CSE130: Principles of Computer Systems Design

Teaching Assistant: Nilufar Ferdous

Socket Programming

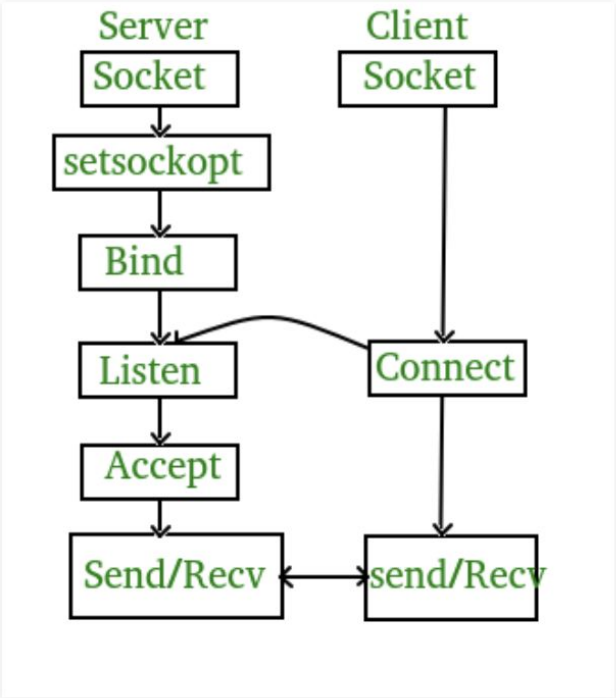
Socket Programming in C/C++

Socket programming is a way of connecting two nodes on a network to communicate with each other:

- One socket(node) listens on a particular port at an IP
- while other socket reaches out to the other to form a connection.
- Server forms the listener socket while client reaches out to the server.

Server-Client Model

State diagram for server and client model



Steps in Socket Programming: Server:Socket()

Socket creation:

int sockfd = socket(domain, type, protocol)

Example: Int sockfd_1= socket(AF_INET, SOCK_STREAM, 0);

- **sockfd_1:** socket descriptor returns an integer
- **domain:** integer, communication domain e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol)
- **type:** communication type
  - SOCK_STREAM: TCP(reliable, connection oriented)
  - SOCK_DGRAM: UDP(unreliable, connectionless)
- **protocol:** Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

SETSOCKOPT()

Setsockopt:

int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);

- This helps in manipulating options for the socket referred by the file descriptor sockfd.
- This is completely optional,
- but it helps in reuse of address and port.
- Prevents error such as: “address already in use”.

Bind()

Bind:

int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

After creation of the socket, bind function binds the socket :

- to the address and
- port number specified in addr. In the example code, for example, bind the server to the localhost

Listen()

Listen:

int listen(int sockfd, int backlog);

- It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection.
- The backlog, defines the maximum length to which the pending connections for sockfd may grow. If a connection request arrives when the queue of connection is full, the client may receive an error with an indication of ECONNREFUSED.

Accept()

Accept:

int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);

Example: int handler = accept(server, (struct sockaddr *)&client, &size);

- It extracts the first connection request on the queue for the listening socket,
- sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket.
- At this point, connection is established between client and server, and they are ready to transfer data.

Stages for Client: Socket Creation

Socket creation:

int sockfd_client = socket(domain, type, protocol)

- **sockfd:** socket descriptor returns an integer
- **domain:** integer, communication domain e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol)
- **type:** communication type
  - SOCK_STREAM: TCP(reliable, connection oriented)
  - SOCK_DGRAM: UDP(unreliable, connectionless)
- **protocol:** Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

Client: Connect()

Connect:

int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

- The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr.
- Server’s address and port is specified in addr.

Send & Receive System calls:

Send ()

Recv()

Server.cpp: socket()

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Hello from server";

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
```

Structure: Sockaddr_in

```
#include <netinet/in.h>

struct sockaddr_in {
    short            sin_family;   // e.g. AF_INET
    unsigned short   sin_port;     // e.g. htons(3490)
    struct in_addr   sin_addr;     // see struct in_addr, below
    char             sin_zero[8];  // zero this if you want to
};

struct in_addr {
    unsigned long s_addr;  // load with inet_aton()
};
```

Bind () & Listen()

```
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( PORT );

// Forcefully attaching socket to the port 8080
if (bind(server_fd, (struct sockaddr *)&address,
                                 sizeof(address))<0)
{
    perror("bind failed");
    exit(EXIT_FAILURE);
}
if (listen(server_fd, 3) < 0)
{
    perror("listen");
    exit(EXIT_FAILURE);
}
```

Accept()

```
if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                    (socklen_t*)&addrlen))<0)
{
    perror("accept");
    exit(EXIT_FAILURE);
}
valread = read( new_socket , buffer, 1024);
printf("%s\n",buffer );
send(new_socket , hello , strlen(hello) , 0 );
printf("Hello message sent\n");
```

Program Functionality:

- code may be either C or C++
- all source files must have a .cpp suffix and be
- compiled by clang++ with no errors or warnings
- flags: clang++ -std=gnu++11 -Wall -Wextra -Wpedantic -Wshadow


Question?


thank you