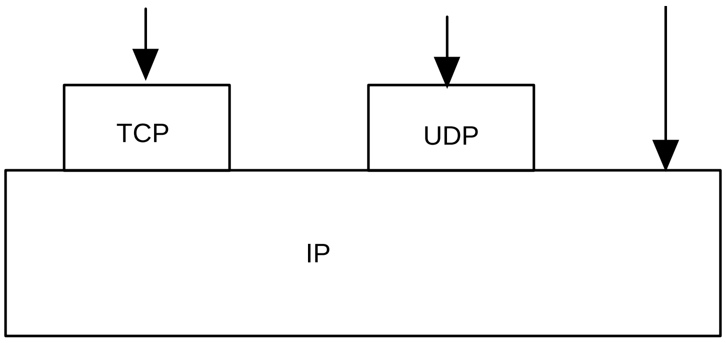
An Introduction to Socket Programming

(Handout for Programming Assignment #1)

TCP/IP implementations offer a programming interface that allows for full duplex communication called socket programming. The socket programming interface was first introduced with the BSD version V UNIX operating system. A number of improvements have been incorporated into the original interface over time. The socket interface provides three TCP/IP services. It can be used for TCP stream communication, UDP datagram communication and for raw datagram submission to the IP layer as shown in Figure, given below.



The TCP socket programming interface typically operates in a client-server model. A typical scenario for a TCP server is that there is a master process that spends most of its time listening for clients. When a client connects, it often is the case that the server creates a new process referred to as the child process to handle the actual work for the client. Typically there are multiple clients connecting to the same server at any given time. The server creates a new process for the client, passes the client over to the new process, and then goes back to listening. Sometimes clients arrive faster than the master process can handle and the server has to refuse connection when this happens. Usually the maximum number of clients that a server can accept by maintaining a queue. Clients that cant be served immediately are put on the queue and served in turn. If the queue is full and another client arrives then the request will not be accepted.

TCP Server Passive Open

For implementing a TCP/IP socket use the following header files using:

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

A system defined structure called sockaddr_in is used to hold the server address information. It is used in the following way:

struct sockaddr_in server address

A server' s passive open command is implemented by the series of calls:

socket_descriptor=socket(address_domain, communications_type, protocol)

The server identifies the type of communication TCP or UDP. The local system collects data required to create a connection and stores it in the Transmission Control Block (TCB). The connection name is a small integer called a socket descriptor or socketid.

Bind(): The server establishes the local IP address and port that it wants to use. Recall that a host may have multiple IP addresses. The server may specify one IP address, or else indicate that it is willing to accept connections arriving at any local IP address. The server may ask for a specific port, or else let the bind call obtain a free port that it can use. This function has the following arguments:

bind(socketid, &serveraddress, length of server address)

Listen(): The server sets the length of the client queue and accepts the socketid and the number of clients as inputs and indicates the creation of a passive socket and the creation of a queue of the required size.

Accept(): The server is ready to accept client connections if the queue is not empty. The accept call creates a new socket descriptor that will be used for this clients connection and will return this new descriptor. The function accept can be called as:

new socket descriptor= accept(socketid, client address structure, length of client address structure)

<u>TCP Client Active OPEN</u>

A client's open command is done using the socket command

sock = socket(type of data communication TCP/UDP, SOCK_STREAM, 0)

for UDP use **SOCK_DGRAM** instead of **SOCK_STREAM**.

The client then uses the connect() when it attempts to establish a connection with the server using the following function call:

connect(socketdescriptor, address_structure, length of address structure).

The host name that is entered by the user e.g. gate can be converted to an IP address by using the gethostname() function. The gethostbyname() function returns a pointer
to a system defined hostent(), i.e., host entry structure. This structure contains the server name and IP address. The following is the syntax fir the gethostbyname() function:

hp=gethostbyname(servername)

The bcopy function is used to copy the IP address (which is in the hp ->h_addr) into server address. The bcopy function has the following syntax:

bcopy(hp->h_addr,&serveraddress.sin_address, hp->h_length)

In addition the htonl() command or host-to-network-long function is used to translate a 32-bit integer stored in the local computer to the Internet format for a 32 bit IP address. This command has the following syntax:

<u>Comrnunication Comrnands</u>:

