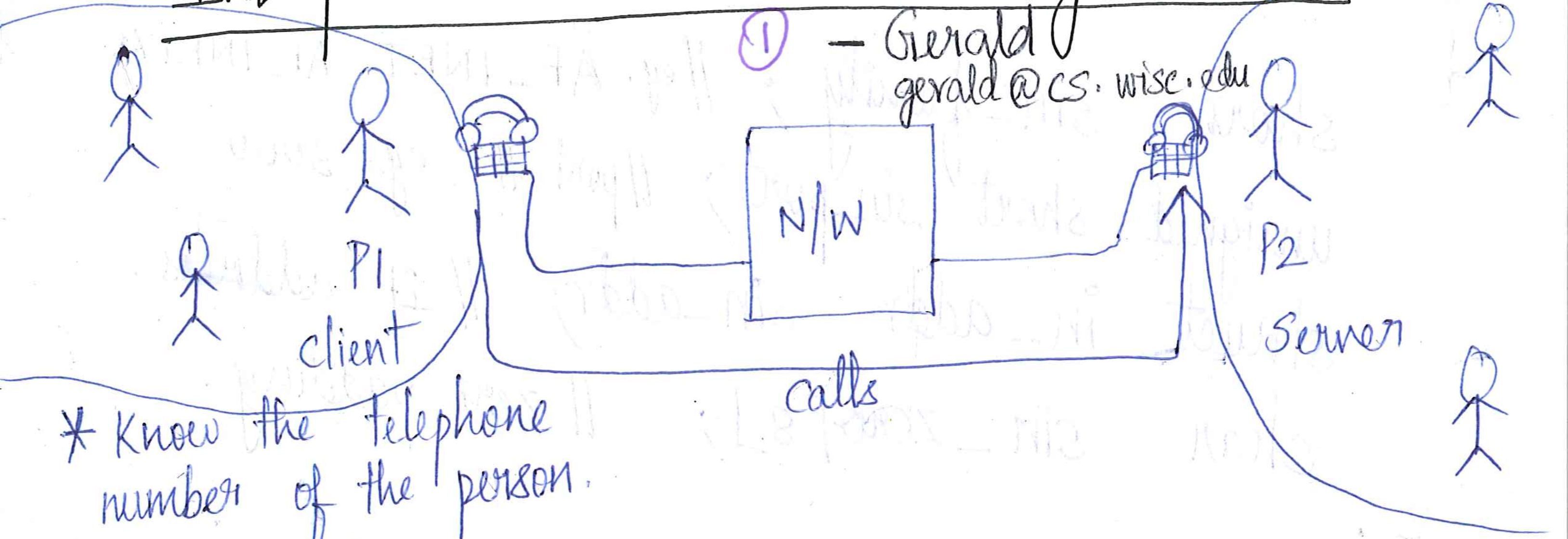
Interprocess Communication using Sockets

