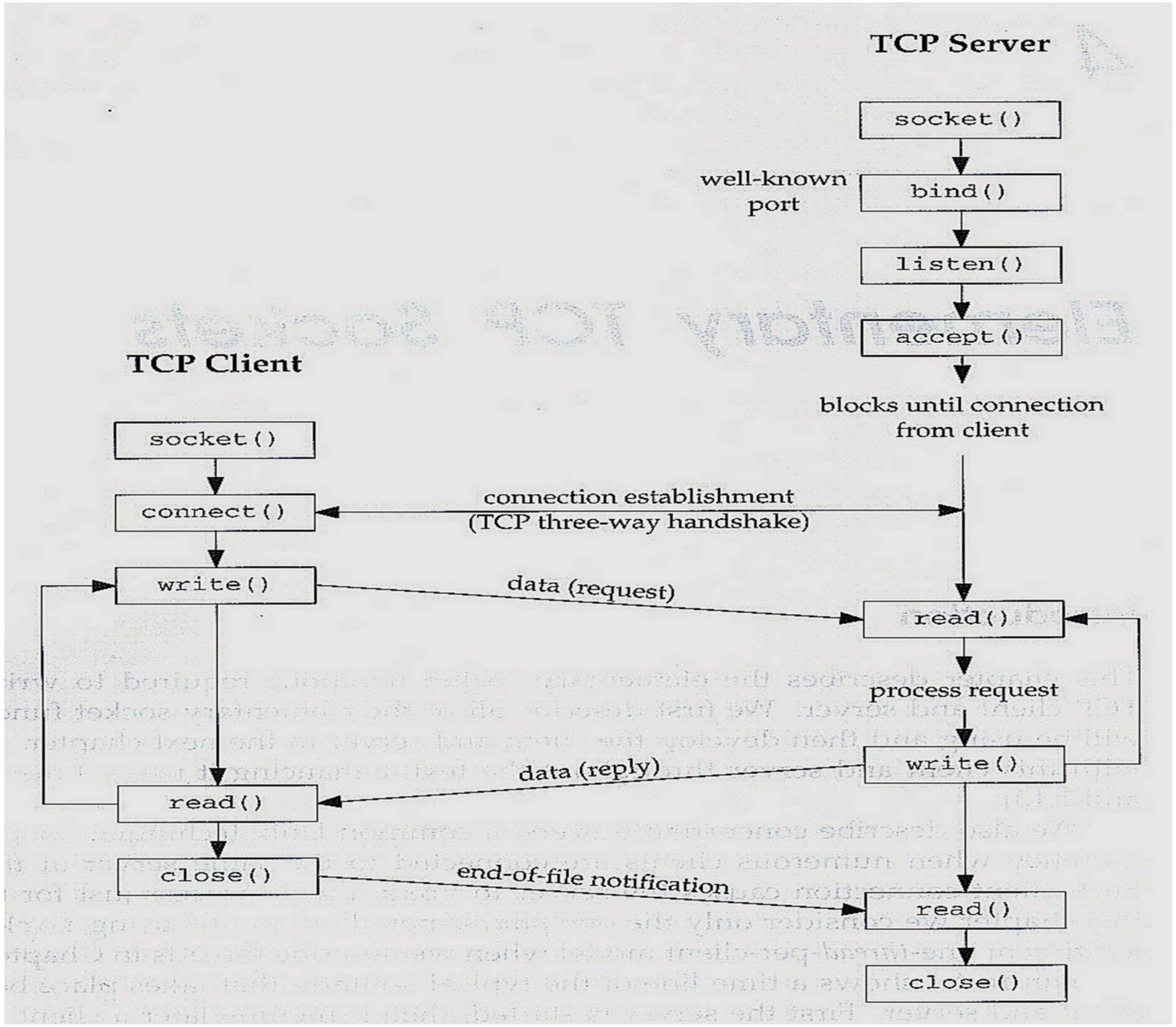