(a) TCP/IP (connection-oriented)

To send information use the following command:

send(socketid, buffer, buffer length, flags)

The flags parameter is set to zero. Buffer is a character array that holds the message to be sent. To receive a message use the following function:

recv(client socket descriptor, buffer, length of buffer, A)

(b) UDP (connectionless)

To send information use the following command:

sendto(socketdescriptor, buffer, length of buffer, flags, destination address structure, length of destination address structure)

To receive information use the following command:

reevfrom(socketdescripter,buffer, length of buffer, flags, address of destination, length of destination address structure)

The other system calls are the same as in TCP/IP.

the flags parameter is set to zero

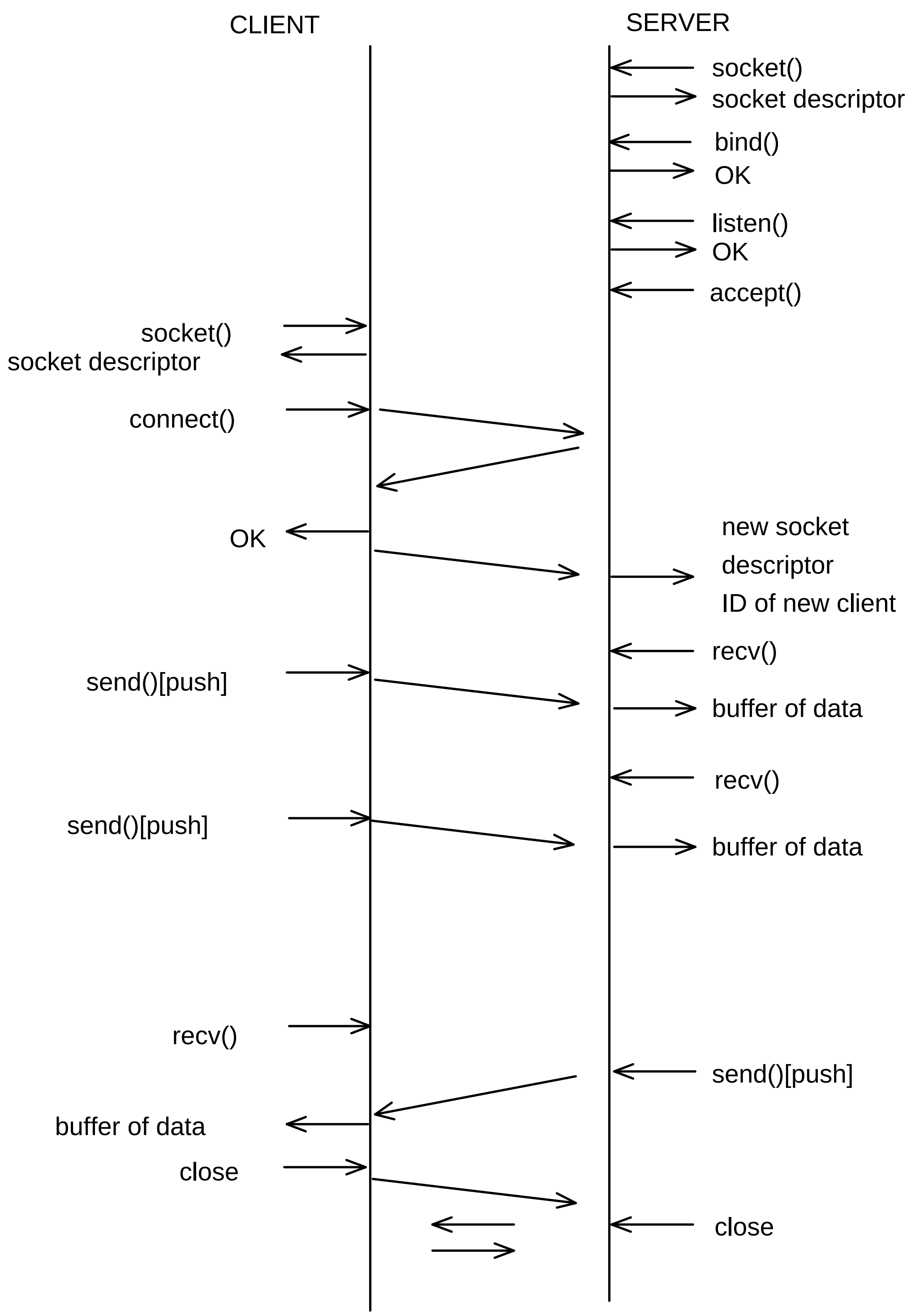
(c) Closing the Socket

The socket that was opened with the client using the accept call is closed using the following comrnand:

