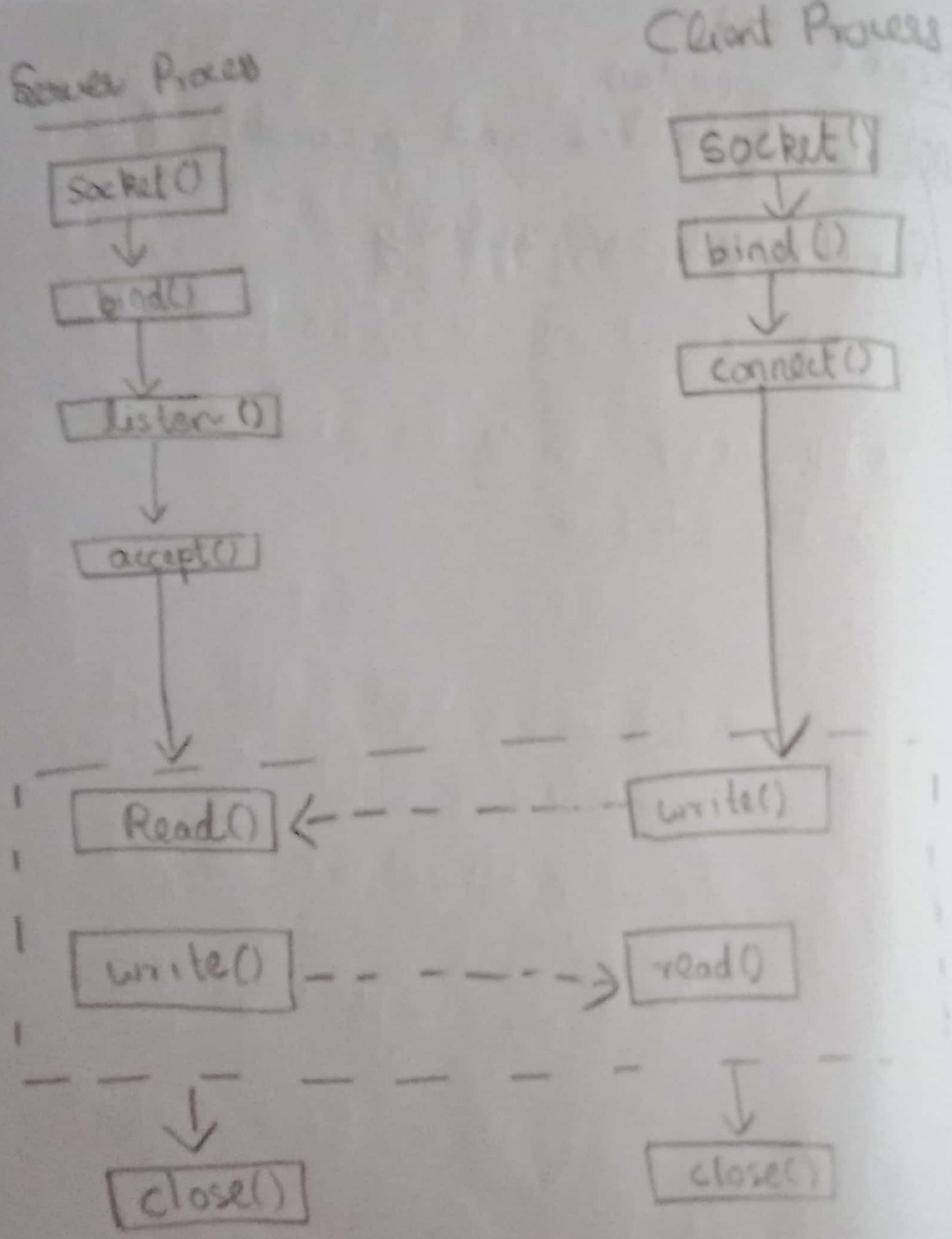

3. Client Server Communication Using Socket Programming

7/3/23

1. Transmissin Control Protocol

~~Algorithm~~ Server

a. header files required
#include <sys/socket.h>
<netitnet/in.h>
<string.h>

b. Variables

		type	for?
~~char~~	buff [100]	- Char -	
	k	- int	- store return value
~~socklen_t~~	sen	~~len~~ socklen_t -	
	sock_desc	int	
	temp_sock_desc	int	
	- sock_addr_in	struct	
		parameter (server, client)	

c. Socket() Creation

Parameters -
- Address Family → AF_INET or AF_UNIX or AF_LOCAL

Output -

Establishing Connection :

Enter a message : Hello

Output venam, enikk time kitteela
ezhuthaan

- TCP so SOCK_STREAM - Bidirectional, reliable, sequenced, unduplicated flow of data without record boundaries.

· return the value to ~~k~~, if ~~k~~ sock_desc == -1, error
sock_desc

else:

4.1 Force binding ~~with~~ socket familes, ports, addresses.

→ server.sin_family = AF_INET

→ server.sin_addr.s_addr = INADDR_ANY

(INADDR_ANY - This is an IP address which is used when we don't want to bind a socket to any specific IP. / if don't know actual IP address)

→ server.sin_port = 3003 ;

→ Repeat same with client

4.2 Create bind()

~~k = bind(sock_desc, (struct sockaddr*) & client, & len);~~

→ k = bind (sockdesc, (struct ~~*~~ sockaddr *)& serve,
sizeof (serve));

Memory pointer and ~~length of~~ & size allocated

→ k == -1, print error

5. Listen()
→ k = listen (sock_desc, 5);

5 is the "backlog" argument defines the max length to which queue of pending connections would grow, any value till 128 is fine.

→ store sizeof (client) in 'len' variable.

6) Accept() for client

temp_sock_desc = accept (sock_desc, struct sockaddr *)&client, &len);

7) recv()

k = recv (temp_sock_desc, buf, 100, 0);
// read incoming data on connections
→ execute program

8) close ~~ten~~ accept();

13

ivdem output venam

7/3/23

Client - Server communication using UDP

Algorithm

Server (UDP)

1. Create socket with
sockfd = socket (AF_INET, SOCK_DGRAM, 0)

2. Force bind, address, INADDR_ANY, port with
htons (atoi (argv[i]));

3. bind (sockfd, (struct sockaddr *) &server, & &server_len)

4. recvfrom (sockfd, buffer, 100, 0), struct sockaddr, &server_len).

Client

1. Socket creation.
2. Force binding.
3. Accept string.

4. ~~Send~~ data to send.
5. sendto (sockfd, buffer, sizeof(buffer), 0,
   (struct sockaddr *)& server, sizeof(server))

~~5~~. // Sends to server

6. ~~to~~ Stop