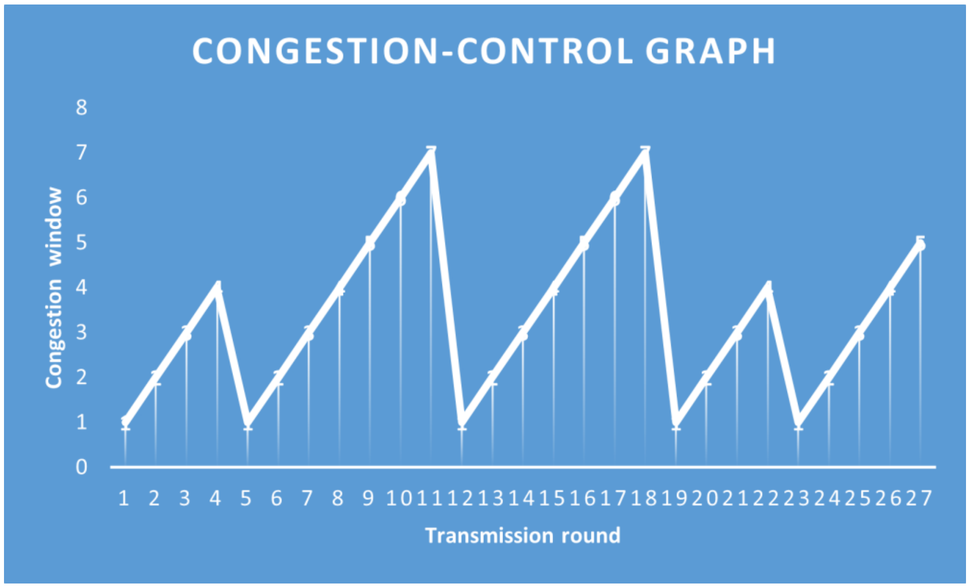
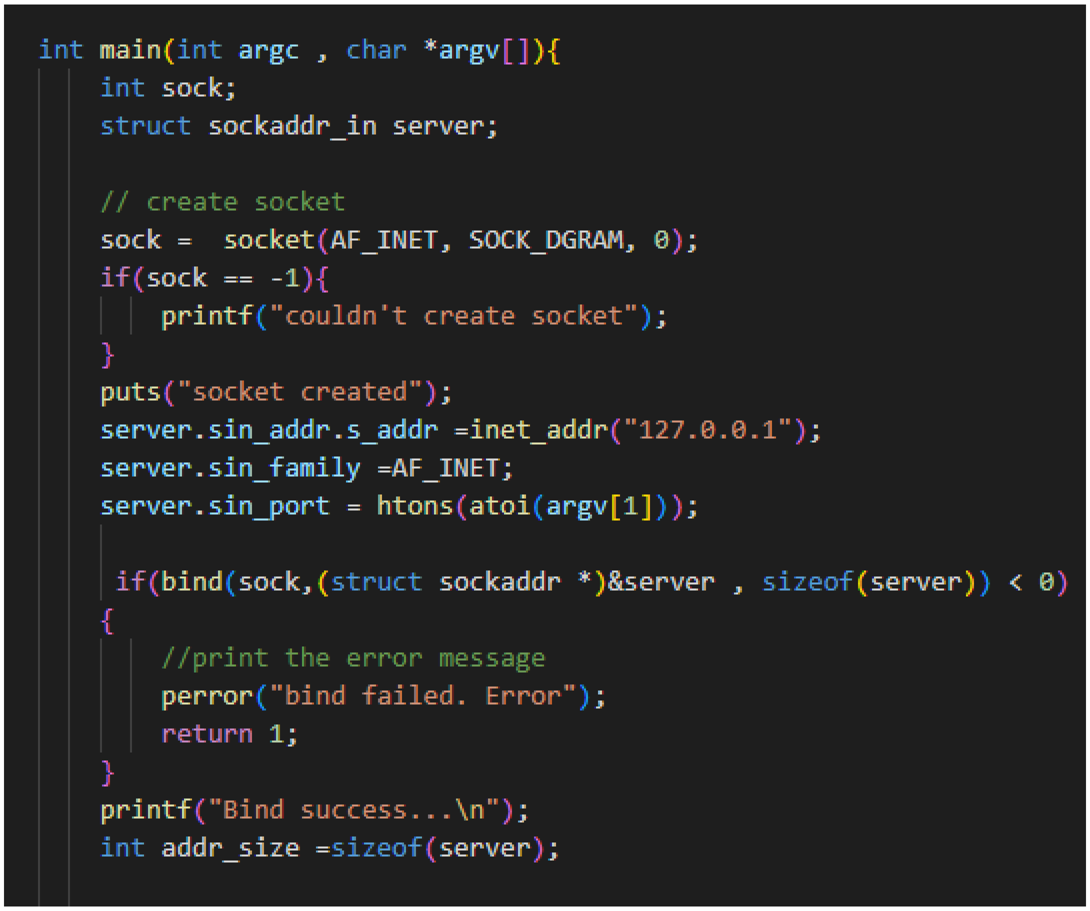
Computer Networks
Assignment 2