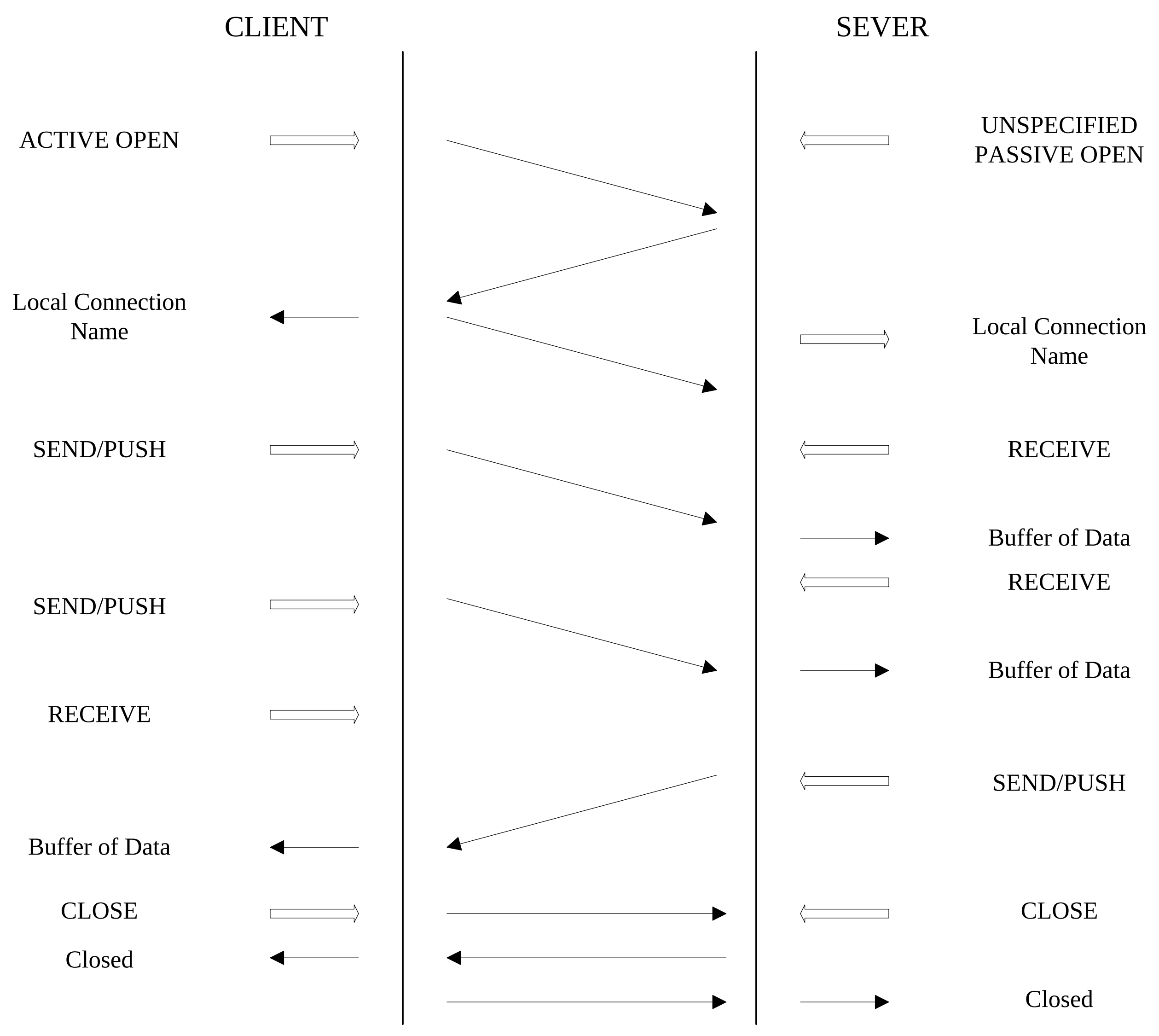
close(socketdescriptor)

This command is issued by the client to close the connection.

