Computer Networks Assignment 3

Ojus Goel 12241190

September 15, 2024

PART 1

1. Topology of two nodes separated by point-to-point link.

```cpp
int main (int argc, char *argv[]) {

    // Creating 2 ndoes
    NodeContainer nodes;
    nodes.Create(2);

    // Setting up p2p link
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("10ms"));

    NetDeviceContainer device = pointToPoint.Install(nodes);

    // Installing Internet stack
    InternetStackHelper stack;
    stack.Install(nodes);

    // Assigning IP addresses
    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign(device);

    Simulator::Stop (Seconds (10.0));

    // Enabling flow monitor
    Ptr<FlowMonitor> flowmon;
    FlowMonitorHelper flowmonHelper;
    flowmon = flowmonHelper.InstallAll ();

    // Run the simulation, print the stats and destroy the simulators
    Simulator::Run ();
    printStats (flowmonHelper, true);
    Simulator::Destroy ();
    return 0;
}
```

- We setup the point-to-point link using "PointToPointHelper".
- We first created 2 nodes with the help of NodeContainer.
- The max bandwidth is 10Mbps and delay is 10ms.
- We then created networking device using NetDeviceContainer.
- Then we installed the Internet Stack and assigned the IPs.
- Finally we tried running the simulation.

2. Now we added UdpClient on Node1 and UdpServer on Node2 using UdpServerHelper of fixed data rate 1Mbps using OnOffHelper.

   - We run the simulation for 12 seconds where the server starts sending packets after seconds from starting.

```
// Assigning IP addresses
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign(device);

// Creating applications
// Create a UDP Server on Node 2
uint16_t port = 9; // UDP port
UdpServerHelper server(port);
ApplicationContainer serverApp = server.Install(nodes.Get(1));
serverApp.Start(Seconds(1.0));
serverApp.Stop(Seconds(10.0));

// Create a UDP Client on Node 1 with a fixed data rate
OnOffHelper client("ns3::UdpSocketFactory", InetSocketAddress(interfaces.GetAddress(1), port));
client.SetConstantRate(DataRate("1Mbps")); // Set Rate1 here
client.SetAttribute("PacketSize", UintegerValue(1024)); // Packet size

ApplicationContainer clientApp = client.Install(nodes.Get(0));
clientApp.Start(Seconds(2.0));
clientApp.Stop(Seconds(10.0));

Simulator::Stop (Seconds (12.0));

// Enabling flow monitor
Ptr<FlowMonitor> flowmon;
FlowMonitorHelper flowmonHelper;
flowmon = flowmonHelper.InstallAll ();

// Enable netanim
AnimationInterface anim("p1_2_netanim_final.xml");

// Run the simulation, print the stats and destroy the simulators
```

3. We now measure the end-to-end throughput while varying the latency

```
ojusg@ojusg-Inspiron-14-5420:~/Desktop/College_Sem_V/Computer_Networks/ass3/12241190/ns-allinone-3.42/ns-3.42$ ./ns3 run scratch
/p1_2.cc
Consolidate compiler generated dependencies of target scratch_p1_2
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
FlowID: 1 (UDP 10.1.1.1 / 49153 --> 10.1.1.2 / 9)
  Tx Bytes: 1026752
  Rx Bytes: 1026752
  Tx Packets: 976
  Rx Packets: 976
  Time LastRxPacket: 10.0062s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 1002.93 Kbps
  Mean{Delay}: 0.0108432
  Mean{Jitter}: 0
Total throughput of System: 1002.93 Kbps
Total packets transmitted: 976
Total packets received: 976
Total packets dropped: 0
Packet Lost Ratio: 0
```

Figure : Delay is 10ms

```
ojusg@ojusg-Inspiron-14-5420:~/Desktop/College_Sem_V/Computer_Networks/ass3/12241190/ns-allinone-3.42/ns-3.42$ ./ns3 run scratch
/p1_2.cc
[  0%] Building CXX object scratch/CMakeFiles/scratch_p1_2.dir/p1_2.cc.o
[  0%] Linking CXX executable ../../build/scratch/ns3.42-p1_2-default
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
FlowID: 1 (UDP 10.1.1.1 / 49153 --> 10.1.1.2 / 9)
  Tx Bytes: 1026752
  Rx Bytes: 1026752
  Tx Packets: 976
  Rx Packets: 976
  Time LastRxPacket: 10.0212s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 1001.06 Kbps
  Mean{Delay}: 0.0258432
  Mean{Jitter}: 0
Total throughput of System: 1001.06 Kbps
Total packets transmitted: 976
Total packets received: 976
Total packets dropped: 0
Packet Lost Ratio: 0
```

