



DOCUMENTACIÓN DEL CÓDIGO.- SIMULACIÓN DE UNA RED NAT

UNIVERSIDAD VERACRUZANA



18 DE DICIEMBRE DE 2023

SISTEMAS OPERATIVOS

Facultad de Estadística e Informática

Mtro. Juan Luis López Herrera

Alumnos: Jesus Lorenzo Tlapa Hernández

Luis Angel López García

Juan Pablo Torres Ortiz

INTRODUCCIÓN

Este conjunto de códigos en lenguaje C implementa una comunicación entre un cliente y un servidor utilizando sockets TCP. La aplicación se compone de tres programas distintos: el servidor, el router, y el cliente. Estos programas se comunican entre sí para transferir información a través de la red, demostrando los principios fundamentales de la comunicación en arquitecturas cliente-servidor. El objetivo principal de este sistema es proporcionar un marco de trabajo para entender los fundamentos de la comunicación en una red utilizando sockets. Cada componente cumple un papel específico en la transmisión de datos, desde la iniciación de la conexión hasta la gestión de la transferencia bidireccional de información.

ESTRUCTURA DEL SISTEMA

1. Servidor

El servidor representa el núcleo del sistema. Espera activamente conexiones entrantes y responde a las solicitudes de los clientes. Este componente encapsula la lógica de negocio y realiza operaciones específicas en respuesta a las peticiones recibidas.

2. Router

El router actúa como un facilitador de la comunicación entre el cliente y el servidor. Gestiona la infraestructura de red, dirigiendo el flujo de datos desde el cliente al servidor y viceversa. Proporciona un punto centralizado para la coordinación de las comunicaciones.

3. Cliente

El cliente inicia la conexión con el router y envía solicitudes al servidor. Representa la interfaz de usuario del sistema y es responsable de la interacción inicial con el usuario final.

REQUISITOS DEL SISTEMA

- Lenguaje C.
- Plataforma de ambiente UNIX/LINUX
- Bibliotecas de Sockets, funciones de red y manejo de archivos en C.

INDICACIONES DE USO

- Compilación:

Utilice un compilador de C compatible para compilar los programas.

- Ejecución:

Inicie el servidor en un terminal.

Inicie el router en otro terminal.

Ejecute el cliente para interactuar con el sistema.

- Configuración de red:

Ajuste las configuraciones de red según sea necesario, como direcciones IP, para adaptarse a su entorno específico.

FUNCIONAMIENTO DETALLADO

FUNCIONAMIENTO

INICIO DE CONEXIÓN

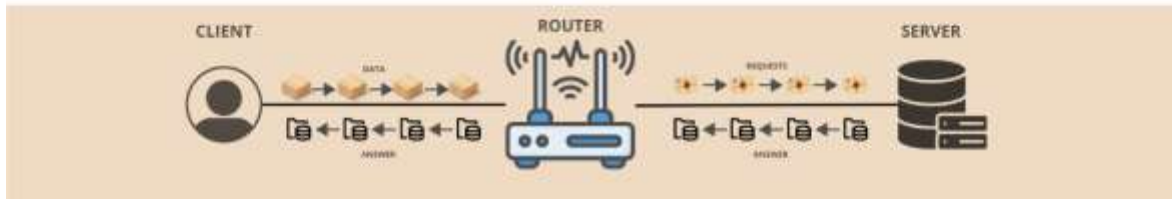
El router establece conexiones simultáneas con el servidor y el cliente. Actúa como un intermediario para facilitar la comunicación entre estos dos componentes.

COMUNICACIÓN CLIENTE-SERVIDOR:

El cliente envía datos al servidor a través del router. El servidor procesa las solicitudes y devuelve las respuestas al cliente a través del mismo router.

INTERMEDIACIÓN DEL ROUTER

El router juega un papel crucial en la comunicación bidireccional. Dirige eficientemente el tráfico de datos, garantizando una conexión estable y confiable entre el cliente y el servidor. El Router también se encarga de la traducción de direcciones (convierte direcciones privadas a públicas).



CODIGO

CLIENTE

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <arpa/inet.h>
8  #include <netdb.h>
9  #include <ctype.h>
10
11
12
13
14  int main(int argc, char *argv[]) {
15      const char protocolo[] = "tcp";
16      int socket_router;
17      struct sockaddr_in dir_router;
18      unsigned short puerto_router = 9090u;
19      char *direccion_router;
20      struct protoent *protoent;
21      struct hostent *hostent;
22      in_addr_t in_addr;
23
24      direccion_router = "127.0.0.1";
25
26
27      //Obtiene la configuración para el socket TCP
28      protoent = getprotobyname(protocolo);
29
30
31
```

