INDEX

EXPT	TITLE	DATE	SIGN
1	Familiarization with o Networking cables a. CAT5, UTP b. Connectors RJ45, T-connector c. Hubs, Switches		
2	TCP Socket Programming Multicast & Broadcast Sockets		
3	UDP Socket Programming Multicast & Broadcast Sockets		
4	Data Link Layer Flow Control Mechanism a. Stop & Wait, b. Sliding Window		
5	Data Link Layer Error Control Mechanism a. Selective Repeat. b. Go Back N		
6	Data Link Layer Error Detection Mechanism a. Cyclic Redundancy Check		

Assignment - 1

Problem Statement : Familiarization with o Networking cables

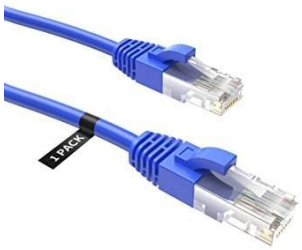
a. CAT5, UTP

b. Connectors RJ45, T-connector

c. Hubs, Switches

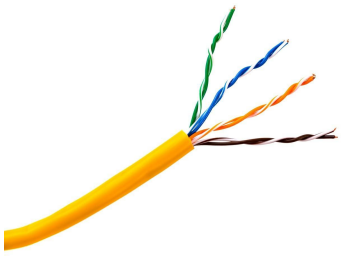
Theory :

CAT5



Category 5 cable (Cat 5) is a twisted pair cable for computer networks. Since 2001, the variant commonly in use is the Category 5e specification (Cat 5e). The cable standard provides performance of up to 100 MHz and is suitable for most varieties of Ethernet over twisted pair up to 2.5GBASE-T[1][2][3][4] but more commonly runs at 1000BASE-T (Gigabit Ethernet) speeds. Cat 5 is also used to carry other signals such as telephone and video.
This cable is commonly connected using punch-down blocks and modular connectors. Most Category 5 cables are unshielded, relying on the balanced line twisted pair design and differential signaling for noise rejection.

UTP:



Unshielded Twisted Pair (UTP) cable is most certainly by far the most popular cable around the world. UTP cable is used not only for networking but also for the traditional telephone (UTP-Cat 1). There are seven different types of UTP categories and, depending on what you want to achieve, you would need the appropriate type of cable. UTP-CAT5e is the most popular UTP

cable which came to replace the old coaxial cable that was not able to keep up with the constant growing need for faster and more reliable networks.

CHARACTERISTICS OF UTP

The characteristics of UTP are very good and make it easy to work with, install, expand and troubleshoot and we are going to look at the different wiring schemes available for UTP, how to create a straight through UTP cable, rules for safe operation and a lot of other cool stuff !
So let's have a quick look at each of the UTP categories available today along with their specifications:

RJ45 or 8P8C connector?

It was originally developed as part of the standardized telecommunication network interface for the purpose of connecting telephone networks. The connectors of the RJ platform were developed as much smaller and cheaper replacements to the older telephone installation methods of hardwired cords or bulky plugs.With the advent of computer networking, a new but very similar connector was developed to carry out data transfer: 8P8C.
8P8C refers to the array of pins, hence the name Eight Position, Eight Contact. In 8P8C connectors, each plug has eight positions that are spaced approximately 1 mm apart. Individual wires are then inserted into these positions. There are a variety of 8P8C connectors out there, with the modern RJ45 Ethernet connector being the most prevalent.
As a result of the close similarities, “RJ45” became the designated informal misnomer for any eight-pinned jack (8P8C modular connector) used in computer networking (Ethernet).
The differences between 8P8C connectors and RJ45 connectors are shown in the following diagram:

HUB, Switches, Routers
Hub:

A Hub is just a connector that connects the wires coming from different sides. There is no signal processing or regeneration. It is an electronic device that operates only on physical layers of the OSI model.It is also known as a repeater as it transmits signal to every port except the port from where signal is received. Also, hubs are not that intelligent in communication and processing information for 2nd and 3rd layer.

Switch:

Switch is a point to point communication device. It operates at the data link layer of OSI model. It uses switching table to find out the correct destination. Basically, it is a kind of bridge that provides better connections. It is a kind of device that set up and stop the connections according to the requirements needed at that time. It comes up with many features such as flooding, filtering and frame transmission.

