RAMAIAH
Institute of Technology

Data Sheet

MARKS : 10

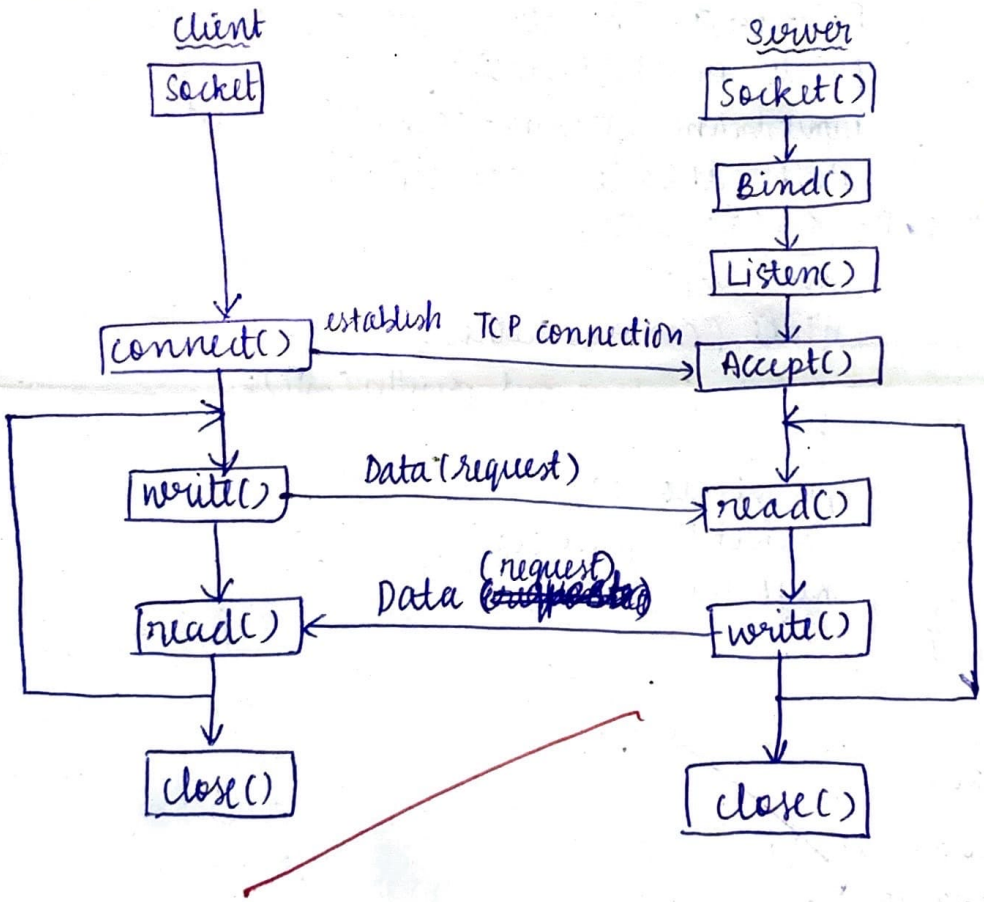
WEEK - 2

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , B
Subject :	DCN LABORATORY	Subject Code:	CSL47

Problem Statement

- To use TCP/UDP sockets and write a client server program in which the client sends the filename in request message and server sends back the contents of the requested file if present.