- Congestion control graph:



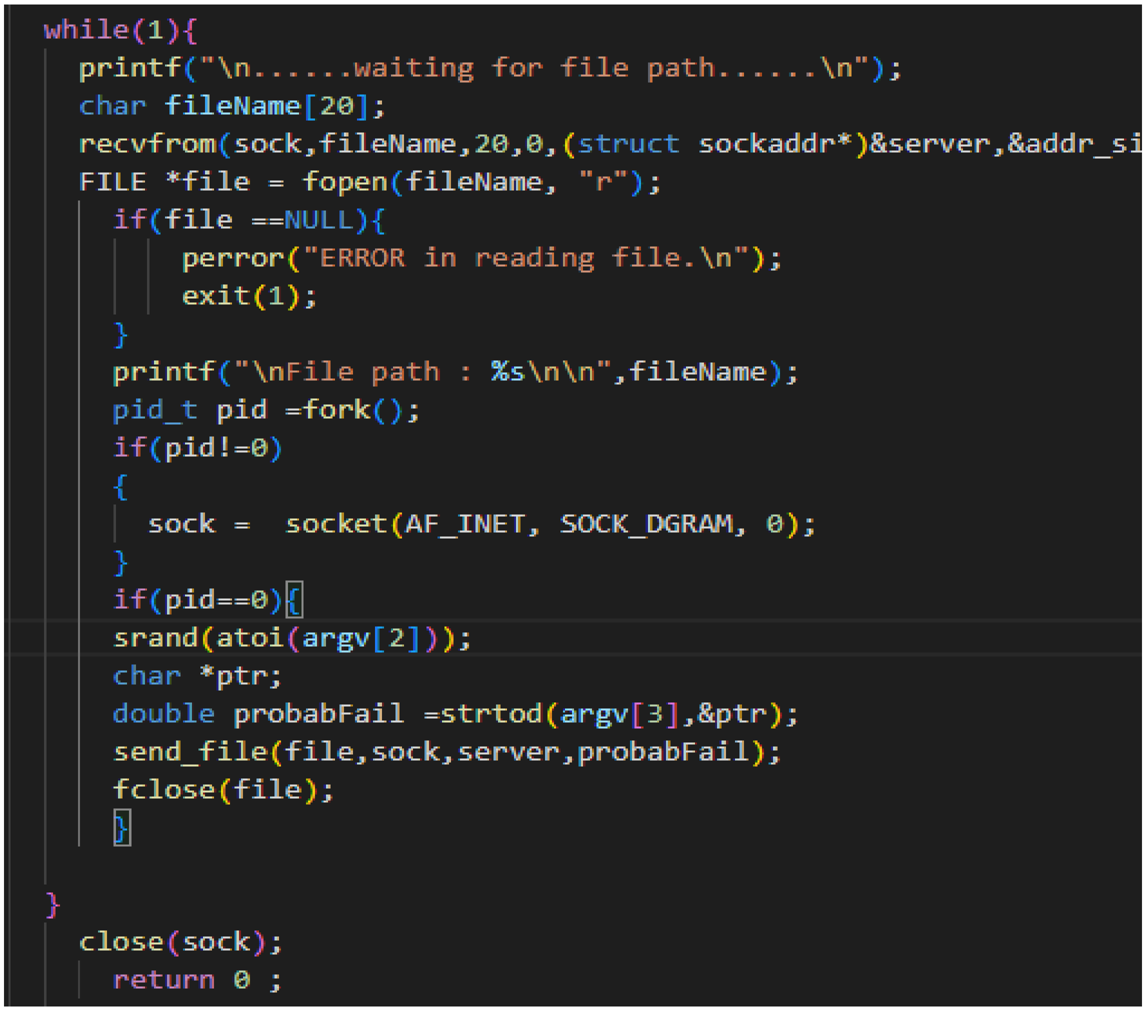
- Code documentation:
  - Server

Socket creation: at first, we initialize socket, then we specify it's attributes like family and address and port number. Then we try to bind it then start listening for incoming requests by incoming file name .



