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

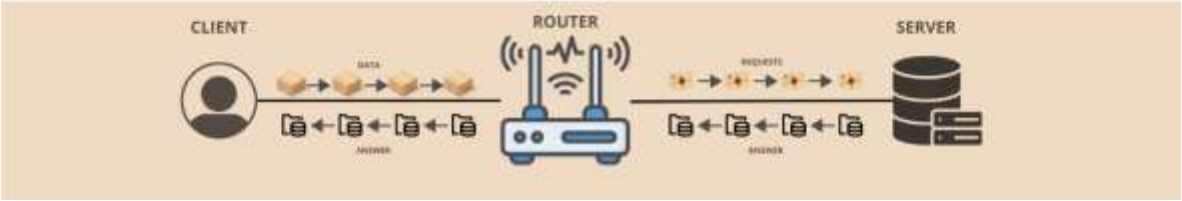
El router establece conexiones simultáneas con el servidor y el cliente. Actúa como un intermediario para facilitar la comunicación entre estos dos componentes.

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 tambien se encarga de la traducción de direcciones (convierte direcciones privadas a públicas).



CODIGO

CLIENTE

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <ctype.h>



int main(int argc, char *argv[]) {
    const char protocolo[]="tcp";
    int socket_router;
    struct sockaddr_in dir_router;
    unsigned short puerto_router = 9090u;
    char *direccion_router;
    struct protoent *protoent;
    struct hostent *hostent;
    in_addr_t in_addr;

    direccion_router = "127.0.0.1";


    //Obtiene la configuración para el socket TCP
    protoent = getprotobyname(protocolo);
```

```
            //Limpia el buffer
            memset(buffer, 0, 200);
            //Se lee una cadena desde el socket cliente
            nbytes = read(socket_servidor, buffer, 200);
            //Se copia el buffer leído hacia la cadena
            strncat(cadena, buffer, nbytes);
            //Si el buffer contiene un ENTER se termina la lectura
            if (strstr(buffer, "\n"))
                enter_presente = 1;
        }

        write(socket_cliente, cadena, nbytes);

        //Se cierra el socket cliente
        close(socket_cliente);
    }

int main(int argc, char *argv[]) {
    const char protocolo[]="tcp";
    int socket_router, socket_cliente;
    struct sockaddr_in dir_router, dir_cliente, dir_servidor;
    unsigned short puerto_router = 9090u, puerto_servidor = 4500u;
    struct protoent *protoent;
    int habilitar_reuso_socket=1;

    //Obtiene la configuraciÃ³n para el socket TCP
    protoent = getprotobyname(protocolo);



    //Crea el socket del dominio Internet, para comunicaciones orientadas a conexiÃ³n y con el protocolo seleccionado
```

```
    //Crea el socket del dominio Internet, para comunicaciones orientadas a conexiÃ³n y con el protocolo seleccion
    socket_router = socket(AF_INET, SOCK_STREAM, protoent->p_proto);
    //Se configura el socket para ser reusado cuando se cierre el programa
    setsockopt(socket_router, SOL_SOCKET, SO_REUSEADDR, &habilitar_reuso_socket, sizeof(int));

    //Se configura la direcciÃ³n en la que escucha el router
    dir_router.sin_family = AF_INET;
    dir_router.sin_addr.s_addr = htonl(INADDR_ANY);
    dir_router.sin_port = htons(puerto_router);

    //Se asocia la direcciÃ³n al socket para escuchar conexiones entrantes
    if (bind(socket_router, (struct sockaddr *)&dir_router, sizeof(struct sockaddr_in)) == -1) {
        perror("Error en la asignaciÃ³n del puerto al socket, el puerto estÃ¡ en uso");
        exit(-1);
    }
    //Se procede a establecer la cola de conexiones entrantes asociada al socket
    listen(socket_router, 5);

    while(1) {
    //vinculado al cliente para poder atenderlo
        socket_cliente = accept(socket_router, (struct sockaddr *) &dir_cliente, &tam);
        pthread_t h1;
    pthread_create(&h1,NULL,hilo,&socket_cliente);
    }
}
```

SERVIDOR

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <ctype.h>



int main(int argc, char *argv[]) {
    const char protocolo[]="tcp";
    int socket_servidor, socket_router;
    struct sockaddr_in dir_servidor, dir_router;
    unsigned short puerto_servidor = 4500u;
    struct protoent *protoent;
    int habilitar_reuso_socket=1;

    //Obtiene la configuraciÃ³n para el socket TCP
    protoent = getprotobyname(protocolo);



    //Crea el socket del dominio Internet, para comunicaciones orientadas a conexiÃ³n y con el protocolo seleccionado
    socket_servidor = socket(AF_INET, SOCK_STREAM, protoent->p_proto);
    //Se configura el socket para ser reusado cuando se cierre el programa
    setsockopt(socket_servidor, SOL_SOCKET, SO_REUSEADDR, &habilitar_reuso_socket, sizeof(int));
```

