**TIME : SERVER**

#include<stdio.h>

#include<string.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<fcntl.h>

#include<stdlib.h>

#include<time.h>

main(int argc, char \* argv[])

{

int n;

int sock\_fd;

int i,j,k;

int childpid;

char buffer[100];

time\_t curtime;

struct sockaddr\_in servaddr, cliaddr;

int len = sizeof(cliaddr);

if(argc != 2)

{

fprintf(stderr, "Usage: ./server port\n");

exit(1);

}

if((sock\_fd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0)

{

printf("Cannot create socket\n");

exit(1);

}

bzero((char\*)&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(atoi(argv[1]));

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if(bind(sock\_fd, (struct sockaddr\*)&servaddr, sizeof(servaddr)) < 0)

{

perror("bind failed:");

exit(1);

}

while(1)

{

if((n = recvfrom(sock\_fd,buffer , sizeof(buffer), 0, (struct sockaddr \*)&cliaddr, &len)) == -1)

{

perror("size not received:");

exit(1);

}

childpid = fork();

if(childpid == 0)

{

time(&curtime);

sprintf( buffer, "= %s", ctime(&curtime));

n = sendto(sock\_fd, buffer, sizeof(buffer),0, (struct sockaddr\*)&cliaddr, sizeof(cliaddr));

if( n < 0)

{

perror("error in sending");

exit(1);

}

exit(1);

}

}

}

**TIME : CLIENT**

#include<stdio.h>

#include<string.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<fcntl.h>

#include<stdlib.h>

main(int argc, char \* argv[])

{

int i,j,n;

int sock\_fd;

struct sockaddr\_in servaddr;

char buff[100];

if(argc != 3)

{

fprintf(stderr, "Usage: ./client IPaddress\_of\_server port\n");

exit(1);

}

if((sock\_fd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0)

{

printf("Cannot create socket\n");

exit(1);

}

bzero((char\*)&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(atoi(argv[2]));

inet\_pton(AF\_INET, argv[1], &servaddr.sin\_addr);

n = sendto(sock\_fd, "", 1,0, (struct sockaddr\*)&servaddr, sizeof(servaddr));

if( n < 0)

{

perror("error in sending");

exit(1);

}

if((n=recvfrom(sock\_fd, buff, sizeof(buff),0, NULL, NULL)) == -1)

{

perror("read error from server:");

exit(1);

}

printf(" the current time of the system is %s\n", buff);

}

OUTPUT :

**Server**

VirtualBox:~/Netverk$ ./s 5000

**Client**

VirtualBox:~/Netverk$ ./c 127.0.0.1 5000

the current time of the system is = Sun Jul 31 11:19:21 2022

**STOP AND WAIT : CLIENT**

#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; //ACK:0, SEQ:1 FIN:2

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: ");

gets(buffer);

//scanf("%s", buffer);

strcpy(frame\_send.packet.data, 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;

}

**STOP AND WAIT : SERVER**

#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; //ACK:0, SEQ:1 FIN:2

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

**Client**

$ ./c 5080

Enter Data: hello i am albin

[+]Frame Send

[+]Ack Received

Enter Data: hi

[+]Frame Send

[+]Ack Received

Enter Data: j

[+]Frame Send

[+]Ack Received

Enter Data: s

[+]Frame Send

[+]Ack Received

Enter Data: end

[+]Frame Send

[+]Ack Received

Enter Data:

**Server**

$./s 5080

[+]Frame Received: hello i am albin

[+]Ack Send

[+]Frame Received: hi

[+]Ack Send

[+]Frame Received: j

[+]Ack Send

[+]Frame Received: s

[+]Ack Send

[+]Frame Received: end

[+]Ack Send

**GO-BACK-N : SERVER**

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#include<time.h>

#include<stdlib.h>

#include<ctype.h>

#include<arpa/inet.h>

#define W 5

#define P1 50

#define P2 10

char a[10];

char b[10];

void alpha9(int);

int main()

{

struct sockaddr\_in ser,cli;

int s,n,sock,i,j,c=1,f;

unsigned int s1;

s=socket(AF\_INET,SOCK\_STREAM,0);

ser.sin\_family=AF\_INET;

ser.sin\_port=6500;

ser.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

bind(s,(struct sockaddr \*) &ser, sizeof(ser));

listen(s,1);

n=sizeof(cli);

sock=accept(s,(struct sockaddr \*)&cli, &n);

printf("\nTCP Connection Established.\n");

s1=(unsigned int) time(NULL);

srand(s1);

strcpy(b,"Time Out ");

recv(sock,a,sizeof(a),0);

f=atoi(a);

while(1)

{

for(i=0;i<W;i++)

{

recv(sock,a,sizeof(a),0);

if(strcmp(a,b)==0)

{

break;

}

}

i=0;

while(i<W)

{

j=rand()%P1;

if(j<P2)

{

send(sock,b,sizeof(b),0);

break;

}

else

{

alpha9(c);

if(c<=f)

{

printf("\nFrame %s Received ",a);

send(sock,a,sizeof(a),0);

}

else

{

break;

}

c++;

}

if(c>f)

{

break;

}

i++;

}

}

close(sock);

close(s);

return 0;

}

void alpha9(int z)

{

int k,i=0,j,g;

k=z;

while(k>0)

{

i++;

k=k/10;

}

g=i;

i--;

while(z>0)

{

k=z%10;

a[i]=k+48;

i--;

z=z/10;

}

a[g]='\0';

}

**GO-BACK-N : CLIENT**

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#include<time.h>

#include<stdlib.h>

#include<ctype.h>

#define W 5

char a[10];

char b[10];

void alpha9(int);

int main()