```

62 //Limpia el buffer
63 memset(buffer, 0, 200);
64 //Se lee una cadena desde el socket cliente
65 nbytes = read(socket_servidor, buffer, 200);
66 //Se copia el buffer leído hacia la cadena
67 strncat(cadena, buffer, nbytes);
68 //Si el buffer contiene un '\n' se termina la lectura
69 if (strstr(buffer, "\n"))
70     enter_presente = 1;
71 }
72
73 write(socket_cliente, cadena, nbytes);
74
75 //Se cierra el socket cliente
76 close(socket_cliente);
77 }
78
79 int main(int argc, char *argv[]) {
80     const char protocolo[] = "tcp";
81     int socket_router, socket_cliente;
82     struct sockaddr_in dir_router, dir_cliente, dir_servidor;
83     unsigned short puerto_router = 9090u, puerto_servidor = 4500u;
84     struct protoent *protoent;
85     int habilitar_reuso_socket=1;
86
87     //Obtiene la configuración para el socket TCP
88     protoent = getprotobyname(protocolo);
89
90
91
92     //Crea el socket del dominio Internet, para comunicaciones orientadas a conexión y con el protocolo seleccionado

```

```

93 socket_router = socket(AF_INET, SOCK_STREAM, protoent->p_proto);
94 //Se configura el socket para ser reusado cuando se cierre el programa
95 setsockopt(socket_router, SOL_SOCKET, SO_REUSEADDR, &habilitar_reuso_socket, sizeof(int));
96
97 //Se configura la dirección en la que escucha el router
98 dir_router.sin_family = AF_INET;
99 dir_router.sin_addr.s_addr = htonl(INADDR_ANY);
100 dir_router.sin_port = htons(puerto_router);
101
102 //Se asocia la dirección al socket para escuchar conexiones entrantes
103 if (bind(socket_router, (struct sockaddr *)&dir_router, sizeof(struct sockaddr_in)) == -1) {
104     perror("Error en la asignación del puerto al socket, el puerto está en uso");
105     exit(-1);
106 }
107 //Se procede a establecer la cola de conexiones entrantes asociada al socket
108 listen(socket_router, 5);
109
110 while(1) {
111     //vinculando al cliente para poder atenderlo
112     socket_cliente = accept(socket_router, (struct sockaddr *)&dir_cliente, &tas);
113     pthread_t hi;
114     pthread_create(&hi, NULL, hilo, &socket_cliente);
115 }
116 }

```

SERVIDOR

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/socket.h>
7 #include <arpa/inet.h>
8 #include <netdb.h>
9 #include <ctype.h>
10
11
12
13
14 int main(int argc, char *argv[]) {
15     const char protocolo[] = "tcp";
16     int socket_servidor, socket_router;
17     struct sockaddr_in dir_servidor, dir_router;
18     unsigned short puerto_servidor = 4500u;
19     struct protoent *protoent;
20     int habilitar_reuso_socket=1;
21
22     //Obtiene la configuración para el socket TCP
23     protoent = getprotobyname(protocolo);
24
25
26
27     //Crea el socket del dominio Internet, para comunicaciones orientadas a conexión y con el protocolo seleccionado
28     socket_servidor = socket(AF_INET, SOCK_STREAM, protoent->p_proto);
29     //Se configura el socket para ser reusado cuando se cierre el programa
30     setsockopt(socket_servidor, SOL_SOCKET, SO_REUSEADDR, &habilitar_reuso_socket, sizeof(int));
31

```

```

32 //Se configura la dirección en la que escucha el servidor
33 dir_servidor.sin_family = AF_INET;
34 dir_servidor.sin_addr.s_addr = htonl(INADDR_ANY);
35 dir_servidor.sin_port = htons(puerto_servidor);
36
37 //Se asocia la dirección al socket para escuchar conexiones entrantes
38 if (bind(socket_servidor, (struct sockaddr *)&dir_servidor, sizeof(struct sockaddr_in)) == -1) {
39     perror("Error en la asignación del puerto al socket, el puerto está en uso");
40     exit(-1);
41 }
42 //Se procede a establecer la cola de conexiones entrantes asociada al socket
43 listen(socket_servidor, 5);
44
45 while(1) {
46     char buffer[200];
47     char cadena[500];
48     int nbytes;
49     int tam = sizeof(struct sockaddr_in);
50     int enter_presente=0;
51
52     //limpia la cadena
53     memset(cadena, 0, 500);
54     //El servidor se detiene a escuchar una conexión entrante a través de la red y devuelve un nuevo socket
55     //vinculado al cliente para poder atenderlo
56     socket_router = accept(socket_servidor, (struct sockaddr *) &dir_router, &tam);
57     while(!enter_presente) {
58         //limpia el buffer
59         memset(buffer, 0, 200);
60         //Se lee una cadena desde el socket cliente
61         nbytes = read(socket_router, buffer, 200);
62         //Se copia el buffer leído hacia la cadena

