# Network Programming Lab

# Name- Nitesh Gupta

# USN- 1NT19IS101

# Batch- B2

**Part - A**

## Bit stuffing

#include<stdio.h> #include<string.h> #include<stdlib.h> void sender();

void receiver(int \*message,int l2); int main(void)

{

sender();

}

void sender()

{

int i,j,n,count=0,zerocounter=0,zero=0; int msg[50];

int result[50];

printf("Enter the number of bits of the message\n"); scanf("%d",&n);

printf("Enter the bits\n"); for(i=0;i<n;i++)

{

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

}

result[0]=0; result[1]=1; result[2]=1; result[3]=1; result[4]=1; result[5]=1; result[6]=1; result[7]=0; j=8;

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

{

if(msg[i]==0)

{

result[j]=msg[i]; j++;

zero=1;

}

else

{

count=0;

if((count==5)&&(zero==1))

{

}

else

{

}

}

}

result[j]=0; zerocounter++; j++;

result[j]=msg[i]; j++;

count=0;

result[j]=msg[i]; j++;

count++;

result[j++]=0; result[j++]=1; result[j++]=1; result[j++]=1; result[j++]=1; result[j++]=1; result[j++]=1; result[j++]=0;

int l1=16+n+zerocounter; printf("The length is: %d\n",l1); printf("The frame is\n"); for(i=0;i<j;i++)

{

printf("%d",result[i]);

}

receiver(result,l1);

}

void receiver(int \*result,int l2)

{

int i,j,counter,l3; int mesg[100]; l3=l2-8;

j=0;

for(i=8;i<l3;i++)

{

if(result[i]==0)

{

if(counter==5)

{

i++;

mesg[j]=result[i]; j++;

counter=0;

}

else

{

mesg[j]=result[i]; j++;

counter=0;

}

}

else

{

mesg[j]=result[i]; j++;

counter++;

}

}

printf("\nReciever side message is:"); for(i=0;i<j;i++)

{

printf("%d",mesg[i]);

}

}

## Byte stuffing

#include<stdio.h> #include<string.h> void reciever(); char frames[1024]; int main()

{

int n,len,i;

char buffer[256],length[10];

printf("How many frames you want to send: "); bzero(buffer,256);

scanf("%d",&n); for(i=0;i<n;i++)

{

printf("Enter frame\n"); scanf("%s",buffer);

printf("String length of buffer is %d\n",strlen(buffer)); len=strlen(buffer);

len=len+1; sprintf(length,"%d",len); strcat(frames,length); strcat(frames,buffer);

}

for(i=0;frames[i]!='\0';i++)

printf("%c",frames[i]); reciever();

return 0;

}

void reciever()

{

int i=0,framelen,lpvar; char leninchar;

printf("\n\nThis is the reciever\n"); printf("\nData recieved is %s",frames); while(frames[i]!='\0')

{

leninchar=frames[i]; framelen=(int)leninchar-(int)'0';

printf("\nLength of this frame is %d\n",framelen); printf("\nFrame >");

lpvar=i+framelen; i=i+1; while(i<lpvar)

{

printf("%c",frames[i++]);

}

printf("\n");

}

}

1. **CRC**

#include<stdio.h> #include<conio.h> int rem(int,int); void main()

{

int i,j,k,dl,dil; int

data[10],div[5],newdata[15],crc[5],datacrc[15],revdata[15],remd[5]; printf("\n Enter the data length= ");

scanf("%d",&dl);

printf("\n Enter the divisor length= "); scanf("%d",&dil);

printf("\n Enter the data : "); for(i=0;i<dl;i++) scanf("%d",&data[i]);

printf("\n Enter the divisor : "); for(i=0;i<dil;i++) scanf("%d",&div[i]);

printf("\n The new data is : "); for(i=0;i<(dl+dil-1);i++)

{

if(i<dl) newdata[i]=data[i]; else

newdata[i]=0; printf("%d",newdata[i]);

}

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

{

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

{

crc[i]=newdata[i+j]; if(crc[0]==1)

newdata[i+j]=rem(newdata[i+j],div[i]); else

newdata[i+j]=rem(newdata[i+j],0);

}

printf("\n The Crc is : "); for(i=0;i<dil-1;i++)

printf("%d",crc[i]);

}

printf("\n The data to be send is : "); for(i=0;i<(dl+dil-1);i++)

{

if(i<dl) datacrc[i]=data[i]; else

datacrc[i]=crc[i-dl]; printf("%d",datacrc[i]);

}

printf("\n Enter the receiver side data : "); for(i=0;i<(dl+dil-1);i++) scanf("%d",&revdata[i]);

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

{

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

{

remd[i]=revdata[i+j]; if(remd[0]==1)

revdata[i+j]=rem(revdata[i+j],div[i]); else

revdata[i+j]=rem(revdata[i+j],0);

