Selective Repeat Client Program: #include<time.h> #include<stdio.h> #include<stdlib.h> #include<sys/socket.h> #include<sys/types.h> #include<netinet/in.h> #include<sys/time.h> #include<sys/wait.h> #include<string.h> #include<unistd.h> #include<arpa/inet.h> int isfaulty(){ int d=rand()%4; return (d>2); } int main() { srand(time(0)); int c sock; c sock = socket(AF INET, SOCK STREAM, 0); struct sockaddr in client; memset(&client, 0, sizeof(client)); client.sin family = AF INET; client.sin_port = htons(9009); client.sin addr.s addr = inet addr("127.0.0.1"); if(connect(c_sock, (struct sockaddr*)&client, sizeof(client)) == -1) { printf("Connection failed"); return 0; printf("\n\tClient -with individual acknowledgement scheme\n\n"); char msg1[50]="akwnowledgemen tof-"; char msg3[50]="negative akwn-";

```
char msg2[50];
  char buff[100];
  int count=-1,flag=1;
  while(count<8){
     bzero(buff,sizeof(buff));
bzero(msg2,sizeof(msg2));
     if(count==7&&flag==1){
        printf("here\n");
        flag=0;
read(c_sock,buff,sizeof(buff));
        continue;
     int n = read(c sock, buff,
sizeof(buff));
     char
i=buff[strlen(buff)-1];
     printf("Message
received from server: %s
\n",buff);
     int isfault=isfaulty();
     printf("correption status:
%d \n",isfault);
     printf("Response/akwn
sent for message \n");
     if(isfault)
        strcpy(msg2,msg3);
        strcpy(msg2,msg1);
        count++;
     msg2[strlen(msg2)]=i;
     write(c_sock,msg2,
sizeof(msg2));
  }
  close(c sock);
  return 0;
}
```

```
Server Program:
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <fcntl.h>
void rsendd(int ch, int
c_sock)
char buff2[60];
bzero(buff2, sizeof(buff2));
strcpy(buff2, "reserver
message:");
buff2[strlen(buff2)] = (ch) +
'0';
buff2[strlen(buff2)] = '\0';
printf("Resending Message to
client: %s \n", buff2);
write(c sock, buff2,
sizeof(buff2));
usleep(1000);
}
int main()
{
int s_sock, c_sock;
s sock = socket(AF INET,
SOCK STREAM, 0);
struct sockaddr in server,
other;
memset(&server, 0,
sizeof(server));
memset(&other, 0,
sizeof(other));
server.sin_family = AF_INET;
server.sin port =
htons(9009);
server.sin addr.s addr =
INADDR ANY;
socklen t add;
if (bind(s sock, (struct
sockaddr *)&server,
sizeof(server)) == -1)
printf("Binding failed\n");
```

return 0;	rv1 = select(c_s
}	&set1, NULL, N
printf("\tServer Up\n Selective	&timeout1);
repeat scheme\n\n");	if (rv1 == -1)
listen(s_sock, 10);	perror("select e
add = sizeof(other);	else if (rv1 == 0
c_sock = accept(s_sock,	{
(struct sockaddr *)&other,	printf("Timeout
&add);	:%d \n", k);
time t t1, t2;	rsendd(k, c_so
char msg[50] = "server	goto qq;
message :";	}
char buff[50];	else
int flag = 0;	{
fd set set1, set2, set3;	read(c_sock, bu
struct timeval timeout1,	sizeof(buff));
timeout2, timeout3;	printf("Message
int rv1, rv2, rv3;	%s\n", buff);
int tot $= 0$;	if (buff[0] == 'n')
int ok[20];	{
memset(ok, 0, sizeof(ok));	printf(" corrupt i
while (tot < 9)	(msg %d) \n", b
{	- 1] - '0');
int toti = tot;	rsendd((buff[str
for (int $j = (0 + toti)$; $j < (3 + toti)$	'0'), c_sock);
toti); j++)	goto qq;
{	}
ເ bzero(buff, sizeof(buff));	else
char buff2[60];	tot++;
bzero(buff2, sizeof(buff2));	}
strcpy(buff2, "server	}
message :");	}
buff2[strlen(buff2)] = (j) + '0';	close(c_sock);
buff2[strlen(buff2)] = '\0';	close(s_sock);
printf("Message sent to client	return 0;
:%s \n", buff2);	}
write(c_sock, buff2,	ſ
sizeof(buff2));	Output:
usleep(1000);	student@ccf
, ,	670G:~/Docu
} for (int k = 0 + toti; k < (toti +	serversel.c
	student@ccf
3); k++)	670G:~/Docu
{ {	./a.out
qq: ED_7ED((((act1));	Server Up
FD_ZERO(&set1);	Selective :
FD_SET(c_sock, &set1);	scheme
timeout1.tv_sec = 2;	DOME
timeout1.tv_usec = 0;	

```
sock + 1,
NULL,
error ");
0)
for message
ck);
uff,
e from Client:
message awk
buff[strlen(buff)
rlen(buff) - 1] -
-Veriton-S2
ments$ gcc
-Veriton-S2
ments$
repeat
```

```
Message sent to client
:server message :0
Message sent to client
:server message :1
Message sent to client
:server message :2
Message from Client:
akwnowledgementof-0
Message from Client:
negative akwn-1
 corrupt message awk
(msg 1)
Resending Message to
client :reserver
message :1
Message from Client:
akwnowledgementof-2
Message from Client:
akwnowledgementof-1
Message sent to client
:server message :3
Message sent to client
:server message :4
Message sent to client
:server message :5
Message from Client:
negative akwn-3
 corrupt message awk
(msg 3)
Resending Message to
client :reserver
message :3
Message from Client:
negative akwn-4
 corrupt message awk
(msg 4)
Resending Message to
client :reserver
message :4
Message from Client:
akwnowledgementof-5
Message from Client:
akwnowledgementof-3
Message from Client:
akwnowledgementof-4
Message sent to client
:server message :6
Message sent to client
:server message :7
```

