Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Review of Sockets Used for Communication Purpose

Sarika Choudhary, Ritika Saroha & Sonal Beniwal
School of Engineering & Sciences
BPS Mahila Vishwavidyalaya
Sonepat, Haryana
India

***Abstract*— Study of this paper will represent a perspective view on sockets which are used for communication purpose between two processes. In this paper we will discuss on types of sockets and their functions. Aim of brief discussion about TCP and UDP protocol sockets and functions.**

***Keywords*—TCP, UDP, IPv4, IPv6, Socket**

I. INTRODUCTION

Socket is the most essential component of computer networking. Basically socket is software not hardware because it opens new connections for the processes allowing data to be read and write. When a computer program wants to connect to a LAN or WAN such as internet, it uses the socket i.e. software component.

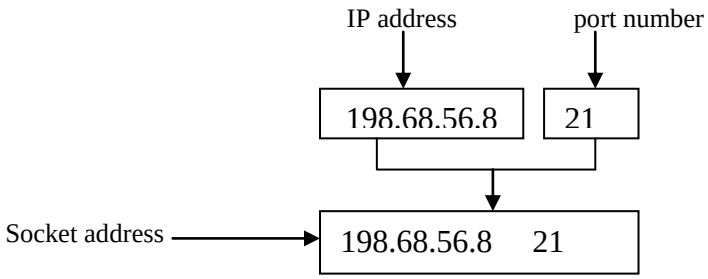
A socket is used to allow one process to speak or connect to another, very much like the telephone is used to allow one person to speak with another. Socket is two way communication links between two processes running on the network. It is an end point of two processes.

There are basically two types of sockets:

a) Elementary TCP Socket- reliable and connection oriented link.
b) Elementary UCP Socket- unreliable and connectionless link.

II. SOCKET ADDRESS

Before going to the discussion with types of sockets we have to know about the address of sockets. Socket address is the combination of two addresses; these are IP address and Port number. Socket provides process-to-process delivery and for this we need these two addresses.



IP address: it is an internet protocol address on the network layer in OSI model which is used to choose host process among many. It is used for host-to-host delivery.

Port number: it works on transport layer which is used to choose particular process among millions. It is used for process-to-process delivery.

III. SOCKET ADDRESS STRUCTURE

In our network each protocol defines its own socket address structure like IPv4, IPv6 etc describes its own structure. Names of these structures begin with sockaddr_ and end with a suffix. Ending of socket address structure is different for each protocol suit. It is used on a given host. This structure is not used for communication purpose like IP address and Port number. Here we will discuss two of them socket address structure:

A. ***IPv4 socket address structure:*** it is also called an "Internet socket address structure". Its name started with sockaddr_in. In this we use <netinet/in.h> header library for executing this structure. Here we shows the IPv4 socket address structure : (sockaddr_in)

```
struct in_addr
{
```



```
In_addr_t       s_addr;
};
Struct sockaddr_in
{
uint8_t         sin_len;
sa_family_t     sin_family;
in_port_t       sin_port;
struct in_addr  sin_addr;
char            sin_zero[8];
};
```

We are listing here three posix-defined Datatypes:

Data types	**Description**	**Header**
int8_t	Signed 8-bit integer	<sys/types.h>
uint8_t	Unsigned 8-bit integer	<sys/types.h>
inet16_t	signed 16-bit integer	<sys/types.h>
uint16_t	Unsigned 16-bit integer	<sys/types.h>
int32_t	signed 32-bit integer	<sys/types.h>
uint32_t	Unsigned 32-bit integer	<sys/types.h>
sa_family_t	Address family of socket address structure	<sys/socket.h>
socklen_t	Length of socket address structure, normally uint32_t	<sys/socket.h>
In_addr_t	Ipv4 address, normally uint32_t	<netinet/in.h>
In_port_t	TCP or UDP port, normally uint16_t	<netinet/in.h>

B. ***IPv6 socket address structure****:* it is slightly different from IPv4. It is Internet Protocol version 6. It includes <netinet/in.h> header. Here we show the IPv6 socket address structure (sockaddr_in6):

```
struct in6_addr
{
uint8_t   s6_addr[10];
};
#define SIN6_LEN                        // required for compile time tests.

Struct sockaddr_in6
{
uint8_t       sin6_len;
sa_family_t           sin6_family;
in_port_t             sin6_port;              //TCP
uint32_t    sin6_flowinfo;          //priority & flow labels
struct in6_addr       sin6_addr;              //IPv6 address
};
```