STATE DIAGRAM



RAMAIAH
Institute of Technology

Data Sheet

MARKS : 10

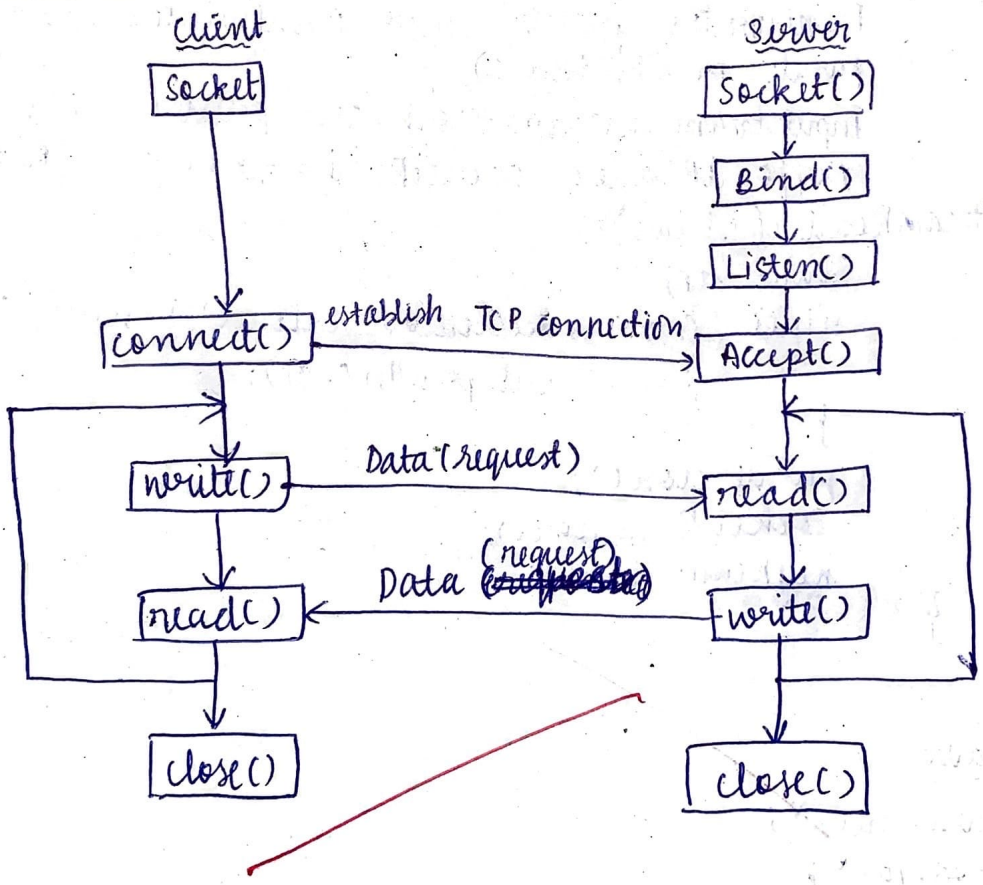
WEEK - 2

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , B
Subject :	DCN LABORATORY	Subject Code:	CSL47

6A Problem Statement

- To use TCP/UDP sockets and write a client server program in which the client sends the filename in request message and server sends back the contents of the requested file if present.

STATE DIAGRAM



IP address: 172.1.6.84

Client.java

```
import java.net.*;
import java.io.*;
public class Client {
    public static void main(String[] args) throws
Exception {
        Socket sock = new Socket("172.1.6.84", 4000);
        System.out.println("Enter the filename: ");
        BufferedReader keyRead = BufferedReader(new
InputStreamReader(System.in));
        String fname = keyRead.readLine();
        OutputStream ostream = sock.getOutputStream();
        PrintWriter pwrite = new PrintWriter(ostream, true);
        pwrite.println(fname);
        InputStream istream = sock.getInputStream();
        BufferedReader socketRead = new BufferedReader(
new InputStreamReader(istream));
        String str;
        while ((str = socketRead.readLine()) != null){
            System.out.println(str);
        }
        pwrite.close();
        socketRead.close();
        keyRead.close();
    }
}
```

Server.java

```
import java.net.*;
import java.io.*;
public class Server{
    public static void main(String[] args){
        ServerSocket servsock = new ServerSocket(4000);
```

```
            System.out.println ("Server is ready for connection");
            Socket sock = sersock.accept();
            System.out.println ("Connection is successful and
waiting for chatting!");
            InputStream istream = sock.getInputStream();
            BufferedReader fileRead = new BufferedReader(
new InputStreamReader(istream));
            String fname = fileRead.readLine();
            OutputStream ostream = sock.getOutputStream();
            PrintWriter pwrite = new PrintWriter(ostream, true);
            String str;
             while ((str = ContentRead.readline()) != null) {
                    pwrite.println(str);
            }
            sock.close();
            sersock.close();
            pwrite.close();
            fileRead.close();
            ContentRead.close();
        }
    }
```

OUTPUT:

```
$ java Server
  Server ready for connection
  Connection is successful and waiting for chatting!

$ java Client
  Enter the filename:
  aloha.txt
  Hello this is DCN Lab.
```

RAMAIAH
Institute of Technology

Data Sheet

MARKS: 10

WEEK-3

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV . B
Subject :	DCN Lab	Subject Code:	CSL407

Trace packets using Wireshark for HTTP and answer the following questions:

1. List up to 10 different protocols that appear in the protocol column in the unfiltered packet-listing window

— i) HTTP
ii) DNS
iii) NBNS
iv) TCP
v) UDP
vi) SSDP
vii) QUIC
viii) MDNS
ix) ARP
x) DHCP

2. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received?

— Time since request: 0.225031648 seconds.

3. What is the Internet address of gaia.cs.umass.edu? What is the Internet address of your computer?

— IP address of gaia.cs.umass.edu:
~~128.119.245.12~~ 185.125.190.98

IP address of my computer:
172.1.6.84

4. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

— Both the browser and server are running HTTP version 1.1.

5. What languages does your browser indicate that it can accept from server? (request)

— Accept-Language: en-GB, en-US;
q=0.9, en;
q=0.8\r\n

6. What is the status code returned from the server to your browser? (response)

— ~~200~~ 540 HTTP/1.1 200 OK

Status codes: 200 OK, 301 Moved Permenantly.
204 No content

7. When was the HTML file that you are retrieving last modified at the server? (response)

— Last-Modified: Fri, 17 May 2024 05:59:02 GMT\r\n

8. How many bytes of content are being returned to your browser? (response)

— 128 bytes

9. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

— If-Modified-Since: Tue, 07 May 2024 05:59:01 GMT\r\n

10. Inspect contents of response. Did server explicitly return contents of file? How can you tell? (response)

— Yes the contents are returned since the bytes of file are returned in ~~200~~ the response with 200 OK status code.
[Here 371 bytes]

11. Now inspect contents of second HTTP GET request from your browser to server. Do you see an "IF-MODIFIED-SINCE" line?

— NO there is no 'if modified since' line in the second GET request.

12. What is the HTTP status code and phrase returned from server in response to this second HTTP GET? Did it explicitly return contents of file? Explain.

— Status code: 204 No Content [HTTP/1.1 204 No content]
It did not return contents of the file since no bytes are specified in the response protocols.

13. How many HTTP GET request messages were sent by browser?
2 request messages
615 GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
153 GET /HTTP/1.1

14. How many data-containing TCP segments were needed to carry the single HTTP response?

15. What is the status code and phrase associated with the response to HTTP GET request?
304 Not Modified.
200 OK

15. How many HTTP GET request messages were sent by your browser? To which Internet address were these GET requests sent?