Router:

Routers are the multiport devices and more sophisticated as compared to repeaters and bridges. It contains a routing table that enables it to make decision about the route i.e. to determine which of several possible paths between the source and destination is the best for a particular transmission. It works on the network layer 3 and used in LANs, MANs and WANs. It stores IP address and maintains address on its own.

Assignment 2

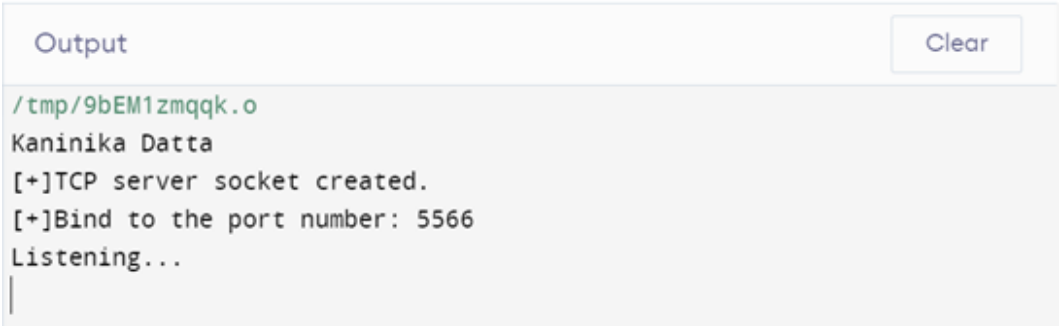
Code :

Server Side Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
int main(){
printf(“Kaninika Datta\n”);
char *ip = "127.0.0.1";
int port = 5566;
int server_sock, client_sock;
struct sockaddr_in server_addr, client_addr;
socklen_t addr_size;
char buffer[1024];
int n;
server_sock = socket(AF_INET, SOCK_STREAM, 0);
if (server_sock < 0){
perror("[-]Socket error");
exit(1);
}
printf("[+]TCP server socket created.\n");
memset(&server_addr, '\0', sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = port;
server_addr.sin_addr.s_addr = inet_addr(ip);
n = bind(server_sock, (struct sockaddr*)&server_addr,sizeof(server_add));
if (n < 0){
perror("[-]Bind error");
exit(1);
}
printf("[+]Bind to the port number: %d\n", port);
listen(server_sock, 5);
printf("Listening...\n");
while(1){
addr_size = sizeof(client_addr);
client_sock = accept(server_sock, (struct sockaddr*)&client_addr, &addr_size);
printf("[+]Client connected.\n");
bzero(buffer, 1024);
recv(client_sock, buffer, sizeof(buffer), 0);
printf("Client: %s\n", buffer);
```