Figure : Delay is 25ms

```
ojusg@ojusg-Inspiron-14-5420:~/Desktop/College_Sem_V/Computer_Networks/ass3/12241190/ns-allinone-3.42/ns-3.42$ ./ns3 run scratch
/p1_2.cc
Consolidate compiler generated dependencies of target scratch_p1_2
[  0%] Building CXX object scratch/CMakeFiles/scratch_p1_2.dir/p1_2.cc.o
[  0%] Linking CXX executable ../../build/scratch/ns3.42-p1_2-default
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
FlowID: 1 (UDP 10.1.1.1 / 49153 --> 10.1.1.2 / 9)
  Tx Bytes: 1026752
  Rx Bytes: 1026752
  Tx Packets: 976
  Rx Packets: 976
  Time LastRxPacket: 10.0962s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 991.773 Kbps
  Mean{Delay}: 0.100843
  Mean{Jitter}: 0
Total throughput of System: 991.773 Kbps
Total packets transmitted: 976
Total packets received: 976
Total packets dropped: 0
Packet Lost Ratio: 0
```

Figure : Delay is 100ms

```
ojusg@ojusg-Inspiron-14-5420:~/Desktop/College_Sem_V/Computer_Networks/ass3/12241190/ns-allinone-3.42/ns-3.42$ ./ns3 run scratch
/p1_2.cc
Consolidate compiler generated dependencies of target scratch_p1_2
[  0%] Building CXX object scratch/CMakeFiles/scratch_p1_2.dir/p1_2.cc.o
[  0%] Linking CXX executable ../../build/scratch/ns3.42-p1_2-default
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
FlowID: 1 (UDP 10.1.1.1 / 49153 --> 10.1.1.2 / 9)
  Tx Bytes: 1026752
  Rx Bytes: 1026752
  Tx Packets: 976
  Rx Packets: 976
  Time LastRxPacket: 10.2462s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 973.714 Kbps
  Mean{Delay}: 0.250843
  Mean{Jitter}: 0
Total throughput of System: 973.714 Kbps
Total packets transmitted: 976
Total packets received: 976
Total packets dropped: 0
Packet Lost Ratio: 0
```

Figure : Delay is 250ms

```
ojusg@ojusg-Inspiron-14-5420:~/Desktop/College_Sem_V/Computer_Networks/ass3/12241190/ns-allinone-3.42/ns-3.42$ ./ns3 run scratch
/p1_2.cc
Consolidate compiler generated dependencies of target scratch_p1_2
[  0%] Building CXX object scratch/CMakeFiles/scratch_p1_2.dir/p1_2.cc.o
[  0%] Linking CXX executable ../../build/scratch/ns3.42-p1_2-default
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
FlowID: 1 (UDP 10.1.1.1 / 49153 --> 10.1.1.2 / 9)
  Tx Bytes: 1026752
  Rx Bytes: 1026752
  Tx Packets: 976
  Rx Packets: 976
  Time LastRxPacket: 10.9962s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 892.463 Kbps
  Mean{Delay}: 1.00084
  Mean{Jitter}: 0
Total throughput of System: 892.463 Kbps
Total packets transmitted: 976
Total packets received: 976
Total packets dropped: 0
Packet Lost Ratio: 0
```