— 2 requests

614 GET /wireshark-labs/HTTP-wireshark-file4.html — 128.119.245.12

153 GET /HTTP/1.1 — 91.189.91.48

16. Can you tell whether your browser downloaded the two images serially or whether they were downloaded from the two websites in parallel?

— Serially (two different requests)

GET /pearson.png HTTP/1.1

GET/8E_cover_small.jpg HTTP/1.1

17. What is the server's response in initial HTTP GET message?

— 771 HTTP/1.1 401 unauthorized

18. What your browser sends the HTTP GET message for second time? What new field is included?

— 153 GET /HTTP/1.1

Referer field

24/5/24

RAMAIAH
Institute of Technology

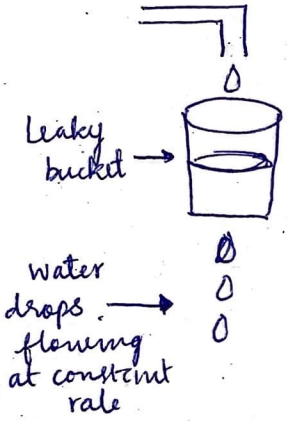
Data Sheet

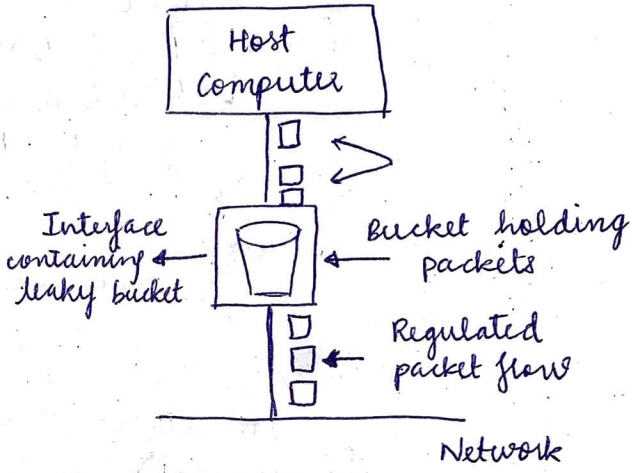
MARKS: 10

WEEK-4

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , B
Subject :	DCN LAB	Subject Code:	CSL47

1) Write a program for congestion control using Leaky Bucket.





Leaky Bucket Algorithm

CODE:

```
import java.util.Scanner;
import java.lang.*;
public class LeakyBucket{
    public static void main(String[] args){
        int i;
        int a[] = new int[20];
        int buckRem=0, buckCap=4, rate=3, sent, recv;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter number of packets");
        int n = in.nextInt();
        System.out.println("Enter the packets:");
        for(i=1; i<=n; i++)
            a[i]=in.nextInt();
        System.out.println("Clock\t packet size\t accept
```

```
lt sent lt remaining ");
for(int i=1; i<=n; i++){
    if(a[i] != 0){
        if (buckRem +a[i] > buck(ap){
            recv = -1; }
        else{
            recv = a[i];
            buckRem += a[i];
        }
    }
    else
        recv = 0;
    if ( buckRem != 0){
        if (buckRem < rate){
            sent = buckRem;
            buckRem = 0;
        }
        else{ sent = rate; buckRem -= rate;
        }
    }
    else
        sent = 0;
    if(recv == -1){
        system.out.println("i + "\t\t" + a[i] + "\t
DROPPED \t" + sent + "\t " + buckRem);
    }
    else{
        system.out.println(i + "\t\t " + a[i] + "\t\t"
+ recv + "\t" + sent + "\t" + buckRem);
    }
}
}
```

OUTPUT:

```
Enter number of packets
10
Enter packets
3
2
```

OUTPUT

4
4
1
2
4
6
3
2

Clock	packet size	accept	sent	remaining
1	3	3	3	0
2	2	2	2	0
3	4	4	3	1
4	4	DROPPED	1	0
5	1	1	1	0
6	2	2	2	0
7	4	4	3	1
8	6	DROPPED	1	0
9	3	3	3	0
10	2	2	2	0

RESOURCE

1. Locate the DNS query and response messages. Are they sent over UDP or TCP?
   - It is sent over UDP (User Datagram Protocol).
2. What is the destination port for the DNS quer

RAMAIAH
Institute of Technology

Data Sheet

MARKS :

WEEK - 5

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , 'B'
Subject :	DCN LAB	Subject Code:	CSL47

WIRESHARK :

1. Locate the DNS query and response messages. Are they sent over UDP or TCP?
   — It is sent over UDP (User Datagram Protocol)

2. What is the destination port for the DNS query message? What is the source port of DNS response message?
   — Dest Port : 53 (for query)
   Src Port : 53 (for response)

3. To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server? Are they same?
   — IP address : 172.1.6.84
   It is the same as my system IP address.

4. Examine the DNS query message. What "Type" of DNS query is it? Does it contain any answers?
   — Type : HTTPS (HTTPS Specific Service Endpoints)
   It does not contain answers.

5. Examine the DNS response message. How many "answer" are provided? What does each answers contain?
   — 2 answers.

Answers:

www.ietf.org : type A , class IN

- Name: www.ietf.org
  [Name Length : 12]
  [Label count : 3]
  Type : A (Host Address) (1)
  Class : IN

www.itef.org : type A , class IN

Name : www.ietf.org
Type : A (Host address)
Class : IN
Time to live : 108
Data Length : 4
Address : 104.16.45.99

5. This webpage contains images. Before retrieving each image, does your host issue new DNS queries?

~~No issues~~ — No new DNS query.

