4a server

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
#include<strings.h>

int main(){
 int serversocket,port;
 struct sockaddr_in serveraddr,clientaddr;
 socklen_t len;
 char message[50];
  serversocket=socket(AF_INET,SOCK_DGRA
,0);
 bzero((char*)&serveraddr,sizeof(serveraddr));
  serveraddr.sin_family=AF_INET;
 printf("Enter the port number ");
 scanf("%d",&port);
serveraddr.sin_port=htons(port);
 serveraddr.sin_addr.s_addr=INADDR_ANY;
 bind(serversocket,(struct
sockaddr*)&serveraddr,sizeof(serveraddr));
 printf("\nWaiting for the client connection\n");
bzero((char*)&clientaddr,sizeof(clientaddr));
 len=sizeof(clientaddr);
recvfrom(serversocket,message,sizeof(message),
0,(struct sockaddr*)&clientaddr,&len);
printf("\nConnection received from client.\n");
 printf("\nThe client has send:\t%s\n",message);
  printf("\nSending message to the client.\n");
 sendto(serversocket,"YOUR MESSAGE
RECEIVED.",sizeof("YOUR
MESSAGERECEIVED."),0,( struct
sockaddr*)&clientaddr,sizeof(clientaddr));
  close(serversocket);}
```

4a client

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<netdb.h>
#include<strings.h>

int main(){
int clientsocket,port;
struct sockaddr_in serveraddr;
socklen_t len;
struct hostent *server;
char message[50];

clientsocket=socket(AF_INET,SOCK_DGRAM,
0);
 bzero((char*)&serveraddr,sizeof(serveraddr));
len=sizeof(serveraddr);
serveraddr.sin_family=AF_INET;
 printf("Enter the port number ");
scanf("%d",&port);
serveraddr.sin_port=htons(port);
fgets(message,2,stdin);
printf("\nSending message for server connection\
n");
sendto(clientsocket,"HI I AM
CLIENT...",sizeof("HI I AM CLIENT...."),0,
(struct
sockaddr*)&serveraddr,sizeof(serveraddr));
 printf("\nReceiving message from server.\n");

recvfrom(clientsocket,message,sizeof(message),0
,(struct sockaddr*)&serveraddr,&len);
printf("\nMessage received:\t%s\n",message);
 close(clientsocket);
}
```

4bserver

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <time.h>
#define S_PORT 43454
#define C_PORT 43455
#define ERROR -1
#define IP_STR "127.0.0.1"

int main(int argc, char const *argv[]) {
int sfd, num;  time_t current_time;
struct sockaddr_in servaddr, clientaddr;
sfd = socket(AF_INET,
SOCK_DGRAM,IPPROTO_UDP);
if (sfd == ERROR) {
perror("Could not open a socket");
return 1;}
memset((char *) &servaddr, 0, sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(S_PORT);
memset((char *) &clientaddr, 0,
sizeof(clientaddr));
clientaddr.sin_family=AF_INET;
clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
clientaddr.sin_port=htons(C_PORT);
if((bind(sfd,(struct sockaddr
*)&servaddr,sizeof(servaddr)))!=0) {
perror("Could not bind socket");
return 2;}
printf("Server is running on %s:%d\n", IP_STR,
S_PORT);
while(1) {
recvfrom(sfd, &num, sizeof(num), 0, (struct
sockaddr *)&clientaddr, (socklen_t
*)&clientaddr);
current_time = time(NULL);
printf("Client at %s:%d asked for time: %s\n",
inet_ntoa(clientaddr.sin_addr),
ntohs(clientaddr.sin_port),
ctime(&current_time));
sendto(sfd, &current_time, sizeof(current_time),
0, (struct sockaddr *)&clientaddr,
sizeof(clientaddr));}
return 0;}
```

4bclient

```
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#define S_PORT 43454
#define C_PORT 43455
#define ERROR -1
#define IP_STR "127.0.0.1"