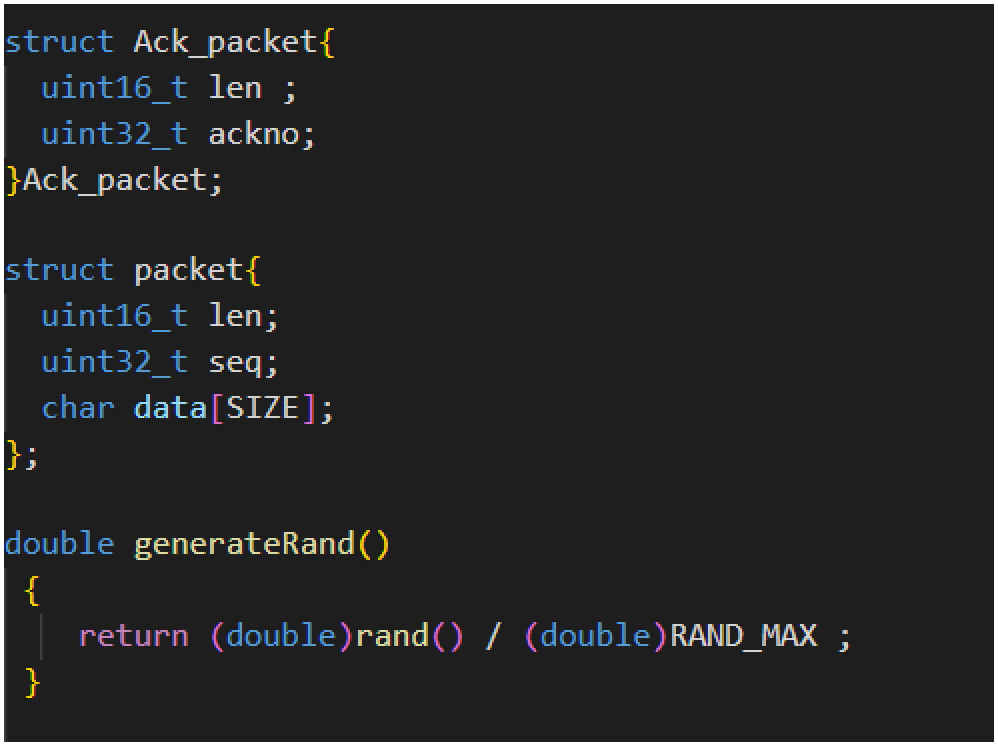
```
int main(int argc , char *argv[]){
    int sock;
    struct sockaddr_in server;

    // create socket
    sock =  socket(AF_INET, SOCK_DGRAM, 0);
    if(sock == -1){
        printf("couldn't create socket");
    }
    puts("socket created");
    server.sin_addr.s_addr =inet_addr("127.0.0.1");
    server.sin_family =AF_INET;
    server.sin_port = htons(atoi(argv[1]));

     if(bind(sock,(struct sockaddr *)&server , sizeof(server)) < 0)
    {
        //print the error message
        perror("bind failed. Error");
        return 1;
    }
    printf("Bind success...\n");
    int addr_size =sizeof(server);
```

process loop: this is the main logic of our code first we give each client socket id and start sending data in send_file function.



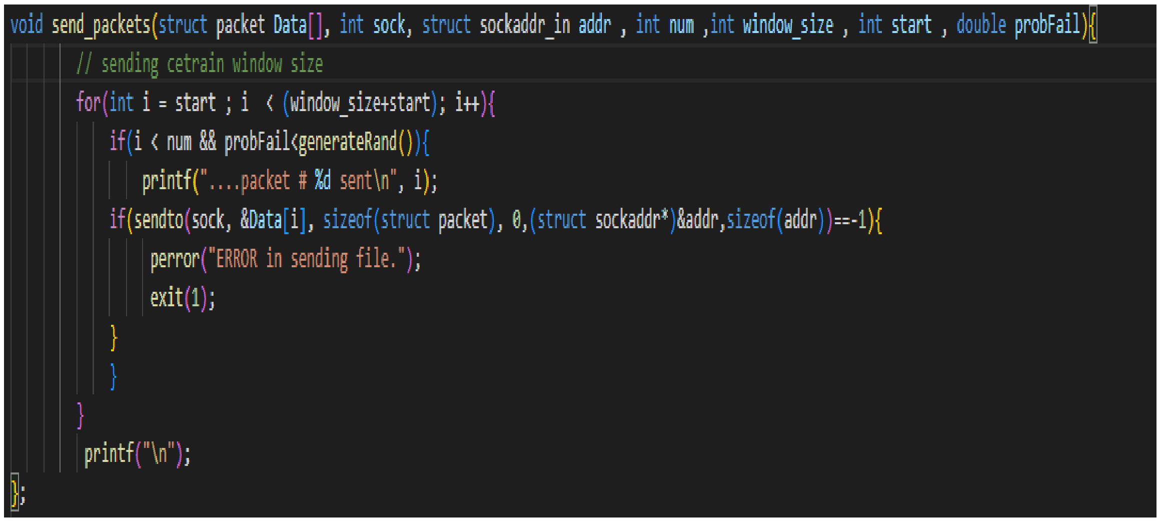
```
while(1){
  printf("\n......waiting for file path......\n");
  char fileName[20];
  recvfrom(sock,fileName,20,0,(struct sockaddr*)&server,&addr_si
  FILE *file = fopen(fileName, "r");
    if(file ==NULL){
        perror("ERROR in reading file.\n");
        exit(1);
    }
    printf("\nFile path : %s\n\n",fileName);
    pid_t pid =fork();
    if(pid!=0)
    {
      sock =  socket(AF_INET, SOCK_DGRAM, 0);
    }
    if(pid==0){
    srand(atoi(argv[2]));
    char *ptr;
    double probabFail =strtod(argv[3],&ptr);
    send_file(file,sock,server,probabFail);
    fclose(file);
    }

}
  close(sock);
    return 0 ;
```

