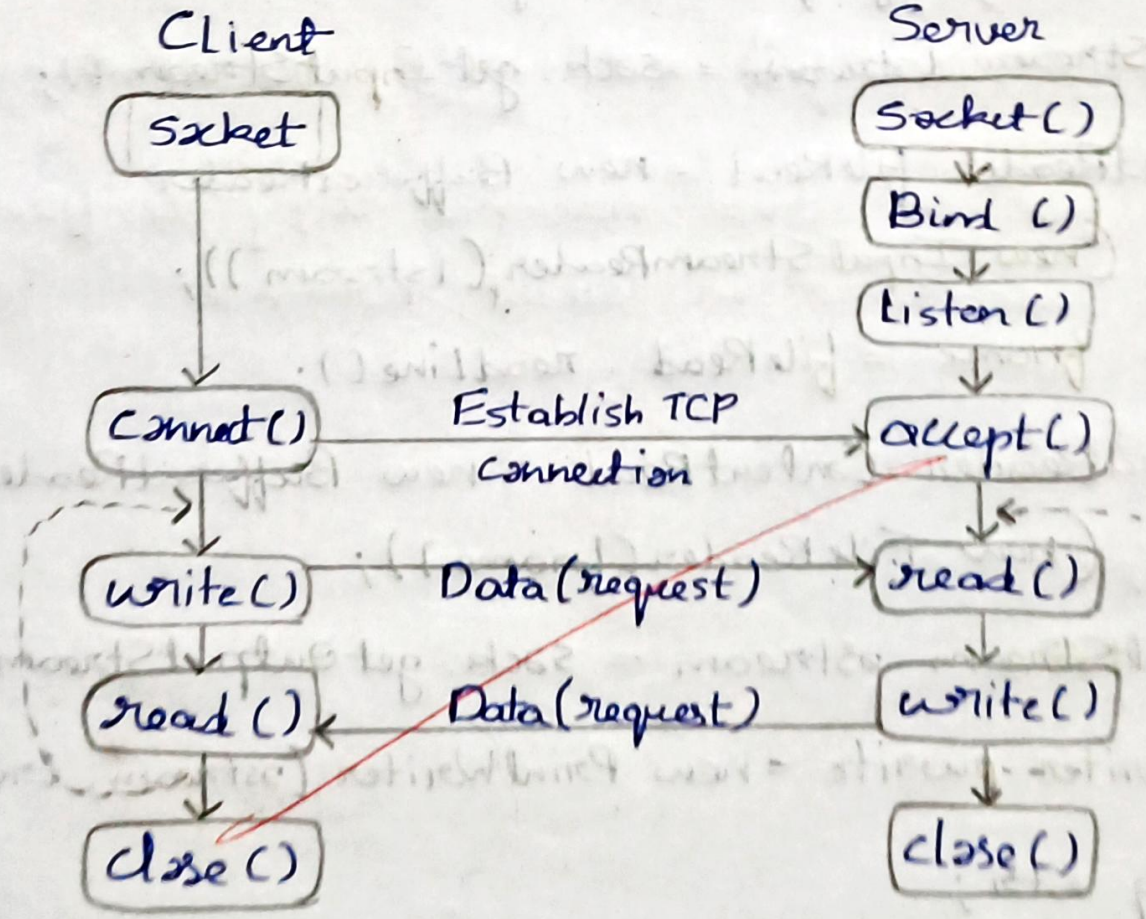
⇒ Problem Statement

Java program / problem to show the Connection or establish connection between server and client, and how communication happens between both using sockets and with the help of TCP (Transmission Control Protocol).