}

printf("\n The reminder is : "); k=0;

for(i=0;i<dil-1;i++)

{

printf("%d",remd[i]); if(remd[i]==0)

k++;

}

}

if(k==dil-1)

printf("\n There is no error found."); else

printf("\n There is error found."); getch();

}

int rem(int x, int y)

{

if(x==y) return 0; else return 1;

}

## Distance vector

#include<stdio.h> struct node

{

unsigned dist[20]; unsigned from[20];

}rt[10];

int main()

{

int dmat[20][20]; int n,i,j,k,count=0;

printf("\nEnter the number of nodes: "); scanf("%d",&n);

printf("\nEnter the cost matrix\n"); for(i=0;i<n;i++)

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

{

scanf("%d",&dmat[i][j]); dmat[i][i]=0; rt[i].dist[j]=dmat[i][j]; rt[i].from[j]=j;

}

do

{

count=0; for(i=0;i<n;i++)

{

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

{

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

{

if(rt[i].dist[j]>dmat[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<n;i++)

{

printf("\n\nState value for router %d is \n",i+1); printf("\nNode \t Via \t Dist. "); for(j=0;j<n;j++)

{

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

}

}

printf("\n\n");

}

1. **Leaky bucket**

#include<stdio.h> #include<stdlib.h>

#define MIN(x,y) (x>y)?y:x int main()

{

int orate,drop=0,cap,x,count=0,inp[10]={0},i=0,nsec,ch; printf("\n enter bucket size : ");

scanf("%d",&cap);

printf("\n enter output rate :"); scanf("%d",&orate);

do{

printf("\n enter number of packets coming at second %d :",i+1); scanf("%d",&inp[i]);

if(inp[i]>cap)

{

printf("Bucket overflow\n"); printf("Packet Discarded\n"); exit(0);

} i++;

printf("\n enter 1 to contiue or 0 to quit ");

scanf("%d",&ch);

}

while(ch); nsec=i;

printf("\n Second \t Recieved \t Sent \t Dropped \tRemained \n"); for(i=0;count || i<nsec;i++)

{

printf(" %d",i+1);

printf(" \t\t%d\t ",inp[i]);

printf(" \t%d\t ",MIN((inp[i]+count),orate)); if((x=inp[i]+count-orate)>0)

{

if(x>cap)

{

count=cap; drop=x-cap;

}

else

{

count=x; drop=0;

}

}

else

{

drop=0; count=0;

}

printf(" \t %d\t %d \n",drop,count);

}

return 0;

}

## Tcp client

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

int main()

{

int sock,bytes\_recv;

struct sockaddr\_in server\_addr;

char recv\_data[1024],send\_data[1024]; struct hostent \*host; host=gethostbyname("127.0.0.1");

if((sock=socket(AF\_INET,SOCK\_STREAM,0))==-1)

{

perror("socket"); exit(1);

}

server\_addr.sin\_family=AF\_INET; server\_addr.sin\_port=htons(6119);

server\_addr.sin\_addr.s\_addr=inet\_addr("127.0.0.1"); if(connect(sock,(struct sockaddr \*)&server\_addr,sizeof(struct

sockaddr))==-1)

{

perror("connect"); exit(1);

}

printf("send Filename to send\n"); gets(send\_data);

if(strcmp(send\_data,"q")!=0) send(sock,send\_data,strlen(send\_data),0);

while((bytes\_recv=recv(sock,recv\_data,1024,0))>0)

{

recv\_data[bytes\_recv]='\0';

//printf("%s\n\n", recv\_data);

//if(strcmp(recv\_data,"q")==0)

// {

// close(sock);

// break;

// }

printf("%s\n", recv\_data);

}

close(sock); return 0;

}

1. **Tcp Server**

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

int main()

{

struct sockaddr\_in server\_addr; struct sockaddr\_in client\_addr; FILE \*fptr;

int sock,connected,bytes\_recv;

char ch,send\_data[1024],recv\_data[1024]; int sin\_size,flag = 0;

if((sock=socket(AF\_INET,SOCK\_STREAM,0))==-1)

{

perror("socket"); exit(1);

}

server\_addr.sin\_family=AF\_INET; server\_addr.sin\_port=htons(6119); server\_addr.sin\_addr.s\_addr=inet\_addr("127.0.0.1"); if(bind(sock,(struct sockaddr \*)&server\_addr, sizeof(struct

sockaddr))==-1)

{

perror("unable to bind"); exit(1);

}

if(listen(sock,5)==-1)

{

perror("lsten"); exit(1);