7-10. Examine DNS response message. What MIT name servers does the response message provided? Does this response message also provide IP addresses of the MIT name servers?

—— ~~asia1~~ Name servers : asia1, use2, ns1-173, use5, ns1-37, eur5, asia2, usw2

RAMAIAH
Institute of Technology

Data Sheet

MARKS : 10

WEEK-6

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , B
Subject :	DCN Lab	Subject Code:	CSL47

1B, 5B

* Design and simulate a wired network with duplex links between 'n' nodes with CDR over UDP. Set the queue size vary the bandwidth and find the number of packets dropped.

[Step 1: go to ns-allinone-3.39 → ns-3.39 → examples → traffic-control → traffic-control.cc]

[Step 2: Copy the cc file → go to ns-3.39 → scratch folder
Don't make any change to original file in the example folder
Rename traffic-control.cc to desired name]

note: There should be no .cc ~~folder~~ files in folder except the desired one. (scratch)

[Step 3: Do the necessary change in the scratch folder]

IFRAH NAAZ
1MS22CS064

TRAFFIC CONTROLLER

```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/traffic-control-module.h"
// This simple example shows how to use TrafficControlHelper to install a
// QueueDisc on a device.
// The default QueueDisc is a pfifo_fast with a capacity of 1000 packets (as in Linux). However, in this example,
// we install a RedQueueDisc with a capacity of 10000 packets.
//
// Network topology
//      10.1.1.0
// n0 ---------- n1
//    point-to-point
// The output will consist of all the traced changes in the length of the RED internal queue and in the length of the
netdevice queue:
//
//    DevicePacketsInQueue 0 to 1
//    TcPacketsInQueue 7 to 8
//    TcPacketsInQueue 8 to 9
//    DevicePacketsInQueue 1 to 0
//    TcPacketsInQueue 9 to 8
//
// plus some statistics collected at the network layer (by the flow monitor) and the application layer. Finally,
// the number of packets dropped by the queuing discipline, the number of packets dropped by the netdevice and
// the number of packets requeued by the queuing discipline are reported.
//
// If the size of the DropTail queue of the netdevice were increased from 1 to a large number (e.g. 1000), one would observe
// that the number of dropped packets goes to zero, but the latency grows in an uncontrolled manner. This is the
// so-called bufferbloat problem, and illustrates the importance of having a small device queue, so that the standing
// queues build in the traffic control layer where they can be managed by advanced queue discs rather than in the device
layer.

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("TrafficControlExample");
/**
 * Number of packets in TX queue trace.
 *
 * \param oldValue Old velue.
 * \param newValue New value.
 */
void
TcPacketsInQueueTrace(uint32_t oldValue, uint32_t newValue)
{
   std::cout << "TcPacketsInQueue " << oldValue << " to " << newValue << std::endl;
}
/**
 * Packets in the device queue trace.
```