⇒ Figure



=> Code

* Server code

```
import java.net.*;
import java.io.*;
public class TCPS
{
  public static void main (String[] args) throws Exception
  {
    ServerSocket sersock = new ServerSocket(4000);
    System.out.println("Server 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();
    BufferedReader ContentRead = new BufferedReader
          (new FileReader(fname));
    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();
    }
}
```

* Client Code

```
import java.net.*;
import java.io.*;
public class TCPC {
  public static void main(String[] args) throws Exception {
    Socket sock = new Socket("127.0.0.1", 4000);
    System.out.println("Enter the filename");
    BufferedReader keyRead = new 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();
}
}
```

⇒ Output

* Server output

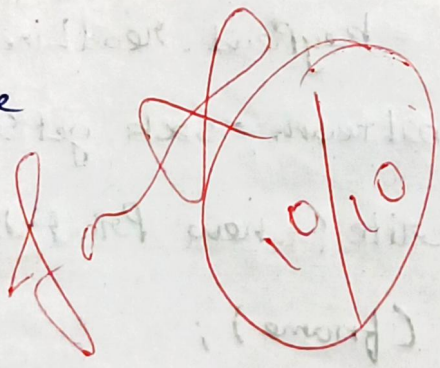
Server ready for connection
Connection is succesfull and waiting for chatting.

// Create a file in server terminal using vi filename.txt and use same in client terminal

* Client Output

Enter filename
demo.txt

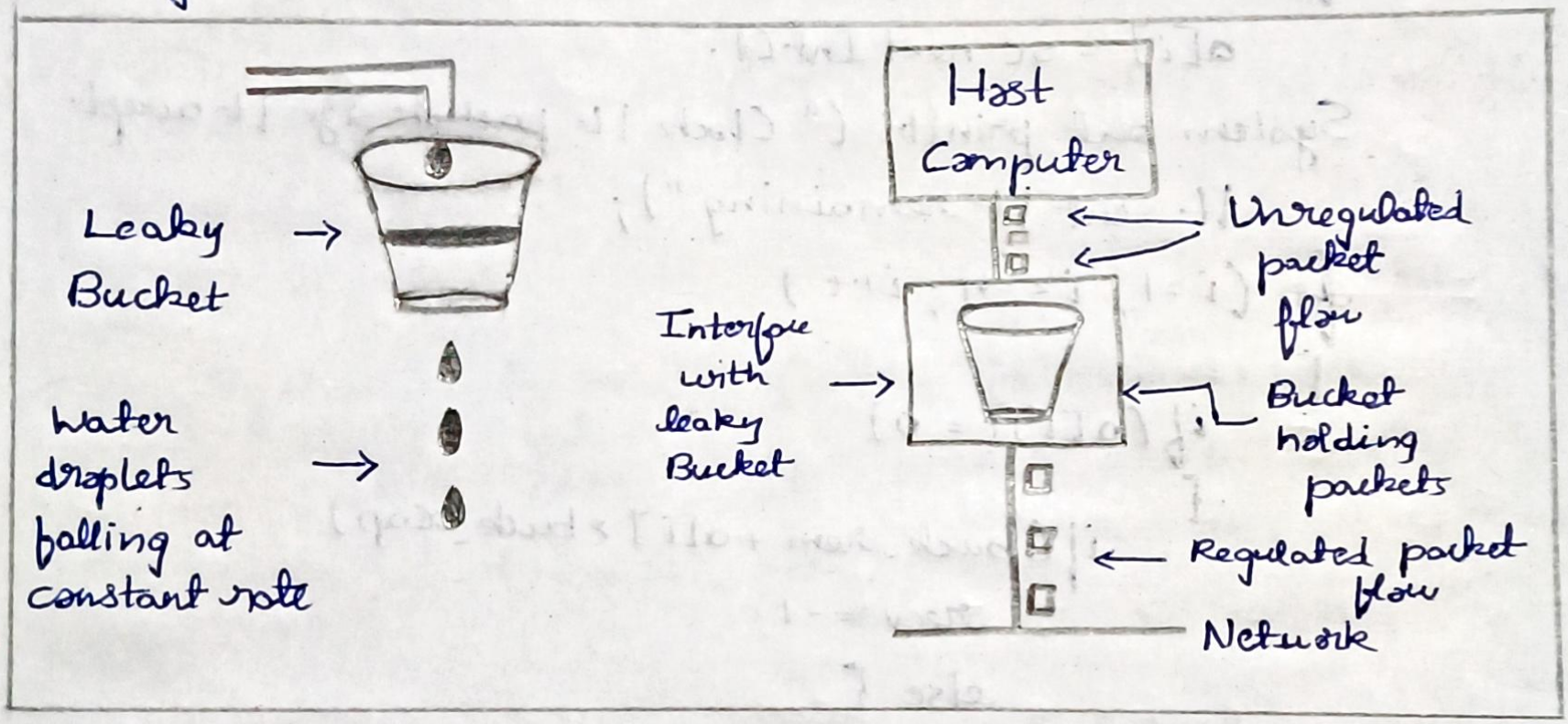
Hi
Hello
How are you ?