IV. SOCKET FUNCTIONS

There are many functions of socket which are used to create the socket bind with well known port and so on. To use sockets we need to include *libraries* these are:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys.socket.h>
#include<netinet/in.h>
```

- ***socket() function :*** for creating a socket we must call this function. This function creates an end point for communication and it is similar to open a file. It returns a file descriptor representing the socket end point and it returns -1 if error occurs.

Syntax: *int socket(int family, int type, int protocol)*
Family: specify the protocol family.
Type: describe which type of service we are using.
Protocol: specify that which type of protocol is used for socket operation.

Family	Description
AF_INET AF_INET6	IPv4 protocol IPv6 protocol
AF_LOCAL AF_ROUTE AF_KEY	Unix domain protocol Routing socket Key socket

Fig: *protocol family constants for socket() function.*

Type	Description
SOCK_STREAM	Stream socket
SOCK_DGRAM	Datagram socket
SOCK_RAW	Raw socket

Fig: type *of socket for socket() function*.

	AF_INET	AF_INET6	AF_LOCAL	AF_ROUTE	AF_KEY
SOCK_STREAM	TCP	TCP	YES		
SOCK_DGRAM	UDP	UDP	YES		
SOCK_RAW	IPv4	IPv6		YES	YES

Fig: *combination of family and type for the socket() function.*

- ***bind() function:*** it assigns a local protocol address to socket. It binds a socket to a well known address for protocol. Means it binds an address and port number. It will return 0 if successful and returns -1 if error occurs.

  Syntax: *int bind(int sockfd, const struct sockaddr *myaddr, socklen_t *addrlen);*

  sockfd : is a socket file descriptor parameter returned by socket() function. A common error from bind function is EDDRINUSE (address already in use).

- ***listen() function:*** after a socket is bound to an address it is still not listening for any incoming connections. Thus, to ensure that no incoming connections are lost, a connection queue must be setup. This allows incoming connections to queue up while the server is dealing with another connection. It makes socket in passive mode.

  Syntax: *int listen(int sockfd, int backlog);*

  This function must call after the socket and bind function but before accept function in TCP protocol.
  Basically kernel maintains two queues:
  1. *Incomplete connection queue*: it contains the entry for arrived requests from the clients. It establish three-way handshaking.
  2. *Complete connection queue:* it contains the entry for each client whom requests or connection has completed.

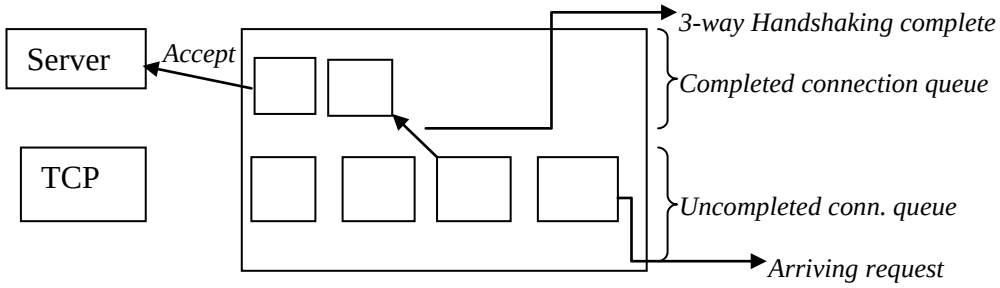


