1. HAMMING CODE

```
#include<stdio.h>

void main() {
   int data[10];
   int dataatrec[10],c,c1,c2,c3,i;
   printf("Enter 4 bits of data one by one\n");
   scanf("%d",&data[0]);
   scanf("%d",&data[1]);
   scanf("%d",&data[2]);
   scanf("%d",&data[4]);
   data[6]=data[0]^data[2]^data[4];
   data[5]=data[0]^data[1]^data[4];
   data[3]=data[0]^data[1]^data[2];
   printf("\nEncoded data is\n");
   for(i=0;i<7;i++)
      printf("%d",data[i]);
   printf("\n\nEnter received data bits one by one\n");
   for(i=0;i<7;i++)
      scanf("%d",&dataatrec[i]);
   c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
   c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
   c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
   c=c3*4+c2*2+c1 ;
   if(c==0) {
      printf("\nNo error while transmission of data\n");
   }
   else {
      printf("\nError on position %d",c);
      printf("\nData sent : ");
      for(i=0;i<7;i++)
      printf("%d",data[i]);
      printf("\nData received : ");
      for(i=0;i<7;i++)
      printf("%d",dataatrec[i]);
      printf("\nCorrect message is\n");
      if(dataatrec[7-c]==0)
         dataatrec[7-c]=1;
      else
         dataatrec[7-c]=0;
      for (i=0;i<7;i++) {
      printf("%d",dataatrec[i]);
      }
   }
}
```

2. CRC-CITT(16-BIT)

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)

char t[28],cs[28],g[]="10001000000100001";
int a,e,c;
void xor(){
   for(c = 1;c < N; c++)
   cs[c] = (( cs[c] == g[c])?'0':'1');
}
```

```
void crc(){
    for(e=0;e<N;e++)
    cs[e]=t[e];
    do{
        if(cs[0]=='1')
        xor();
        for(c=0;c<N-1;c++)
            cs[c]=cs[c+1];
        cs[c]=t[e++];
    }while(e<=a+N-1);
}
int main() {
    printf("\nEnter data : ");
    scanf("%s",t);
    printf("\n----------------------------------------");
    printf("\nGeneratng polynomial : %s",g);
    a=strlen(t);
    for(e=a;e<a+N-1;e++)
        t[e]='0';
    printf("\n----------------------------------------");
    printf("\nModified data is : %s",t);
    printf("\n----------------------------------------");
    crc();
    printf("\n CRC checksum is : %s",cs);
    for(e=a;e<a+N-1;e++)
        t[e]=cs[e-a];
    printf("\n----------------------------------------");
    printf("\nFinalcodeword transmitted is : %s",t);
    printf("\n----------------------------------------");
    printf("\nTest error detection 0(yes) 1(no)? : ");
    scanf("%d",&e);
    if(e==0) {
        do{
            printf("\nEnter the position where error is to be inserted : ");
            scanf("%d",&e);
        }while(e==0 || e>a+N-1);
        t[e-1]=(t[e-1]=='0')?'1':'0';
        printf("\n----------------------------------------");
        printf("\nErroneous data : %s\n",t);
    }
    crc();
    for(e=0;(e<N-1) && (cs[e]!='1');e++);
    if(e<N-1) {
        printf("\n CRC checksum is : %s",cs);
        printf("\nError detected\n\n");
    }
    else
    {
        printf("\n CRC checksum is : %s",cs);
        printf("\nNo error detected\n\n");
    }
    printf("\n----------------------------------------\n");
    return 0;
}
```

3. Socket Programming

Server File

```
#include<stdio.h>
#include<arpa/inet.h>
```

```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv[])
{
    int sockfd, newsockfd, clength;
    struct sockaddr_in serv_addr,cli_addr;
    char t[MAX], str[MAX];
    strcpy(t,"exit");
    sockfd=socket(AF_INET, SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    printf("\nBinded");
    bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
    printf("\nListening...");
    listen(sockfd, 5);
    clength=sizeof(cli_addr);
    newsockfd=accept(sockfd,(struct sockaddr*)&cli_addr, &clength);
    close(sockfd);
    read(newsockfd, &str, MAX);
    printf("\nClient message\n File Name : %s\n", str);
    f1=fopen(str, "r");
    while(fgets(buff, 4096, f1)!=NULL) {
        write(newsockfd, buff,MAX);
        printf("\n");
    }
    fclose(f1);
    printf("\nFile Transferred\n");
    return 0;
}
```

Client

