CS6111 - COMPUTER NETWORKS LAB 2

SOCKET PROGRAMMING

Name : Ajitesh M

Reg. No : 2019103503

Aim :

To explore socket programming and establish a TCP connection between server and client.

STUDY OF SOCKET PROGRAMMING

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.

The system calls required for socket programming is present in sys/socket.h

Socket() :

Used to create a socket.

```
int sockfd = socket(domain, type, protocol)
```

Parameters :

domain – communication domain. For TCP IPv4 – AF_INET

type – communication type. For TCP – SOCK_STREAM

protocol – protocol value for IP is 0.

Return value :

Sockfd – Socket file descriptor

If sockfd < 0 => Error in socket creation.

Setsockopt() :

This helps in manipulating options for the socket referred by the file descriptor sockfd.

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

Returns 0 on success, -1 on failure.

This is completely optional, but it helps in reuse of address and port.

Prevents error such as: “address already in use”.

Bind() :

After creating a socket, we have to bind the socket to the address specified in the sockaddr structure which holds the values of socket family, port, and IP address.

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

Returns 0 on success, -1 on failure.

Listen():

Puts the server socket in a passive mode – waits for the client requests.

Backlog – maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error.

```
int listen(int sockfd, int backlog);
```

Returns 0 on success, -1 on failure.

Accept():

Awaits for client connection request, and extracts the first connection request from the queue of pending connections, creates a new connected socket to serve up the particular client and returns the same. Returns -1 on failure.

At this point, connection is established between client and server, and they are ready to transfer data.

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

Connect():

Used to connect the socket referred by sockfd to the address specified by addr. Usually the client socket is mentioned and the server address is specified in addr. Returns -1 on failure.

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

Read() , Recv() :

Both are used to read incoming data on the socket into a buffer, returning the count of number of bytes read.
Blocking in nature.

```
ssize_t read(int fs, void *buf, ssize_t N); // Defined in unistd.h
ssize_t recv(int sockfd, void *buf, size_t len, int flags); // Defined in sys/socket.h
```

Write(), send() :

Both are used to write/send data to the socket into a buffer, returning the count of number of bytes sent.
Blocking in nature.

```
ssize_t write(int fs, void *buf, ssize_t N); // Defined in unistd.h
ssize_t send(int sockfd, void *buf, size_t len, int flags); // Defined in sys/socket.h
```

Code :

Server.c

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

#define PORT 5000
#define MAX 2048

int main()
{
    int sockfd;
    struct sockaddr_in addr;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    if (sockfd < 0)
    {
        printf("Socket creation failed\n");
        exit(1);
    }
    else
    {
        printf("Socket created\n");
    }

    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(sockfd, (struct sockaddr *)&addr, sizeof(addr)) < 0)
    {
        printf("Socket bind failed\n");
        exit(1);
    }
    else
    {
        printf("Socket binded.\n");
    }

    if (listen(sockfd, 5) != 0)
    {
        printf("Listen failed\n");
        exit(1);
    }
    else
    {
        printf("Socket listening \n");
```

```
    }

    int newsockfd;
    struct sockaddr_in clientaddr;
    int len = sizeof(clientaddr);
    newsockfd = accept(sockfd, (struct sockaddr *)&clientaddr, &len);

    char buffer[1024];
    read(newsockfd, buffer, 1024);
    printf("From client : %s\n", buffer);

char* msg = "Welcome client!!";
    write(newsockfd, msg, strlen(msg));
    close(newsockfd);
    return 0;
}
```

Client.c

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

#define PORT 5000
#define MAX 2048

int main()
{
    int sockfd;
    struct sockaddr_in addr;

    // Socket Creation
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        printf("Socket creation failed\n");
        exit(1);
    }
    else
    {
        printf("Socket created.\n");
    }

    // Server socket connection
    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    if (connect(sockfd, (struct sockaddr *)&addr, sizeof(addr)) < 0)
    {
        printf("Server connection failed.\n");
```

```
        exit(1);
    }
    else
    {
        printf("Server connected.\n");
    }

    char* msg = "Hello Server !! ";
    int wcnt = write(sockfd, msg, strlen(msg));
    if(wcnt == 0)
          printf("Unable to write to the server\n");

    char buffer[1024];
    read(sockfd, buffer, 1024);
    printf("From Server : %s\n", buffer);
    close(sockfd);
    return 0;
}
```

Output :