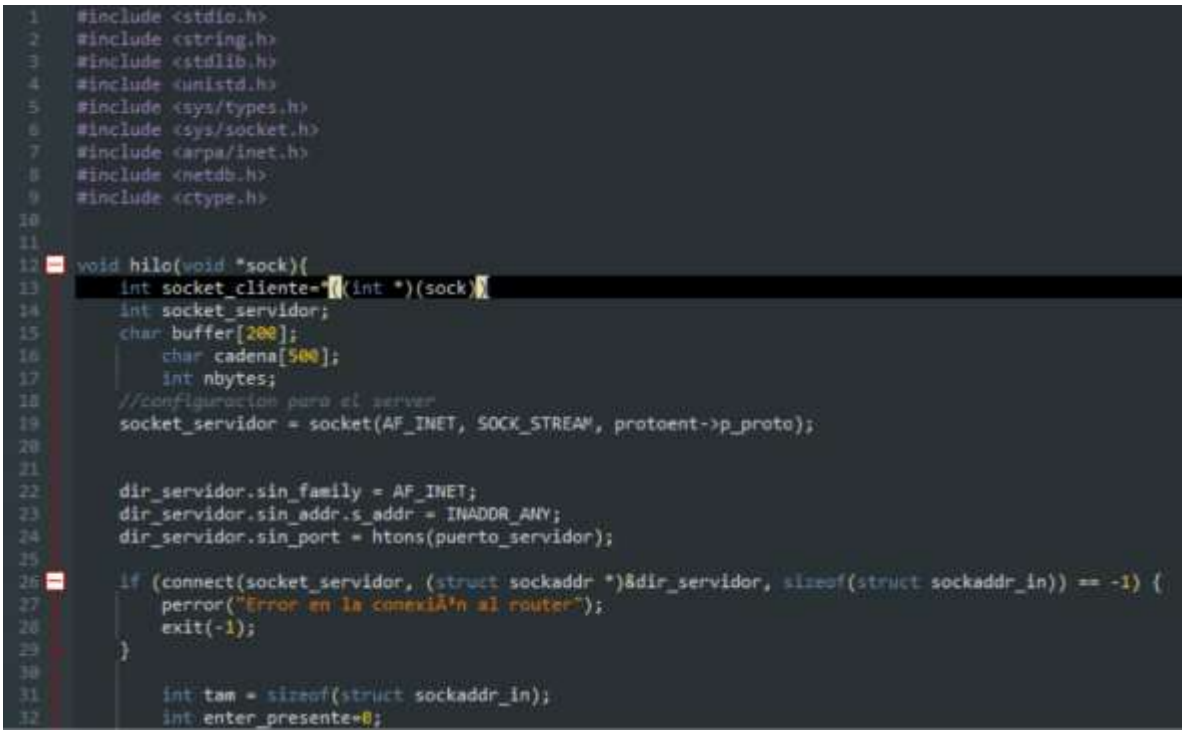
```
    //Se configura la direcciÃ³n en la que escucha el servidor
    dir_servidor.sin_family = AF_INET;
    dir_servidor.sin_addr.s_addr = htonl(INADDR_ANY);
    dir_servidor.sin_port = htons(puerto_servidor);

    //Se asocia la direcciÃ³n al socket para escuchar conexiones entrantes
    if (bind(socket_servidor, (struct sockaddr *)&dir_servidor, sizeof(struct sockaddr_in)) == -1) {
        perror("Error en la asignaciÃ³n del puerto al socket, el puerto estÃ¡ en uso");
        exit(-1);
    }
    //Se procede a establecer la cola de conexiones entrantes asociado al socket
    listen(socket_servidor, 5);

    while(1) {
        char buffer[200];
        char cadena[500];
        int nbytes;
        int tam = sizeof(struct sockaddr_in);
        int enter_presente=0;

        //Limpia la cadena
        memset(cadena, 0, 500);
        //El servidor se detiene a escuchar una conexiÃ³n entrante a través de la red y devuelve un nuevo socket
        //vinculado al cliente para poder atenderlo
        socket_router = accept(socket_servidor, (struct sockaddr *) &dir_router, &tam);
        while(!enter_presente) {
            //Limpia el buffer
            memset(buffer, 0, 200);
            //Se lee una cadena desde el socket cliente
            nbytes = read(socket_router, buffer, 200);
            //Se copia el buffer leido hacia la cadena
            strncat(cadena, buffer, nbytes);
            //Si el buffer contiene un ENTER se termina la lectura
            if (strstr(buffer, "\n"))
                enter_presente = 1;
        }
        //Se calcula la longitud de la cadena completa memset
        nbytes = strlen(cadena);
        //Se imprime la cadena leida desde la conexiÃ³n
        printf("El router ha enviado: %s", cadena);
        //La cadena se convierte en mayÃºsculas
        for(int i=0; i<nbytes; i++)
            cadena[i] = toupper(cadena[i]);
        //Se devuelve un mensaje al cliente
        write(socket_router, cadena, nbytes);
        //Se cierra el socket cliente
        close(socket_router);
    }
}
```