Message sent to client :server message :8 Message from Client: negative akwn-6 corrupt message awk (msq 6)Resending Message to client :reserver message :6 Message from Client: akwnowledgementof-7 Message from Client: akwnowledgementof-8 Timeout for message :8 Resending Message to client :reserver message :8 Message from Client: negative akwn-8 corrupt message awk (msq 8) Resending Message to client :reserver message :8 Message from Client: akwnowledgementof-8 student@ccf-Veriton-S2

student@ccf-Veriton-S2
670G:~/Documents\$ gcc
clientsel.c
student@ccf-Veriton-S2
670G:~/Documents\$
./a.out

Client -with individual acknowledgement scheme

Message received from server : server message :0 correption status : 0 Response/akwn sent for message Message received from server : server message :1 correption status : 1

Response/akwn sent for message Message received from server : server message :2 correption status : 0 Response/akwn sent for message Message received from server : reserver message :1 correption status : 0 Response/akwn sent for message Message received from server : server message :3 correption status : 1 Response/akwn sent for message Message received from server : server message :4 correption status : 1 Response/akwn sent for message Message received from server : server message :5 correption status : 0 Response/akwn sent for message Message received from server : reserver message :3 correption status : 0 Response/akwn sent for message Message received from server : reserver message :4

correption status : 0

Message received from

correption status : 1

server : server

message

message :6

Response/akwn sent for

Response/akwn sent for message Message received from server : server message :7 correption status : 0 Response/akwn sent for message Message received from server : server message :8 correption status : 0 Response/akwn sent for message here Message received from server : reserver message :8 correption status : 1 Response/akwn sent for message Message received from server : reserver message :8 correption status : 0 Response/akwn sent for message