Here we initialize our structs we have 2 structs on for acknowledge number it contain ack number and length of acknowledge struct, the second one is for sending the bucket itself and it contain the array of bytes data and sequence number of that packet (seq). generateRand() function used to generate random number between 0-1



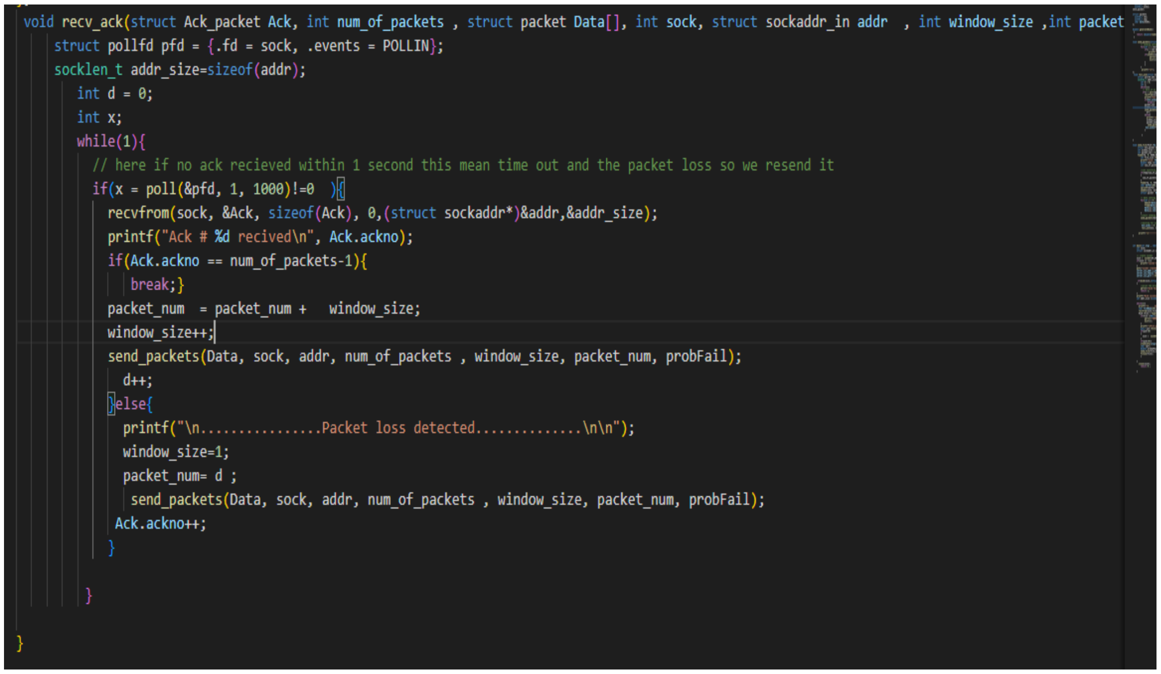
```c
struct Ack_packet{
  uint16_t len ;
  uint32_t ackno;
}Ack_packet;

struct packet{
  uint16_t len;
  uint32_t seq;
  char data[SIZE];
};

double generateRand()
 {
    return (double)rand() / (double)RAND_MAX ;
 }
```

Send_packets: first it loop through widow size and send packets it loss the packet which rand number (between (0-1)) is less then probability of fail .