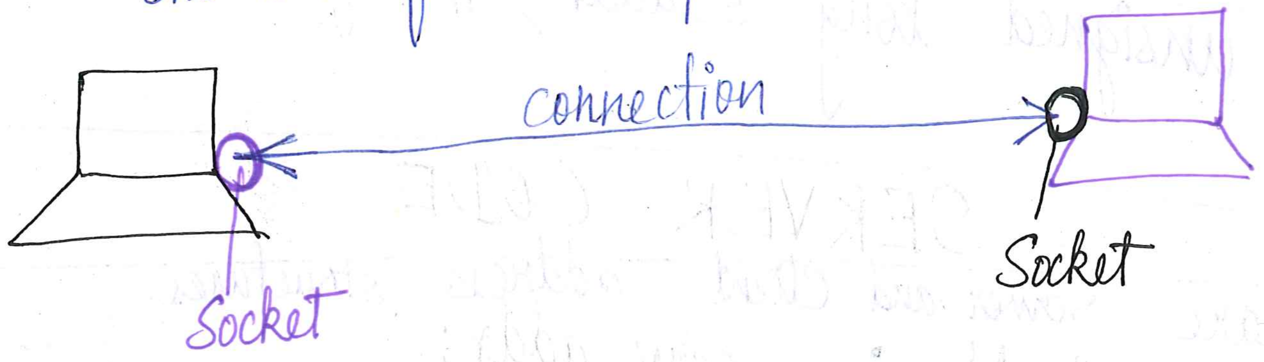
① – Gerald
gerald@cs.wisc.edu



* Know the telephone number of the person.
* name of the person

Socket

One end of an interprocess communication channel.



```
struct sockaddr
{
    unsigned short  sa_family;   //address family
    char            sa_data[14];

};
```

```
struct sockaddr_in ②
{
    short sin_family;          // eg. AF_INET, AF_INET6.
    unsigned short sin_port;   // port # eg. 3000
    struct in_addr sin_addr;   // IP address.
    char sin_zero[8];          // zero-padding.
};

struct in_addr
{
    unsigned long s_addr;  // 4-bytes IP address.
}
```

SERVER CODE

① Declare Server and Client address structures.

```
struct sockaddr_in serv_addr;
struct sockaddr_in cli_addr;
```

② Create a new socket.

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
                   ↓           ↓        ↓
                AF_UNIX    SOCK_DGRAM  protocol
                AF_INET6                ↙
                               0 = OS chooses for us
                               TCP/UDP.
```

③. Fill details in the serv_addr structure.

```
serv_addr. sin_family = AF_INET;
serv_addr. sin_addr. s_addr = INADDR_ANY;
serv_addr. sin_port = htons(portno);
```

$\Downarrow$

```
portno = atoi(argv[1]);
```

④ Bind the server address to the socket.

(||^{r} to assigning a phone number to a telephone (or) a SIM card to a mobile handset.)

```
bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr))
```

* check return value for error.

ret value < 0 $\Rightarrow$ error in bind.

⑤ Listen on the socket.

$\downarrow$ IP + port (assigned).

```
listen( sockfd, 5);
```

$\downarrow$ size of the backlog queue.

4

6. Accept a connection from a client and create a newsockfd to handle that connection.

```
newsockfd = accept ( sockfd,
                     (struct sockaddr *) &cli_addr,
                     &clilen );
```

↓ check for error.

clilen = sizeof(cli_addr);

7. Read data from client and store it in a buffer.

```
char buffer[256];
bzero(buffer,256);

n = read(newsockfd, buffer, 255);
// check for error (n < 0 ⇒ error in reading).

// print message.
```

8. Write some data for client to read.

```
n = write(newsockfd, "Got it!", 8);
// check for error in writing.
```

9. END the server.

CLIENT CODE

1. Declare serv_addr and server host entry

```
struct sockaddr_in serv_addr;
struct hostent *server; // has a h_addr member.
```

2. Create a new client socket

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

3. Get Server's host entry using server's name

$\downarrow$ phone number.

```
server = gethostbyname(argv[1]);
// if server == NULL ⇒ no such host.
```

4. Set the fields in the serv_addr structure.

```
bzero((char *) & serv_addr, sizeof(serv_addr));
serv_addr. sin_family = AF_INET;
```

P.T.O.

```
bcopy((char *) server -> h_addr,
      (char *) &serv_addr.sin_addr.s_addr,
      server -> h_length);
```

* using bcopy since server→h_addr is a char * (string).

```
void bcopy(char *s1, char *s2, int len);
```

source — dest. — no. of bytes.

```
serv_addr.sin_port = htons(portno);
```

(5) Establish a connection to the server.