⇒ **Problem Statement**

Write a program for congestion control using leaky bucket algorithm.

⇒ **Figure**




⇒ **Code**

```
import java.util.Scanner;
import java.lang.*;
public class bucket {
    public static void main (String[] args ) {
```

```
int i;
int a[] = new int[20];
int buck_rem = 0, buck_cap = 4, rate = 3,
    sent, recv;
Scanner sc = new Scanner(System.in);
System.out.println("Enter number of packets");
int n = sc.nextInt();
System.out.println("Enter the packets");
for(i = 1; i<=n; i++)
    a[i] = sc.nextInt();
System.out.println("Clock \t packet size \t accept
    \t sent \t remaining");
for(i=1; i<=n; i++)
{
    if(a[i] != 0)
    {
        if(buck_rem + a[i] > buck_cap)
            recv = -1;
        else {
            recv = a[i];
            buck_rem += a[i];
        }
    }
    else
        recv = 0;
    if(buck_rem != 0) {
        if(buck_rem < rate)
```

```
            {
                sent = buck_rem;
                buck_rem = 0;
            }
            else {
                sent = rate;
                buck_rem = buck_rem - rate;
            }
        }
        else
            sent = 0;
    if (recv == -1)
        System.out.println (+i + "\t\t" +a[i] + "\tdropped\t"
        + sent + "\t\t" + buck_rem);

    else
        System.out.println (+i + "\t\t" + a[i] +
            "\t" + recv + "\t" + sent + "\t"
                + buck_rem);

        }
    }
}
```

⇒ Output Sample

Enter number of packets

5

Enter the packets

2 4 1 5 3

| Clock | packet size | accept | sent | remaining |
|---|---|---|---|---|
| 1 | 2 | 2 | 2 | 0 |
| 2 | 4 | 4 | 3 | 1 |
| 3 | 1 | 1 | 2 | 0 |
| 4 | 5 | dropped | 0 | 0 |
| 5 | 3 | 3 | 3 | 0 |

⇒ Output

Enter number of packets

6

Enter the packets

4 6 5 3 2 7

| Clock | packet size | accept | sent | remaining |
|---|---|---|---|---|
| 1 | 4 | 4 | 3 | 1 |
| 2 | 6 | dropped | 1 | 0 |
| 3 | 5 | dropped | 0 | 0 |
| 4 | 3 | 3 | 3 | 0 |
| 5 | 2 | 2 | 2 | 0 |
| 6 | 7 | dropped | 0 | 0 |

→ Problem Statement,

Trace Hypertext Transfer Protocal using packet siniffer and packet analyser.

(Using Wireshark software)

→ Website address,

http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html

→ Lab Exercise Questions

i) Is your browser running HTTP 1.0 or 1.1 ? What version of http is the server running ?

Ans: Both the server and browser are working based on HTTP 1.1.

ii) What language does your browser indicate that it can accept from the server ?

Ans: The language that the browser accepts is,
en-GB or en-US (English)

iii) What is the status code returned from server to your browser?

Ans: 200 OK if connection is successful
404 not found if there is fail in connection.

iv) When was the HTML file, that you are retreiving last modified at the server?

Ans: The HTML file being retrieved was last modified at Fri, 26 May 23, 05:50:01 GMT.

v) How many bytes of content are being returned to your browser?