Fig: *two queues maintained by TCP for listening function.*

- ***connect() function:*** after creating a socket the client needs establish a connection with the server. It specifies the remote address that client wants to connect. Once it is done the socket is placed in active mode. It returns 0 if successful and returns -1 if error occurs.

Syntax: *int connect(int sockfd, const struct sockaddr * servaddr, socklen_t *addrlen);*

- ***accept() function:*** it is called by TCP server. It accepts the connection from the completed connection queue. To accept one we must call accept(). Once a connection is established then source and destination processes can exchange data. If the completed connection queue is empty then the process is put on to sleep or we can say socket connection is blocked till the next requests come.

  Syntax: *int accept(int sockfd, struct sockaddr *saraddr, socklen_t *addrlen);*

  addrlen- is the length size of socket address.
  If accept() function is successful then it return a non-negative descriptor and returns -1 if an error occurs.

- ***close() function:*** this function is used to close the connection when client or server is finished their work. Basic action of close is that it mark the socket as it closed and no action will performed anymore and returns to the process.

  Syntax: *int close(int sockfd);*

V. TYPES OF SOCKET

There are various type of sockets are used. The type of sockets depends upon the purpose and platforms of applications. There are three types of sockets:

a. Unix Domain Socket
b. Internet Domain Socket
   i. TCP socket
   ii. UDP socket
   iii. Raw IP socket
c. Ns Domain Socket

Internet domain socket supports all the platforms so they are widely used across all the platforms.

Internet domain socket: These types of sockets are used by all the applications which are communicating over the network. There are three types of internet domain socket:

i. *TCP Socket:* we all know that the TCP is a reliable protocol. It creates virtual connection between two TCP's to send data.
ii. *UDP Socket:* we all know that UDP is the user datagram protocol. It provides unreliable and connection less services. There is no connection established to send data.
iii. *Raw IP Socket:* it is non formatted protocol. It works on transport and network layer in OSI model. In TCP or UDP, they just only receive the payload or data whereas the raw socket receives header information of packet along with data.

VI. ELEMENTARY TCP SOCKET