```
connect(sockfd, &serv_addr, sizeof(serv_addr));
// check for error.
```

↓ IP + port no.

(6) Write some data to a buffer and send it to the server.

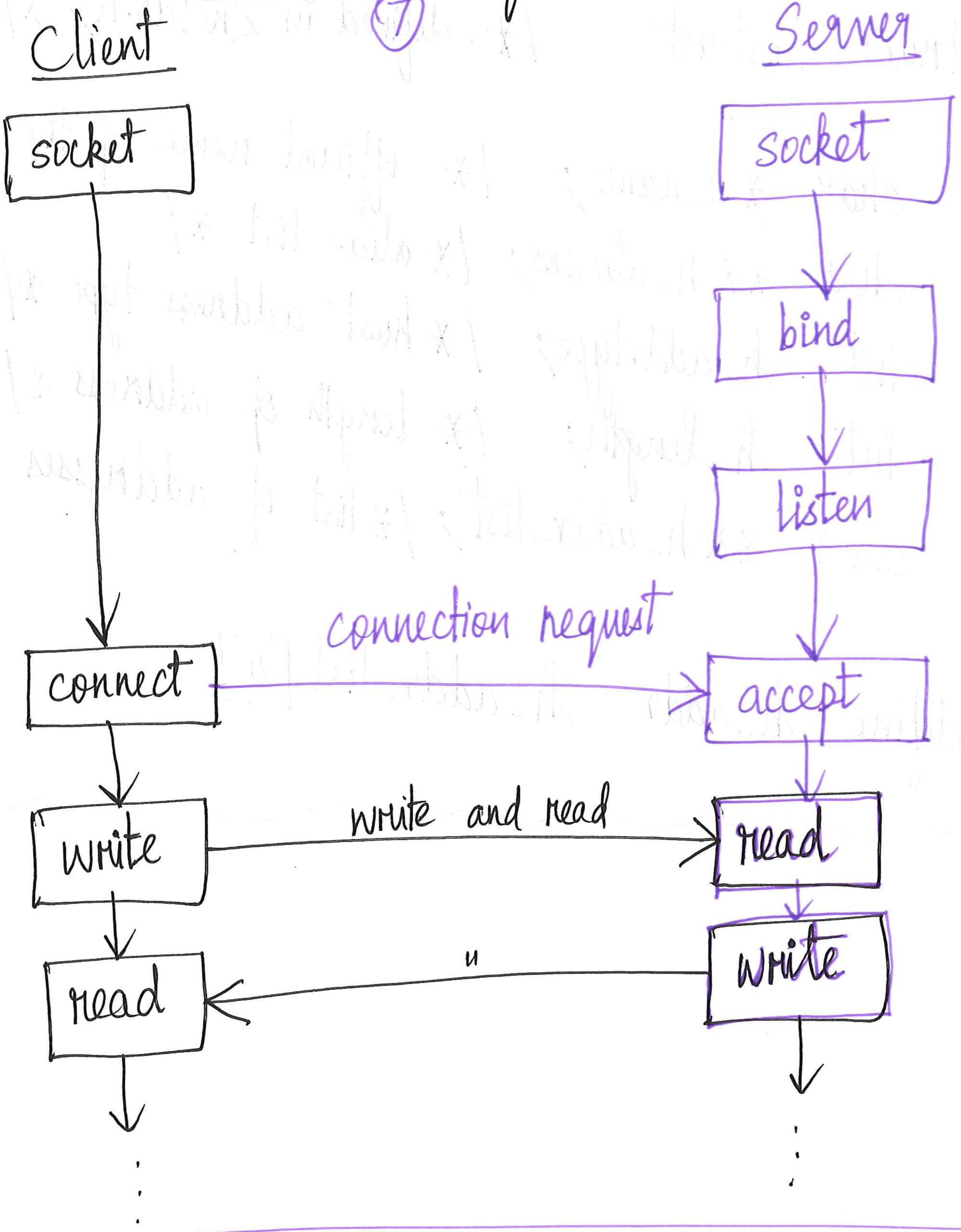
```
n = write(sockfd, buffer, strlen(buffer));
```

(7) Read some data sent by the server.