frame_send.ack = 0;			
#include <stdio.h> #include <stdio.h> #include <stdio.h> #include <string.h> #include <arpa inet.h=""> #int main(int arg. char** #argy(a): #argy(b): #argy(b): #argy(b): #int sq_no; #int addr_size = #int main(int arg. char** #argy(b): #argy(b): #argy(b): #argy(b): #int sockfd; #int textory #int addr_size = #int main(int arg. char** #argy(b): #argy(b): #int sockfd; #int textory #argy(b): #int addr_size = #int main(int arg. char** #argy(b): #argy(b): #int addr_size = #int main(int arg. char** #argy(b): #argy(b): #int sockfd; #int textory #argy(b): #int addr_size = #int main(int arg. char** #argy(b): #int addr_size = #int part atoi(argy(1)): #int port = atoi(argy(1): #int addr_size = #int fame_id = 1 #int addr_size = #int fame_id = 1 #int addr_size = #int fame_id =</arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></string.h></stdio.h></stdio.h></stdio.h>	_	_	
#include <stdilb.h> #include <string,h> #include <string,h> #include <arpa inet.h=""> #include</arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></string,h></string,h></stdilb.h>	Client Program:	printf("Enter Data: ");	int frame_kind;
#include <string.h> #include string.h> #include string.h> #include string.h> #include staps/inet.h> #include saps/inet.h> #include s</string.h>		scanf("%s", buffer);	int sq_no;
#include <unistd.h> #include sarpa/inet.h> #i</unistd.h>	#include <stdlib.h></stdlib.h>	strcpy(frame_send.packet.dat	int ack;
#include <arpa inet.h=""> #include <arpa inet.h=""></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa>	#include <string.h></string.h>	a, buffer);	Packet packet;
#include <sys socket.h=""> typedef struct packet(</sys>	#include <unistd.h></unistd.h>	sendto(sockfd, &frame_send,	}Frame;
sizeof(serverAddr);	#include <arpa inet.h=""></arpa>	sizeof(Frame), 0, (struct	int main(int argc, char**
printf("[+]Frame Send\n"); printf("Usage: %s <port>", argy(0); exit(0); printf("Usage: %s <port>", argy(0); int port = atoi(argy(1]); int sockfd; struct sockaddr; serverAddr, newAddr; char buffer[1024]; socklen_t addr_size; int frame_id=0; rame_id=0; rame_id=10; printf("Lack Rote sockaddr]); serverAddr; char buffer[1024]; socklen_t addr_size; int frame_send; printf("Lack Rote sockaddr]); serverAddr; char buffer[1024]; socklen_t addr_size; int frame_send; socklen_t addr_size; int frame_send; socklen_t addr_size; int frame_send; socklen_t addr_size; int frame_send; serverAddr; serverAddr: serverAddr: serverAddr; se</port></port></port></port></port></port>	#include <sys socket.h=""></sys>	sockaddr*)&serverAddr,	argv){
Packet; typedef struct frame{	typedef struct packet{	sizeof(serverAddr));	if (argc != 2){
int addr_size = sizeof(serverAddr); int sq_no; int f_recv_size = sizeof(serverAddr); int sq_no; int f_recv_size = sizeof(serverAddr); int sq_no; int f_recv_size = sizeof(serverAddr); int sockfd; struct sockaddr_in serverAddr, saddr_size); if (argo = 2){	char data[1024];	printf("[+]Frame Send\n");	printf("Usage: %s <port>",</port>
sizeof(serverAddr); int f_recv_size = int port = atoi(argv[1]); int sockfd; struct sockaddr_in serverAddr, condition serverAddr, condi	}Packet;	}	argv[0]);
int sq_no; int ack; Packet packet; }Frame; int main(int argc, char 'argv[I]{	typedef struct frame{	int addr_size =	exit(0);
int ack;	int frame_kind;	sizeof(serverAddr);	}
Packet packet; Frame; Struct sockaddr InserverAddr, rame Struct sockaddr InserverAddr, sackadf Struct sockaddr InserverAddr, rame Struct sockaddr InserverAddr, sackadf InserverAddr, rame Struct sockaddr InserverAddr, sackadf InserverAddr, sackadf InserverAddr, rame InserverAdd	int sq_no;	int f_recv_size =	int port = atoi(argv[1]);
SFrame, sizeof(frame_recv), 0, (struct sockaddr, newAddr; char buffer[1024]; socklen_t addr_size); if (argc != 2){ printf("Usage: %s <port>", argv[0]); frame_recv.ack == frame_id=0; frame_recv.ack == frame_send; sockfd = socket(AF_INET, sockfd; struct sockaddr_in serverAddr; seceived\n"); ack_recv = 0; serverAddr.sin_port == htons(port); serverAddr.sin_addr.s_addr == inet_addr("127.0.0.1"); serverAddr.sin_port == htons(port); serverAddr.sin_port == htons(port); serverAddr.sin_family = AF_INET, sockfd = socket(AF_INET, sockfd = socket(AF_INET, sockfd = socket(AF_INET, serverAddr.sin_port == frame_id=0; frame_id=0;</port>	int ack;	recvfrom(sockfd,	int sockfd;
int main(int argc, char *argv[]){ if (argc!= 2){ if (frecv_size > 0 && frame_recv.qno== 0 && frame_recv;	Packet packet;	&frame_recv,	struct sockaddr_in
*argv[]){ if (argc != 2){ printf("Usage: %s <port>", argv[0]); exit(0); } frame_recv.ack == exit(0); } frame_id+1){ printf("I+]Ack Received\n"); ack_recv = 1; struct sockaddr_in serverAddr; char buffer[1024]; socklen_t addr_size; int frame_send; serverAddr; char buffer[1024]; socklen_t addr_size; int frame_send; serverAddr; char buffer[1024]; socklen_t addr_size; int frame_id=0; frame_id+1){ printf("I+]Ack Received\n"); ack_recv = 1; slese{ printf("I-]Ack Not serverAddr; char buffer[1024]; socklen_t addr_size; int frame_id=0; frame_id+1){ serverAddr: sin_family = AF_INET; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ serverAddr.sin_port = htons(port); serverAddr.sin_family = AF_INET; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ int frame_id=0; frame_id++; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ int frame_id=0; frame_id++; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ int frame_id=0; frame_id++; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ int frame_id=0; frame_id=0; frame_id++; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ int frame_id=0; frame_id=0; frame_id++; serverAddr.sin_addr.s_addr = int_addr("127.0.0.1"); while(1){ int frame_id=0; frame_send.s_nor frame_recv.g_no == 0 && frame_fecv; frame_recv.inet=0; sockld = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr); serverAddr.sin_addr.s_addr = intet_addr("127.0.0.1"); while(1){ int frame_id=0; int frame_send; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr); serverAddr.sin_addr.s_addr = intet_addr("127.0.0.1"); while(1){ inter_id=0} inter=0 & & int</port>	}Frame;	sizeof(frame_recv), 0 ,(struct	serverAddr, newAddr;