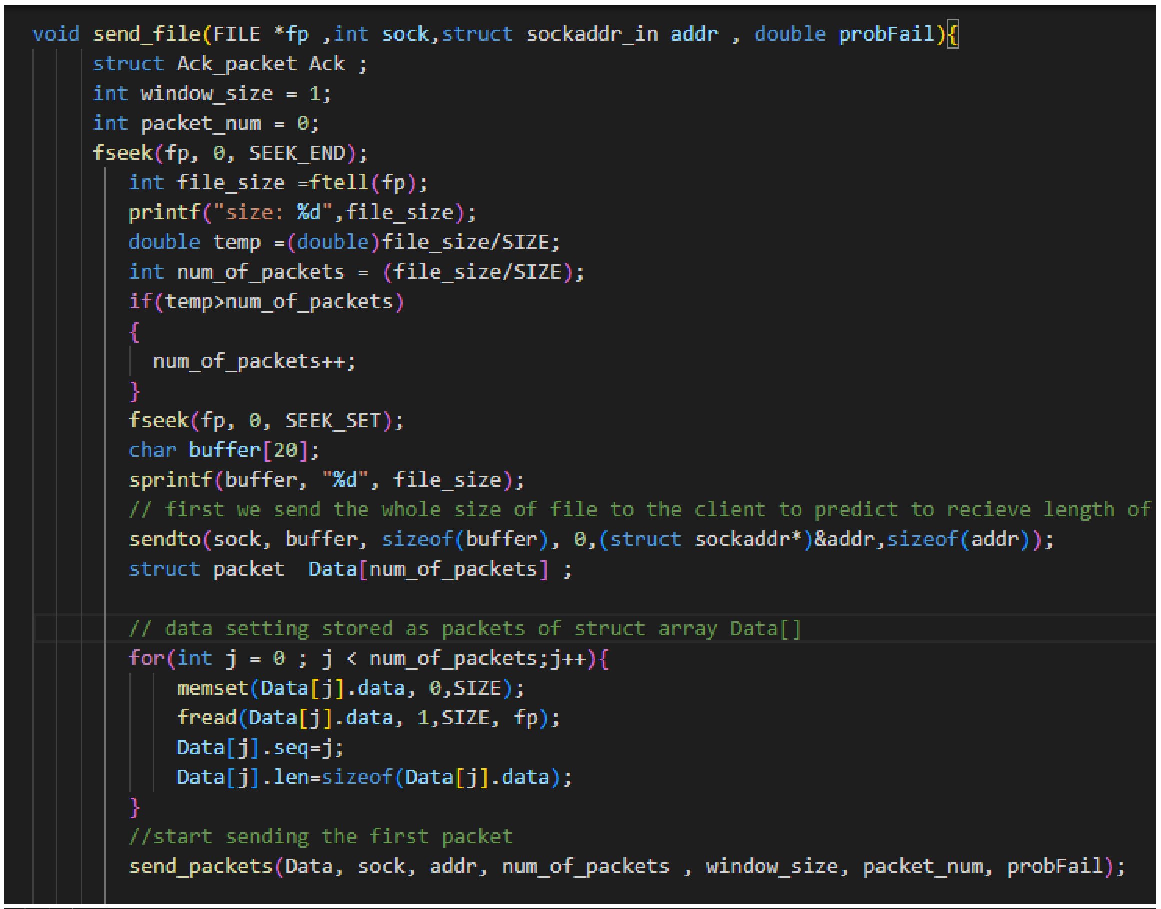
```c
void send_packets(struct packet Data[], int sock, struct sockaddr_in addr , int num ,int window_size , int start , double probFail){
      // sending cetrain window size
      for(int i = start ; i  < (window_size+start); i++){
          if(i < num && probFail<generateRand()){
              printf("....packet # %d sent\n", i);
          if(sendto(sock, &Data[i], sizeof(struct packet), 0,(struct sockaddr*)&addr,sizeof(addr))==-1){
               perror("ERROR in sending file.");
               exit(1);
          }
          }
      }
       printf("\n");
};
```

Recv_ack : in this function we wait for ack to receive from client side for 1 second id ack not received within this time this mean time out occurs so we resend the packet and decrease window size to start from one

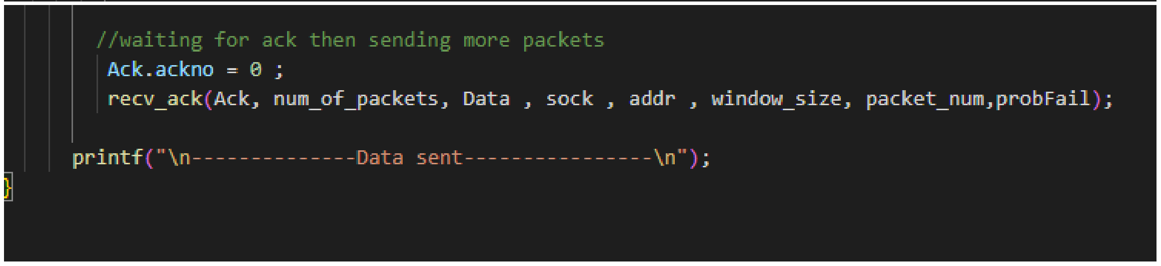


```
void recv_ack(struct Ack_packet Ack, int num_of_packets , struct packet Data[], int sock, struct sockaddr_in addr  , int window_size ,int packet
    struct pollfd pfd = {.fd = sock, .events = POLLIN};
    socklen_t addr_size=sizeof(addr);
      int d = 0;
      int x;
      while(1){
        // here if no ack recieved within 1 second this mean time out and the packet loss so we resend it
        if(x = poll(&pfd, 1, 1000)!=0  ){
          recvfrom(sock, &Ack, sizeof(Ack), 0,(struct sockaddr*)&addr,&addr_size);
          printf("Ack # %d recived\n", Ack.ackno);
          if(Ack.ackno == num_of_packets-1){
             break;}
          packet_num  = packet_num +   window_size;
          window_size++;
          send_packets(Data, sock, addr, num_of_packets , window_size, packet_num, probFail);
            d++;
          }else{
            printf("\n...............Packet loss detected.............\n\n");
            window_size=1;
            packet_num= d ;
             send_packets(Data, sock, addr, num_of_packets , window_size, packet_num, probFail);
           Ack.ackno++;
          }

      }

}
```

