SELECTIVE REPEAT SERVER

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <unistd.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
struct timeval timeout;
void func(int connfd)
{
   char buff[MAX];
   int f, c, ack, next = 0;
   while (1)
   {
      sleep(1);
      bzero(buff, MAX);
      recv(connfd, buff, MAX, 0);
      if (strcmp("Exit", buff) == 0)
      {
         printf("Exit\n");
         break;
      }
      f = atoi(buff);
      c = rand() % 3;
      switch (c){
      case 0:
         printf("Frame %d not received\n",f);
         ack=-1;
         printf("Negative Acknowledgement sent: %d\n",f);
         bzero(buff, MAX);
         snprintf(buff, sizeof(buff), "%d", ack);
         send(connfd, buff, sizeof(buff), 0);
         break;
      case 1:
         ack = f;
         sleep(2);
         printf("Frame %d received\nAcknowledgement sent: %d\n", f, ack);
         bzero(buff, MAX);
```

```
            snprintf(buff, sizeof(buff), "%d", ack);
            send(connfd, buff, sizeof(buff), 0);
            next = ack + 1;
            break;
         case 2:
            ack = f;
            printf("Frame %d received\nAcknowledgement sent: %d\n", f, ack);
            bzero(buff, MAX);
            snprintf(buff, sizeof(buff), "%d", ack);
            send(connfd, buff, sizeof(buff), 0);
            next = ack + 1;
            break;
         }
      }
   }
   void main()
   {
      int sockfd, connfd, len;
      struct sockaddr_in servaddr, cli;
      sockfd = socket(AF_INET, SOCK_STREAM, 0);
      if (sockfd == -1)
      {
         printf("Socket creation failed\n");
         exit(0);
      }
      else
         printf("Socket successfully created\n");
      bzero(&servaddr, sizeof(servaddr));
      servaddr.sin_family = AF_INET;
      servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
      servaddr.sin_port = htons(PORT);
      if ((bind(sockfd, (SA *)&servaddr, sizeof(servaddr))) != 0)
      {
         printf("socket bind failed\n");
         exit(0);
      }
      else
         printf("Socket successfully binded\n");
      if ((listen(sockfd, 5)) != 0)
      {
         printf("Listen failed\n");
         exit(0);
      }
      else
```

```
        printf("Server listening\n");
    len = sizeof(cli);
    connfd = accept(sockfd, (SA *)&cli, &len);
    if (connfd < 0)
    {
        printf("Server accept failed\n");
        exit(0);
    }
    else
        printf("Server accept the client\n");
    func(connfd);
    close(sockfd);
}
```

SELECTIVE REPEAT CLIENT

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <time.h>
#include <sys/time.h>
#include <unistd.h>
#include <arpa/inet.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
struct timeval timeout;
void func(int sockfd, int nf, int ws)
{
   char buff[MAX];
   int ack, i = 0, n, k, w1 = 0, w2 = ws - 1, j, flag = 0, count_ack=0;
   if (setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, (const char *)&timeout, sizeof(timeout)) < 0)
      perror("setsockopt(SO_RCVTIMEO) failed");

   for (i = 0; i < nf && i <= w2; i++)
   {
      bzero(buff, sizeof(buff));
      snprintf(buff, sizeof(buff), "%d", i);
      k = send(sockfd, buff, sizeof(buff), 0);
      printf("Frame %d sent\n", i);
   }
   while (1)
   {
      if(count_ack == nf)
      {
         strcpy(buff, "Exit");
         k = send(sockfd, buff, sizeof(buff), 0);
         break;
      }
      if (w2 - w1 != ws - 1 && flag == 0 && i != nf)
      {
         bzero(buff, sizeof(buff));
         snprintf(buff, sizeof(buff), "%d", i);
         k = send(sockfd, buff, sizeof(buff), 0);
         printf("Frame %d sent\n", i);
         w2++;
         i++;
```

```
      }
      flag = 0;
      bzero(buff, sizeof(buff));
      n = recv(sockfd, buff, MAX, 0);
      ack = atoi(buff);

      if (n > 0)
      {
         if (ack + 1 == nf)
         {
            printf("Acknowledgement received: %d\n", ack);
            count_ack++;
            bzero(buff, sizeof(buff));
            /*strcpy(buff, "Exit");
            k = send(sockfd, buff, sizeof(buff), 0);
            break;*/
         }
         else if(ack == -1){
            printf("Acknowledgement not received for %d\nResending frame\n", w1);
            bzero(buff, sizeof(buff));
            snprintf(buff, sizeof(buff), "%d", w1);
            k = send(sockfd, buff, sizeof(buff), 0);
            printf("Frame sent: %d\n",w1);
         }
         else{
            w1++;
            printf("Acknowledgement received: %d\n", ack);
            count_ack++;
         }
      }
   }
}
void main()
{
   int sockfd, connfd, f, w;
   struct sockaddr_in servaddr, cli;
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd == -1)
   {
      printf("Socket creation failed\n");
      exit(0);
   }
   else
      printf("Socket successfully created\n");
```

```
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    timeout.tv_sec = 3;
    timeout.tv_usec = 0;
    if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) != 0)
    {
        printf("Connection with the server failed\n");
        exit(0);
    }
    else
        printf("Connected to the server\n");
    printf("Enter the number of frames: ");
    scanf("%d", &f);
    printf("Enter the window size: ");
    scanf("%d", &w);
    func(sockfd, f, w);
    close(sockfd);
}
```

```
aront@Dell-Vostro:~/lab$ gcc selective_server.c -o sel_c
aront@Dell-Vostro:~/lab$ ./sel_c
Socket successfully created
Socket successfully binded
Server listening
Server accept the client
Frame 0 received
Acknowledgement sent: 0
Frame 1 received
Acknowledgement sent: 1
Frame 2 not received
Negative Acknowledgement sent: 2
Frame 3 received
Acknowledgement sent: 3
Frame 4 received
Acknowledgement sent: 4
Frame 2 received
Acknowledgement sent: 2
Exit
```

```
aront@Dell-Vostro:~/lab$ gcc selective_client.c -o sel_c
aront@Dell-Vostro:~/lab$ ./sel_c
Socket successfully created
Connected to the server
Enter the number of frames: 5
Enter the window size: 3
Frame 0 sent
Frame 1 sent
Frame 2 sent
Acknowledgement received: 0
Frame 3 sent
Acknowledgement received: 1
Frame 4 sent
Acknowledgement not received for 2
Resending frame
Frame sent: 2
Acknowledgement received: 3
Acknowledgement received: 4
Acknowledgement received: 2
```

