TCP SOCKET(EXP.04)

Server Side

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

int main(void)
{
   int socket_desc, client_sock, client_size;
   struct sockaddr_in server_addr, client_addr;
   char server_message[2000], client_message[2000];

   // Clean buffers:
   memset(server_message, '\0',
sizeof(server_message));
   memset(client_message, '\0',
sizeof(client_message));

   // Create socket:
   socket_desc = socket(AF_INET, SOCK_STREAM, 0);

   if(socket_desc < 0){
      printf("Error while creating socket\n");
      return -1;
  }
   printf("Socket created successfully\n");

   // Set port and IP:
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(2000);
   server_addr.sin_addr.s_addr =
inet_addr("127.0.0.1");

   // Bind to the set port and IP:
   if(bind(socket_desc, (struct
sockaddr*)&server_addr, sizeof(server_addr))<0){
      printf("Couldn't bind to the port\n");
      return -1;
   }
   printf("Done with binding\n");

   // Listen for clients:
   if(listen(socket_desc, 1) < 0){
      printf("Error while listening\n");
      return -1;
   }
   printf("\nListening for incoming
connections.....\n");

   // Accept an incoming connection:
   client_size = sizeof(client_addr);
   client_sock = accept(socket_desc, (struct
sockaddr*)&client_addr, &client_size);

   if (client_sock < 0){
      printf("Can't accept\n");
      return -1;
   }
   printf("Client connected at IP: %s and port: %i\n",
inet_ntoa(client_addr.sin_addr),
ntohs(client_addr.sin_port));

   // Receive client's message:
   if (recv(client_sock, client_message,
sizeof(client_message), 0) < 0){
      printf("Couldn't receive\n");
      return -1;
   }
   printf("Msg from client: %s\n", client_message);
printf("\nEnter the Server Message:");
scanf("%s",server_message);

   // Respond to client:
   strcpy(server_message, "This is the server's
message.");

   if (send(client_sock, server_message,
strlen(server_message), 0) < 0){
      printf("Can't send\n");
      return -1;
   }

   // Closing the socket:
   close(client_sock);
   close(socket_desc);

   return 0;
}
```

<u>Client Side</u>

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

int main(void)
{
   int socket_desc;
   struct sockaddr_in server_addr;
```

```
    char server_message[2000], client_message[2000];

    // Clean buffers:

memset(server_message,'\0',sizeof(server_message))
;

memset(client_message,'\0',sizeof(client_message));

    // Create socket:
    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");

    // Set port and IP the same as server-side:
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr =
inet_addr("127.0.0.1");

    // Send connection request to server:
    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");

    // Get input from the user:
    printf("Enter message: ");
    gets(client_message);

    // Send the message to server:
    if(send(socket_desc, client_message,
strlen(client_message), 0) < 0){
        printf("Unable to send message\n");
        return -1;
    }

    // Receive the server's response:
    if(recv(socket_desc, server_message,
sizeof(server_message), 0) < 0){
        printf("Error while receiving server's msg\n");
        return -1;
    }

    printf("Server's response: %s\n",server_message);
```

```
// Close the socket:
close(socket_desc);

return 0;}
```

UDP(EXP.05)

Server Side

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char **argv){

  if (argc != 2) {
    printf("Usage: %s <port>\n", argv[0]);
    exit(0);
  }

  char *ip = "127.0.0.1";
  int port = atoi(argv[1]);

  int sockfd;
  struct sockaddr_in server_addr, client_addr;
  char buffer[1024];
  socklen_t addr_size;
  int n;