```c
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int main(int arg,char*argv[])
{
    int sockfd,n;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char send[MAX],recvline[MAX],s[MAX],name[MAX];
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr =inet_addr("127.0.0.1");
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
    printf("\nEnter the source file name : \n");
    scanf("%s",send);
    write(sockfd,send,MAX);
    while((n=read(sockfd,recvline,MAX))!=0) {
       printf("%s",recvline);
    }
    close(sockfd);
    return 0;
}
```

Steps to run TCP/IP socket program ->

```
Gcc server.c -o serfile
Gcc client.c -o clifile
```

NS3 Program Changes

1st Program

Steps: -

1. Remove the comments
2. No changes in header file includes
3. Remove everything above int main() except namespace
4. Remove parameters inside main()
5. Remove cmd line till NodeContainer and also the line below simulation time
6. Socket Type to be : ns3::UdpSocketFactory
7. Change number of nodes to 3
8. Modify NetDevice Container like so:

```
NetDeviceContainer dev01;
 dev01= p2p.Install (nodes.Get(0),nodes.Get(1));
 NetDeviceContainer dev12;
 dev12= p2p.Install (nodes.Get(1),nodes.Get(2));
```

8. Remove lines from TrafficControlHelper till the line above IPV4AddressHelper
9. Replicate the lines below IPV4AddressHelper and change interface names to if01 and if02 and add both the devices respectively. Wiz dev01 and dev12
Change the address of dev12 to "10.1.2.0"
10. Populate the routing tables with this line : Ipv4GlobalRoutingHelper::PopulateRoutingTables();
11. Remove payload size in ono and config
12. Set the sink to nodes.get(2)
13. Set rmt of node (1)
14. Remove rmt.settos
15. Modify apps.add line to node (0)
16. Remove everything below "***FlowMonitorStatistics***"
17. Add the following :

```
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter = stats.begin (); iter != stats.end
(); ++iter)
 {
 Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);
 std::cout << "Flow ID: " << iter->first << " Src Addr " << t.sourceAddress << " Dst Addr " <<
t.destinationAddress<< std::endl;
std::cout << "Lost Packets = " << iter->second.lostPackets<< std::endl;
 }
```

18. Remove everything between Destroy and return 0

2nd Program

Steps:
1. Same as 1st program, changes follow
2. Modify the number of nodes to 4
3. Remove the queue and also set data rate to 5mbps of the p2p
4. Make 3 devices -> 0-2, 1-2 and 2-3 to map out the topology
5. Create 3 interfaces mapped out to the 3 devices and change the addresses of 2 and 3 to 10.1.2.0 and 10.1.3.0
6. Populate the global routing table examples/tutorials/third.cc
7. In UDP flow set sink node to 3
8. And make changes as 1st program
9. This time keep the payload
10. In rmt getAddress of interface3.GetAddress(1)
11. Apps.add Get(0)
12. Now replicate this for tcp flow
13. Here change socket type to “ns3::TcpSocketFactory”
14. Rename all variables and the places they are used in
15. Set port_tcp = 9
16. sinkapp_tcp.Start(Seconds (0.5))
17. apps_tcp.add Get(1)
18. apps_tcp.Start(1.5)
19. Then flowmon as usual -> show tx packets -> second.txPackets

---

Third Program

1. Include -> “ns3/flow-monitor-module.h” and “ns3/wifi-module.h“
2. Remove all the comments
3. Remove variables verbose and tracing
4. Remove command line ans verbose check
5. Add double simulationTime = 10;
6. Add std::string socketType = “ns3::UdpSocketFactory”
7. Keep the nWifi check
8. Add these 2 lines: double simulationTime = 10; //seconds
std::string socketType = "ns3::UdpSocketFactory";
9. Modify The check to check for nWifi and nCsma nodes > 250
10. Comment out lines from UdpEchoServer till clientApps.Stop()
11. Remove if(tracing) block
12. Above populating routing table add the //Flow bock until apps.Stop from traffic-control.cc
13. In sinkApp line while creating change nodes.Get to csmaNodes.Get(nCsma)
14. Similarly in rmt change the interfaces.GetAddress to csmaInterfaces.GetAddress(nCsma)
15. Remove rmt.settos
16. apps.add -> wifiStaNodes.Get(nWifi - 1)
17. Add the first 2 lines of FlowMonitor from traffic-control before simulator::run and after simulator:: Stop
18. After run add this line: monitor->CheckForLostPackets ();
19. Then add till “**FlowMonitor Statistics **”
20. Then add the for loop as in first program
21. Steps 16 through 18 should be in between run and destroy