if (argc != 2){ if (argc != 2){ if (f recv_size > 0 && frame_recv.sq_no == 0 && frame_frame_send; sq_no = frame_recv.sq_no == 0 && frame_recv.sq_no == 0 && frame_frame_send; sq_no = 0 && frame_recv.sq_no == 0 && frame_frame_send; sq_no = 0 && frame_recv.sq_no == 0 && frame_frame_send; sq_no = 0 && frame_frame_recv.sq_no == 0 && frame_frame_send; sq_no = 0 && frame_recv.sq_no == 0 && frame_frame_send; sq_no = 0 && frame_frame_recv.sq_no == 0 && frame_frame_frame_send; sq_no = 0 && frame_fram	int main(int argc, char	sockaddr*)&serverAddr,	char buffer[1024];
if (argc != 2){ printf("Usage: %s <port>", argv[0]); frame_recv.sq_no == 0 && frame_recv; frame_frame_secv; frame_frame_secv; frame_frame_secv; frame_id+1){ sockfd: sockfd: socked(AF_INET, sockfd: sockaddr_in serverAddr; printf("[-]Ack Not serverAddr]); serverAddr; printf("[-]Ack Not serverAddr]); serverAddr.sin_family = AF_INET; serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); bind(sockfd, (struct sockaddr)); serverAddr.sin_addr.s_addr = sinet_addr("127.0.0.1"); serverAddr.sin_family = AF_INET; serverAddr.sin_family = sinet_addr("127.0.0.1"); serverAddr.sin_family = sinet_addr(sin_port = sinet_addr(sin</port>	*argv[]){	&addr_size);	socklen_t addr_size;
argv[0]); frame_recv.ack == Frame frame_send; svit(0); frame_id+1){ sockfd = socket(AF_INET, printf("[+]Ack Received\n"); sockfd = socket(AF_INET, sockdd; printf("[-]Ack Not sizeof(serverAddr, '\0', struct sockaddr_in printf("[-]Ack Not serverAddr.sin_family = serverAddr; Received\n"); serverAddr.sin_family = serverAddr; AF_INET; serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); frame frame_recv; close(sockfd); bind(sockfd, (struct int ack_recv = 1; return 0; sockaddr*)&serverAddr.sin_addr.s_in_addr.s_inder sockfd = socket(AF_INET, Sockfd(serverAddr.sin_addr.s_in_in_addr.s_in_addr.s_in_addr.s_in_addr.s_in_addr.s_in_addr.s_in_ad	if (argc != 2){	if(f_recv_size > 0 &&	
exit(0); } frame_id+1){ printf("[+]Ack Received\n"); ack_recv = 1; } sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, \0'), sizeof(serverAddr)); serverAddr: int frame_id = 0; Frame frame_send; Frame frame_recv; int ack_recv = 1; sockled_socked\n"); sockfd; serverAddr.sin_family = AF_INET; serverAddr.sin_port = htons(port); sizeof(serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); sizeof(serverAddr.sin_addr.s_addr = inet_size = inet_	printf("Usage: %s <port>",</port>	frame_recv.sq_no == 0 &&	Frame frame_recv;
printf("[+]Ack Received\n"); int port = atoi(argv[1]); int sockfd; int sockfd; int sockaddr_in serverAddr; char buffer[1024]; socklen_t addr_size; int frame_id = 0; Frame frame_send; Frame frame_recv; int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, \0'); serverAddr.sin_family = AF_INET; serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr)); addr_size = sizeof(newAddr); serverAddr.sin_family = AF_INET; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr)); addr_size = sizeof(newAddr); sizeof(serverAddr)); serverAddr.sin_family = AF_INET; sizeof(serverAddr.sin_addr); while(1){ int f_recv_size = recvfrom(sockfd, Aframe_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, schert = AF_INET; serverAddr.sin_family = AF_INET; serverAddr.sin_family = sockaddr*)&serverAddr.sin_family = recvf(serverAddr.sin_family = recvfrom(sockfd, Aframe_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, saddr_size); if (f_recv_size > 0 && frame_recv.frame_kind = 1 && frame_recv.frame_kind = 1 && frame_recv.frame_kind = 1 && frame_recv.sq_no == frame_id){ frame_send.sq_no = frame_id; frame_adar("127.0.1"); frame_send.sq_no = frame_id;	argv[0]);	frame_recv.ack ==	Frame frame_send;
printf("[+]Ack Received\n"); int port = atoi(argv[1]); ack_recv = 1; ack_recv = 1; sizeof(serverAddr, \0'), sizeof(serverAddr)); serverAddr: Received\n"); AF_INET; serverAddr.sin_addr.size sizeof(serverAddr); serverAddr.sin_addr.sin_addr.size sockled_t = addr("127.0.0.1"); sockled_t = sockle(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr.sin_family = AF_INET; serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); bind(sockfd, (struct sockaddr*)&serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); bind(sockfd, (struct sockaddr*)&serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); sizeof(serverAddr)); sizeof(serverAddr)); serverAddr.sin_family = AF_INET; serverAddr.sin_family = #include < stdio.h> sizeof(serverAddr)); serverAddr.sin_port = #include < stdio.h> serverAddr.sin_port = #include < stdio.h> serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); serverAddr.sin_addr.s_addr = #include < stdio.h> serverAddr.sin_erecv, sizeof(Frame), serverAddr.sin_addr.s_addr = #include < sys/systat.h> sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include < sys/systat.h> sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include < sys/systat.h> sircude < sys/systat.h> sircude < sys/systat.h> sircude < sirce > 0 && frame_recv.frame_kind = 1 && frame_recv.frame_kind = 1 && frame_recv.frame_kind = 1 && frame_recv.sq_no = = frame_id; sharp struct packet{ char data[1024]; struct packet{	exit(0);	frame_id+1){	sockfd = socket(AF_INET,