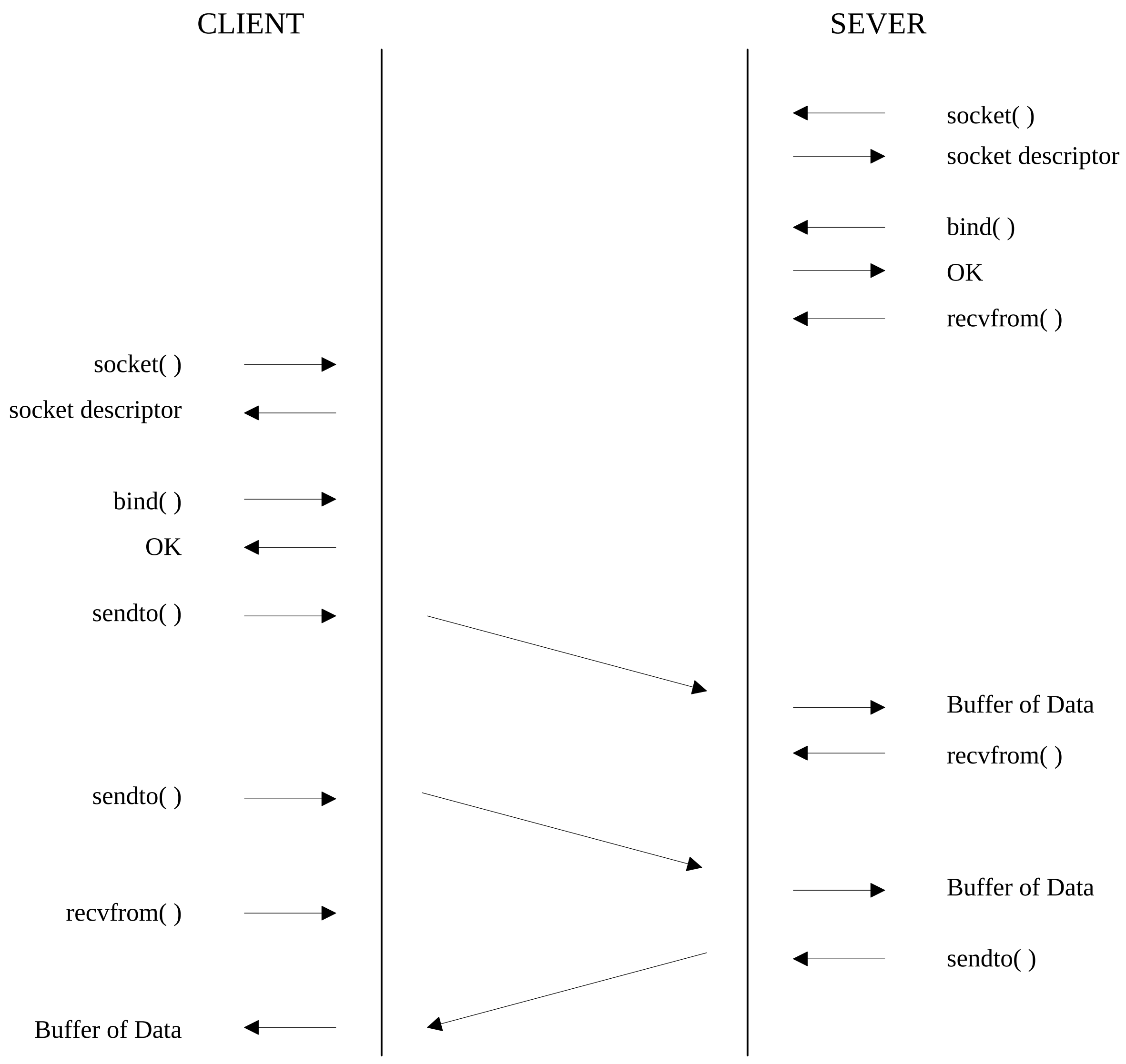
Refer to the enclosed figures describing the TCP/IP and UDP protocols.



TCP/IP System calls for Socket Connection



A typical sequence of TCP primitives



Typical UDP socket calls