```
bzero(buffer, 1024);
strcpy(buffer, "HI, THIS IS SERVER. HAVE A NICE DAY!!!");
printf("Server: %s\n", buffer);
send(client_sock, buffer, strlen(buffer), 0);
close(client_sock);
printf("[+]Client disconnected.\n\n");
}
return 0;
}
```

Output

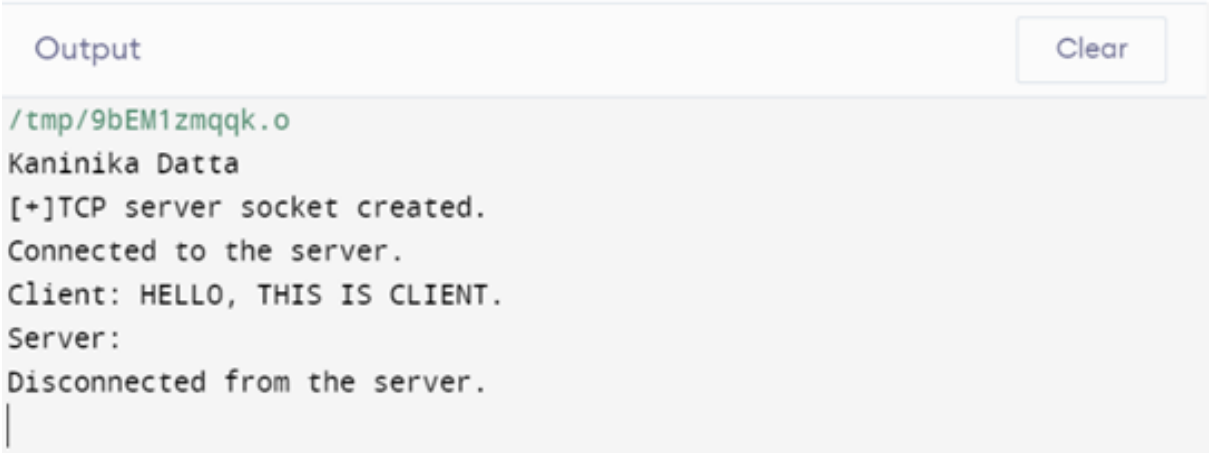


Client Side Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
int main(){
printf("Kaninika Datta\n");
char *ip = "127.0.0.1";
int port = 5566;
int sock;
struct sockaddr_in addr;
socklen_t addr_size;
char buffer[1024];
int n;
sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock < 0){
perror("[-]Socket error");
exit(1);
```

```
}
printf("[+]TCP server socket created.\n");
memset(&addr, '\0', sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_port = port;
addr.sin_addr.s_addr = inet_addr(ip);
connect(sock, (struct sockaddr*)&addr, sizeof(addr));
printf("Connected to the server.\n");
bzero(buffer, 1024);
strcpy(buffer, "HELLO, THIS IS CLIENT.");
printf("Client: %s\n", buffer);
send(sock, buffer, strlen(buffer), 0);
bzero(buffer, 1024);
recv(sock, buffer, sizeof(buffer), 0);
printf("Server: %s\n", buffer);
close(sock);
printf("Disconnected from the server.\n");
return 0;
 }
```

Output



Assignment 3

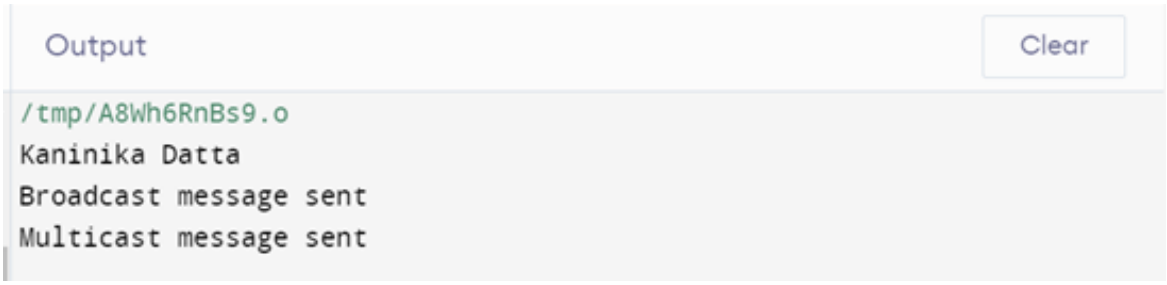
Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>
#include <netdb.h>
#define BUF_SIZE 256
#define BROADCAST_PORT 8888
#define MULTICAST_ADDR "224.0.0.1"
#define MULTICAST_PORT 9999
void error(char *msg) { perror(msg);
exit(1);
}
int main(int argc, char *argv[]) { printf("Kaninika Datta\n");
int sockfd_broadcast, sockfd_multicast, n;
struct sockaddr_in broadcast_addr, multicast_addr; struct in_addr localInterface;
struct hostent *group; char buffer[BUF_SIZE];
    sockfd_broadcast = socket(AF_INET, SOCK_DGRAM,
    0);
 if (sockfd_broadcast < 0)
error("ERROR opening broadcast socket");
 int broadcastEnable = 1;
setsockopt(sockfd_broadcast, SOL_SOCKET, SO_BROADCAST, &broadcastEnable, sizeof(broadcastEnable));
bzero(&broadcast_addr, sizeof(broadcast_addr));
broadcast_addr.sin_family = AF_INET; broadcast_addr.sin_port = htons(BROADCAST_PORT);
if (inet_aton("255.255.255.255", &broadcast_addr.sin_addr) == 0)
error("ERROR setting broadcast address");
sockfd_multicast = socket(AF_INET, SOCK_DGRAM, 0);
 if (sockfd_multicast < 0)
error("ERROR opening multicast socket");
localInterface.s_addr = inet_addr("127.0.0.1");
setsockopt(sockfd_multicast, IPPROTO_IP, IP_MULTICAST_IF, (char *)&localInterface, sizeof(localInterface));
bzero(&multicast_addr, sizeof(multicast_addr));
multicast_addr.sin_family = AF_INET;
multicast_addr.sin_port = htons(MULTICAST_PORT);
group = gethostbyname(MULTICAST_ADDR);
```