Send file: this is the main core of our program as we pass to this function file name to send from and socket address and the probability of fail he first send client the size of file and then stole whole file in array of packet struct each packet is of certain size around 512 byte then pass these packets to send_packets() function we explain early and then call function recv_ack() that wait for acknowledge for certain time we also explain this function early in recv_ack() .

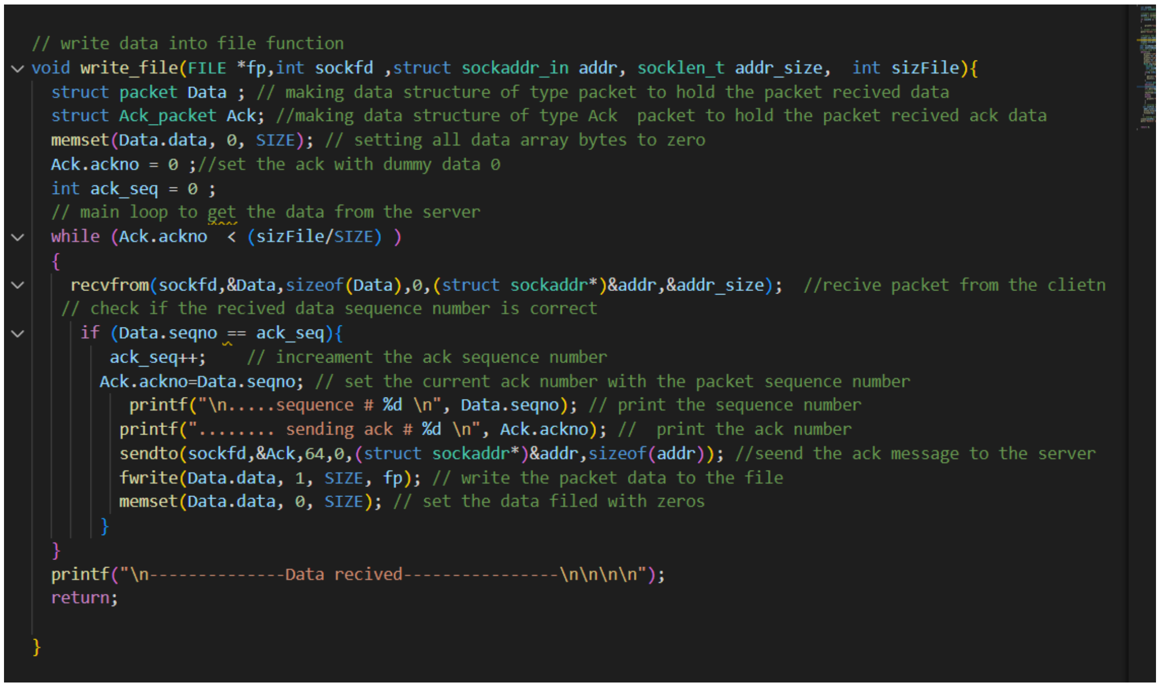


```
void send_file(FILE *fp ,int sock,struct sockaddr_in addr , double probFail){
        struct Ack_packet Ack ;
        int window_size = 1;
        int packet_num = 0;
        fseek(fp, 0, SEEK_END);
           int file_size =ftell(fp);
           printf("size: %d",file_size);
           double temp =(double)file_size/SIZE;
           int num_of_packets = (file_size/SIZE);
           if(temp>num_of_packets)
           {
             num_of_packets++;
           }
           fseek(fp, 0, SEEK_SET);
           char buffer[20];
           sprintf(buffer, "%d", file_size);
           // first we send the whole size of file to the client to predict to recieve length of
           sendto(sock, buffer, sizeof(buffer), 0,(struct sockaddr*)&addr,sizeof(addr));
           struct packet  Data[num_of_packets] ;

           // data setting stored as packets of struct array Data[]
           for(int j = 0 ; j < num_of_packets;j++){
               memset(Data[j].data, 0,SIZE);
               fread(Data[j].data, 1,SIZE, fp);
               Data[j].seq=j;
               Data[j].len=sizeof(Data[j].data);
           }
           //start sending the first packet
           send_packets(Data, sock, addr, num_of_packets , window_size, packet_num, probFail);
```