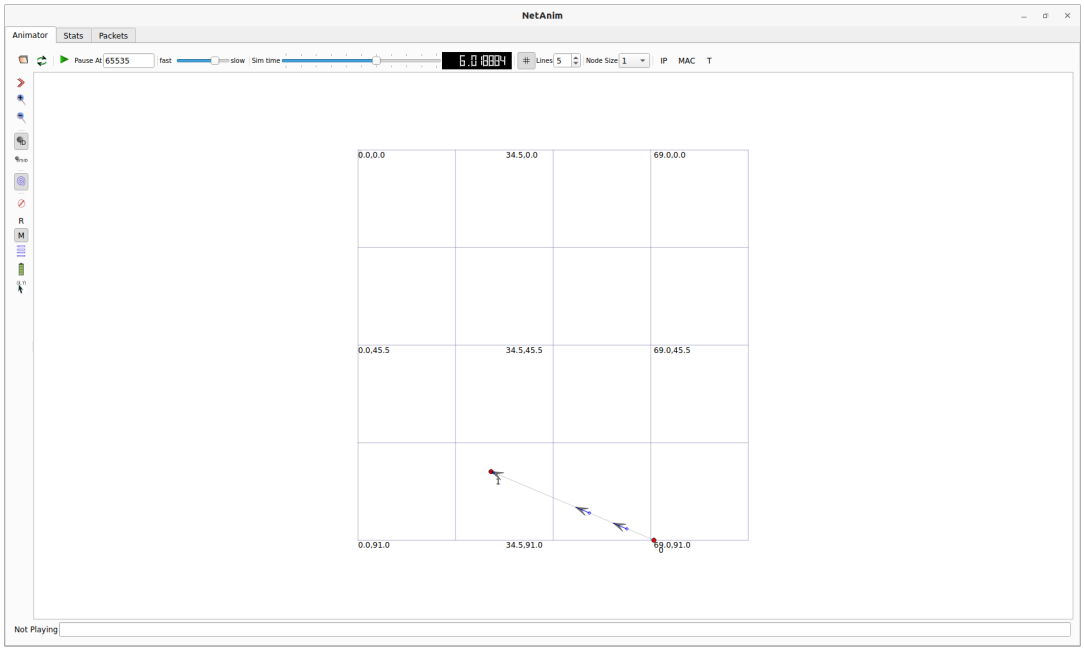
Figure : Delay is 1000ms

We varied the latency values as 10ms, 25ms, 100ms, 250ms and finally 1000ms. On observing closely, we can notice that the **throughput decreases on increasing the delay.** The reason being that since delay is increasing, it takes the packets longer to reach the destination which therefore decreases the throughput.

4. Now adding another client application to Node1 and a server instance to Node2. **To ensure that we have no conflicts each client-server pair must run on different ports. One pair is running on port 9 and other pair is running on port 10 here.**

```
// Creating applications

// First UDP Server on Node 2, listening on port 9
uint16_t port1 = 9; // UDP port for first client-server pair
UdpServerHelper server1(port1);
ApplicationContainer serverApp1 = server1.Install(nodes.Get(1));
serverApp1.Start(Seconds(1.0));
serverApp1.Stop(Seconds(7.0));

// First UDP Client on Node 1, sending to port 9
OnOffHelper client1("ns3::UdpSocketFactory", InetSocketAddress(interfaces.GetAddress(1), port1));
client1.SetConstantRate(DataRate("1Mbps")); // Set Rate1 here
client1.SetAttribute("PacketSize", UintegerValue(1024)); // Packet size
ApplicationContainer clientApp1 = client1.Install(nodes.Get(0));
clientApp1.Start(Seconds(2.0));
clientApp1.Stop(Seconds(7.0));

// Second UDP Server on Node 2, listening on a different port (10)
uint16_t port2 = 10; // UDP port for the second client-server pair
UdpServerHelper server2(port2);
ApplicationContainer serverApp2 = server2.Install(nodes.Get(1));
serverApp2.Start(Seconds(5.0));
serverApp2.Stop(Seconds(10.0));

// Second UDP Client on Node 1, sending to port 10
OnOffHelper client2("ns3::UdpSocketFactory", InetSocketAddress(interfaces.GetAddress(1), port2));
client2.SetConstantRate(DataRate("1Mbps")); // Set Rate2 here
client2.SetAttribute("PacketSize", UintegerValue(1024)); // Packet size
ApplicationContainer clientApp2 = client2.Install(nodes.Get(0));
clientApp2.Start(Seconds(6.0));
clientApp2.Stop(Seconds(10.0));


Simulator::Stop (Seconds (20.0));
```

This is the topology of nodes that we have set up. Corresponding animations can be viewed using .xml files. p1_2_netanim_final.xml for viewing part 2 of this question and p1_4_netanim.xml for visualising this part.


NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
1
IP
MAC
T
0.0,0.0
34.5,0.0
69.0,0.0
0.0,45.5
34.5,45.5
69.0,45.5
0.0,91.0
34.5,91.0
69.0,91.0
Not Playing

PART 2

PART A

First of all we create the custom network topology stated to us in the question. nodesP1 consists of nodes n0, n1 and n2. nodesP2 consits of nodes n3, n4, n5 and nodesP3 consists of nodes n6 and n7. Followed same naming convention throughout the code. We created the nodes and we set up the fixed positions of the routers, nodes for easy visualisation of the network.

```
int main(int argc, char *argv[]) {
    // Setting the default time resolution
    Time::SetResolution(Time::NS);

    // Creating nodes
    NodeContainer NodesP1, routers, NodesP2, NodesP3;
    NodesP1.Create(3);  // Nodes n0, n1, n2
    NodesP2.Create(3);  // Nodes n3, n4, n5
    NodesP3.Create(2);  // Nodes n6, n7
    routers.Create(3);  // Routers router0, router1, router2

    // Setting constant positions for all nodes (Previously we had error/warning for position of nodes,so correcting it this time)
    MobilityHelper mobility;
    Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();

    // Setting positions for NodesP1
    positionAlloc->Add(Vector(0.0, 10.0, 0.0));  // n0
    positionAlloc->Add(Vector(0.0, 20.0, 0.0)); // n1
    positionAlloc->Add(Vector(0.0, 30.0, 0.0)); // n2

    // Setting positions for NodesP2
    positionAlloc->Add(Vector(30.0, 0.0, 0.0));  // n3
    positionAlloc->Add(Vector(30.0, 10.0, 0.0)); // n4
    positionAlloc->Add(Vector(30.0, 20.0, 0.0)); // n5

    // Setting positions for NodesP3
    positionAlloc->Add(Vector(40.0, 30.0, 0.0)); // n6
    positionAlloc->Add(Vector(40.0, 40.0, 0.0)); // n7

    // Setting positions for routers
    positionAlloc->Add(Vector(10.0, 20.0, 0.0)); // router0
    positionAlloc->Add(Vector(20.0, 5.0, 0.0));  // router1
    positionAlloc->Add(Vector(20.0, 35.0, 0.0)); // router2

    mobility.SetPositionAllocator(positionAlloc);
    mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");

    // Installing mobility model on all nodes
    mobility.Install(NodesP1);
    mobility.Install(NodesP2);
```

Now we setup the required links. We set up the data rate as 10Mbps and delay as 20ms. Code snippet is shown below: We set up the parameters and install all the required links. For links between routers we set up data rate as 4Mbps and delay as 40ms. We chose these values to create traffic in our network. Since connection between routers is slower than between nodes and routers, we can observe packet losses easily.

```
    // Installing mobility model on all nodes
    mobility.Install(NodesP1);
    mobility.Install(NodesP2);
    mobility.Install(NodesP3);
    mobility.Install(routers);

    // Setting up Point-to-Point link parameters
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("10ms"));

    // Installing P2P links from NodesP1 to router0
    NetDeviceContainer DevicesP1;
    for (uint32_t i = 0; i < NodesP1.GetN(); ++i) {
        NetDeviceContainer link = pointToPoint.Install(NodesP1.Get(i), routers.Get(0));
        DevicesP1.Add(link);
    }

    // Installing P2P links from NodesP2 to router1
    NetDeviceContainer DevicesP2;
    for (uint32_t i = 0; i < NodesP2.GetN(); ++i) {
        NetDeviceContainer link = pointToPoint.Install(NodesP2.Get(i), routers.Get(1));
        DevicesP2.Add(link);
    }

    // Installing P2P links from NodesP3 to router2
    NetDeviceContainer DevicesP3;
    for (uint32_t i = 0; i < NodesP3.GetN(); ++i) {
        NetDeviceContainer link = pointToPoint.Install(NodesP3.Get(i), routers.Get(2));
        DevicesP3.Add(link);
    }

    // Installing P2P links between routers (router0 to router1 and router2)
    PointToPointHelper routerLink;
    routerLink.SetDeviceAttribute("DataRate", StringValue("4Mbps"));
    routerLink.SetChannelAttribute("Delay", StringValue("40ms"));

    NetDeviceContainer routerLinks;
    routerLinks.Add(routerLink.Install(routers.Get(0), routers.Get(1)));
    routerLinks.Add(routerLink.Install(routers.Get(0), routers.Get(2)));
```