Ans: The byte length or content length being returned to browser is 128 bytes.

| Name : | Anosh. M. Somayaji | Branch: | CSE |
|---|---|---|---|
| USN/Roll No. : | 1MS21CS016 | Sem/Sec: | IV 1st sem, A |
| Subject : | DCN Laboratory | Subject Code: | CSL47 |

→ Problem Statement,

Trace Domain Name Server using packet sniffer and packet analyzer.

→ Objective,

To understand working of Domain Name Service.

→ nslookup - query about some DNS server.

Visit → http://www.ietf.org.

→ Lab Exercise Questions,

i) Locate the DNS query & response messages. Are they sent over UDP or TCP?

Ans: The messages were located and they are sent over UDP (User Datagram Protocol)

ii) What is the destination port for DNS query message? What is the source port of DNS response message?

Ans. The destination port is (query response)

→ 53

The source port of DNS response message is

→ 53 (Both are same)

iii) To what IP address is the DNS query message sent ? Use ipconfig to determine the ip address of your local DNS server ? Are these two ip addresses the same ?

Ans: IP address of query message → 172.1.2.2

IP address of Local DNS server → 172.1.6.67

Both ip addresses are different.

iv) Examine the DNS query message ? What 'type' of message is it ? Does the query have any answers ?

Ans: It is of the type A, class IN

Answers of the query, are zero
Answers in response are 2

v) Examine DNS response msg. How many answers are provided? What does these answers contain?

Ans

Answers provided in response are 2.
Answers contain,

www.ietf.org : type CNAME, class IN, cname www.ietf.org.cdn.cloudflare.net.

www.ietf.org.cdn.cloudflare.net : type A, class IN, addr 104.16.44.99.

www.ietf.org.cdn.cloudflare.net : type A, class IN, addr 104.16.45.99.

vi) Consider the subsequent TCP SYN packet sent by host. Does the ip address of SYN packet correspond to any of the IP address provided in DNS response message?

Ans: One TCP SYN message has the same IP address as in DNS response message (172.1.6.67)
The other TCP syn message has different ip address than DNS response msg (104.16.45.99)

vii) This website has images. Before retrieving each image, does your host issue new DNS queries.

Ans: No, all images are loaded from the website, so additional DNS queries are not required.

| Name : | ANEESH M SOMAYAJI | Branch: | CSE |
|---|---|---|---|
| USN/Roll No. : | 1MS21CS016 | Sem/Sec: | IV$^{th}$, A |
| Subject : | DCN Laboratory | Subject Code: | CSL47 |

→ Problem Statement,

Trace Internet Protocol and Internet Control Message Protocol using packet sniffer and packet analyser.

→ Questions | Answers

1. Select the ICMP echo request msg sent by your computer and expand the IP part of packet in packet details window. What is IP address of your computer ?

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

3. How many bytes are in the IP header ? How many bytes are in payload of IP datagram ? Explain how u determine of payload bytes.

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

Ans 1. The IP address of the computer is 172.1.6.72

Ans 2: The protocol is ICMP and the value is 1

Ans 3: There are 20 bytes in the IP header. The total length is 84 bytes. Therefore the length of payload is (84-20) → 64 bytes.

Payload length = 64 bytes.

Ans 4: No, IP datagram hasn't been fragmented. We can find this out because the D bit here is set to 1 and M bit is set to zero. which means it is the last fragment bit. Hence it can't be fragmented. (Fragment offset field is zero)

Q5. Which field in IP datagram changes from one datagram to other within this series of ICMP msgs sent by your computer.

Q6. Which field stays constant?

Q7. Find ICMP Echo msg sent by your computer after changing packet size to 2000. Has it been fragmented?

Q8. Print out first fragment of IP datagram. What info in the IP header indicates that data has been fragmented. What info in IP header indicates whether this is first or latter fragment? How long is IP datagram?

Q9. What info in IP header indicates that this is not first fragment? Are there more fragment? How can u tell?

Q10. What fields change in IP address between first & second fragment?

Ans 5: The fields changing are,

i) Identification

~~ii) Time to live~~

iii) Header Checksum.