int sockfd; struct sockaddr_in serverAddr; char buffer[1024]; socklen_t addr_size; int frame_id = 0; Frame frame_send; Frame frame_recv; int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_addr.s_addr #include <string.h> #include <string.h> #include <sys types.h=""> serverAddr.sin_addr.s_addr #include <sys stat.h=""> if(ack_recv = 1){ #include <arpa inet.h=""> frame_recv, sizeof(serverAddr)); fframe_recv; int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = #include <stdo.h> #include <sys types.h=""> #include <sys types.h=""> #include <sys types.h=""> #include <sys stat.h=""> #include <sys stat.h=""> #include <sys socket.h=""> #include <sys socket.h=""> #include <arpa inet.h=""> #include <arpa ine<="" td=""><td>}</td><td>printf("[+]Ack Received\n");</td><td>SOCK_DGRAM, 0);</td></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></arpa></sys></sys></sys></sys></sys></sys></sys></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></stdo.h></arpa></sys></sys></string.h></string.h>	}	printf("[+]Ack Received\n");	SOCK_DGRAM, 0);
struct sockaddr_in printf("[-]Ack Not serverAddr.sin_family = serverAddr; Received\n"); AF_INET; socklen_t addr_size; } htons(port); int frame_id = 0; frame_id++; serverAddr.sin_addr.s_addr = Frame frame_send; } inet_addr("127.0.0.1"); Frame frame_recv; close(sockfd); bind(sockfd, (struct int ack_recv = 1; return 0; sockaddr*)&serverAddr, sock_DGRAM, 0); sizeof(serverAddr); addr_size = sizeof(newAddr); serverAddr.sin_family = #include <stdio.h> while(1){ int_recv_size = recv_riosockfd, frame_recv, sizeof(Frame), o, (struct serverAddr.sin_port = #include <string.h> sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys types.h=""> sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys socket.h=""> kaddr_size); inet_addr("127.0.0.1"); #include <sys socket.h=""> kaddr_size); if (f_recv_size > 0 && frame_recv.frame_kind == 1 kaddr_size); if (f_recv_size > 0 && frame_send.sq_no = typedef struct packet{<td>int port = atoi(argv[1]);</td><td>ack_recv = 1;</td><td>memset(&serverAddr, '\0',</td></sys></sys></sys></string.h></stdio.h>	int port = atoi(argv[1]);	ack_recv = 1;	memset(&serverAddr, '\0',
serverAddr; Received\n"); AF_INET; char buffer[1024]; ack_recv = 0; serverAddr.sin_port = socklen_t addr_size; } htons(port); int frame_id = 0; frame_id++; serverAddr.sin_addr.s_addr = Frame frame_send; } inet_addr("127.0.0.1"); Frame frame_recv; close(sockfd); bind(sockfd, (struct int ack_recv = 1; return 0; sockaddr*)&serverAddr, sock_DGRAM, 0); sizeof(serverAddr)); addr_size = sizeof(newAddr); serverAddr.sin_family = #include <stdio.h> int f_recv_size = serverAddr.sin_family = #include <stdib.h> kframe_recv, sizeof(Frame), serverAddr.sin_port = #include <strig.h> sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys types.h=""> sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys socket.h=""> if (f_recv_size > 0 && inet_addr("127.0.0.1"); #include <sys socket.h=""> kaddr_size); if(ack_recv = 1){ #include <arpa inet.h=""> kaframe_recv.sq_no == frame_send.sq_no = typedef struct packet{ frame_id){ <td>int sockfd;</td><td>}else{</td><td>sizeof(serverAddr));</td></arpa></sys></sys></sys></strig.h></stdib.h></stdio.h>	int sockfd;	}else{	sizeof(serverAddr));
char buffer[1024]; socklen_t addr_size; int frame_id = 0; Frame frame_send; Frame frame_recv; int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = #include <stdib.h> finclude <stdib.h> finclude <stdib.h> finclude <stdib.h> finclude <stdib.h> finclude <sys stat.h=""> inet_addr("127.0.0.1"); bind(sockfd, (struct) sockaddr*)&serverAddr, sizeof(serverAddr,)); addr_size = sizeof(newAddr)); while(1){ int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(Frame), 0, (struct) sockaddr*)&newAddr, sizeof(serverAddr)); serverAddr.sin_family = #include <stdib.h> finclude <string.h> serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ if(ack_recv = 1){ firame_send.sq_no = frame_id; ack_recv = 0; serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); #include <sys stat.h=""> include <sys stat.h=""> if (f_recv_size > 0 && frame_recv.frame_kind == 1 && frame_recv.sq_no == frame_id){ frame_id){ frame_id}</sys></sys></string.h></stdib.h></sys></stdib.h></stdib.h></stdib.h></stdib.h></stdib.h>	struct sockaddr_in	printf("[-]Ack Not	serverAddr.sin_family =
socklen_t addr_size; } htons(port); int frame_id = 0; frame_id++; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); Frame frame_send; } inet_addr("127.0.0.1"); bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr,)); sockaddr*, sizeof(serverAddr,); sockaddr*, sizeof(serverAddr,sizeof(serverAddr,sizeof(serverAddr,sizeof(serverAddr,size	serverAddr;	Received\n");	AF_INET;
int frame_id = 0; Frame frame_send; Frame frame_recv; int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = AF_INET; serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = #include <stdio.h> #in</stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h>	char buffer[1024];	ack_recv = 0;	serverAddr.sin_port =
Frame frame_send; Frame frame_recv; int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = #include <stdlib.h> htons(port); serverAddr.sin_addr.s_addr = int_addr("127.0.0.1"); met_addr("127.0.0.1"); while(1){ #include <sys syes.h=""> #include <sys socket.h=""> while(1){ #include <unistd.h> if(ack_recv = 1){ frame_send.sq_no = frame_id; } inet_addr("127.0.0.1"); bind(sockfd, (struct) sockaddr*)&serverAddr, sin_cluet sockaddr*)&serverAddr, sizeof(serverAddr); sizeof(serverAddr)); addr_size = sizeof(newAddr); while(1){ int f_recv_size = recvfrom(sockfd,</unistd.h></sys></sys></stdlib.h>	socklen_t addr_size;	}	htons(port);
Frame frame_recv; int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = AF_INET; serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ #include <stdio.h> #include <sys types.h=""> #include <sys stat.h=""> #include <sys socket.h=""> #include <sys socket.h=""> #include <unistd.h> #include <arpa inet.h=""> #rame_recv.frame_kind == 1 #frame_id; #frame_id){ frame_id){ char data[1024];</arpa></unistd.h></sys></sys></sys></sys></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h>	int frame_id = 0;	frame_id++;	serverAddr.sin_addr.s_addr =