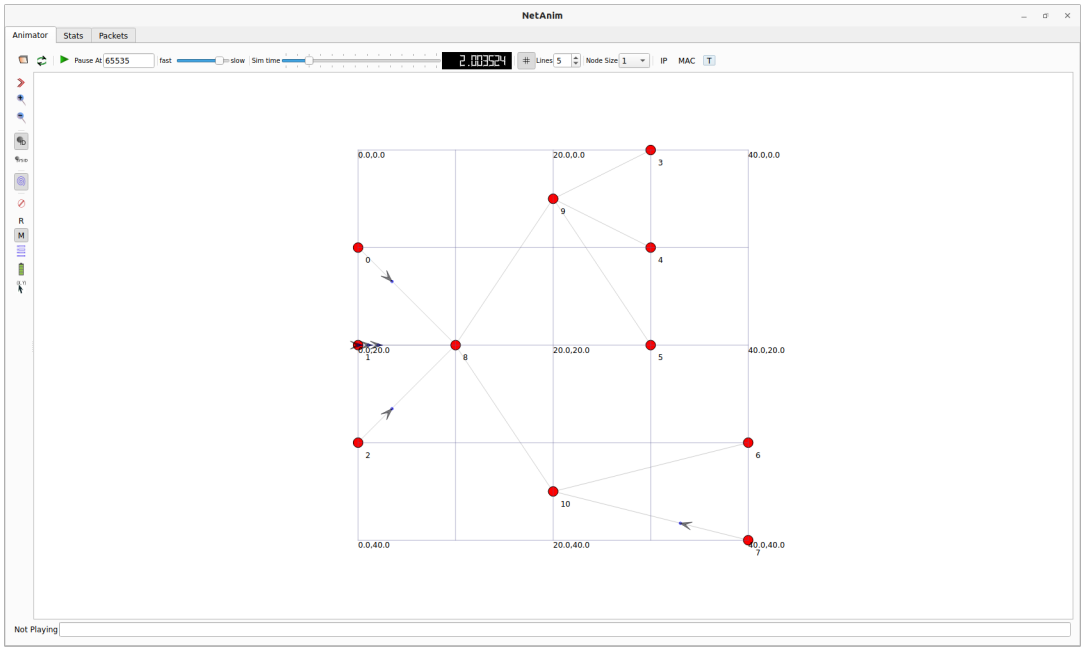
Then we assign the IP address to the nodes and routers.

```
Ipv4InterfaceContainer InterfacesP1, InterfacesP2, InterfacesP3, routerInterfaces;

// Assigning IP addresses to P1 nodes (n0, n1, n2) connected to router0
for (uint32_t i = 0; i < 3; ++i) {
    std::ostringstream subnet;
    subnet << "10.1." << i + 1 << ".0";
    address.SetBase(Ipv4Address(subnet.str().c_str()), "255.255.255.252");
    InterfacesP1.Add(address.Assign(DevicesP1.Get(i * 2)));
    address.Assign(DevicesP1.Get(i * 2 + 1));
}

// Assigning IP addresses to P2 nodes (n3, n4, n5) connected to router1
for (uint32_t i = 0; i < 3; ++i) {
    std::ostringstream subnet;
    subnet << "10.2." << i + 1 << ".0";
    address.SetBase(Ipv4Address(subnet.str().c_str()), "255.255.255.252");
    InterfacesP2.Add(address.Assign(DevicesP2.Get(i * 2)));  // Assign IP to node's device
    address.Assign(DevicesP2.Get(i * 2 + 1));  // Assign IP to router's device
}

// Assigning IP addresses to P3 nodes (n6, n7) connected to router2
for (uint32_t i = 0; i < 2; ++i) {
    std::ostringstream subnet;
    subnet << "10.3." << i + 1 << ".0";
    address.SetBase(Ipv4Address(subnet.str().c_str()), "255.255.255.252");
    InterfacesP3.Add(address.Assign(DevicesP3.Get(i * 2)));  // Assign IP to node's device
    address.Assign(DevicesP3.Get(i * 2 + 1));  // Assign IP to router's device
}

// Assigning IP addresses to router-router links
for (uint32_t i = 0; i < 2; ++i) {
    std::ostringstream subnet;
    subnet << "10.4." << i + 1 << ".0";
    address.SetBase(Ipv4Address(subnet.str().c_str()), "255.255.255.252");
    routerInterfaces.Add(address.Assign(routerLinks.Get(i*2)));
    address.Assign(routerLinks.Get(i*2+1));
}

// Populating routing tables
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

Now we define the flow as stated to us in the question. Showing the screenshot only for 2 flows. We followed the same method for all the flows. Here we also set up the data rate and packet size for each flow. For each flow, I have used different values for better understanding and easy visualisation.