DISTANCE VECTOR ROUTING

```
#include <stdio.h>
struct node{
   unsigned dist[20];
   unsigned from[20];
} rt[10];
void main(){
   int costmat[20][20];
   int nodes, i, j, k, count = 0;
   printf("\nEnter the number of nodes: ");
   scanf("%d", &nodes);
   printf("\nEnter the cost matrix:          -1 for infinite cost\n");
   for (i = 0; i < nodes; i++){
      for (j = 0; j < nodes; j++){
         scanf("%d", &costmat[i][j]);
         if(costmat[i][j]==-1){
            costmat[i][j]=9999;
         }
         costmat[i][i] = 0;
         rt[i].dist[j] = costmat[i][j];
         rt[i].from[j] = j;
      }
   }
   do {
      count = 0;
      for (i = 0; i < nodes; i++){
         for (j = 0; j < nodes; j++){
            for (k = 0; k < nodes; k++){
               if (rt[i].dist[j] > costmat[i][k] + rt[k].dist[j]) {
                  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\nRouting Table 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");
```

}

```
aront@Dell-Vostro:~/lab$ gcc dvr.c -o dvr
aront@Dell-Vostro:~/lab$ ./dvr

Enter the number of nodes: 4

Enter the cost matrix:           -1 for infinite cost
0 2 -1 1
2 0 3 7
-1 3 0 11
1 7 11 0


Routing Table For Router 1

Node 1 via 1 Distance: 0
Node 2 via 2 Distance: 2
Node 3 via 2 Distance: 5
Node 4 via 4 Distance: 1

Routing Table For Router 2

Node 1 via 1 Distance: 2
Node 2 via 2 Distance: 0
Node 3 via 3 Distance: 3
Node 4 via 1 Distance: 3

Routing Table For Router 3

Node 1 via 2 Distance: 5
Node 2 via 2 Distance: 3
Node 3 via 3 Distance: 0
Node 4 via 2 Distance: 6

Routing Table For Router 4

Node 1 via 1 Distance: 1
Node 2 via 1 Distance: 3
Node 3 via 1 Distance: 6
Node 4 via 4 Distance: 0
```

FTP_SERVER

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>

int main(void){
   FILE *fp;
   char name[100],fileread[100],fname[100],ch,file[100],rcv[100];
   int n;
   int socket_desc,client_sock,client_size;
   struct sockaddr_in server_addr,client_addr;

   socket_desc = socket(AF_INET, SOCK_STREAM, 0);
   if(socket_desc < 0){
      printf("Unable to create socket\n");
      return -1;
   }
   printf("Socket created successfully\n");

