Lab Assignment : 1

IT-304 Lab 2024-2025

List of Topics: Socket Programming

Socket Programming:

In computer networking, an Internet socket or network socket is an endpoint of a bidirectional inter process communication flow across an internet protocol based computer network such as the internet. The term internet socket is also used as a name for an application programming interface (API) for the TCP/IP protocol stack, usually provided by the operating system. Internet sockets constitute a mechanism for delivering incoming data packets to the appropriate application process or thread, based on a combination of local and remote IP address and port numbers. Each socket is mapped by the operating system to a communicating application process or thread. A socket address is the combination of an IP address (the location of the computer) and a port (which is mapped to the application program process) into a single identity, much like one end of a telephone connection is the combination of a phone number and a particular extension.

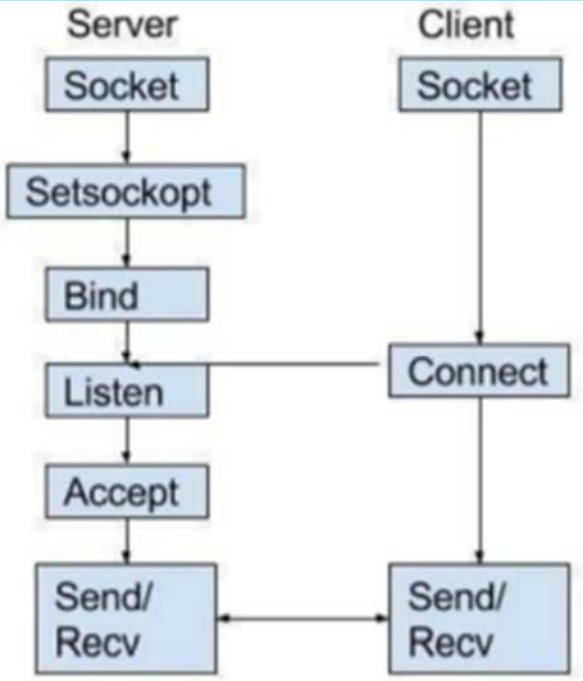
Two types of internet sockets:

1. **Stream sockets:** They are connection oriented reliable sockets also called TCP sockets.
2. **Datagram sockets:** They are connection-less unreliable sockets also called UDP sockets

TCP client server model

Most inter-process communication across the network uses the client server model. These terms refer to the two processes which will be communicating with each other. One of the two processes, the client, connects to the other process, the server, typically to make a request for information. Notice that the client needs to know of the existence of and the address of the server, but the server does not need to know the address of (or even the existence of) the client prior to the connection being established.

Notice also that once a connection is established, both sides can send and receive information. The system calls for establishing a connection are somewhat different for the client and the server, but both involve the basic construct of a socket. A socket is one end of an inter-process communication channel. The two processes each establish their own socket as shown in figure 1. The steps involved in establishing a socket on the client side are as follows:



Server side commands

1. **Socket creation:**

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

- sockfd: socket descriptor, an integer (like a file-handle)
- domain: integer, communication domain e.g., AF INET (IPv4 pro tocol) , AF INET6 (IPv6 protocol). (Here AF stands for Address Family.)
- 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 the protocol field in the IP header of a packet.

2. **Setsockopt:**set the socket options

```
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 de scriptor sockfd. An application program can use setsockopt() to allocate buffer space, control timeouts, or permit socket data broadcasts.

3. **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(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR ANY to specify the IP address.

4. **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 queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

5. **Accept:**

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

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

2.2 Client side commands

1. **Socket connection:** Exactly same as that of server's socket creation
2. **Connect:**

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

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

3. **For sending and receiving data:**
    a. **read()** or send() / sendto()
    b. **write()** or recv() / recvfrom()

system calls are used.

2.3 How to run

1. Open a terminal
2. Type “gcc server.c -o server”, and press enter.
3. Type “./server”, and press enter.
4. Open 2nd terminal
5. Type “gcc client.c -o client”, and press enter.
6. Type “./client”, and press enter.
7. Both the server and client are running simultaneously.
8. Note: Always run the server first.

Exercise:

1. Create TCP server and client using sockets library. Make them communicate with each other by making a Q&A system between them. Make a list of 4-5 Q&A pairs in a file and use these for the application.
2. Create UDP server and client using socket programming.
    a. Since UDP doesn’t support reliability, sometimes packets may be received with errors. Errors can be detected using either CRC or checksums. Use a function to compute a 32 bit checksum over the data packet and include it in the packet header. Let the server and client check for error before accepting a packet else, responding with an error message.
    b. Since error rates on LAN is extremely low, test your code by artificially introducing introducing error by either changing a character in the message or adding a wrong CRC header.

Submission:

1. Submit assignment with the report consists of an input file and output file with proper explanation of each output of all the exercises in pdf format.
2. Add all the outputs and a brief description of the commands used in the given demo scripts in the report.
3. Submitted code in a report should be well commented.
4. Submit a zip file with your student id which will consist of the folder for each script & respective outputs. one common report in a parent directory.

    Eg:

group-id-2022***.zip

5. Submission Deadline: 8 Aug,2024

References:

Here are some nice links for better understanding of sockets programming:

1. https://beej.us/guide/bgnet/html/index-wide.html
2. https://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html
3. https://www.geeksforgeeks.org/socket-programming-cc/