```
if (group == NULL)

error("ERROR getting multicast group");
memcpy(&multicast_addr.sin_addr.s_addr, group->h_addr, group->h_length);
sprintf(buffer, "Broadcast message from client");
n = sendto(sockfd_broadcast, buffer, strlen(buffer), 0, (struct sockaddr *)&broadcast_addr,
sizeof(broadcast_addr));
if (n < 0)
error("ERROR sending broadcast message"); printf("Broadcast message sent\n");
sprintf(buffer, "Multicast message from client");
n = sendto(sockfd_multicast, buffer, strlen(buffer), 0, (struct sockaddr *)&multicast_addr,
sizeof(multicast_addr));
if (n < 0)
error("ERROR sending multicast message");
printf("Multicast message sent\n"); close(sockfd_broadcast);
close(sockfd_multicast);
return 0;
}
```

Output



Assignment 4

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int k,time,win=2,i2=0,frame=0,a[20],b[20],i,j,s,r,ack,c,d;
int send(int,int);
int receive();
int checsum(int *);

main()
{
printf(“Kaninika Datta\n”);
int i1=0,j1=0,c1;
printf("Enter the frame size\n");
scanf("%d",&frame);
printf("Enter the window size\n");
scanf("%d",&win);
j1=win;
for(i=0;i<frame;i++)
{
a[i]=rand();
}
k=1;
while(i1<frame)
{
if((frame-i1)<win)
j1=frame-i1;
printf("\n\ntransmit the window no %d\n\n",k);
c1=send(i1,i1+j1);
ack=receive(i1,i1+j1,c1);
if (ack!=0)
{printf("\n\n1.Selective window\n");
printf("2.Go back N\n");
scanf("%d",&ack);
switch(ack)
{
case 1:
printf("\n\n\t Selective window \t\nEnter the faulty frame no\n");
scanf("%d",&i2);
printf("\n\n Retransmit the frame %d \n",i2);
send(i2,i2+1);
break;
```

```
case 2:
printf("\n\n\t Go back n\t\n\n");
printf("\nRetransmit the frames from %d to %d\n",i1,i1+j1);
send(i1,i1+j1);
break;
}
}
i1=i1+win;
k++;
}
}

int send(c,d)
{
int t1;
for(i=c;i<d;i++)
{
b[i]=a[i];
printf("frame %d is sent\n",i);
}
s=checsum(&a[c]);
return(s);
}

int receive(c,d,c2)
int c2;
{
r=checsum(&b[c]);
if(c2==r)
{
return(0);
}
else
return(1);
}

int checsum(int *c)
{
int sum=0;
for(i=0;i<win;i++)
sum=sum^(*c);
return sum;
}
```

Output

Output

Clear

```
/tmp/bxk4PuI9M6.o
Kaninika Datta
Enter the frame size
50
Enter the window size
5
transmit the window no 1
frame 0 is sent
frame 1 is sent
frame 2 is sent
frame 3 is sent
frame 4 is sent

transmit the window no 2
frame 5 is sent
frame 6 is sent
frame 7 is sent
frame 8 is sent
frame 9 is sent

transmit the window no 3
frame 10 is sent
frame 11 is sent
frame 12 is sent
frame 13 is sent
frame 14 is sent

transmit the window no 4
frame 15 is sent
frame 16 is sent
frame 17 is sent
frame 18 is sent
frame 19 is sent

transmit the window no 5
frame 35005211 is sent

1.Selective window
2.Go back N
```