Ans 6: Constant fields are,

i) Total length ii) Protocol iii) Differentiated service field iv) Source & Destination IP, Time to Live

v) Version (IPv4) vi) Header length (20bytes)

Ans 7. After changing the packet size to 2000 we get to know that the ECHO msg request has been fragmented & D bit is set & M bit is 0.

Ans 8: The M bit value for particular fragment is 1 which means it can be first or any middle fragment.

The offset field tells us whether it is first or any other fragment if offset = 0, then it's first fragment.

Ans 9: If the M bit is not zero then there are more fragment. Also if offset field is not zero in IP header then it identifies that it isn't the first fragment.

Ans 10: The total length, more fragments bits and the fragment offset field changes whereas identification and time to live is same.

Done
30/06/23

| Name : | ANEESH M SOMAYAJI | Branch: | CSE |
|---|---|---|---|
| USN/Roll No. : | 1MS21CS016 | Sem/Sec: | IV$^{th}$ sem, A |
| Subject : | DCN Laboratory | Subject Code: | CSL47 |

→ Problem Statement,

Trace Dynamic host control protocol using packet sniffer & packet analyser.

⇒ Questions,

i) Are DHCP messages sent over UDP or TCP ?

ii) What is the link-layer address of your host ?

iii) What is the value of DHCP discover msg differentiate this msg from DHCP request msg ?

iv) What is the value of transaction ID. first four discover/offer DHCP msges ? What is the value of second set of DHCP msges ? What is purpose of Transaction-ID field ?

v) A host uses DHCP to obtain IP address, but host IP isn't confirmed until end of 4 msges set, If IP address is not set until end of 4 msgs then what are values in IP datagrams of 4 msges, For each of 4 DHCP msges, initiate source & destination IP address.

Ans 1) DHCP messages are sent over User Datagram Protocol (UDP)

Ans 2) Link Layer address of DHCP host is 2c:f0:5d:17:6b:60.

Ans 3) DHCP message type helps in differentiating between discover & request msg, Discover msg has value as 1 & request msg has value as 3.

Ans 4) The value of Transactim IP of first four Request/Discover/offer/ACK msg is 0xc82ff54c. The value of second set of DHCP msgs is 0x47aee3d7

A Transactim IP field is used so that DHCP server can differentiate between Client requests during client prequest process.

Ans 5)

| | Source IP | Destination IP |
|---|---|---|
| Discover | 0.0.0.0 | 255.255.255.255 |
| Offer | 172.1.6.1 | 172.1.6.72 |
| Request | 0.0.0.0 | 255.255.255.255 |
| ACK | 172.1.6.1 | 172.1.6.72 |

vi) What is IP address of DHCP server ?

vii) What IP address is DHCP server offering in DHCP offer msg ? Indicate which msg contains this address.

viii) In the example there is no relay between host & DHCP server, what value indicates absence of relay agent ? Is there relay agent in your exp ? If so what is the IP address.

ix) Explain purpose of lease time. How long is the lease time in ur exp ?

x) What is purpose of DHCP release msg ? Does DHCP server give acknowledge receipt of client's DHCP request.

Ans 6) IP address of DHCP server - 172.1.6.1

Ans 7) IP address of DHCP offer msg is 172.1.6.72 (Your client IP address)

"DHCP message type = Offer" indicates the msg contains this address.

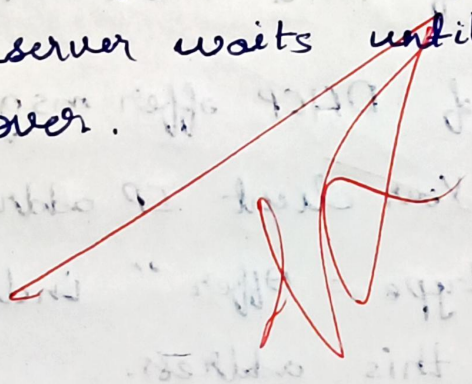
Ans 8) The value of Relay agent IP address indicates the absence of relay agent if the address is (0.0.0.0). As relay agent address is not zero there is relay agent in this exp.