int ack_recv = 1; sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ #include <stdio.h> #include <stine.h> htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ #include <sys stat.h=""> while(1){ #include <sys socket.h=""> while(1){ #include <arpa inet.h=""> frame_send.sq_no = frame_id; *sockaddr*)&serverAddr, size = sizeof(newAddr); while(1){ int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, &addr_size); if (f_recv_size > 0 && frame_recv.frame_kind == 1 && frame_recv.sq_no == frame_id){ char data[1024];</arpa></sys></sys></stine.h></stdio.h>	-	}	inet_addr("127.0.0.1");
sockfd = socket(AF_INET, SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = #include <stdlib.h> #include <sting.h> #include <styftypes.h> #include <sys types.h=""> #include <sys stat.h=""> #inc</sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></sys></styftypes.h></sting.h></stdlib.h>	Frame frame_recv;	close(sockfd);	bind(sockfd, (struct
SOCK_DGRAM, 0); memset(&serverAddr, '\0', sizeof(serverAddr)); serverAddr.sin_family = AF_INET; serverAddr.sin_port = htons(port); serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); while(1){ if(ack_recv == 1){ frame_send.sq_no = frame_id; Server Program: while(1){ imt f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, &addr_size); if (f_recv_size > 0 && frame_recv.frame_kind == 1 && frame_id; saddr_size = sizeof(newAddr); while(1){ int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, &addr_size); if (f_recv_size > 0 && frame_recv.frame_kind == 1 && frame_id){ frame_id){ frame_id){ frame_id}	int ack_recv = 1;	return 0;	sockaddr*)&serverAddr,
memset(&serverAddr, '\0', sizeof(serverAddr)); #include <stdio.h> int f_recv_size = recvfrom(sockfd, %frame_recv, sizeof(Frame), %frame_recv,</stdio.h>	, _	}	sizeof(serverAddr));
sizeof(serverAddr)); #include <stdio.h> serverAddr.sin_family = #include <stdlib.h> AF_INET; #include <string.h> serverAddr.sin_port = #include <time.h> htons(port); #include <sys types.h=""> serverAddr.sin_addr.s_addr = #include <sys stat.h=""> inet_addr("127.0.0.1"); #include <sys socket.h=""> while(1){ #include <unistd.h> if(ack_recv == 1){ #include <arpa inet.h=""> frame_send.sq_no = typedef struct packet{ frame_id; #include <stdio.h> int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, &frame_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, &frame_recv_size); if (f_recv_size); if (f_recv_size); if (f_recv_size) = 0 && frame_recv.frame_kind == 1 &frame_id){</stdio.h></arpa></unistd.h></sys></sys></sys></time.h></string.h></stdlib.h></stdio.h>	SOCK_DGRAM, 0);		addr_size = sizeof(newAddr);
serverAddr.sin_family = #include <stdlib.h> recvfrom(sockfd, %frame_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys types.h=""> #include <sys stat.h=""> serverAddr.sin_addr.s_addr = #include <sys stat.h=""> #include <sys socket.h=""> #include <sys socket.h=""> inet_addr("127.0.0.1"); #include <unistd.h> frame_recv.frame_kind == 1</unistd.h></sys></sys></sys></sys></sys></stdlib.h>	•	_	while(1){
AF_INET; #include <string.h> &frame_recv, sizeof(Frame), 0, (struct sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys types.h=""> #include <sys stat.h=""> &addr_size); if (f_recv_size > 0 && while(1){ #include <unistd.h> frame_recv.frame_kind == 1</unistd.h></sys></sys></string.h>	771		
serverAddr.sin_port = #include <time.h> 0, (struct sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys types.h=""> sockaddr*)&newAddr, &addr_size); if (f_recv_size > 0 && while(1){ #include <sys socket.h=""> frame_recv.frame_kind == 1</sys></sys></time.h>	serverAddr.sin_family =	#include <stdlib.h></stdlib.h>	recvfrom(sockfd,
htons(port); #include <sys types.h=""> sockaddr*)&newAddr, serverAddr.sin_addr.s_addr = #include <sys stat.h=""> &addr_size); inet_addr("127.0.0.1"); #include <sys socket.h=""> if (f_recv_size > 0 && while(1){ #include <unistd.h> frame_recv.frame_kind == 1</unistd.h></sys></sys></sys>	AF_INET;	_	&frame_recv, sizeof(Frame),
serverAddr.sin_addr.s_addr = #include <sys stat.h=""> &addr_size); inet_addr("127.0.0.1"); #include <sys socket.h=""> if (f_recv_size > 0 && while(1){ #include <unistd.h> frame_recv.frame_kind == 1 #include <arpa inet.h=""> &addr_size); if (f_recv_size > 0 && frame_recv.frame_kind == 1 #include <arpa inet.h=""> &addr_size); if (f_recv_size > 0 && frame_recv.frame_kind == 1 #include <arpa inet.h=""> frame_recv.sq_no == #include <arpa inet.h=""> frame_recv.frame_kind == 1 #include <arpa inet.h=""> frame_recv.sq_no == #include <a #include="" 127.0.0.1");="" <sys="" href="https://www.new.new.new.new.new.new.new.new.new.</td><td></td><td></td><td>0, (struct</td></tr><tr><td>inet_addr(" socket.h=""> if (f_recv_size > 0 && while(1){ #include <unistd.h> frame_recv.frame_kind == 1</unistd.h></arpa></arpa></arpa></arpa></arpa></unistd.h></sys></sys>	,	, ,	•
<pre>while(1){ if(ack_recv == 1){ frame_send.sq_no = frame_id;</pre>		-	
if(ack_recv == 1){		-	
frame_send.sq_no = typedef struct packet{ frame_id){ char data[1024];	* / *		
frame_id; char data[1024];	•		
			frame_id){
frame_send.frame_kind = 1; }Packet;	_		
	trame_send.frame_kind = 1;	}Packet;	

```
printf("[+]Frame Received:
%s\n",
frame_recv.packet.data);
frame send.sq no = 0;
frame_send.frame_kind = 0;
frame send.ack =
frame recv.sq no + 1;
sendto(sockfd, &frame_send,
sizeof(frame send), 0, (struct
sockaddr*)&newAddr,
addr size);
printf("[+]Ack Send\n");
}else{
printf("[+]Frame Not
Received\n");
frame id++;
}
close(sockfd);
return 0;
}
Output:
ccf@ccf-Veriton-526700
:-/Desktops gcc -o
server serverstop.c
ccf@ccf-Verlton-S2670G
:-/Desktop$ ./server
5100
[+]Frame Received: 1
[+]Ack Send
[+]Frame Received: 2
[+]Ack Send
[+] Frame Received: 3
[+]Ack Send
[+] Frame Received: 4
[+]Ack Send
ccf@ccf-veriton-52670G
:-/Desktop$ gcc -o
client clientstop.c
ccf@ccf-Veriton-526700
:-/Desktop$ ./client
5108
```