This code is for point to point connection between two nodes.

```
 *
 * \param oldValue Old velue.
 * \param newValue New value.
 */
void
DevicePacketsInQueueTrace(uint32_t oldValue, uint32_t newValue)
{
  std::cout << "DevicePacketsInQueue " << oldValue << " to " << newValue << std::endl;
}
/**
 * TC Soujoun time trace.
 *
 * \param sojournTime The soujourn time.
 */
void
SojournTimeTrace(Time sojournTime)
{
  std::cout << "Sojourn time " << sojournTime.ToDouble(Time::MS) << "ms" << std::endl;
}
int
main(int argc, char* argv[])
{
  double simulationTime = 10; // seconds // time taken to run the simulation
  std::string transportProt = "Tcp"; //we can choose either TCP or UDP for the transport protocol
  std::string socketType;

  CommandLine cmd(__FILE__);
  cmd.AddValue("transportProt", "Transport protocol to use: Tcp, Udp", transportProt);
  cmd.Parse(argc, argv);

  if (transportProt == "Tcp")
  {
    socketType = "ns3::TcpSocketFactory";    //If TCP is selected
  }
  else
  {
    socketType = "ns3::UdpSocketFactory";    //If UDP is selected
  }

  NodeContainer nodes;
  nodes.Create(2);//The value is based on number of nodes in the network.
                  // (can be 2,3,4,5,.....)
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute("DataRate", StringValue("10Mbps"));  // Setting the attributes of devices
  pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));       // and channel of network
  pointToPoint.SetQueue("ns3::DropTailQueue", "MaxSize", StringValue("1p"));

  NetDeviceContainer devices;          // We create device objects for each point-to-point
  devices = pointToPoint.Install(nodes); // connection between two nodes.

  InternetStackHelper stack;           // If (0)——(1)——(2) is the network, there we will
                                       // be two devices created [b/w 0 and 1, b/w 1 and 2]
```

```
stack.Install(nodes);

TrafficControlHelper tch;
tch.SetRootQueueDisc("ns3::RedQueueDisc");
QueueDiscContainer qdiscs = tch.Install(devices);  // to manage congestion
Ptr<QueueDisc> q = qdiscs.Get(1);
q->TraceConnectWithoutContext("PacketsInQueue", MakeCallback(&TcPacketsInQueueTrace));
Config::ConnectWithoutContext(
  "/NodeList/1/$ns3::TrafficControlLayer/RootQueueDiscList/0/SojournTime",
  MakeCallback(&SojournTimeTrace));

Ptr<NetDevice> nd = devices.Get(1);
Ptr<PointToPointNetDevice> ptpnd = DynamicCast<PointToPointNetDevice>(nd);
Ptr<Queue<Packet>> queue = ptpnd->GetQueue();
queue->TraceConnectWithoutContext("PacketsInQueue", MakeCallback(&DevicePacketsInQueueTrace));
// Assign IP address to the devices with parameters (IP address, mask address)
Ipv4AddressHelper address;                  // If multiple devices, we assign different IP
address.SetBase("10.1.1.0", "255.255.255.0"); // addresses like 10.1.1.0, 10.1.2.0, for each each
                                            // link
Ipv4InterfaceContainer interfaces = address.Assign(devices);
// We create interface for each device [node to node connection]
// Flow
uint16_t port = 7;
Address localAddress(InetSocketAddress(Ipv4Address::GetAny(), port));
PacketSinkHelper packetSinkHelper(socketType, localAddress);
ApplicationContainer sinkApp = packetSinkHelper.Install(nodes.Get(0)); // Node 0 is set as receiver of
                                                                       // packets
sinkApp.Start(Seconds(0.0));
sinkApp.Stop(Seconds(simulationTime + 0.1));

uint32_t payloadSize = 1448;
Config::SetDefault("ns3::TcpSocket::SegmentSize", UintegerValue(payloadSize));

OnOffHelper onoff(socketType, Ipv4Address::GetAny());  // Used to generate traffic according to on &
onoff.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));  // off pattern
onoff.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute("PacketSize", UintegerValue(payloadSize));
onoff.SetAttribute("DataRate", StringValue("50Mbps")); // bit/s
ApplicationContainer apps;

InetSocketAddress rmt(interfaces.GetAddress(0), port); // it creates an interface by getting IP address
rmt.SetTos(0xb8);                   // of device connected to receiver. Here it is node
AddressValue remoteAddress(rmt);    // only one device interface
onoff.SetAttribute("Remote", remoteAddress);//
apps.Add(onoff.Install(nodes.Get(1))); // Node 1 is the sender of the packets
apps.Start(Seconds(1.0));           // Consider 0 — 1 — 2 network with Node0
apps.Stop(Seconds(simulationTime + 0.1));  // as sender and Node 2 as receiver. So
                                    // we mention the interface of device connected
FlowMonitorHelper flowmon;          // to receiver with IP address associated
Ptr<FlowMonitor> monitor = flowmon.InstallAll();  // to Node 2. i.e. node intermediate to
                                                  // receiver.
```

```
Simulator::Stop(Seconds(simulationTime + 5));
Simulator::Run();
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>(flowmon.GetClassifier());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats();
std::cout << std::endl << "*** Flow monitor statistics ***" << std::endl;
std::cout << "  Tx Packets/Bytes:   " << stats[1].txPackets << " / " << stats[1].txBytes
      << std::endl;
std::cout << "  Offered Load: "
      << stats[1].txBytes * 8.0 /
          (stats[1].timeLastTxPacket.GetSeconds() -
           stats[1].timeFirstTxPacket.GetSeconds()) /
          1000000
      << " Mbps" << std::endl;
std::cout << "  Rx Packets/Bytes:   " << stats[1].rxPackets << " / " << stats[1].rxBytes
      << std::endl;
uint32_t packetsDroppedByQueueDisc = 0;
uint64_t bytesDroppedByQueueDisc = 0;
if (stats[1].packetsDropped.size() > Ipv4FlowProbe::DROP_QUEUE_DISC)
{
  packetsDroppedByQueueDisc = stats[1].packetsDropped[Ipv4FlowProbe::DROP_QUEUE_DISC];
  bytesDroppedByQueueDisc = stats[1].bytesDropped[Ipv4FlowProbe::DROP_QUEUE_DISC];
}
std::cout << "  Packets/Bytes Dropped by Queue Disc:   " << packetsDroppedByQueueDisc << " / "
      << bytesDroppedByQueueDisc << std::endl;
uint32_t packetsDroppedByNetDevice = 0;
uint64_t bytesDroppedByNetDevice = 0;
if (stats[1].packetsDropped.size() > Ipv4FlowProbe::DROP_QUEUE)
{
  packetsDroppedByNetDevice = stats[1].packetsDropped[Ipv4FlowProbe::DROP_QUEUE];
  bytesDroppedByNetDevice = stats[1].bytesDropped[Ipv4FlowProbe::DROP_QUEUE];
}
std::cout << "  Packets/Bytes Dropped by NetDevice:   " << packetsDroppedByNetDevice << " / "
      << bytesDroppedByNetDevice << std::endl;
std::cout << "  Throughput: "
      << stats[1].rxBytes * 8.0 /
          (stats[1].timeLastRxPacket.GetSeconds() -
           stats[1].timeFirstRxPacket.GetSeconds()) /
          1000000
      << " Mbps" << std::endl;
std::cout << "  Mean delay:   " << stats[1].delaySum.GetSeconds() / stats[1].rxPackets
      << std::endl;
std::cout << "  Mean jitter:   " << stats[1].jitterSum.GetSeconds() / (stats[1].rxPackets - 1)
      << std::endl;
auto dscpVec = classifier->GetDscpCounts(1);
for (auto p : dscpVec)
{
  std::cout << "  DSCP value:  0x" << std::hex << static_cast<uint32_t>(p.first) << std::dec
       << "  count:   " << p.second << std::endl;
}
Simulator::Destroy();
std::cout << std::endl << "*** Application statistics ***" << std::endl;
```