```
n = read(sockfd, read_buffer, 255);
```

(8) END of client.

Sockets Interface



CLOSE CONNECTION.

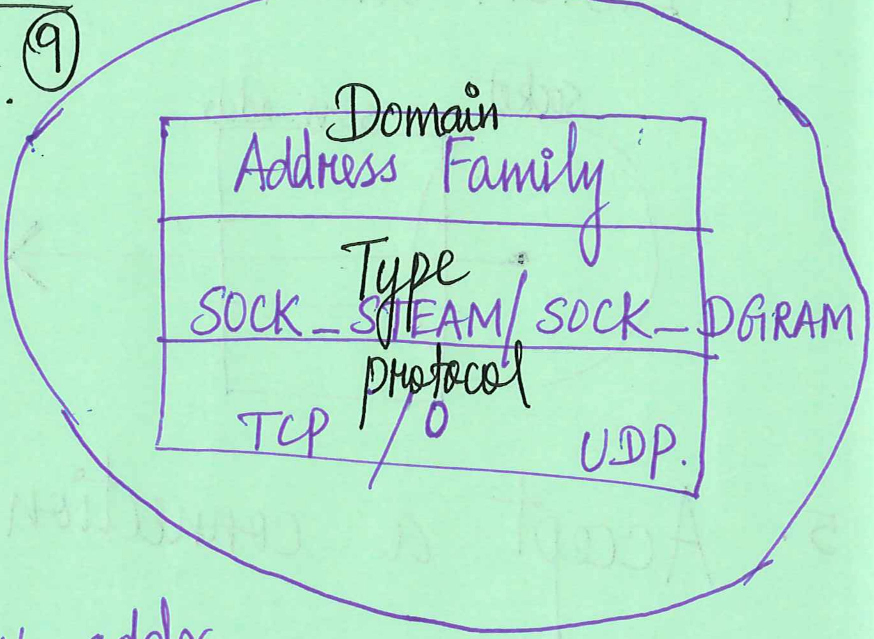
ADDITIONAL ⑧ NOTES