   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(2000);
   server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

   if(bind(socket_desc,(struct sockaddr*)&server_addr, sizeof(server_addr)) < 0){
      printf("Could'nt bind to port\n");
      return -1;
   }
   printf("Binding Completed\n");

   if(listen(socket_desc,1) < 0){
      printf("Error while listening\n");
      return -1;
   }
   printf("Listening for Connections\n");

   client_size = sizeof(client_addr);
   client_sock = accept(socket_desc, (struct sockaddr*)&client_addr, &client_size);

   if(client_sock < 0){
      printf("Can't Accept");
      return -1;
```

```
    }
    printf("Client connected at IP: %s and port:
%i\n",inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));
    printf("Connection Accepted\n");

    n=recv(client_sock,rcv,100,0);
    rcv[n]='\0';

    fp=fopen(rcv,"r");

    if(fp==NULL){
       send(client_sock,"error",5,0);
       close(client_sock);
    }
    else{
       while(fgets(fileread,sizeof(fileread),fp)){
          if(send(client_sock,fileread,sizeof(fileread),0)<0){
            printf("Can't send file contents\n");
          }
          sleep(1);
       }

       if(!fgets(fileread,sizeof(fileread),fp)){
          printf("Done..\n");
          send(client_sock,"completed",9,0);
       }
    }
    close(client_sock);
    close(socket_desc);

    return 0;
}
```

FTP_CLIENT

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>

int main(void){
   FILE *fp;
   int n,s;
   char name[100],rcvmsg[100],rcvg[100],fname[100];
   int socket_desc;
   struct sockaddr_in server_addr;

   socket_desc = socket(AF_INET, SOCK_STREAM, 0);
   if(socket_desc < 0){
      printf("Unable to create socket\n");
      return -1;
   }
   printf("Socket created successfully\n");

   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(2001);
   server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

   if(connect(socket_desc,(struct sockaddr*)&server_addr, sizeof(server_addr)) < 0){
      printf("Unable to Connect\n");
      return -1;
   }
   printf("Connected with server successfully\n");

   printf("Enter filename ");
   scanf("%s",name);
   printf("Enter the new filename\t");
   scanf("%s",fname);

   fp=fopen(fname,"w");
   send(socket_desc,name,sizeof(name),0);

   while(1){
      s=recv(socket_desc,rcvg,100,0);
      rcvg[s]='\0';
```

```
    if(strcmp(rcvg,"error")==0){
       printf("File is not available\n");
       exit(1);
    }
    if(strcmp(rcvg,"completed")==0){
       printf("\nFile is transferred...\n");
       fclose(fp);
       break;
    }
    else{
       printf("The file contents are: ");
       fputs(rcvg,stdout);
       fprintf(fp,"%s",rcvg);
    }
  }
  close(socket_desc);
  return 0;
}
```

```
aront@Dell-Vostro:~/lab$ gcc ftpserver.c -o ftp_s
aront@Dell-Vostro:~/lab$ ./ftp_s
Socket created successfully
Binding Completed
Listening for Connections
Client connected at IP: 127.0.0.1 and port: 49460
Connection Accepted
Done..
```

```
aront@Dell-Vostro:~/lab$ gcc ftpclient.c -o ftp_c
aront@Dell-Vostro:~/lab$ ./ftp_c
Socket created successfully
Connected with server successfully
Enter filename text.txt
Enter the new filename  copy.txt
The file contents are: Hi! How's life?
File is transferred...
```

LEAKY BUCKET

```
#include<stdio.h>
void main(){
   int in,out,bsize,n,bucket=0;
   printf("Enter the bucket size: ");
   scanf("%d",&bsize);
   printf("Enter the no of inputs: ");
   scanf("%d",&n);
   printf("Enter the packet outgoing rate: ");
   scanf("%d",&out);
   while(n!=0){
      printf("Enter the incoming packet size: ");
      scanf("%d",&in);
      if(in<=(bsize-bucket)){
         bucket+=in;
         printf("  Bucket status: %d out of %d\n",bucket,bsize);
      }
      else{
         printf("  Dropped packet:%d \n",in-(bsize-bucket));
         bucket=bsize;
         printf("  Bucket status: %d out of %d\n",bucket,bsize);
      }
      bucket=bucket-out;
      printf("  After outgoing,bucket status: %d out of %d\n",bucket,bsize);
      n--;
   }
}
```

```
aront@Dell-Vostro:~/lab$ gcc leaky.c -o leaky
aront@Dell-Vostro:~/lab$ ./leaky
Enter the bucket size: 4
Enter the no of inputs: 3
Enter the packet outgoing rate: 2
Enter the incoming packet size: 4
  Bucket status: 4 out of 4
  After outgoing,bucket status: 2 out of 4
Enter the incoming packet size: 2
  Bucket status: 4 out of 4
  After outgoing,bucket status: 2 out of 4
Enter the incoming packet size: 3
  Dropped packet:1
  Bucket status: 4 out of 4
  After outgoing,bucket status: 2 out of 4
```