The contents depend on the question asked (regarding dropping packets)

RAMAIAH
Institute of Technology

Data Sheet

MARKS : 10

WEEK-7

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , 'B'
Subject :	DCN LAB	Subject Code:	CSL47

1. Point-to-Point Connection for 3-node Topology

CODE: [Changes done in the code]

```
NodeContainer nodes;
nodes.Create(3); // For three node point-to-point connection
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
pointToPoint.SetQueue("ns3::DropTailQueue", "MaxSize",
StringValue("1p"));
NetDeviceContainer devices01;
devices01 = pointToPoint.Install(nodes.Get(0), nodes.Get(1));
NetDeviceContainer devices12;
devices12 = pointToPoint.Install(nodes.Get(1), nodes.Get(2));

InternetStackHelper Stack;
Stack.Install(nodes);

IpV4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
IpV4InterfaceContainer interfaces01 = address.Assign(devices01);

IpV4 address.SetBase("10.1.2.0", "256.255.255.0");
IpV4InterfaceContainer interfaces12 = address.Assign(devices12);

Ipv4GlobalRoutingHelper::PopulateRoutingTables();
// examples → tutorial → third.cc → line 173
```

```
// Flow
ApplicationContainer sinkApp = packetSinkHelper.Install
  (nodes.Get(2)); // Set the receiver node

InetSocketAddress rmt (interfaces12.GetAddress(1), port); // remote interfa
apps.Add(onoff.Install(nodes.Get(0))); // sender node

// Remove the last line
// std::cout << q->GetStats(x< std::endl;
```

OUTPUT: [In Terminal]

```
*** Flow monitor statistics ***
  Tx Packets/Bytes: 1222/1830108
  Offered Load: 10.0141 Mbps
  Rx Packets/Bytes: 1209/1810608
  Packets/Bytes Dropped by Queue Disc: 6/9000
  Packets/Bytes Dropped by NetDevice: 0/0
  Throughput: 9.92211 Mbps
  Mean delay: 0.01789
  Mean jitter: 0.00111849
  DSCP Value: 0x2e count: 1222
```

Note: To run the simulation:

```
./ns3 run filename --vis
```

PB. 6B

2. Design and simulate a 4 node point-to-point network and connect the link as follows:

n0 → n1 , n1 → n2, n3 → n1

Apply UDP agent between n0 → n2 and TCP agent between n3 → n2.

Apply relevant applications over TCP and UDP agents by changing the parameters. Determine the number of packets sent by TCP and UDP.

***

```
PointToPointHelper p2p1;
p2p1.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
p2p1.SetChannelAttribute ("Delay", StringValue ("1ms"));
// UDP interface
  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  NetDeviceContainer devices;
  devices = p2p1.Install (nodes.Get(0), nodes.Get(1));
  Ipv4InterfaceContainer interfaces = address.Assign (devices);
  devices = p2p1.Install (nodes.Get(1), nodes.Get(2));
  address.SetBase ("10.1.2.0", "255.255.255.0");
  interfaces = address.Assign(devices);
// TCP interfaces
  Ipv4AddressHelper address1;
  address1.SetBase("10.1.3.0", "255.255.255.0");
  NetDeviceContainer devices1;
  devices1 = p2p1.Install (nodes.Get(3), nodes.Get(1));
  Ipv4InterfaceContainer interfaces1 = address1.Assign(devices1);
  devices1 = p2p1.Install (nodes.Get(1), nodes.Get(2));
  address1.SetBase("10.1.4.0", "255.255.255.0");
  interfaces1 = address1.Assign(devices1);
  Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

