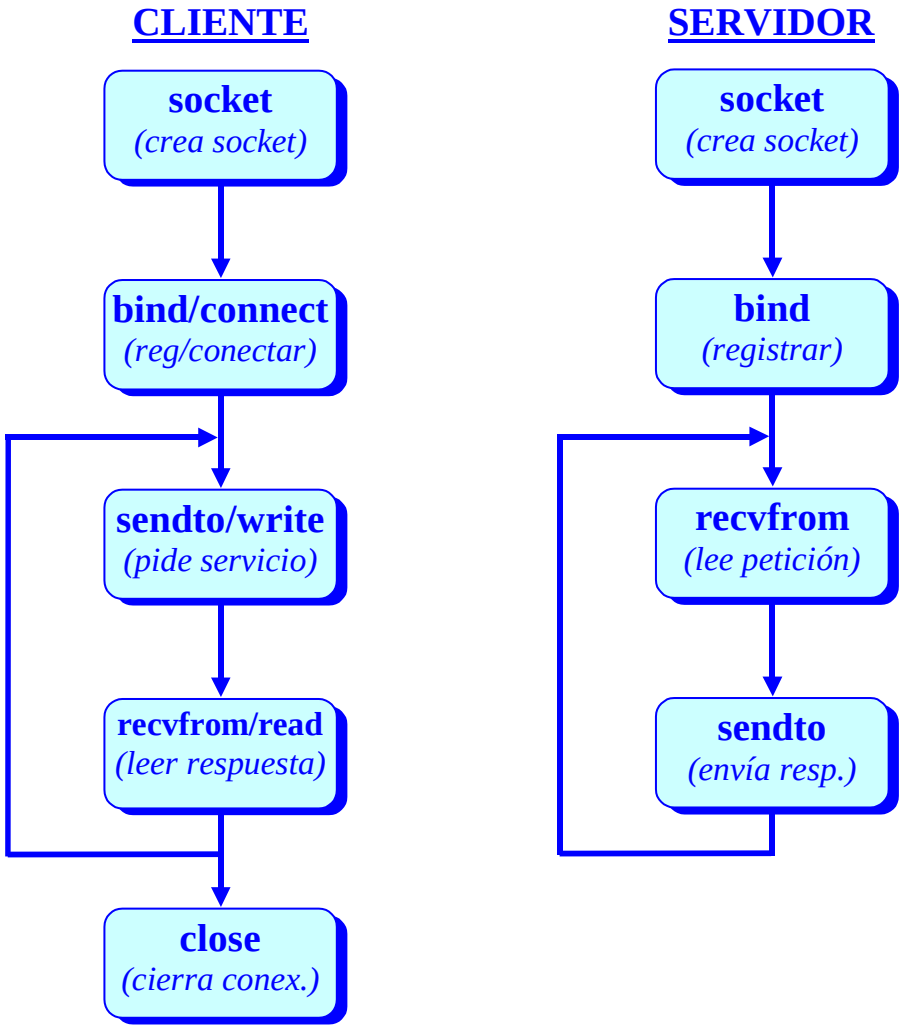
❻Leer “petición”

❼ ¡Normalmente nunca cerrar `sservice`!

Tema 6. Sistemas C/S Básicos con Sockets

Índice

Implementación de un C/S sobre UDP



Tema 6. Sistemas C/S Básicos con Sockets

Índice

Bibliografía

- **Internetworking with TCP/IP (Vol. 1)**
  Comer, D.E. Prentice-Hall. 1991.
- **Internetworking with TCP/IP (Vol 3)**
  Comer D.E., Stevens, D.L. Prentice-Hall. 1993.
- **Unix Network Programming**
  Stevens, W.R. Prentice-Hall. 1990.
- **Computer Networks and Internets (2nd edition)**
  Comer, D.G. Prentice-Hall. 1999
- **A Guide to the TCP/IP Protocol Suite**
  Wilder, F. Artech House. 1993.