As we know TCP is a connection oriented protocol. This socket is used as an end point for communication. Socket address consists: <IP address, port>. Port numbers will defined as an integer above 1024 to 65535. Because below 1024 ports are well-defined. TCP connection consists of a pair of sockets.

For example: a web server listens at port 13 for incoming requests. When a client wants to make a connection with a server socket, the client is assigned a port from the local hosts. If client's IP address is 196.56.68.8 and web server address is 196.68.56.20 and local host assigned port number is 2311 so the connection between client and server defined by the pair of sockets.

<196.56.68.8: 2311, 196.68.56.20: 13>

- *TCP Client and Server:* TCP server listens on a well-known port and accept the connection. TCP client take a lead to connection request to a TCP server. A real TCP server can accept multiple requests coming from the clients.

  Functions used by the TCP clients:
  1. Socket () 2. Connect() 3. Read() or write() 4.close()

  Functions used by the TCP server:
  1. Socket() 2. Bind() 3. Listen() 4. Accept() 5. Read() or write()
  6. close()

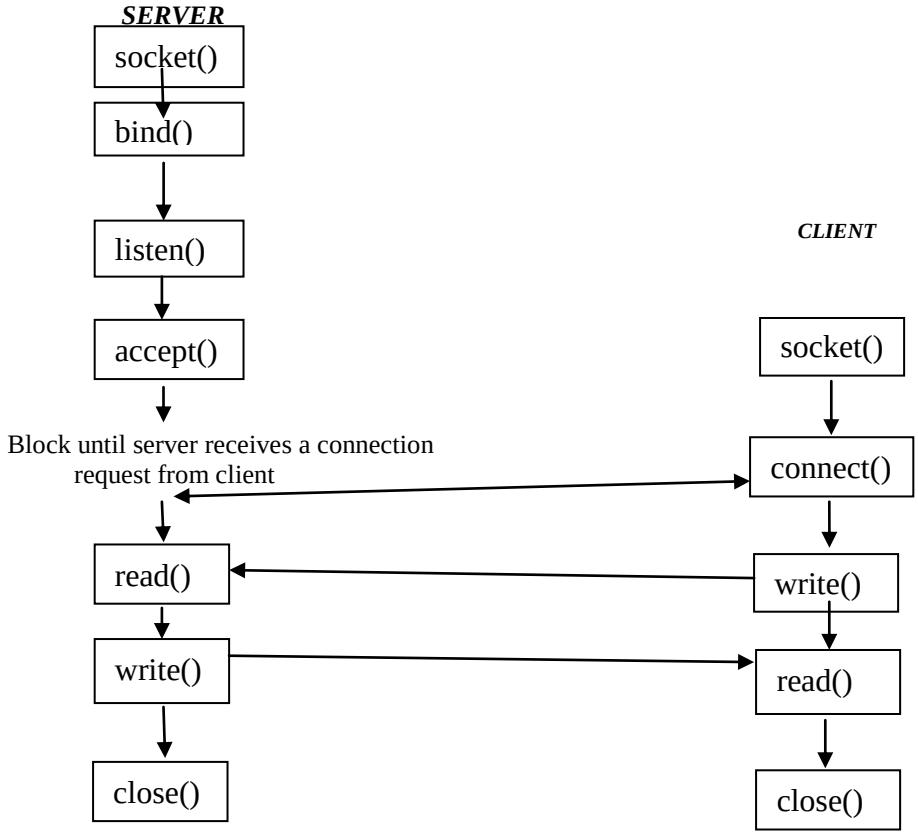


Fig: *TCP Client Server Model*

VII. ELEMENTARY UDP SOCKET

UDP is connection less protocol. It uses datagram for communication. Applications of UDP are: DNS (Domain Name System), NFS (Network File System), and SNMP (Simple Network Management Protocol). In UDP connection client does not establish a connection with the server so server does not accept a connection from client. It uses sendto() or recvfrom() function instead of write() and read() respectively.

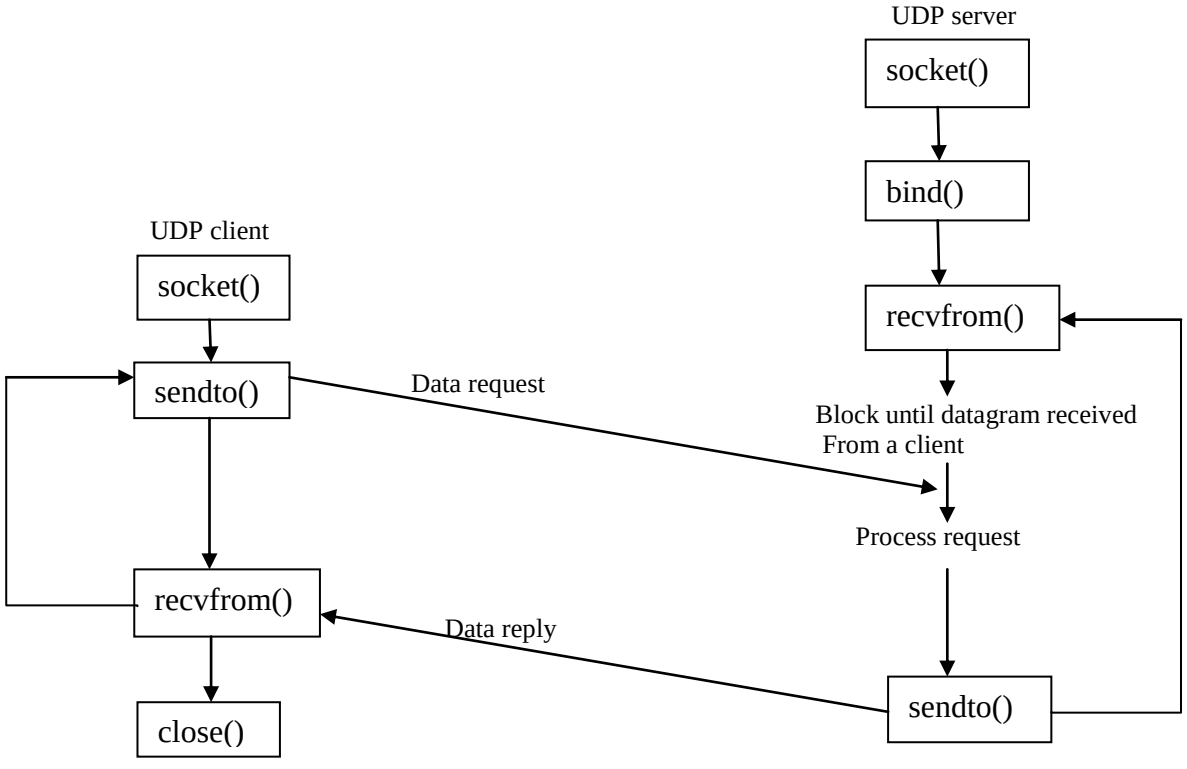


Fig: *UDP client server model.*

Functions used by the UDP clients:

1. socket() 2. sendto() 3. recvfrom() 4. close()

Functions used by the UDP server:

1. socket() 2. Bind() 3. Sendto() 4. Recvfrom() 5. Close()

VIII. CONCLUSION

In this paper we discussed about socket functions and its types. If we have a deep knowledge of sockets then we can create a connection between two processes in an efficient manner. This type of knowledge about all type of socket helps us to create a socket and implement it properly.

ACKNOWLEDGEMENT

We would like to give our sincere gratitude to our guide Mrs. Sonal Beniwal who guides us to pursue this topic and help us for completion of this topic.

REFERENCES

[1] W. Richard Stevens "UNIX Network Programming volume 1", 2nd edition Prentice Hall.
[2] Douglas E-comer "Internetworking with TCP/IP vol-I" PHI publication.
[3] Douglas E-comer, David L. Stevens "Internetworking with TCP/IP vol III", PHI publication.