```
struct hostent      /* defined in <netdb.h> */
{
    char  *h_name;       /* official name of the host*/
    char  **h_aliases;   /* alias list */
    int   h_addrtype;    /* host address type */
    int   h_length;      /* length of address */
    char  **h_addr_list; /* list of addresses */
}
#define h_addr  h_addr_list[0]
```

SERVER

1. Create a socket.

sockfd = socket

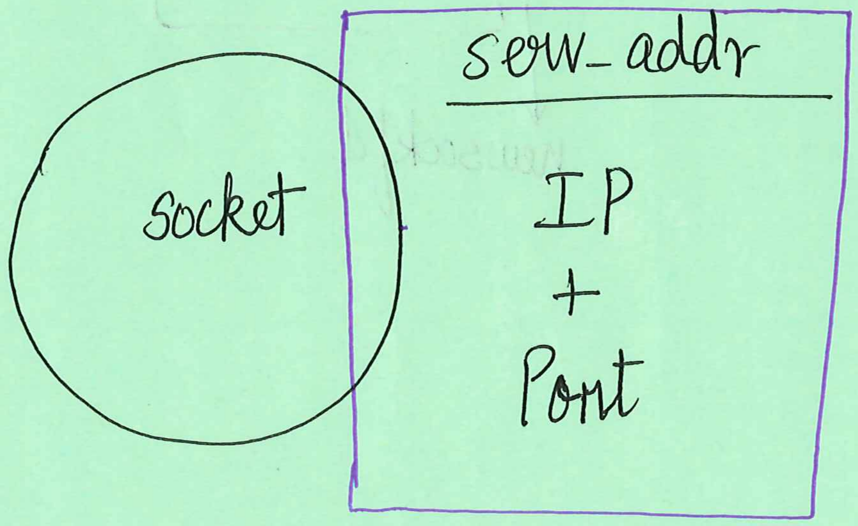


2. Fill serv_addr

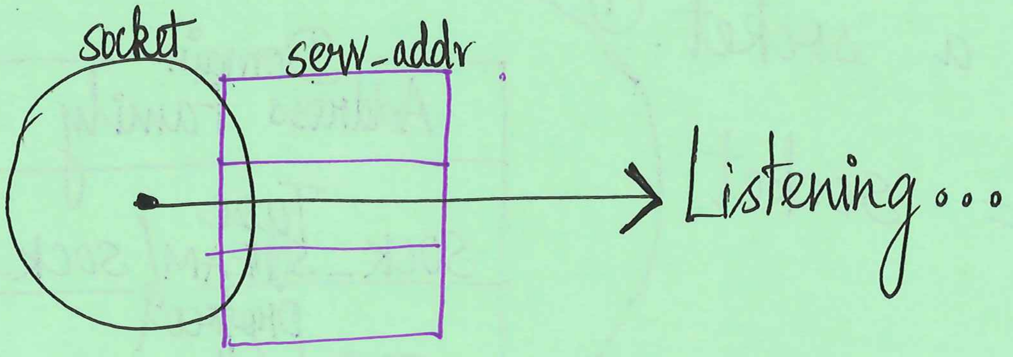
serv_addr

Family	AF_INET
IP address	INADDR_ANY
Port	3000

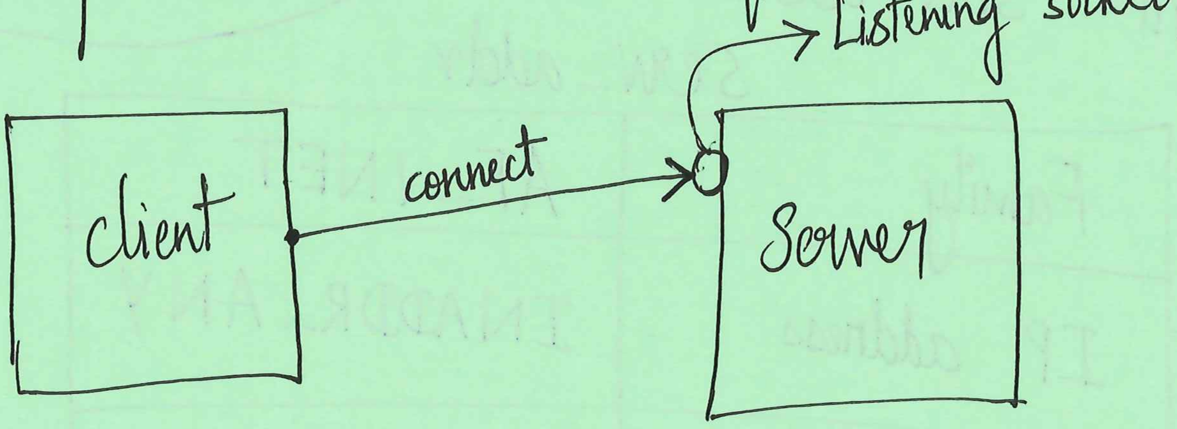
3. Bind serv_addr to the socket

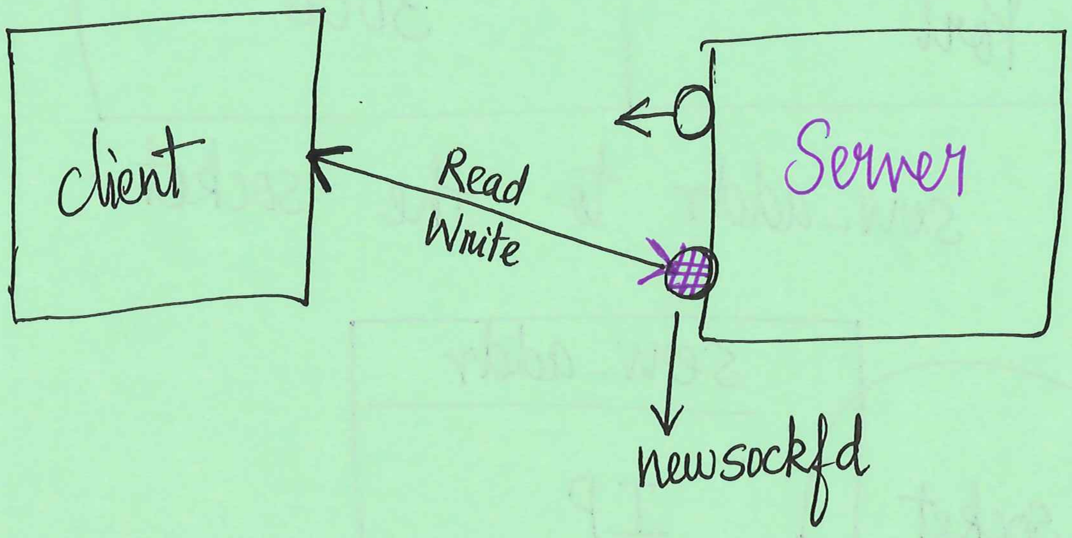


4. Listen on the socket



5. Accept a connection from a client.

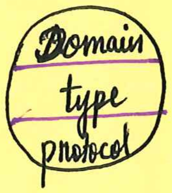