int main(int argc, char const *argv[]) {
int sfd;     int num = 1;
time_t start_time, rtt, current_time;
struct sockaddr_in servaddr, clientaddr;
socklen_t addrlen; sfd = socket(AF_INET,
SOCK_DGRAM,IPPROTO_UDP);
if (sfd == ERROR) {
perror("Could not open a socket");
return 1;}
memset((char *) &servaddr, 0, sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr(IP_STR);
servaddr.sin_port=htons(S_PORT);
memset((char *) &clientaddr, 0,
sizeof(clientaddr));
clientaddr.sin_family=AF_INET;
clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
clientaddr.sin_port=htons(C_PORT);
if((bind(sfd,(struct sockaddr
*)&clientaddr,sizeof(clientaddr)))!=0) {
perror("Could not bind socket");
return 2;}
printf("Client is running on %s:%d\n", IP_STR,
C_PORT);
start_time = time(NULL);
sendto(sfd, &num, sizeof(num), 0, (struct
sockaddr *)&servaddr, sizeof(servaddr));
addrlen = sizeof(clientaddr);
recvfrom(sfd, &current_time,
sizeof(current_time), 0, (struct sockaddr
*)&clientaddr, &addrlen);
rtt = time(NULL) - start_time;
current_time += rtt / 2;
printf("Server's Time: %s\n",
ctime(&current_time));
return 0;}
```

stopserver

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <signal.h>
#include <sys/time.h>
#define PORT 8080
#define MAXLINE 1024
#define TIMEOUT_SEC 2
#define TERMINATE_SIGNAL "END"

void die(char *s) { perror(s);   exit(1);}
int sockfd; struct sockaddr_in servaddr;
char buffer[MAXLINE];
char ack[MAXLINE];  int timeout_flag = 0;
void timeout_handler(int signum) {
   timeout_flag = 1;}

int main() {
if ((sockfd = socket(AF_INET,
SOCK_DGRAM, 0)) < 0)
 die("socket creation failed");
memset(&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(PORT);
servaddr.sin_addr.s_addr = INADDR_ANY;
struct timeval timeout;
timeout.tv_sec = TIMEOUT_SEC;
timeout.tv_usec = 0;
if (setsockopt(sockfd, SOL_SOCKET,
SO_RCVTIMEO, &timeout,
sizeof(timeout)) < 0)
die("setsockopt failed");
signal(SIGALRM, timeout_handler);
while (1) {
printf("Enter frame data (type 'END' to
terminate): ");
fgets(buffer, MAXLINE, stdin);
buffer[strcspn(buffer, "\n")] = 0;
 if (strcmp(buffer,
TERMINATE_SIGNAL) == 0) {
 printf("Terminating data transfer.\n");
break; }   timeout_flag = 0;
alarm(TIMEOUT_SEC);
sendto(sockfd, (const char *)buffer,
strlen(buffer),
 0, (const struct sockaddr *)&servaddr,
 sizeof(servaddr));  printf("Sent: %s\n", buffer);
int n = recvfrom(sockfd, (char *)ack,
MAXLINE,0, (struct sockaddr *)&servaddr,
 &(socklen_t){sizeof(servaddr)});  alarm(0);
 if (n > 0) {
printf("Received acknowledgment: %s\n", ack);
} else if (timeout_flag) {
printf("Timeout occurred, resending frame.\n");
continue;  } }   close(sockfd);   return 0;}
```

stopclient

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 8080
#define MAXLINE 1024
#define TERMINATE_SIGNAL "END"

void die(char *s) { perror(s);  exit(1);}

int main() {   int sockfd;
struct sockaddr_in servaddr, cliaddr;
char buffer[MAXLINE];
 char ack[MAXLINE];   int len, n;
if ((sockfd = socket(AF_INET, SOCK_DGRAM,
0)) < 0)
die("socket creation failed");
memset(&servaddr, 0, sizeof(servaddr));
 memset(&cliaddr, 0, sizeof(cliaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = INADDR_ANY;
 servaddr.sin_port = htons(PORT);
if (bind(sockfd, (const struct sockaddr
*)&servaddr, sizeof(servaddr)) < 0)
 ,die("bind failed");
len = sizeof(cliaddr);      while (1) {
n = recvfrom(sockfd, (char *)buffer, MAXLINE,
0, (struct sockaddr *)&cliaddr, &len);
buffer[n] = '\0';  printf("Received: %s\n", buffer);
if(strcmp(buffer, TERMINATE_SIGNAL) == 0)
{printf(" Terminating data transfer.\n");
 break; }     sprintf(ack, "ACK");
sendto(sockfd, (const char *)ack, strlen(ack),
 0, (const struct sockaddr *)&cliaddr,len);
  printf("Sent acknowledgment: %s\n", ack); }
 close(sockfd);  return 0; }
```

gosender