{

int num;

printf("ENTER NUMBER :");

scanf("%d",&num);

alpha9(num);

printf("A :%s",a);

/\*int s,f,wl,c=1,x,i=0,j,n,p=0,e=0;

struct sockaddr\_in ser;

s=socket(AF\_INET,SOCK\_STREAM,0);

ser.sin\_family=AF\_INET;

ser.sin\_port=6500;

ser.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

connect(s,(struct sockaddr \*) &ser, sizeof(ser));

printf("\nTCP Connection Established.\n");

printf("\nEnter the number of Frames: ");

scanf("%d",&f);

alpha9(f);

send(s,a,sizeof(a),0);

strcpy(b,"Time Out ");

while(1)

{

for(i=0;i<W;i++)

{

alpha9(c);

send(s,a,sizeof(a),0);

if(c<=f)

{

printf("\nFrame %d Sent",c);

c++;

}

}

i=0;

wl=W;

while(i<W)

{

recv(s,a,sizeof(a),0);

p=atoi(a);

if(strcmp(a,b)==0)

{

e=c-wl;

if(e<f)

{

printf("\nTime Out, Resent Frame %d onwards",e);

}

break;

}

else

{

if(p<=f)

{

printf("\nFrame %s Acknowledged",a);

wl--;

}

else

{

break;

}

}

if(p>f)

{

break;

}

i++;

}

if(wl==0 && c>f)

{

send(s,b,sizeof(b),0);

break;

}

else

{

c=c-wl;

wl=W;

}

}

close(s);

return 0;

\*/

}

void alpha9(int z)

{

int k,i=0,j,g;

k=z;

while(k>0)

{

i++;

k=k/10;

}

g=i;

i--;

while(z>0)

{

k=z%10;

a[i]=k+48;

i--;

z=z/10;

}

a[g]='\0';

}

OUTPUT

**Server**

$ ./s 3000

TCP Connection Established.

Frame 1 Received

Frame 2 Received

Frame 3 Received

Frame 4 Received

Frame 5 Received

Frame 6 Received

Frame 7 Received

Frame 8 Received

Frame 9 Received

**Client**

$ ./c 3000

ENTER NUMBER : 70826179

A :70826179

TCP Connection Established.

Enter the number of Frames: 10

Frame 1 Sent

Frame 2 Sent

Frame 3 Sent

Frame 4 Sent

Frame 5 Sent

Frame 1 Acknowledged

Frame 2 Acknowledged

Time Out, Resent Frame 3 onwards

Frame 3 Sent

Frame 4 Sent

Frame 5 Sent

Frame 6 Sent

Frame 7 Sent

Frame 3 Acknowledged

Frame 4 Acknowledged

Time Out, Resent Frame 5 onwards

Frame 5 Sent

Frame 6 Sent

Frame 7 Sent

Frame 8 Sent

Frame 9 Sent

Frame 5 Acknowledged

Frame 6 Acknowledged

Frame 7 Acknowledged

Frame 8 Acknowledged

Frame 9 Acknowledged

Frame 10 Sent

**SELECTIVE REPEAT : SERVER**

#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++){

//printf("%d %d %d \n",tot,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,tot,j);

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;} // a timeout occured

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++;

//printf("%d %d %d \n",tot,toti,k);

}

}

}

close(c\_sock);

close(s\_sock);

return 0;

}

**SELECTIVE REPEAT : CLIENT**

#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(){ //simulating corruption of message

int d=rand()%4;

//printf("%d\n",d);

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){

//sleep(3);

printf("here\n"); //simulate loss

//i--;

flag=0;

read(c\_sock,buff,sizeof(buff));

//printf("aa %s \n",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;

}

OUTPUT :

**Client**

$ ./c 3000

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 : 1

Response/akwn sent for message

Message received from server : reserver 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 : 0

Response/akwn sent for message

Message received from server : server 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

here

Message received from server : reserver message :8

correption status : 0

Response/akwn sent for message

**Server**

$ ./s 3000

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: negative akwn-2

corrupt message awk (msg 2)

Resending Message to client :reserver message :2

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: 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

Message from Client: akwnowledgementof-8

**DISTANCE VECTOR ROUTING:**

#include<stdio.h>

struct node

{

unsigned dist[20];

unsigned from[20];

}rt[10];

int main()

{

int costmat[20][20];

int nodes,i,j,k,count=0;

printf("\nEnter the number of nodes : ");

scanf("%d",&nodes);//Enter the nodes

printf("\nEnter the cost matrix :\n");

for(i=0;i<nodes;i++)

{

for(j=0;j<nodes;j++)

{

scanf("%d",&costmat[i][j]);

costmat[i][i]=0;

rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix

rt[i].from[j]=j;

} }

do

{

count=0;

for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i to k using the cost matrix

//and add the distance from k to node j

for(j=0;j<nodes;j++)

for(k=0;k<nodes;k++)

if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

{//We calculate the minimum distance

rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];

rt[i].from[j]=k;

count++;

}

}while(count!=0);

for(i=0;i<nodes;i++)

{

printf("\n\n For router %d\n",i+1);

for(j=0;j<nodes;j++)

{

printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);

}

}

printf("\n\n");

}

OUTPUT :

Enter the number of nodes :

3

Enter the cost matrix :

0 2 7

2 0 1

7 1 0

For router 1

node 1 via 1 Distance 0

node 2 via 2 Distance 2

node 3 via 3 Distance 3

For router 2

node 1 via 1 Distance 2

node 2 via 2 Distance 0

node 3 via 3 Distance 1

For router 3

node 1 via 1 Distance 3

node 2 via 2 Distance 1

node 3 via 3 Distance 0