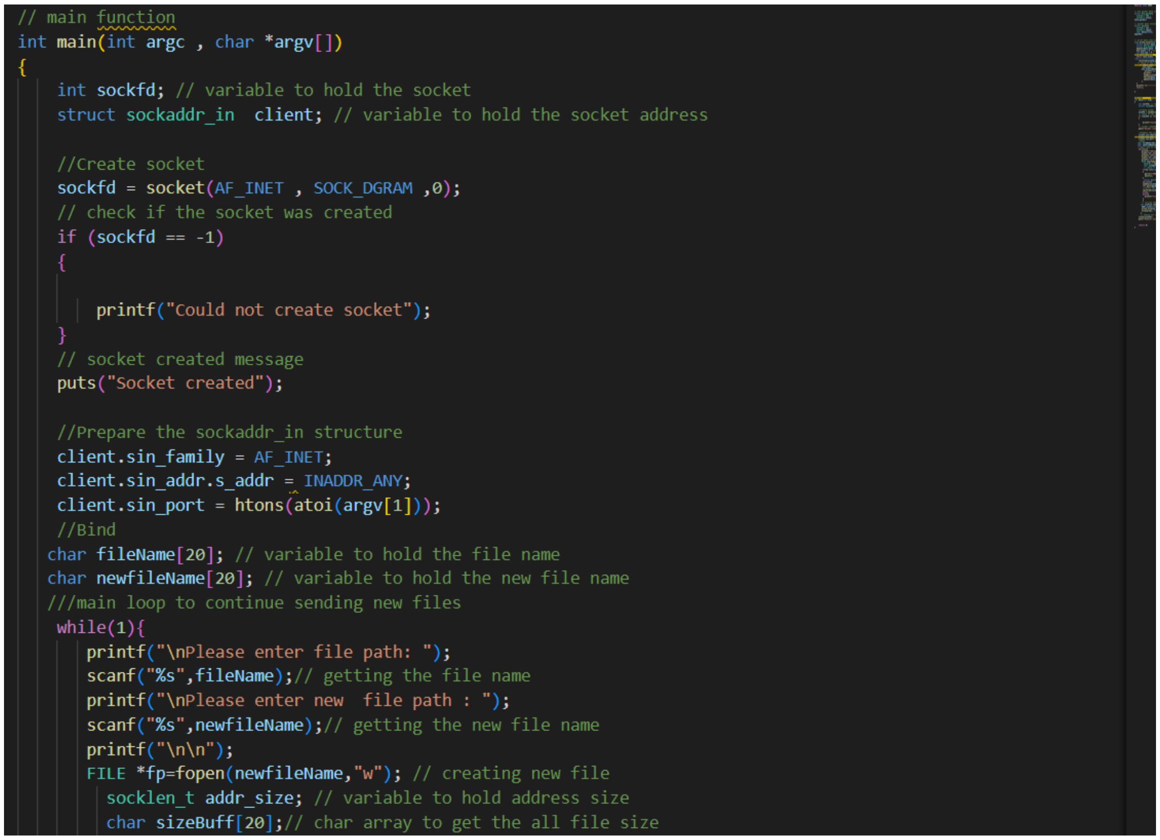


```
        //waiting for ack then sending more packets
         Ack.ackno = 0 ;
         recv_ack(Ack, num_of_packets, Data , sock , addr , window_size, packet_num,probFail);

      printf("\n--------------Data sent----------------\n");
}
```