```
#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>
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;}
    printf("\tServer Up\n Go back n (n=3) used to
send 10 messages \n\n");
    listen(s_sock, 10);  add = sizeof(other);
    c_sock = accept(s_sock, (struct sockaddr
*)&other, &add);
time_t t1, t2;    char msg[50] = "servermessage :";
    char buff[50];    int flag = 0;
   fd_set set1, set2, set3;
    struct timeval timeout1, timeout2, timeout3;
    int rv1, rv2, rv3;   int i = -1;
qq:
    i = i + 1;
    bzero(buff, sizeof(buff));   char buff2[60];
    bzero(buff2, sizeof(buff2));
    strcpy(buff2, "server message :");
    buff2[strlen(buff2)] = i + '0';
    buff2[strlen(buff2)] = '\0';
    printf("Message sent to client :%s \n", buff2);
write(c_sock, buff2, sizeof(buff2)); usleep(1000);
    i = i + 1;    bzero(buff2, sizeof(buff2));
strcpy(buff2, msg);    buff2[strlen(msg)] = i + '0';  }
    printf("Message sent to client :%s \n", buff2);
 write(c_sock, buff2, sizeof(buff2)); i = i + 1;
  usleep(1000);
qqq:
    bzero(buff2, sizeof(buff2)); strcpy(buff2, msg);
    buff2[strlen(msg)] = i + '0';
    printf("Message sent to client :%s \n", buff2);
    write(c_sock, buff2, sizeof(buff2));
    FD_ZERO(&set1);   FD_SET(c_sock, &set1);
    timeout1.tv_sec = 2;  timeout1.tv_usec = 0;
 rv1 = select(c_sock + 1, &set1, NULL, NULL,
&timeout1);
  if (rv1 == -1)     perror("select error ");
    else if (rv1 == 0){
   printf("Going back from %d:timeout \n", i);
     i = i - 3; goto qq;}
    else {
      read(c_sock, buff, sizeof(buff));
      printf("Message from Client: %s\n", buff);
      i++;  if (i <= 9)    goto qqq;}
qq2:
    FD_ZERO(&set2); FD_SET(c_sock, &set2);
    timeout2.tv_sec = 3;  timeout2.tv_usec = 0;
    rv2 = select(c_sock + 1, &set2, NULL, NULL,
&timeout2);
    if (rv2 == -1)
       perror("select error ");
    else if (rv2 == 0)
    {   printf("Going back from %d:timeout on last
2\n", i - 1);
       i = i - 2;
       bzero(buff2, sizeof(buff2));
       strcpy(buff2, msg);
       buff2[strlen(buff2)] = i + '0';
       write(c_sock, buff2, sizeof(buff2));
       usleep(1000);
       bzero(buff2, sizeof(buff2));
       i++;
       strcpy(buff2, msg);
       buff2[strlen(buff2)] = i + '0';
       write(c_sock, buff2, sizeof(buff2));
       goto qq2;
    }
    else
    {read(c_sock, buff, sizeof(buff));
       printf("Message from Client: %s\n", buff);
       bzero(buff, sizeof(buff));
       read(c_sock, buff, sizeof(buff));
       printf("Message from Client: %s\n", buff); }
     close(c_sock);    close(s_sock);   return 0;
```

gorecvr

```
#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(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] = "akwnowledgementof-";
    char msg2[50];  char buff[100];
    int flag = 1, flg = 1;
    for (int i = 0; i <= 9; i++)  {flg = 1;
bzero(buff, sizeof(buff));  bzero(msg2,
sizeof(msg2));
 if (i == 8 && flag == 1) {
   printf("here\n"); i--;
 flag = 0;  read(c_sock, buff, sizeof(buff));}
       int n = read(c_sock, buff, sizeof(buff));
       if (buff[strlen(buff) - 1] != i + '0')
       {  flg=0;  i--;   }  else {
printf("Message received from server : %s \n",
buff);
printf("Aknowledgement sent for message \n");
     strcpy(msg2, msg1);
 msg2[strlen(msg2)] = i + '0';
 write(c_sock, msg2, sizeof(msg2)); }}
close(c_sock)  return 0;}
```

selectsender

```
#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;}
printf("\tServer Up\n Selective repeat scheme\n\
n");
listen(s_sock, 10);
add = sizeof(other);
c_sock = accept(s_sock, (struct sockaddr
*)&other, &add);
time_t t1, t2;
char msg[50] = "server message :";
char buff[50];
int flag = 0;
fd_set set1, set2, set3;
struct timeval timeout1, timeout2, timeout3;
```

```
int rv1, rv2, rv3;
int tot = 0;   int ok[20];
memset(ok, 0, sizeof(ok));
while (tot < 9){  int toti = tot;
for (int j = (0 + toti); j < (3 + toti); j++)
{bzero(buff, sizeof(buff));
char buff2[60];
bzero(buff2, sizeof(buff2));
strcpy(buff2, "server message :");
buff2[strlen(buff2)] = (j) + '0';
buff2[strlen(buff2)] = '\0';
printf("Message sent to client :%s \n", buff2);
write(c_sock, buff2, sizeof(buff2));
usleep(1000);}
for (int k = 0 + toti; k < (toti + 3); k++){
qq:
FD_ZERO(&set1);
FD_SET(c_sock, &set1);
timeout1.tv_sec = 2;   timeout1.tv_usec = 0;
rv1 = select(c_sock + 1, &set1, NULL, NULL,
&timeout1);
if (rv1 == -1)
perror("select error ");
else if (rv1 == 0){
printf("Timeout for message :%d \n", k);
rsendd(k, c_sock);   goto qq;}
else{  read(c_sock, buff, sizeof(buff));
printf("Message from Client: %s\n", buff);
if (buff[0] == 'n'){
printf(" corrupt message awk (msg %d) \n",
buff[strlen(buff) - 1] - '0');
rsendd((buff[strlen(buff) - 1] - '0'), c_sock);
goto qq;}    else
tot++;}}
close(c_sock); close(s_sock);
return 0;}
```

selectrecvr

