Assignment No. 3: Socket Programming in C/C++ on Linux

Problem Statement :

Socket Programming in C/C++ on Linux

A] TCP Client, TCP Server

B] UDP Client, UDP Server

Theory:

A **network socket** is an endpoint of an inter-process communication flow across a computer network. Today, most communication between computers is based on the Internet Protocol; therefore most network sockets are **Internet sockets**.

A **socket API** is an application programming interface (API), usually provided by the operating system, that allows application programs to control and use network sockets. Internet socket APIs are usually based on the Berkeley sockets standard.

A **socket address** is the combination of an IP address and a port number, much like one end of a telephone connection is the combination of a phone number and a particular extension. Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

Socket types

Several Internet socket types are available:

- Datagram sockets, also known as connectionless sockets, which use User Datagram Protocol (UDP).
- Stream sockets, also known as connection-oriented sockets, which use Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP).
- Raw sockets (or *Raw IP sockets*), typically available in routers and other network equipment. Here the transport layer is bypassed, and the packet headers are made accessible to the application.

TCP Client -Server

TCP Socket API

The sequence of function calls for the client and a server participating in a TCP connection is presented in Figure 1.

As shown in the figure, the steps for establishing a TCP socket on the client side are the following:

- Create a socket using the socket() function;
- Connect the socket to the address of the server using the connect() function;
- Send and receive data by means of the read() and write() functions.
- Close the connection by means of the close() function.

The steps involved in establishing a TCP socket on the server side are as follows:

- Create a socket with the socket() function;
- Bind the socket to an address using the bind() function;
- Listen for connections with the listen() function;
- Accept a connection with the accept() function system call. This call typically blocks until a client connects with the server.

- Send and receive data by means of send() and receive().
- Close the connection by means of the close() function.

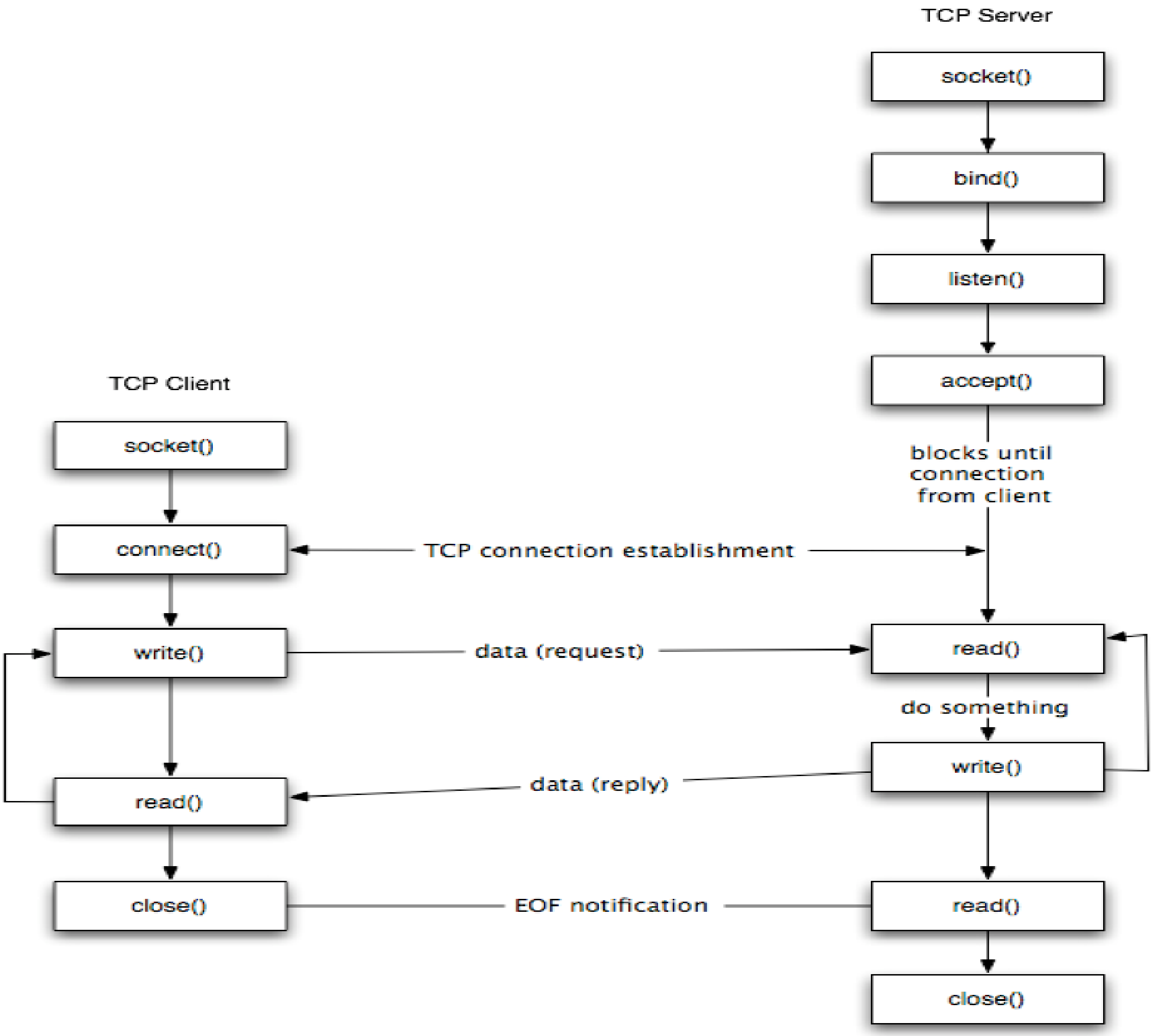

Figure 1: TCP client-server.