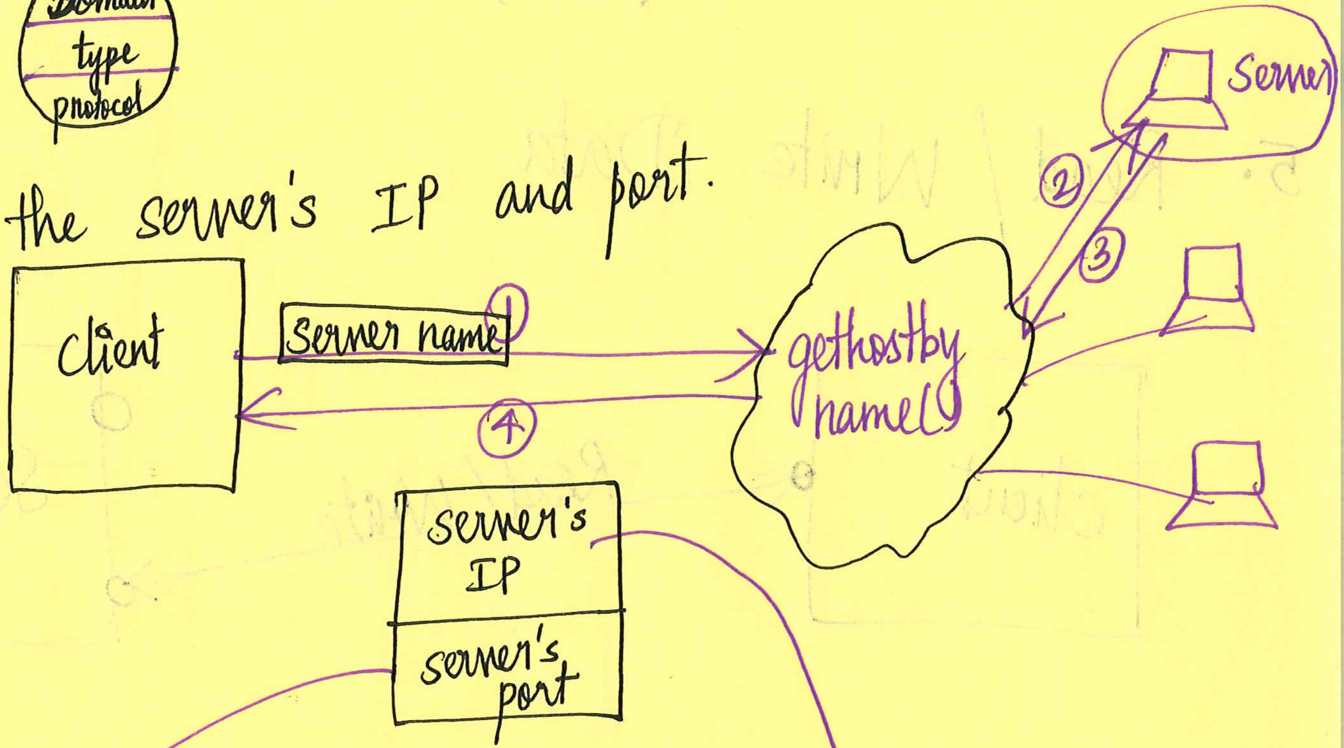
~~CLIBENT~~

CLIENT

1. Socket



2. Find the server's IP and port.

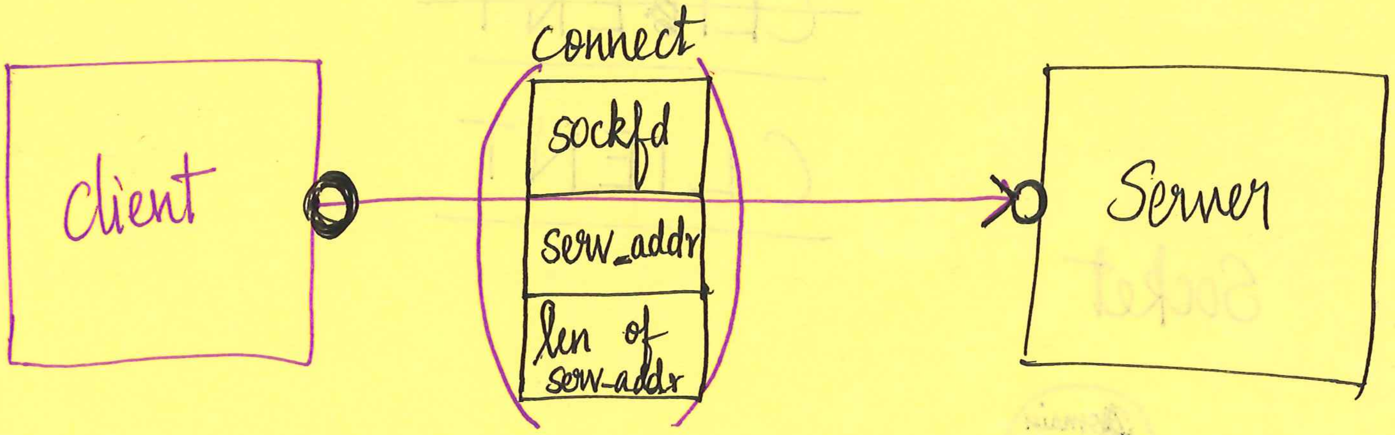


3. Set the fields in serv_addr.

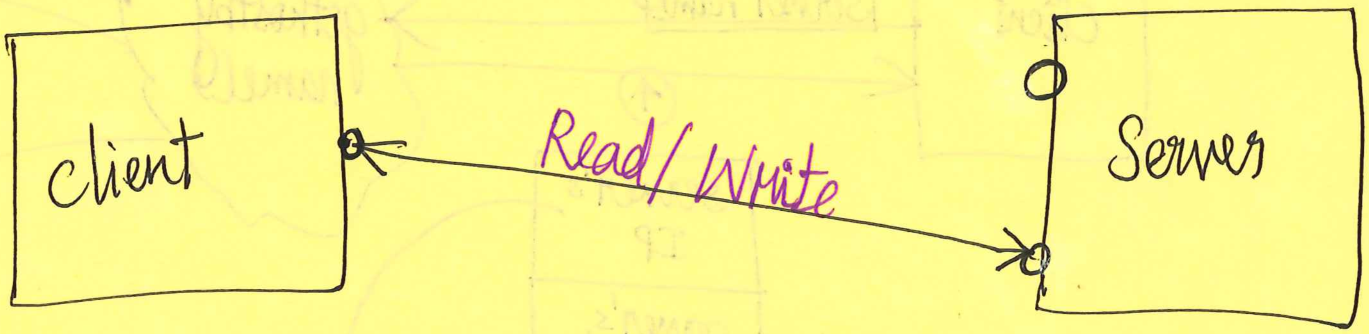
serv_addr

Family	AF_INET
IP address	
Port #	

4. Connect to the (12) server.



5. Read / Write Data


client
Read/Write
Server