Enter Data: 1

[+] Frame Send
[+]Ack Received
Enter Data: 2
[+] Frame Send
[+]Ack Received
Enter Data: 3
[+]Frame Send
[+]Ack Received
Enter Data: 4
[+]Frame Send
[+]Ack Received

```
Go Back n
Client Program:
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
int main()
int c sock;
c sock = socket(AF INET,
SOCK STREAM, 0);
struct sockaddr_in client;
memset(&client, 0,
sizeof(client));
client.sin family = AF INET;
client.sin port = htons(5100);
client.sin addr.s addr =
inet addr("127.0.0.1");
if (connect(c sock, (struct
sockaddr *)&client,
sizeof(client)) == -1)
printf("Connection failed");
return 0;
printf("\n\tClient -with
individual acknowledgement
scheme\n\n");
char msg1[50] =
"akwnowledgementof-";
char msg2[50];
char buff[100];
int flag = 1, flg = 1;
for (int i = 0; i \le 9; i++)
flq = 1;
bzero(buff, sizeof(buff));
bzero(msg2, sizeof(msg2));
if (i == 8 \&\& flag == 1)
printf("here\n");
flag = 0;
```

read(c_sock, buff,	memset(&other, 0,	strcpy(buff2, msg);
sizeof(buff));	sizeof(other));	buff2[strlen(msg)] = i + '0';
}	server.sin_family = AF_INET;	printf("Message sent to client
int n = read(c_sock, buff,	server.sin_port =	:%s \n", buff2);
sizeof(buff));	htons(5100);	write(c_sock, buff2,
if (buff[strlen(buff) - 1] != i +	server.sin_addr.s_addr =	sizeof(buff2));
'0')	INADDR_ANY;	i = i + 1;
{	socklen_t add;	usleep(1000);
printf("Discarded as out of	if (bind(s_sock, (struct	ddd:
order \n");	sockaddr *)&server,	bzero(buff2, sizeof(buff2));
i;	sizeof(server)) == -1)	strcpy(buff2, msg);
}	{	buff2[strlen(msg)] = i + '0';
else	printf("Binding failed\n");	printf("Message sent to client
{	return 0;	:%s \n", buff2);
printf("Message received	}	write(c_sock, buff2,
from server : %s \n", buff);	printf("\tServer Up\n Go back	sizeof(buff2));
printf("Aknowledgement sent	n (n=3) used to send 10	FD_ZERO(&set1);
for message \n");	messages \n\n");	FD_SET(c_sock, &set1);
strcpy(msg2, msg1);	listen(s_sock, 10);	timeout1.tv_sec = 2;
msg2[strlen(msg2)] = i + '0';	add = sizeof(other);	timeout1.tv_usec = 0;
write(c_sock, msg2,	c_sock = accept(s_sock,	rv1 = select(c_sock + 1,
sizeof(msg2));	(struct sockaddr *)&other,	&set1, NULL, NULL,
}	&add);	&timeout1);
	time_t t1, t2;	if (rv1 == -1)
close(c_sock);	char msg[50] = "server	perror("select error ");
return 0;	message :";	else if (rv1 == 0)
}	char buff[50];	{ printf/"Coing book from
Comicar Broadom:	int flag = 0;	printf("Going back from
Server Program: #include <stdio.h></stdio.h>	fd_set set1, set2, set3;	%d:timeout \n", i);
#include <stdib.h></stdib.h>	struct timeval timeout1,	i = i - 3;
	timeout2, timeout3;	goto qq;
#include <sys socket.h=""></sys>	int rv1, rv2, rv3; int i = -1;	} else
#include <sys types.h=""> #include <sys time.h=""></sys></sys>	·	r
,	qq:	{
#include <netinet in.h=""></netinet>	i = i + 1;	read(c_sock, buff,
#include <string.h> #include <unistd.h></unistd.h></string.h>	bzero(buff, sizeof(buff));	sizeof(buff)); printf("Message from Client:
	char buff2[60];	%s\n", buff);
#include <arpa inet.h=""> #include <fcntl.h></fcntl.h></arpa>	bzero(buff2, sizeof(buff2)); strcpy(buff2, "server	, ,,
int main()	message :");	i++; if (i <= 9)
f ()	buff2[strlen(buff2)] = i + '0';	` '
int s_sock, c_sock;	buff2[strlen(buff2)] = '\0';	goto qqq; เ
s_sock = socket(AF_INET,	printf("Message sent to client	, qq2:
SOCK_STREAM, 0);	:%s \n", buff2);	FD_ZERO(&set2);
struct sockaddr_in server,	write(c_sock, buff2,	FD_SET(c_sock, &set2);
other;	sizeof(buff2));	timeout2.tv_sec = 3 ;
memset(&server, 0,	usleep(1000);	timeout2.tv_sec = 0;
sizeof(server));	i = i + 1;	
5,2551(551751),	bzero(buff2, sizeof(buff2));	
	beologbane, sieconbane),	