  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  if (sockfd < 0) {
    perror("[-]socket error");
    exit(1);
  }

  memset(&server_addr, '\0', sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(port);
  server_addr.sin_addr.s_addr = inet_addr(ip);

  n = bind(sockfd, (struct sockaddr*)&server_addr,
sizeof(server_addr));
  if (n < 0){
    perror("[-]bind error");
    exit(1);
  }

  bzero(buffer, 1024);
  addr_size = sizeof(client_addr);
  recvfrom(sockfd, buffer, 1024, 0, (struct
sockaddr*)&client_addr, &addr_size);
  printf("[+]Data recv: %s\n", buffer);

  bzero(buffer, 1024);
  strcpy(buffer, "Welcome to the UDP Server.");
  sendto(sockfd, buffer, 1024, 0, (struct
sockaddr*)&client_addr, sizeof(client_addr));
  printf("[+]Data send: %s\n", buffer);

  return 0;
}
```

Client Side

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char **argv){

  if (argc != 2) {
    printf("Usage: %s <port>\n", argv[0]);
    exit(0);
  }

  char *ip = "127.0.0.1";
  int port = atoi(argv[1]);

  int sockfd;
  struct sockaddr_in addr;
  char buffer[1024];
  socklen_t addr_size;

  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  memset(&addr, '\0', sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_port = htons(port);
  addr.sin_addr.s_addr = inet_addr(ip);

  bzero(buffer, 1024);
  strcpy(buffer, "Hello World!");
  sendto(sockfd, buffer, 1024, 0, (struct
sockaddr*)&addr, sizeof(addr));
  printf("[+]Data send: %s\n", buffer);

  bzero(buffer, 1024);
  addr_size = sizeof(addr);
  recvfrom(sockfd, buffer, 1024, 0, (struct
sockaddr*)&addr, &addr_size);
  printf("[+]Data recv: %s\n", buffer);

  return 0;
}
```

STOP AND WAIT(EXP.06)

```
#include<stdio.h>
#include<stdlib.h>
int ack()
{
int k;
k=rand();
if(k%2==0)
return 1;
else
return 0;
}
void main()
{
int n,i,test;
printf("Enter the number of packet you need to stimulate\n");
scanf("%d",&n);
for(i=0;i<=n;i++)
{
x:test=ack();
printf("%d\n",test);
if(test==1)
{
printf("Success ack received for pack - %d -  sending next packet\n",i);
}
else
{
printf("Failed ack not received for pack %d – sending packet again\n",i);
goto x;
}
}
}
```

GO BACK N(EXP.07)

```
#include<stdio.h>

#include<stdlib.h>

void main()

{

int
temp1,temp2,temp3,temp4,i,winsize=8,noframes,mo
reframes;

int receiver(int);

int simulate(int);

temp4=0,temp1=0,temp2=0,temp3=0;

noframes=10;

winsize=8;

moreframes=noframes;

printf(“Number of frames = %d\n”,noframes);

while(moreframes>0)

{

temp1=simulate(winsize);

winsize-=temp1;

temp4+=temp1;

if(temp4>noframes)

{

temp4=noframes;

}

for(i=temp3+1;i<=temp4;i++)

printf(“\nSending frame %d”,i);

temp2=receiver(temp1);

temp3+=temp2;

if(temp3>noframes)

temp3=noframes;

printf(“\nacknowledgement for frame upto
%d”,temp3);

moreframes-=temp2;
temp4=temp3;
if(winsize<=0)
winsize=8;
}
printf("\nEnd of sliding window protocol");
}
int receiver(int temp1)
{
int i;
for(i=0;i<100;i++)
rand();
i=rand()%temp1;
return i;
}
int simulate(int winsize)
{
int temp1,i;
for(i=0;i<50;i++)
temp1=rand();
if(temp1==0)
temp1=simulate(winsize);
i=temp1%winsize;
if(i==0)
return winsize;
else
return temp1%winsize;
}
```

SLIDING WINDOW(EXP.08)

```
#include<studio.h>

#include<string.h>

#include<stdlib.h>

void main()

{

char sender[50],receiver[50];

int i,winsize;

printf("Enter the window size:");

scanf("%d",& winsize);

printf("\n sender window is expected to store
message\n");

printf("Enter the data to be sent:");

flush(stdin);

scanf("%s",sender);

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

receiver[i] =sender[i];

receiver[i]=NULL;

printf("\n window size of receiver is expanded \n");

printf("\n Acknowledgement from receiver \n");

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

printf("\n ack:%d",i);

printf("\n Msg received is %s \n",receiver);

printf("\n window size of receiver shrinked \n");

}
```