}

printf("tcp server is waiting for client on port XXXX\n"); sin\_size=sizeof(struct sockaddr\_in); connected=accept(sock,(struct sockaddr \*)&client\_addr,&sin\_size); while(1)

{

bytes\_recv=recv(connected,recv\_data,1024,0); recv\_data[bytes\_recv]='\0';

printf("reciecved data is %s\n\n\n",recv\_data);

fptr=fopen(recv\_data,"r"); if(fptr==NULL)

{

strcpy(send\_data,"FILE"); send(connected,send\_data,strlen(send\_data),0);

}

ch = fgetc(fptr);

while(ch != EOF)//this loop searches the for the current word

{

// fscanf(fptr,"%s",send\_data); send\_data[flag] = ch;

flag++;

ch = fgetc(fptr);

//send(connected,send\_data,strlen(send\_data),0);

}

send(connected,send\_data,strlen(send\_data),0);

//send\_data[0] = 'q';

//strcpy(send\_data,"q");

//send(connected,send\_data,strlen(send\_data),0); close(connected);

break;

}

}

## UDP client

#include <stdio.h> #include <sys/socket.h> #include <netinet/in.h> #include <string.h>

int main(){

int clientSocket, portNum, nBytes; char buffer[1024];

struct sockaddr\_in serverAddr; socklen\_t addr\_size;

/\*Create UDP socket\*/

clientSocket = socket(PF\_INET, SOCK\_DGRAM, 0);

/\*Configure settings in address struct\*/ serverAddr.sin\_family = AF\_INET; serverAddr.sin\_port = htons(8893);

serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

/\*Initialize size variable to be used later on\*/ addr\_size = sizeof serverAddr;

while(1){

printf("Type a sentence to send to server:\n"); fgets(buffer,1024,stdin);

printf("You typed: %s",buffer); nBytes = strlen(buffer) + 1;

/\*Send message to server\*/ sendto(clientSocket,buffer,nBytes,0,(struct sockaddr

\*)&serverAddr,addr\_size);

/\*Receive message from server\*/

nBytes = recvfrom(clientSocket,buffer,1024,0,NULL,

NULL);

printf("Received from server: %s\n",buffer);

}

return 0;

}

1. **UDP server**

#include <stdio.h> #include <sys/socket.h> #include <netinet/in.h> #include <string.h> #include <stdlib.h>

int main(){

int udpSocket, nBytes; char buffer[1024];

struct sockaddr\_in serverAddr, clientAddr; struct sockaddr\_storage serverStorage; socklen\_t addr\_size, client\_addr\_size;

int i;

/\*Create UDP socket\*/

udpSocket = socket(PF\_INET, SOCK\_DGRAM, 0);

/\*Configure settings in address struct\*/ serverAddr.sin\_family = AF\_INET; serverAddr.sin\_port = htons(8893);

serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

/\*Bind socket with address struct\*/

bind(udpSocket, (struct sockaddr \*) &serverAddr, sizeof(serverAddr));

/\*Initialize size variable to be used later on\*/ addr\_size = sizeof serverStorage;

while(1){

/\* Try to receive any incoming UDP datagram. Address and port of

\* requesting client will be stored on serverStorage variable \*/ nBytes = recvfrom(udpSocket,buffer,1024,0,(struct sockaddr

\*)&serverStorage, &addr\_size);

/\*Convert message received to uppercase\*/ for(i=0;i<nBytes-1;i++)

buffer[i] = toupper(buffer[i]);

/\*Send uppercase message back to client, using serverStorage as the address\*/

sendto(udpSocket,buffer,nBytes,0,(struct sockaddr

\*)&serverStorage,addr\_size);

}

return 0;

}

# Part - B

## Part b 1

/\*

* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;

\*

* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.

\*

* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111- 1307 USA

\*/

#include "ns3/netanim-module.h" #include "ns3/core-module.h" #include "ns3/network-module.h" #include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h" #include "ns3/applications-module.h" using namespace ns3;

int

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

{

Time::SetResolution (Time::NS); NodeContainer nodes; nodes.Create (2); PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps")); pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms")); NetDeviceContainer devices;

devices = pointToPoint.Install (nodes); InternetStackHelper stack; stack.Install (nodes); Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices); UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9); echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); ApplicationContainer clientApps = echoClient.Install (nodes.Get (0)); clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0)); AnimationInterface anim ("first.xml"); Simulator::Run ();

Simulator::Destroy (); return 0;

}

## Part b 2

/\* -\*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -\*- \*/

/\*

* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;

\*

* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.