The IP address of relay agent is 172.1.6.1.

Ans 9) Lease time is used to tell the amount of time a DHCP server assigns an IP address to client.

The least time in this experiment is 8 days, 8 hours. (720000s)

Ans 10) DHCP release message is used to cancel the lease on the IP address given to it by the DHCP server. The DHCP server does not send the acknowledgement receipt of DHCP's request made by client. If DHCP release msg is lost then DHCP server waits until the lease period time is over.



-> Problem Statement,

Design and simulate a wired network with duplex links between n nodes with CBR over UDP, set the queue size, vary the bandwidth and find no of packets dropped.

-> Program,

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/application-module.h"
#include "ns3/traffic-control-module.h"
#include "ns3/flow-monitor-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("TrafficControlExample");
int main (int argc, char *argv[])
{
    double simulationTime = 10;
    std::string TransportProt = "Udp";
    std::string socketType;
```

```
CommandLine cmd;
cmd.AddValue ("transportProt", "Transport
protocol to use: Tcp, Udp", transportProt);
cmd.Parse (argc, argv);
if (transportProt.compare ("Tcp") == 0)
  { SocketType = "ns3:TcpSocketFactory"; }
else {
  SocketType = "ns3:TcpSocketFactory"; }
NodeContainer nodes;
nodes.Create (3);
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);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces12 = address.Assign (devices12);
```

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
//Flow
 uint16_t port = 7;
 Address localAddress (InetSocketAddress (Ipv4Address::GetAny(), port));
 PacketSinkHelper packetSinkHelper (socketType, localAddress);
 ApplicationContainer sinkApp = packetSinkHelper.Install (nodes.Get(2));
 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());
onoff.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable [const=1]"));
onoff.SetAttribute ("OffTime", StringValue("ns3::ConstantRandomVariable [const=0]"));
onoff.SetAttribute ("PacketSize", UintegerValue (payloadSize));
onoff.SetAttribute ("DataRate", StringValue ("50mbps"));
 ApplicationContainer apps;
 InetSocketAddress rmt (interfaces12.GetAddress(1), port);
 rmt.SetTos(0xb8);
 AddressValue remoteAddress (rmt);
 onoff.SetAttribute ("Remote", remoteAddress");
 apps.Add (onoff.Install(nodes.Get(0)));
 apps.Start (Seconds(1.0));
 apps.Stop (Seconds (simulationTime + 0.1));
 FlowMonitorHelper flowmon;
 Ptr<FlowMonitor> monitor = flowmon.InstallAll();
 Simulator::Stop (Seconds (simulatorTime + 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[i].txBytes * 8.0 /
        (stats[i].timeLastTxPacket.GetSeconds() -
    stats[i].timeFirstTxPacket.GetSeconds()) / 1000000 << "Mbps" <<
                    std::endl;
std::cout << "  Rx Packets/Bytes:   " << stats[i].rxPackets << "/"
    << stats[i].rxBytes << std::endl;
uint32_t packetDroppedByQueueDisc = 0;
uint64_t bytesDroppedByQueueDisc = 0;
if (stats[i].packetDropped.size() > Ipv4FlowProbe::DROP_QUEUE_DISC)
  {
    packetsDroppedByQueueDisc = stats[i].packetDropped[Ipv4FlowProbe::
                                    DROP_QUEUE_DISC];
    bytesDroppedByQueueDisc = stats[i].bytesDropped[Ipv4FlowProbe::
                                    DROP_QUEUE_DISC];
  }
std::cout << "  Packets/Bytes Dropped by Queue Disc:" << packetsDroppedByQueueDisc
        << "/" << bytesDroppedByQueue << std::endl;
Simulator::Destroy();
return 0;
}
```

⇒ Output:

```
*** Flow Monitor Statistics ***
  Tx Packets/Bytes:   27624/40775024
  Offered Load:   50.9687 Mbps
  Rx Packets/Bytes:   5408/7982208
  Packets/Bytes Dropped by Queue Disc:   11970/17667720
```