Assignment 5

Code:

Selective Repeat:

```
#include<stdio.h>
int main()
{
printf("Kaninika Datta\n");
int w,i,f,frames[50];
printf("Enter window size: ");
scanf("%d",&w);
printf("\nEnter number of frames to transmit: ");
scanf("%d",&f);
printf("\nEnter %d frames: ",f);
for(i=1;i<=f;i++)
scanf("%d",&frames[i]);
printf("\nWith sliding window protocol the frames will be sent in the following manner
(assuming no corruption of frames)\n\n");
printf("After sending %d frames at each stage sender waits for acknowledgement sent
by the receiver\n\n",w);
for(i=1;i<=f;i++)
{
if(i%w==0)
{
printf("%d\n",frames[i]);
printf("Acknowledgement of above frames sent is received by sender\n\n");
}
else
printf("%d ",frames[i]);
}
if(f%w!=0)
printf("\nAcknowledgement of above frames sent is received by sender\n");
return 0;
}
```

Output:

```
Output                                                    Clear
/tmp/bxk4PuI9M6.o
Kaninika Datta
Enter window size: 3
Enter number of frames to transmit: 5
Enter 5 frames: 12 5 89 4 6
With sliding window protocol the frames will be sent in the following
    manner(assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent
    by the receiver

12 5 89
Acknowledgement of above frames sent is received by sender

4 6
Acknowledgement of above frames sent is received by sender
```

Code:

Go Back N:

```
#include<stdio.h>
int main()
{
printf("Kaninika Datta\n");
int windowsize,sent=0,ack,i;
printf("enter window size\n");
scanf("%d",&windowsize);
while(1)
{
for( i = 0; i < windowsize; i++)
{
printf("Frame %d has been transmitted.\n",sent);
sent++;
if(sent == windowsize)
break;
}
printf("\nPlease enter the last Acknowledgement received.\n");
scanf("%d",&ack);
if(ack == windowsize)
break;
else
sent = ack;
}
return 0;
}
```

Output

Output Clear

```
/tmp/bxk4PuI9M6.o
Kaninika Datta
enter window size
8
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received.
7
Frame 7 has been transmitted.

Please enter the last Acknowledgement received.
8
```

Assignment 6

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

char* calculate(char* mes, const char* gen)
{
int m = strlen(mes);
int n = strlen(gen);
char* message = (char*) malloc((m+n) * sizeof(char));
strcpy(message, mes);
strcat(message, "00000000");
for (int i = 0; i <= m-n; i++)
{
if(message[i]!='0')
{
for (int j = 0; j < n; j++)
{
if(message[i+j] == gen[j])
message[i+j] = '0';
else
message[i+j] = '1';
}
}
}
return message+m;
}

int main()
{
printf("Kaninika Datta\n");
char gen[] = "1011";
char mes[100];
printf("Enter message: ");
scanf("%s", mes);
char* crc = calculate(mes, gen);
char* mesWithCRC = (char*) malloc((strlen(mes)+strlen(crc)+1) * sizeof(char));
strcpy(mesWithCRC, mes);
strcat(mesWithCRC, crc);
char* rmessage = (char*) malloc((strlen(mes)+strlen(crc)+1) * sizeof(char));
strcpy(rmessage, mesWithCRC);
```

```
srand(time(0));
int modulo = strlen(mesWithCRC);
int errorIndex = rand() % modulo;
// If the original bit was a 0, it is flipped to 1; otherwise, it is flipped to 0.
if (mesWithCRC[errorIndex] == '0')
{
rmessage[errorIndex] = '1';
}
else
{
rmessage[errorIndex] = '0';
}
char* receivedMessage = (char*) malloc((strlen(mes)+1) * sizeof(char));
strncpy(receivedMessage, rmessage, strlen(mes));
receivedMessage[strlen(mes)] = '\0';
char* receivedCRC = calculate(receivedMessage, gen);
if (strcmp(receivedCRC, rmessage+strlen(mes)) == 0)
{
printf("No error detected!\n");
}
else
{
printf("Error detected.\n");
}
free(crc);
free(mesWithCRC);
free(rmessage);
free(receivedMessage);
return 0;
}
```

Output