ROUTER



```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <ctype.h>


void hilo(void *sock){
    int socket_cliente=*(int *)(sock);
    int socket_servidor;
    char buffer[200];
        char cadena[500];
        int nbytes;
    //configuracion para el server
    socket_servidor = socket(AF_INET, SOCK_STREAM, protoent->p_proto);


    dir_servidor.sin_family = AF_INET;
    dir_servidor.sin_addr.s_addr = INADDR_ANY;
    dir_servidor.sin_port = htons(puerto_servidor);

    if (connect(socket_servidor, (struct sockaddr *)&dir_servidor, sizeof(struct sockaddr_in)) == -1) {
        perror("Error en la conexiÃ³n al router");
        exit(-1);
    }

        int tam = sizeof(struct sockaddr_in);
        int enter_presente=0;
```

```c
        int enter_presente=0;

        //Limpia la cadena
        memset(cadena, 0, 500);
        //El router se detiene a escuchar una conexiÃ³n entrante a travÃ©s de la red y devuelve un nuevo socket

        while(!enter_presente) {
            //Limpia el buffer
            memset(buffer, 0, 200);
            //Se lee una cadena desde el socket cliente
            nbytes = read(socket_cliente, buffer, 200);
            //Se copia el buffer leido hacia la cadena
            strncat(cadena, buffer, nbytes);
            //Si el buffer contiene un ENTER se termina la lectura
            if (strstr(buffer, "\n"))
                enter_presente = 1;
        }
        //Se calcula la longitud de la cadena completa memset
        nbytes = strlen(cadena);
        //Se imprime la cadena leida desde la conexiÃ³n
        printf("El cliente ha enviado: %s", cadena);

        //Se envia al servidor
        write(socket_servidor, cadena, nbytes);


        //se lee del servidor y se envia al cliente
        memset(cadena, 0, 500);
        enter_presente = 0;
        while(!enter_presente) {
            //Limpia el buffer
```

```
    int enter_presente=0;

    //Limpia la cadena
    memset(cadena, 0, 500);
    //El router se detiene a escuchar una conexiÃ³n entrante a través de la red y devuelve un nuevo socket
    //vinculado al cliente para poder atenderlo
    socket_cliente = accept(socket_router, (struct sockaddr *) &dir_cliente, &tam);
    while(!enter_presente) {
        //Limpia el buffer
        memset(buffer, 0, 200);
        //Se lee una cadena desde el socket cliente
        nbytes = read(socket_cliente, buffer, 200);
        //Se copia el buffer leido hacia la cadena
        strncat(cadena, buffer, nbytes);
        //Si el buffer contiene un ENTER se termina la lectura
        if (strstr(buffer, "\n"))
            enter_presente = 1;
    }
    //Se calcula la longitud de la cadena completa memset
    nbytes = strlen(cadena);
    //Se imprime la cadena leida desde la conexiÃ³n
    printf("El cliente ha enviado: %s", cadena);

    //Se envia al servido
    write(socket_servidor, cadena, nbytes);
```

```
        //se lee del servidor y se envia al cliente
        memset(cadena, 0, 500);
        enter_presente = 0;
        while(!enter_presente) {
            //Limpia el buffer
            memset(buffer, 0, 200);
            //Se lee una cadena desde el socket cliente
            nbytes = read(socket_servidor, buffer, 200);
            //Se copia el buffer leido hacia la cadena
            strncat(cadena, buffer, nbytes);
            //Si el buffer contiene un ENTER se termina la lectura
            if (strstr(buffer, "\n"))
                enter_presente = 1;
        }

        write(socket_cliente, cadena, nbytes);

        //Se cierra el socket cliente
        close(socket_cliente);
    }
}
```

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.