```
#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]="akwnowledgementof-";
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);
else{strcpy(msg2,msg1);
count++;} msg2[strlen(msg2)]=i;
write(c_sock,msg2, sizeof(msg2)); }
close(c_sock);   return 0;}
```

4a

server.c
Enter the port number 8088
Waiting for the client connection
Connection received from client.
The client has send:HI I AM CLIENT...
Sending message to the client.

Client.c
Enter the port number 8088
Sending message for server connection
Receiving message from server.
Message received:YOUR MESSAGE RECEIVED.

4b

server.c
Server is running on 127.0.0.1:43454
Client at 127.0.0.1:43455 asked for time: Wed Apr 17 15:09:43 2024

client.c
Client is running on 127.0.0.1:43455
Server's Time: Wed Apr 17 15:09:43 2024

stop&wait

server.c
Enter frame data (type 'END' to terminate): 12
Sent: 12
Received acknowledgment: ACK
Enter frame data (type 'END' to terminate):
END
Terminating data transfer.

Client.c
Received: 12
Sent acknowledgment: ACK

gobackn

sender.c
Server Up
Go back n (n=3) used to send 10 messages

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 sent to client :server message :3
Message from Client: akwnowledgementof-1
Message sent to client :server message :4
Message from Client: akwnowledgementof-2
Message sent to client :server message :5
Message from Client: akwnowledgementof-3
Message sent to client :server message :6
Going back from 6:timeout
Message sent to client :server message :4
Message sent to client :server message :5
Message sent to client :server message :6
Message from Client: akwnowledgementof-4
Message sent to client :server message :7
Message from Client: akwnowledgementof-5
Message sent to client :server message :8
Message from Client: akwnowledgementof-6
Message sent to client :server message :9
Message from Client: akwnowledgementof-7
Going back from 9:timeout on last 2
Message from Client: akwnowledgementof-8
Message from Client: akwnowledgementof-9

receiver.c
Client -with individual acknowledgement scheme
Message received from server : server message :0
Aknowledgement sent for message
Message received from server : server message :1
Aknowledgement sent for message
Message received from server : server message :2
Aknowledgement sent for message
Message received from server :server message :3
Aknowledgement sent for message
Discarded as out of order
Discarded as out of order
Discarded as out of order
Message received from server:server message :4
Aknowledgement sent for message
Message received from server:server message :5
Aknowledgement sent for message
Message received from server:server message :6
Aknowledgement sent for message
Message received from server:server message :7
Aknowledgement sent for message
here
Discarded as out of order
Message received from server:server message :8
Aknowledgement sent for message
Message received from server:server message :9
Aknowledgement sent for message

selectiverepeat

sender.c

```
Server Up
 Selective repeat scheme

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: akwnowledgementof-1
Message from Client: akwnowledgementof-2
Message sent to client :server message :3
Message sent to client :server message :4
Message sent to client :server message :5
Message from Client: akwnowledgementof-3
Message from Client: negative akwn-4
 corrupt message awk (msg 4)
Resending Message to client :reserver
message :4
Message from Client: negative akwn-5
 corrupt message awk (msg 5)
Resending Message to client :reserver
 message :5
Message from Client: akwnowledgementof-4
Message from Client: akwnowledgementof-5
Message sent to client :server message :6
Message sent to client :server message :7
Message sent to client :server message :8
Message from Client: akwnowledgementof-6
Message from Client: akwnowledgementof-7
Timeout for message :8
Resending Message to client :reserver message :8 here
Message from Client: akwnowledgementof-8
```

receiver.c

```
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 : 0
Response/akwn sent for message
Message received from server : server message :2
correption status : 0
Response/akwn sent for message
Message received from server : server message :3
correption status : 0
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 : 1
Response/akwn sent for message
Message received from server : reserver
message :4
correption status : 0
Response/akwn sent for message
Message received from server : reserver
message :5
correption status : 0
Response/akwn sent for message
Message received from server : server message :6
correption status : 0
Response/akwn sent for message
Message received from server : server message :7
correption status : 0
Response/akwn sent for message
Message received from server : reserver
message :8
correption status : 0
Response/akwn sent for message
```