UDP Socket API

There are some fundamental differences between TCP and UDP sockets. UDP is a connection-less, unreliable, datagram protocol (TCP is instead connection-oriented, reliable and stream based). There are some instances when it makes to use UDP instead of TCP. Some popular applications built around UDP are DNS, NFS, SNMP and for example, some Skype services and streaming media.

Figure 2 shows the the interaction between a UDP client and server. First of all, the client does not establish a connection with the server. Instead, the client just sends a datagram to the server using the sendtofunction which requires the address of the destination as a parameter. Similarly, the server does not accept a connection from a client. Instead, the server just calls the recvfrom function, which waits until data arrives from some client. recvfrom returns the IP address of the client, along with the datagram, so the server can send a response to the client.

As shown in the Figure, the steps of establishing a UDP socket communication on the client side are as follows:

□ Create a socket using the socket() function;

□ Send and receive data by means of the recvfrom() and sendto() functions.

The steps of establishing a UDP socket communication on the server side are as follows:

□ Create a socket with the socket() function;

□ Bind the socket to an address using the bind() function;

□ Send and receive data by means of recvfrom() and sendto().

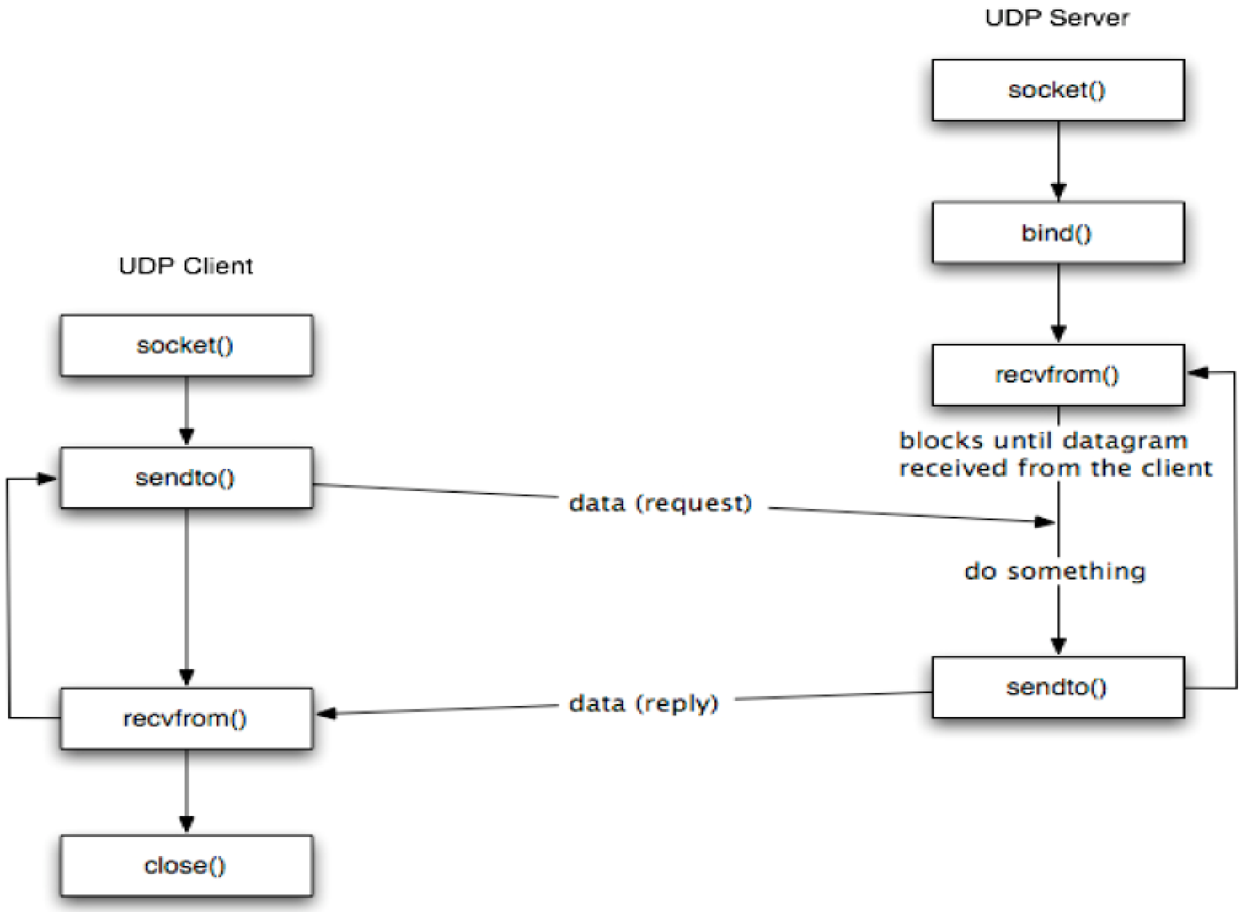

Figure 2: UDP client-server.

Conclusion – Thus we have successfully studied socket Programming with TCP and UDP socket.