**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);
      else{
         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;
}
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;
}
```

**Output**:

```
student@ccf-Veriton-S2
670G:~/Documents$ gcc
serversel.c
student@ccf-Veriton-S2
670G:~/Documents$
./a.out
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:
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
(msg 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
(msg 8)
Resending Message to
client :reserver
message :8
Message from Client:
akwnowledgementof-8

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
Response/akwn sent for
message
Message received from
server : server
message :6
correption status : 1
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
```

**Stop and Wait:**
**Client Program:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
typedef struct packet{
char data[1024];
}Packet;
typedef struct frame{
int frame_kind;
int sq_no;
int ack;
Packet packet;
}Frame;
int main(int argc, char
*argv[]){
if (argc != 2){
printf("Usage: %s <port>",
argv[0]);
exit(0);
}
int port = atoi(argv[1]);
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_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;
frame_send.frame_kind = 1;
frame_send.ack = 0;
printf("Enter Data: ");
scanf("%s", buffer);
strcpy(frame_send.packet.dat
a, buffer);
sendto(sockfd, &frame_send,
sizeof(Frame), 0, (struct
sockaddr*)&serverAddr,
sizeof(serverAddr));
printf("[+]Frame Send\n");
}
int addr_size =
sizeof(serverAddr);
int f_recv_size =
recvfrom(sockfd,
&frame_recv,
sizeof(frame_recv), 0 ,(struct
sockaddr*)&serverAddr,
&addr_size);
if( f_recv_size > 0 &&
frame_recv.sq_no == 0 &&
frame_recv.ack ==
frame_id+1){
printf("[+]Ack Received\n");
ack_recv = 1;
}else{
printf("[-]Ack Not
Received\n");
ack_recv = 0;
}
frame_id++;
}
close(sockfd);
return 0;
}
```

**Server Program:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
typedef struct packet{
char data[1024];
}Packet;
typedef struct frame{
int frame_kind;
int sq_no;
int ack;
Packet packet;
}Frame;
int main(int argc, char**
argv){
if (argc != 2){
printf("Usage: %s <port>",
argv[0]);
exit(0);
}
int port = atoi(argv[1]);
int sockfd;
struct sockaddr_in
serverAddr, newAddr;
char buffer[1024];
socklen_t addr_size;
int frame_id=0;
Frame frame_recv;
Frame frame_send;
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 =
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,
&addr_size);
if (f_recv_size > 0 &&
frame_recv.frame_kind == 1
&& frame_recv.sq_no ==
frame_id){
```

```
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**

**<u>Client Program:</u>**

```
#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 <= 9; i++)
{
flg = 1;
bzero(buff, sizeof(buff));
bzero(msg2, sizeof(msg2));
if (i == 8 && flag == 1)
{
printf("here\n");
flag = 0;
read(c_sock, buff,
sizeof(buff));
}
int n = read(c_sock, buff,
sizeof(buff));
if (buff[strlen(buff) - 1] != i +
'0')
{
printf("Discarded as out of
order \n");
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;
}
```

**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>
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(5100);
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] = "server
message :";
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;
}
```

**Output**:

```
student@ccf-Veriton-S2
670G:~/Documents$ gcc
clientgo.c
student@ccf-Veriton-S2
670G:~/Documents$
./a.out
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
Message received from
server : server
message :4
Aknowledgement sent
for message
Message received from
server : server
message :5
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 :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

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

```
Message from Client:
akwnowledgementof-4
Message sent to client
:server message :7
Message from Client:
akwnowledgementof-5
Message sent to client
:server message :8
Going back from
8:timeout
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 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
```

**Link State Routing:**

```
#include<stdio.h>
int cost[10][10];
int dist[10];
int arr[20];
void djikstra(int);
int search(int);
int length_of( int * );
void print_route(int, int);
int prev[20];
int order_arr[20];
int src;
main()
{
int num_nodes,i, j, d;
printf("Enter number of
nodes:");
scanf("%d", &num_nodes);
printf("Enter the source
node:");
scanf("%d", &src);
printf("Enter the cost matrix,
For infinity enter 999\n");
for(i=0; i <num_nodes;
i++) for(j=0; j
<num_nodes; j++)
scanf("%d", &cost[i][j]);
djikstra( num_nodes);
}
void djikstra(int _num_nodes
)
{
int i, min_val, l;
int k =0;
int m =0;
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;
for(i = 0; i <_num_nodes; i++)
{
if(i != src)
{
if( cost[src][i] < 999)
{
dist[i] = cost[src][i];
prev[i] = src;
}
else
dist[i] = 999;
}
else
{
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]])
{
dist[neighbour[m]] =
dist[min_val] +
cost[min_val][neighbour[m]];
prev[neighbour[m]] = min_val;
}
m++;
}
m=0;
for(i = 0; i <_num_nodes; i++)
neighbour[i] = -1;
min =999;
min_val = -1;
}while( length_of(arr) !=
_num_nodes);
i =1;
l=1;
while( i < _num_nodes)
{
print_route(i, l);
printf("[ distance = %d]",
dist[i]);
printf("\n");
i++;
l++;
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
i++;
}
if(arr[i] == -1)
return 0;
else
return 1;
}
int length_of( int _arr[])
{
int i=0;
while(_arr[i] != -1)
i++;
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]
```