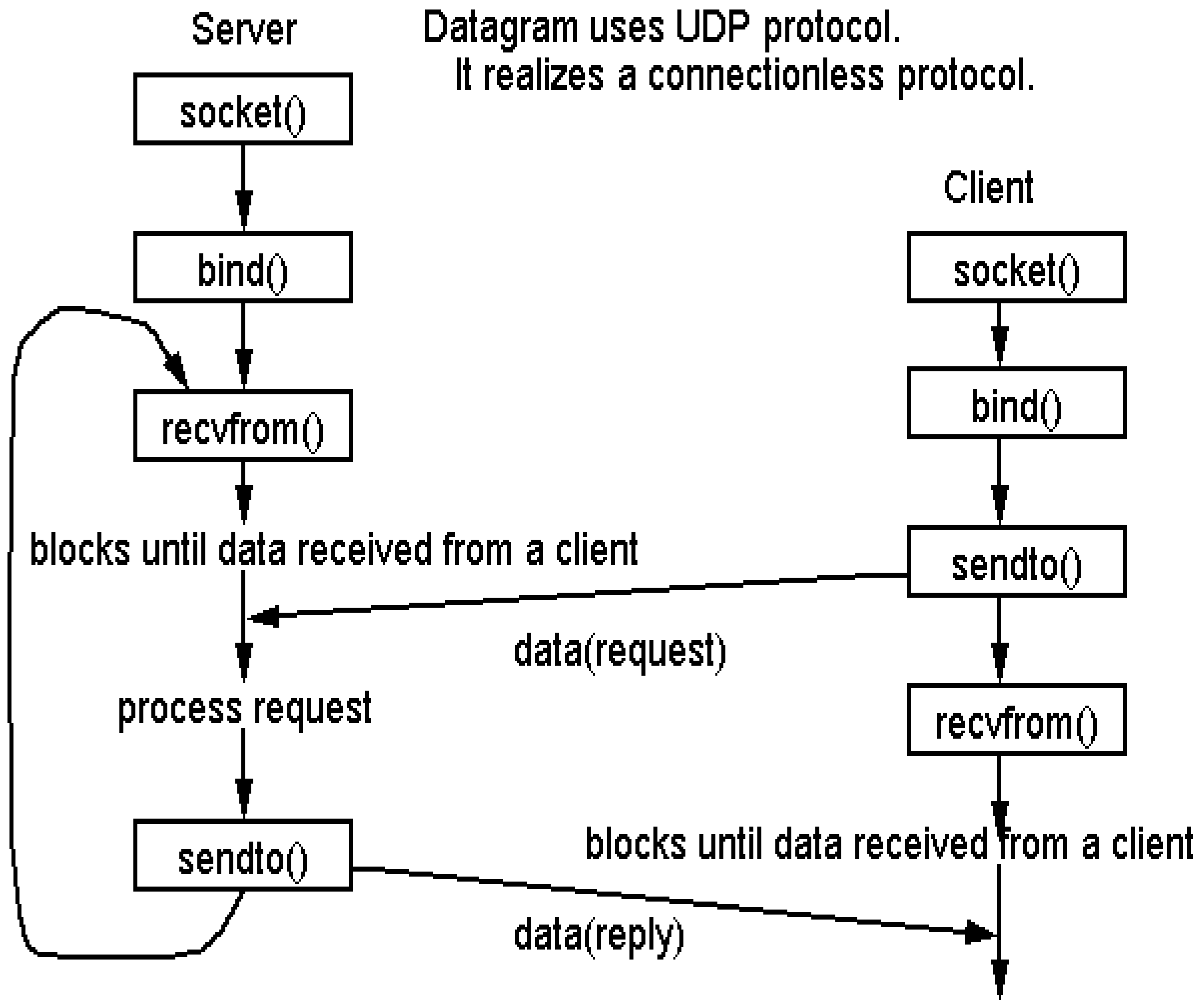
Socket Programming