```

```

62         //Se copia el buffer leído hacia la cadena
63         strncat(cadena, buffer, nbytes);
64         //Si el buffer contiene un ENTER se termina la lectura
65         if (strstr(buffer, "\n"))
66             enter_presente = 1;
67     }
68     //Se calcula la longitud de la cadena completa memset
69     nbytes = strlen(cadena);
70     //Se imprime la cadena leída desde la conexión
71     printf("El router ha enviado: %s", cadena);
72     //La cadena se convierte en mayúsculas
73     for(int i=0; i<nbytes; i++)
74         cadena[i] = toupper(cadena[i]);
75     //Se devuelve un mensaje al cliente
76     write(socket_router, cadena, nbytes);
77     //Se cierra el socket cliente
78     close(socket_router);
79 }
80 }

```

ROUTER

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <arpa/inet.h>
8  #include <netdb.h>
9  #include <ctype.h>
10
11
12 void hilo(void *sock){
13     int socket_cliente=((int *)sock);
14     int socket_servidor;
15     char buffer[200];
16     char cadena[500];
17     int nbytes;
18     //configuracion para el server
19     socket_servidor = socket(AF_INET, SOCK_STREAM, protoent->p_proto);
20
21
22     dir_servidor.sin_family = AF_INET;
23     dir_servidor.sin_addr.s_addr = INADDR_ANY;
24     dir_servidor.sin_port = htons(puerto_servidor);
25
26     if (connect(socket_servidor, (struct sockaddr *)&dir_servidor, sizeof(struct sockaddr_in)) == -1) {
27         perror("Error en la conexión al router");
28         exit(-1);
29     }
30
31     int tam = sizeof(struct sockaddr_in);
32     int enter_presente=0;
```

```
32     int enter_presente=0;
33
34     //limpia la cadena
35     memset(cadena, 0, 500);
36     //El router se detiene a escuchar una conexión entrante o trabaja de la red y devuelve un nuevo socket
37
38     while(!enter_presente) {
39         //limpia el buffer
40         memset(buffer, 0, 200);
41         //Se lee una cadena desde el socket cliente
42         nbytes = read(socket_cliente, buffer, 200);
43         //Se copia el buffer leído hacia la cadena
44         strncat(cadena, buffer, nbytes);
45         //Si el buffer contiene un ENTER se termina la lectura
46         if (strstr(buffer, "\n"))
47             enter_presente = 1;
48     }
49     //Se calcula la longitud de la cadena completa memset
50     nbytes = strlen(cadena);
51     //Se imprime la cadena leída desde la conexión
52     printf("El cliente ha enviado: %s", cadena);
53
54     //Se envía al servidor
55     write(socket_servidor, cadena, nbytes);
56
57     //se lee del servidor y se envía al cliente
58     memset(cadena, 0, 500);
59     enter_presente = 0;
60     while(!enter_presente) {
61         //limpia el buffer
62     }
```

```

62     int enter_presente=0;
63
64     //Limpia la cadena
65     memset(cadena, 0, 500);
66     //El router se dedica a escuchar una conexión entrante a través de la red y devuelve un nuevo socket
67     //vinculado al cliente para poder atenderlo
68     socket_cliente = accept(socket_router, (struct sockaddr *) &dir_cliente, &tar);
69     while(!enter_presente) {
70         //Limpia el buffer
71         memset(buffer, 0, 200);
72         //Se lee una cadena desde el socket cliente
73         nbytes = read(socket_cliente, buffer, 200);
74         //Se copia el buffer leído hacia la cadena
75         strncat(cadena, buffer, nbytes);
76         //Si el buffer contiene un ENTER se termina la lectura
77         if (strstr(buffer, "\n"))
78             enter_presente = 1;
79     }
80     //Se calcula la longitud de la cadena completa memset
81     nbytes = strlen(cadena);
82     //Se imprime la cadena leída desde la conexión
83     printf("El cliente ha enviado: %s", cadena);
84
85     //Se envía al servidor
86     write(socket_servidor, cadena, nbytes);
87

```

```

88
89     //se lee del servidor y se envía al cliente
90     memset(cadena, 0, 500);
91     enter_presente = 0;
92     while(!enter_presente) {
93         //Limpia el buffer
94         memset(buffer, 0, 200);
95         //Se lee una cadena desde el socket cliente
96         nbytes = read(socket_servidor, buffer, 200);
97         //Se copia el buffer leído hacia la cadena
98         strncat(cadena, buffer, nbytes);
99         //Si el buffer contiene un ENTER se termina la lectura
100        if (strstr(buffer, "\n"))
101            enter_presente = 1;
102    }
103
104    write(socket_cliente, cadena, nbytes);
105
106    //Se cierra el socket cliente
107    close(socket_cliente);
108
109 }

```

CONCLUSIONES

Este conjunto de código nos sirve como un recurso educativo para quienes quieran experimentar con el manejo de redes con sockets, también para aquellos que quieran visualizar la manera de manejar redes en c y saber tanto datos generales como específicos del código. La documentación proporciona una visión detallada del sistema, sus componentes y su funcionamiento, facilitando el aprendizaje y la aplicación práctica de los conceptos de programación de red en un entorno de sockets TCP.