rv2 = select(c_sock + 1, &set2, NULL, NULL,	Client -with individual	Aknowledgement sent for message
&timeout2);	acknowledgement scheme	here
if (rv2 == -1)		Discarded as out of
perror("select error ");	Message received from	order
else if (rv2 == 0)	server : server	Message received from
s	message :0	server : server
printf("Going back from	Aknowledgement sent	message :8
%d:timeout on last 2\n", i - 1);	for message	Aknowledgement sent
i = i - 2;	Message received from	for message
bzero(buff2, sizeof(buff2));	server : server	Message received from
strcpy(buff2, msg);	message :1	server : server
buff2[strlen(buff2)] = i + '0';	Aknowledgement sent	message :9
write(c_sock, buff2,	for message	Aknowledgement sent
sizeof(buff2));	Message received from	for message
usleep(1000);	server : server	_
bzero(buff2, sizeof(buff2));	message :2	student@ccf-Veriton-S2
i++;	Aknowledgement sent	670G:~/Documents\$ gcc
strcpy(buff2, msg);	for message	servergo.c
buff2[strlen(buff2)] = i + '0';	Message received from	student@ccf-Veriton-S2
- , ,-	server : server	670G:~/Documents\$
write(c_sock, buff2, sizeof(buff2));	message :3	./a.out
	Aknowledgement sent	Server Up
goto qq2;	for message	Go back n (n=3) used
} else	Message received from	to send 10 messages
eise	server : server	
road(a cook buff	message :4	Message sent to client
read(c_sock, buff,	Aknowledgement sent	:server message :0
sizeof(buff));	for message	Message sent to client
printf("Message from Client:	Message received from	:server message :1
%s\n", buff);	server : server	Message sent to client
bzero(buff, sizeof(buff));	message :5	:server message :2
read(c_sock, buff,	Aknowledgement sent	Message from Client:
sizeof(buff));	for message	akwnowledgementof-0
printf("Message from Client:	Discarded as out of	Message sent to client
%s\n", buff);	order	:server message :3
}	Discarded as out of	Message from Client:
close(c_sock);	order	akwnowledgementof-1
close(s_sock);	Discarded as out of	Message sent to client
return 0;	order	:server message :4
}	Message received from	Message from Client:
	server : server	akwnowledgementof-2
Output:	message :6	Message sent to client
student@ccf-Veriton-S2	Aknowledgement sent	:server message :5
670G:~/Documents\$ gcc	for message	Message from Client:
clientgo.c	Message received from	akwnowledgementof-3
student@ccf-Veriton-S2	server : server	Message sent to client
670G:~/Documents\$	message :7	:server message :6
./a.out		

Message from Client: void print route(int, int); akwnowledgementof-4 int prev[20]; Message sent to client int order_arr[20]; :server message :7 int src; Message from Client: main() akwnowledgementof-5 Message sent to client int num nodes,i, j, d; :server message :8 printf("Enter number of Going back from nodes:"); 8:timeout scanf("%d", &num_nodes); Message sent to client printf("Enter the source :server message :6 node:"); Message sent to client scanf("%d", &src); :server message :7 printf("Enter the cost matrix, Message sent to client For infinity enter 999\n"); :server message :8 for(i=0; i < num nodes; Message from Client: i++) for(j=0; jakwnowledgementof-6 <num_nodes; j++) Message sent to client scanf("%d", &cost[i][j]); :server message :9 djikstra(num nodes); Message from Client: } akwnowledgementof-7 void djikstra(int num nodes Going back from 9:timeout on last 2 { Message from Client: int i, min val, I; akwnowledgementof-8 int k = 0: Message from Client: int m = 0; akwnowledgementof-9 int min =999; int last; int neighbour[20]; for(i = 0; i < 20; i++) arr[i] = -1;neighbour[i] = -1; order arr[i] = -1; prev[i] = -1;} arr[0] = src;last =0;

Link State Routing:

#include<stdio.h>

int cost[10][10];

int search(int);

int length_of(int *);

int dist[10];

int arr[20];
void djikstra(int);

dist[i] = 0;} do { for(i=0; i < _num_nodes; i++) if(search(i) == 0)if(dist[i] < min) min = dist[i]; min val = i;} } } last++; arr[last] = min val; for(i=0; i<_num_nodes; i++) if(search(i) == 0)if(cost[min val][i] < 999) neighbour[m] = i; m++; } } } m = 0; while(neighbour[m] != -1) if(dist[min val] + cost[min val][neighbour[m]] < dist[neighbour[m]]) $for(i = 0; i < num_nodes; i++)$ dist[neighbour[m]] = if(i != src) dist[min val] + cost[min val][neighbour[m]]; if(cost[src][i] < 999) prev[neighbour[m]] = min_val; dist[i] = cost[src][i]; m++; prev[i] = src; } m=0:

else

else

}

dist[i] = 999;

```
for(i = 0; i < num nodes; i++)
neighbour[i] = -1;
min =999;
min val = -1;
}while( length_of(arr) !=
num nodes);
i = 1;
I=1;
while(i < num nodes)
print route(i, I);
printf("[ distance = %d]",
dist[i]);
printf("\n");
j++;
|++;
for(k = 0; k < 20; k++)
order_arr[k] = -1;
}
}
void print_route( int _i, int _l)
int begin;
int * ptr;
int h, len, temp;
static int inc[20];
if(i == src)
ptr = order_arr;
while(*ptr != -1)
ptr++;
len = ptr-order arr;
for(h = 0; h < len/2;
h++)
temp = order_arr[h];
order arr[h] = order arr[len -
h -1];
order_arr[len-h-1] = temp;
ptr = order_arr;
printf("%d", src);
while(*ptr != -1)
printf("->%d ", *ptr);
ptr++;
return;
```

```
else
order arr[inc[ l]] = i;
inc[_l]++;
print_route(prev[_i],
_l);
}
}
int search(_i)
int i = 0;
while (arr[i] != -1)
{if(_i == arr[i]) break;
else
j++;
}
if(arr[i] == -1)
return 0;
else
return 1;
int length_of( int _arr[])
int i=0;
while(_arr[i] != -1)
j++;
return i;
}
Output:
student@ccf-Veriton-S2
670G:~/Documents$ gcc
link.c
student@ccf-Veriton-S2
670G:~/Documents$
./a.out
Enter number of
nodes:7
Enter the source
node:3
Enter the cost matrix,
For infinity enter 999
0 2 999 3 999 999 999
2 0 5 999 4 999 999
999 5 0 999 999 4 3
3 999 999 0 5 999 999
999 4 999 5 0 2 999
```

```
999 999 4 999 2 0 1

999 999 3 999 999 1 0

3->0 ->1 [ distance = 5]

3->0 ->1 ->2 [

distance = 10]

3[ distance = 0]

3->4 [ distance = 5]

3->4 ->5 [ distance = 7]

3->4 ->5 ->6 [

distance = 8]
```