TCP Sockets


TCP Server
socket()
well-known port
bind()
listen()
accept()
blocks until connection from client
TCP Client
socket()
connect()
connection establishment (TCP three-way handshake)
write()
data (request)
read()
process request
write()
data (reply)
read()
close()
end-of-file notification
read()
close()

UDP Sockets


Server
Datagram uses UDP protocol.
It realizes a connectionless protocol.
socket()
bind()
recvfrom()
blocks until data received from a client
process request
sendto()
Client
socket()
bind()
sendto()
data(request)
recvfrom()
blocks until data received from a client
data(reply)

Socket

- int sockfd=socket (domain, type, protocol);

where

- sockfd: socket descriptor, an integer (like a file-handle)
- domain: integer, communication domain
  - e.g., AF_INET (IPv4 protocol) or AF_UNIX
- type: communication type
  - SOCK_STREAM: reliable, 2-way, connection-based service
  - SOCK_DGRAM: unreliable, connectionless
- protocol: e.g., TCP or UDP
  - use IPPROTO_TCP or IPPROTO_UDP to send/receive TCP or UDP packets

Example : sockfd = socket(AF_INET, SOCK_STREAM, 0);

Bind

- associates an IP address and port for use by the socket
- bind(sockfd,(structsockaddr*)&serv_addr, sizeof(serv_addr))
  - sockfd: socket being used
  - serv_addr: address structure uses sockaddr_in
  - sizeof: the size (in bytes) of the serv_addr structure

- struct sockaddr_in

```
{
 sa_family_t    sin_family; /* address family: AF_INET */
 u_int16_t      sin_port;   /* port in network byte order */
 struct in_addr  sin_addr;  /* internet address */
};
```

- struct in_addr   /* Internet address */

```
  {
   u_int32_t     s_addr;     /* address in network byte order */
  };
```

Listen

- The listen system call allows the process to listen on the socket for connections.

- listen(sockfd, queue_length)
  - sockfd: socket being used
  - queue_length: number of active participants that can “wait” for a connection

  Example : listen(sockfd, 5);

accept

- Use the accept function to accept a connection request from a remote host
- The function returns a socket corresponding to the accepted connection
- newsockfd=accept(sockfd,(structsockaddr*) &cli_addr,&clength);
  - newsockfd : new socket used for data-transfer
  - sockfd: original socket being listened on (e.g., server)
  - cli_addr: address structure of the active participant (e.g., client)
    - The accept function updates/returns the sockaddr structure with the client's address information
  - clength: size (in bytes) of the client sockaddr structure
    - The accept function updates/returns this value

Connect

- The connect function is used by a client program to establish communication with a remote entity

- connect(sockfd,(structsockaddr*)&serv_addr,sizeof(serv_addr));
  - sockfd: client's socket to be used in connection
  - serv_addr: server's address structure
  - sizeof: size (in bytes) of the serv_addr structure

Sending / Receiving Data

```
#include <fcntl.h>
size_t read (int fd, void* buf, size_t cnt);
```

- fd: file descripter
- buf: buffer to read data from
- cnt: length of buffer

```
size_t write (int fd, void* buf, size_t cnt);
```

- fd: file descripter
- buf: buffer to write data to
- cnt: length of buffer

close

- When finished using a socket, the socket should be closed:
- close(sockfd);
  - sockfd: the file descriptor (socket being closed)

TCP Server

```
/*SERVER - Create a file called hello.txt in the current
    directory and pass that as the file name for server */
#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60

int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv)
{
int sockfd, newsockfd, clength;
struct sockaddr_in serv_addr,cli_addr;
char t[MAX], str[MAX];
strcpy(t,"exit");
sockfd=socket(AF_INET, SOCK_STREAM,0);
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=INADDR_ANY;
serv_addr.sin_port=htons(SERV_TCP_PORT);
printf("\nBinded");
bind(sockfd,(struct sockaddr*)&serv_addr,
    sizeof(serv_addr));
printf("\nListening...");
listen(sockfd, 5);
clength=sizeof(cli_addr);
newsockfd=accept(sockfd,(struct sockaddr*)
    &cli_addr,&clength);
close(sockfd);
read(newsockfd, &str, MAX);
printf("\nClient message\n File Name : %s\n", str);
f1=fopen(str, "r");
while(fgets(buff, 4096, f1)!=NULL) {
write(newsockfd, buff,MAX);
printf("\n"); }
fclose(f1);
printf("\nFileTransferred\n");
return 0;
}
```

TCP client

```
//CLIENT
#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60

int main(int arg,char*argv[])
{
int sockfd,n;
struct sockaddr_in serv_addr;
struct hostent*server;
char send[MAX],recvline[MAX],s[MAX],name[MAX];
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
serv_addr.sin_port=htons(SERV_TCP_PORT);
connect(sockfd,(struct
    sockaddr*)&serv_addr,sizeof(serv_addr));
printf("\nEnter the source file name : \n");
scanf("%s",send);
write(sockfd,send,MAX);
while((n=read(sockfd,recvline,MAX))!=0) {
printf("%s",recvline);
}
close(sockfd);
return 0;
}
```

END