\*

* You should have received a copy of the GNU General Public License
  + along with this program; if not, write to the Free Software
  + Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111- 1307 USA

\*/

// Network topology

//

// n0 n1 n2 n3

// | | | |

// =================

// LAN

//

// - UDP flows from n0 to n1 and back

// - DropTail queues

// - Tracing of queues and packet receptions to file "udp-echo.tr" #include <fstream>

#include "ns3/core-module.h" #include "ns3/csma-module.h" #include "ns3/applications-module.h" #include "ns3/internet-module.h" #include "ns3/netanim-module.h" using namespace ns3;

int

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

{

Address serverAddress; NodeContainer n;

n.Create (4); InternetStackHelper internet;

internet.Install (n); CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));

csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2))); csma.SetDeviceAttribute ("Mtu", UintegerValue (1400)); NetDeviceContainer d = csma.Install (n);

Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (d); serverAddress = Address(i.GetAddress (1));

uint16\_t port = 9; // well-known echo port number UdpEchoServerHelper server (port);

ApplicationContainer apps = server.Install (n.Get (1)); apps.Start (Seconds (1.0));

apps.Stop (Seconds (10.0)); uint32\_t packetSize = 1024; uint32\_t maxPacketCount = 1;

Time interPacketInterval = Seconds (1.); UdpEchoClientHelper client (serverAddress, port);

client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount)); client.SetAttribute ("Interval", TimeValue (interPacketInterval)); client.SetAttribute ("PacketSize", UintegerValue (packetSize)); apps = client.Install (n.Get (0));

apps.Start (Seconds (2.0));

apps.Stop (Seconds (10.0));

#if 0

client.SetFill (apps.Get (0), "Hello World"); client.SetFill (apps.Get (0), 0xa5, 1024);

uint8\_t fill[] = { 0, 1, 2, 3, 4, 5, 6};

client.SetFill (apps.Get (0), fill, sizeof(fill), 1024); #endif

AnimationInterface anim ("second.xml"); Simulator::Run ();

Simulator::Destroy ();

}

## Part b 3

#include "ns3/core-module.h"

#include "ns3/point-to-point-module.h" #include "ns3/network-module.h" #include "ns3/applications-module.h" #include "ns3/wifi-module.h"

#include "ns3/mobility-module.h" #include "ns3/csma-module.h" #include "ns3/internet-module.h" #include "ns3/netanim-module.h"

// Default Network Topology

//

// 10.1.1.0

// n0 -------------- n1 n2 n3 n4

// point-to-point | | | |

// ================

// LAN 10.1.2.0

using namespace ns3; int

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

{

uint32\_t nCsma = 3; NodeContainer p2pNodes; p2pNodes.Create (2);

NodeContainer csmaNodes; csmaNodes.Add (p2pNodes.Get (1)); csmaNodes.Create (nCsma); PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps")); pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms")); NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes); CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps")); csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560))); NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack; stack.Install (p2pNodes.Get (0)); stack.Install (csmaNodes); Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer p2pInterfaces; p2pInterfaces = address.Assign (p2pDevices); address.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer csmaInterfaces; csmaInterfaces = address.Assign (csmaDevices); UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9); echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0)); Ipv4GlobalRoutingHelper::PopulateRoutingTables (); pointToPoint.EnablePcapAll ("second");

csma.EnablePcap ("second", csmaDevices.Get (1), true); AnimationInterface anim ("third.xml");

Simulator::Run (); Simulator::Destroy (); return 0;

}

## Part b 4

#include <string> #include <fstream>

#include "ns3/core-module.h"

#include "ns3/point-to-point-module.h" #include "ns3/internet-module.h"

#include "ns3/applications-module.h"

#include "ns3/network-module.h" #include "ns3/packet-sink.h" #include "ns3/netanim-module.h" using namespace ns3;

int

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

{

uint32\_t maxBytes = 0; NodeContainer nodes; nodes.Create (2); PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps")); pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));

NetDeviceContainer devices;

devices = pointToPoint.Install (nodes); InternetStackHelper internet; internet.Install (nodes); Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (devices); uint16\_t port = 9; // well-known echo port number BulkSendHelper source ("ns3::TcpSocketFactory", InetSocketAddress (i.GetAddress (1), port));

source.SetAttribute ("MaxBytes", UintegerValue (maxBytes)); ApplicationContainer sourceApps = source.Install (nodes.Get (0)); sourceApps.Start (Seconds (0.0));

sourceApps.Stop (Seconds (10.0)); PacketSinkHelper sink ("ns3::TcpSocketFactory",

InetSocketAddress (Ipv4Address::GetAny (), port)); ApplicationContainer sinkApps = sink.Install (nodes.Get (1)); sinkApps.Start (Seconds (0.0));

sinkApps.Stop (Seconds (10.0));

Simulator::Stop (Seconds (10.0)); AnimationInterface anim ("fourth.xml"); anim.EnablePacketMetadata(true); Simulator::Run ();

Simulator::Destroy ();

}