```
// Configure UDP
  uint32_t payloadSize = 1448;
  OnOffHelper onoff ("ns3::UdpSocketFactory", Ipv4Address::GetAny());
  onoff.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable
  [Constant=1]"));
  onoff.SetAttribute ("OffTime", StringValue("ns3::ConstantRandomVariable
  [Constant=0]"));
  onoff.SetAttribute ("PacketSize", StringValue(payloadSize));
  onoff.SetAttribute ("DataRate", StringValue ("50 Mbps"));
  uint16_t port = 7;
  Address localAddress (InetSocketAddress (Ipv4Address::GetAny(), port));
  PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory", localAddress);
  ApplicationContainer sinkApp = packetSinkHelper.Install (nodes.Get(2));
  sinkApp.Start(Seconds(0.0));
  sinkApp.Stop(Seconds(5.0));
  ApplicationContainer apps;
  AddressValue remoteAddress(InetSocketAddress (interfaces.GetAddress(1), port));
  onoff.SetAttribute ("Remote", remoteAddress);
  apps.Add(onoff.Install(nodes.Get(0)));
  apps.Start(Seconds(1.0));
  apps.Stop(Seconds(5.0));
// Configure TCP
  OnOffHelper onoff1("ns3::TcpSocketFactory", Ipv4Address::GetAny());
  onoff1.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable
  [Constant=1]"));
  onoff1.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable
  [Constant=0]"));
  onoff1.SetAttribute("PacketSize", UIntegerValue(payloadSize));
  onoff1.SetAttribute("DataRate", StringValue("50Mbps"));
  uint16_t port = 9;
  Address localAddress1(InetSocketAddress(Ipv4Address::GetAny())
  PacketSinkHelper packetSinkHelper1("ns3::TcpSocketFactory", localAddress1);
  ApplicationContainer sinkApp1 = packetSinkHelper1.Install(nodes.Get(2));
  ApplicationContainer apps1;
  AddressValue remoteAddress1(InetSocketAddress(interfaces1.GetAddress(1),
  onoff1.SetAttribute("Remote", remoteAddress1);                    port));
  apps1.Add(onoff.Install(nodes.Get(3)));
```

RAMAIAH
Institute of Technology

Data Sheet

MARKS :

WEEK-7

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , 'B'
Subject :	DCN Lab	Subject Code:	CSL47

OUTPUT :

Flow ID : 1 Src Addr 10.1.1.1 Dst Addr 10.1.2.2
Tx Packets = 17265
Rx Packets = 4269
lost Packets Packets = 7396
Throughput : 4874.1 Kbps
Flow ID : 2 Src Addr 10.1.3.1 Dst Addr 10.1.4.2
Tx Packets = 3
Rx Packets = 3
lost Packets Packets = 0
Throughput : 0.355688 Kbps
Flow ID : 3 Src Addr 10.1.4.2 Dst Addr 10.1.3.1
Tx Packets = 2
Rx Packets = 2
lost Packets Packets = 0
Throughput : 0.208185 Kbps

RAMAIAH
Institute of Technology

Data Sheet

MARKS: 10

WEEK-8

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV, B
Subject :	DCN Lab	Subject Code:	CSL47

37, 8A Trace packets using Wireshark for IP, ICMP and Answer the following:

1) Select the first ICMP Echo request message sent by your computer and expand the internet protocol part of packet. What is the IP address of your computer?

— Src: 172.1.6.51 (IP Address of my computer)

2) Within the IP packet header, what is the value in the upper layer protocol field?

— Protocol: ICMP (1)

3) How many bytes are in the IP header? How many bytes are in the payload of IP datagram? Explain how you determined the number of payload bytes.

— Header Length : 20 bytes (5)
Payload Length : 528 bytes
Reason: Total length = Header Length + Payload Length
Payload Length = Total - header length
= 548 - 20
= 528 bytes

4) Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented?

— Yes it is ~~not~~ fragmented. This is because the Don't fragment bit is not set (0).

5) Which fields in the IP datagram always change from one datagram to the next within this series of ICMP messages sent by your computer?

— Header Length, Identification, Header checksum, Total Length, Time to live, Flags, source address, Destination address

6) Which fields stay constant?

— Differentiated Service Field, Protocol, Version

7) Find the first ICMP Echo Request message that was sent by your computer after you changed the packet size to be 2000. Has the message been fragmented across more than one IP datagram?

— Yes, it has been fragmented as the Don't fragment bit is not set.

[2 IPv4 Fragments (2008 bytes): #8(1480), #9(528)]

8) Write the first fragment of the fragmented IP datagram. What information in IP header indicates whether this is the first fragment versus latter fragment? How long is this IP datagram?

— *Frame: 2154, payload: 0-1479 (1480 bytes)
* Fragment Offset tells whether it is the first or latter fragment. If it @ is 0, then the first fragment else it will be latter fragment.
* Total length: 548 bytes

9) What information in the IP header indicates that this is not the first datagram fragment? Are there more fragments? How can you tell?

— Fragment offset tells us whether it is the fragment

one, or middle fragment.
More fragments flag bit tell if there are more fragments. When it is not set, it means it is the last fragment. else there are more fragments.

10) What fields change in the IP header between the first and second?
— Identification, Header checksum, Header Length, Total length, Time to Live

RAMAIAH
Institute of Technology

Data Sheet

MARKS :

WEEK-8

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV, B
Subject :	DCN Lab	Subject Code:	CSL47

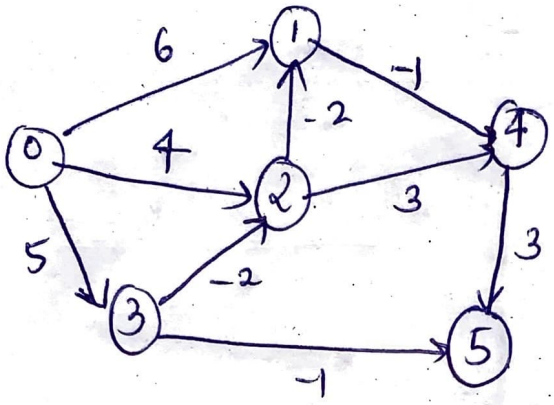
4A

BELLMAN FORD

```
import java.util.Arrays;
import java.util.Scanner;
class Edge{
    int s, d, w;
    Edge(int s, int d, int w){
        this.s = s
        this.d = d
        this.w = w
}   }
public class BellmanFord{
    public static void Bellmanford (int v, Edge[] edges, int s){
        int[] dist = new int[vertices];
        Arrays.fill (distance, Integer.MAX_VALUE);
        dist[s] = 0;
        for(int i = 0; i < v-1; i++){
            for (Edge edge : edges){
                if(dist[edge.s] ~~+ edge.w~~ != Integer.MAX_VALUE &&
                dist[edge.s] + edge.w < dist[edge.d]){
                    dist[edge.d] = dist[edge.s] + edge.w;
            } }
        }
        for (Edge edge : edges){
            if (dist[edge.s] != Integer.MAX_VALUE
                && dist[edge.s] + edge.w < dist[edge.d]){
                System.out.println ("Graph contains negative neeight
                    cycle");
            } return;
        }
    printSol(dist);
}
```