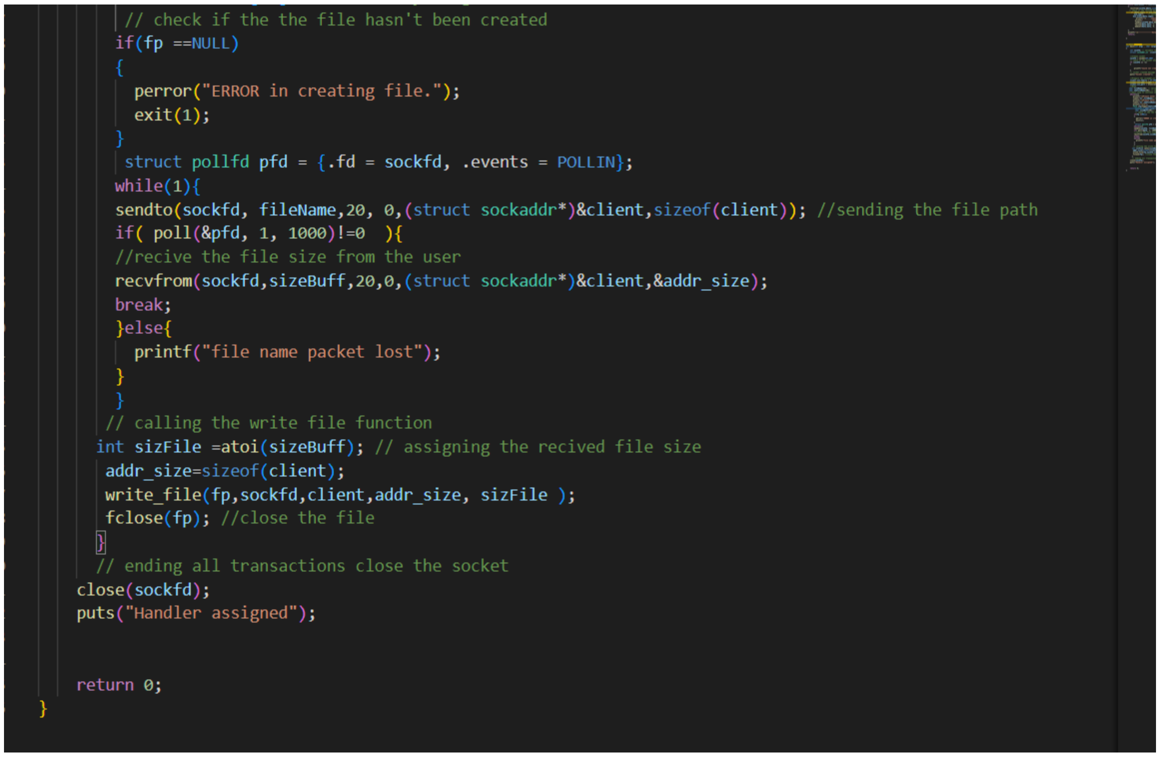
➢ Client



```c
// write data into file function
void write_file(FILE *fp,int sockfd ,struct sockaddr_in addr, socklen_t addr_size,  int sizFile){
  struct packet Data ; // making data structure of type packet to hold the packet recived data
  struct Ack_packet Ack; //making data structure of type Ack  packet to hold the packet recived ack data
  memset(Data.data, 0, SIZE); // setting all data array bytes to zero
  Ack.ackno = 0 ;//set the ack with dummy data 0
  int ack_seq = 0 ;
  // main loop to get the data from the server
  while (Ack.ackno  < (sizFile/SIZE) )
  {
    recvfrom(sockfd,&Data,sizeof(Data),0,(struct sockaddr*)&addr,&addr_size);  //recive packet from the clietn
   // check if the recived data sequence number is correct
     if (Data.seqno == ack_seq){
        ack_seq++;    // increament the ack sequence number
      Ack.ackno=Data.seqno; // set the current ack number with the packet sequence number
         printf("\n.....sequence # %d \n", Data.seqno); // print the sequence number
        printf("........ sending ack # %d \n", Ack.ackno); //  print the ack number
        sendto(sockfd,&Ack,64,0,(struct sockaddr*)&addr,sizeof(addr)); //seend the ack message to the server
        fwrite(Data.data, 1, SIZE, fp); // write the packet data to the file
        memset(Data.data, 0, SIZE); // set the data filed with zeros
     }
  }
  printf("\n-------------Data recived---------------\n\n\n\n");
  return;

}
```



```c
// main function
int main(int argc , char *argv[])
{
    int sockfd; // variable to hold the socket
    struct sockaddr_in  client; // variable to hold the socket address

    //Create socket
    sockfd = socket(AF_INET , SOCK_DGRAM ,0);
    // check if the socket was created
    if (sockfd == -1)
    {

        printf("Could not create socket");
    }
    // socket created message
    puts("Socket created");

    //Prepare the sockaddr_in structure
    client.sin_family = AF_INET;
    client.sin_addr.s_addr = INADDR_ANY;
    client.sin_port = htons(atoi(argv[1]));
    //Bind
   char fileName[20]; // variable to hold the file name
   char newfileName[20]; // variable to hold the new file name
   ///main loop to continue sending new files
    while(1){
       printf("\nPlease enter file path: ");
       scanf("%s",fileName);// getting the file name
       printf("\nPlease enter new  file path : ");
       scanf("%s",newfileName);// getting the new file name
       printf("\n\n");
       FILE *fp=fopen(newfileName,"w"); // creating new file
         socklen_t addr_size; // variable to hold address size
         char sizeBuff[20];// char array to get the all file size
```



```
         // check if the the file hasn't been created
        if(fp ==NULL)
        {
          perror("ERROR in creating file.");
          exit(1);
        }
         struct pollfd pfd = {.fd = sockfd, .events = POLLIN};
        while(1){
        sendto(sockfd, fileName,20, 0,(struct sockaddr*)&client,sizeof(client)); //sending the file path
        if( poll(&pfd, 1, 1000)!=0  ){
        //recive the file size from the user
        recvfrom(sockfd,sizeBuff,20,0,(struct sockaddr*)&client,&addr_size);
        break;
        }else{
          printf("file name packet lost");
        }
        }
       // calling the write file function
      int sizFile =atoi(sizeBuff); // assigning the recived file size
       addr_size=sizeof(client);
       write_file(fp,sockfd,client,addr_size, sizFile );
       fclose(fp); //close the file
      }
      // ending all transactions close the socket
    close(sockfd);
    puts("Handler assigned");


    return 0;
}
```