```
private static void printSol(int[] dist){
    System.out.println("Vertex distance from source");
    for (int i=0 ; i< dist.length ; i++){
        System.out.println(i +"\t\t" +dist[i]);
    }
}
public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter number of vertices: ");
    int v = scanner.nextInt();
    System.out.print("Enter number of edges: ");
    int edgesCount = scanner.nextInt();
    Edge[] edges = new Edge[edgesCount];
    System.out.println("Enter each edges as u v w
    format: ");
    for(int i=0; i<edgesCount; i++){
        int s = scanner.nextInt();
        int d = scanner.nextInt();
        int w = scanner.nextInt();
        edges[i] = new Edge(s, d, w);
    }
    System.out.println("Enter source vertex: ");
    int source = scanner.nextInt();
    bellmanford(v, edges, source);
}
}
```

OUTPUT:



Enter number of vertices : 6
Enter number of edges: 9
Enter each edge as u v w format:
0 1 6
0 2 4
0 3 5
1 4 -1
2 1 -2
3 2 -2
2 4 3
3 5 -1
4 5 3
Enter source vertex: 0

Vertex	Distance from source
0	0
1	1
2	3
3	5
4	0
5	3

RAMAIAH Institute of Technology

Data Sheet

MARKS :

WEEK-9

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV . B
Subject :	CBD DCN LAB	Subject Code:	CSL47

4B, 8B, 10B

Q) Design and simulate infrastructure less network, generate two traffic flow nodes and analyse its performance.

base file : Examples → tutorial → third.cc
reference file : Examples → Traffic control → traffic-control.cc

CHANGES:

* Add header files

```
#include "ns3/flow-monitor-module.h"
#include "ns3/traffic-control-module.h"
```

* Initialize

```
double simulationTime = 10;
```

* Remove the UDP Echo application

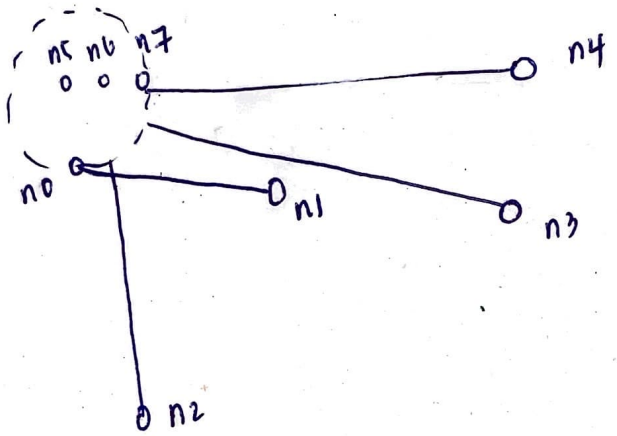
```
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

* copy the flow monitor settings from traffic-control.cc

```
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
localAddress); // Socket Type
ApplicationContainer sinkApp = packetSinkHelper.Install
(csmaNodes.Get(2)); //Destination in Bus topology
Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue
(payloadSize));
OnOffHelper onoff ("ns3::TcpSocketFactory", Ipv4Address::
GetAny()); // Socket Type
InetSocketAddress rmt(csmaInterfaces.GetAddress(nCsma,port)); //Interface with bus topology
apps.Add(onoff.Install(wifiStaNodes.Get(0))); // Source node from wifi node through AP
```

OUTPUT:

*** Flow monitor statistics ***
Tx Packets/Bytes: 4/220
Offered Load : 0.000193407 Mbps
Rx Packets/Bytes: 4/220
Packets/Bytes Dropped by Queue Disc: 0/0



RAMAIAH
Institute of Technology

Data Sheet

MARKS :

WEEK-9

Name :	IFRAH NAAZ	Branch:	CSE
USN/Roll No. :	1MS22CS064	Sem/Sec:	IV , B
Subject :	DCN Lab	Subject Code:	CSL47

Q) Design and simulate simple Extended Service Set with transmitting nodes in wireless LAN and determine the performance with respect to transmission of packets.

base file : examples → wireless → wifi-simple-adhoc.cc
Reference file : Examples → traffic-control → traffic-control.cc

- Copy the header files from reference to base file

```
#include "ns3/traffic-control-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
```

- Initialize

```
double simulationTime = 10;
```

- Change number nodes based on network

```
c.Create(4); // creates 4 nodes.
```

- Create vector for each node in Mobility Helper

```
positionAlloc → Add(Vector(0.0, 0.0, 0.0));
positionAlloc → Add(Vector(0.0, 0.0, 0.0));
positionAlloc → Add(Vector(0.0, 10.0, 0.0));
positionAlloc → Add(Vector(10.0, 10.0, 0.0));
// Don't change z-axis (not required)
```

⊛ After interface, add Routing Table and Flow Statistics from third.cc and traffic-control.c respectively

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

⊛ Mention the socketType.

```
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
localAddress);
```

⊛ Change the objectcreated for node and interface in source and destination

```
ApplicationContainer sinkApp = packetSinkHelper.Install
(c.Get(3)); // Desired destination
InetSocketAddress rmt (i.GetAddress(3), port);
// interface (remote)
apps.Add(onoffInstall (c.Get(1))); // Source node
```

⊛ Remove the excess code after ~~infero~~ interface creation till Simulator::Run();

OUTPUT:

```
*** Flow monitor statistics ***
  Tx Packets/Bytes:   759/1108700
  Offered Load: 0.893634 ~~Bytes~~ Mbps
  Rx Packets/Bytes:   749/1048660
  Packets/Bytes Dropped by Queue Disc: 0/0
```