```
// Create TCP traffic flows
uint16_t tcpPort1 = 8080;
uint16_t tcpPort2 = 8081;
uint16_t tcpPort3 = 8082;

// Creating UDP traffic flow
uint16_t udpPort1 = 8084;

// TCP Flow 1: From n7 to n4
Address tcpSinkAddress1(InetSocketAddress(InterfacesP2.GetAddress(1), tcpPort1));
PacketSinkHelper tcpSinkHelper1("ns3::TcpSocketFactory", tcpSinkAddress1);
ApplicationContainer tcpSinkApp1 = tcpSinkHelper1.Install(NodesP2.Get(1));
tcpSinkApp1.Start(Seconds(1.0));
tcpSinkApp1.Stop(Seconds(11.0));

OnOffHelper tcpClient1("ns3::TcpSocketFactory", tcpSinkAddress1);
tcpClient1.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));
tcpClient1.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));
tcpClient1.SetAttribute("DataRate", DataRateValue(DataRate("0.5Mbps")));
tcpClient1.SetAttribute("PacketSize", UintegerValue(2096));
ApplicationContainer tcpClientApp1 = tcpClient1.Install(NodesP3.Get(1));
tcpClientApp1.Start(Seconds(2.0));
tcpClientApp1.Stop(Seconds(11.0));

// UDP Flow: From n1 to n5
Address udpSinkAddress1(InetSocketAddress(InterfacesP2.GetAddress(2), udpPort1));
PacketSinkHelper udpSinkHelper1("ns3::UdpSocketFactory", udpSinkAddress1);
ApplicationContainer udpSinkApp1 = udpSinkHelper1.Install(NodesP2.Get(2));
udpSinkApp1.Start(Seconds(1.0));
udpSinkApp1.Stop(Seconds(11.0));

OnOffHelper udpClient("ns3::UdpSocketFactory", udpSinkAddress1);
udpClient.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));
udpClient.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));
udpClient.SetAttribute("DataRate", DataRateValue(DataRate("4Mbps")));
udpClient.SetAttribute("PacketSize", UintegerValue(512));
ApplicationContainer udpClientApp = udpClient.Install(NodesP1.Get(1));
udpClientApp.Start(Seconds(2.0));
udpClientApp.Stop(Seconds(11.0));
```

This is how the topology we defined in this part looks like.



Output on running the code is shown below (All the stats are collected using Flow monitor):

```
FlowID: 1 (TCP 10.1.1.1 / 49153 --> 10.2.1.1 / 8082)
  Tx Bytes: 1569932
  Rx Bytes: 1565228
  Tx Packets: 2717
  Rx Packets: 2709
  Time LastRxPacket: 11.7813s
  Lost Packets: 8
  Pkt Lost Ratio: 0.00294442
  Throughput: 1.22088 Mbps
  Mean{Delay}: 0.180583
  Mean{Jitter}: 0.00200458
FlowID: 2 (TCP 10.1.3.1 / 49153 --> 10.3.1.1 / 8081)
  Tx Bytes: 1218924
  Rx Bytes: 1218924
  Tx Packets: 2118
  Rx Packets: 2118
  Time LastRxPacket: 11.1204s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 1.01966 Mbps
  Mean{Delay}: 0.0642223
  Mean{Jitter}: 0.00169933
FlowID: 3 (TCP 10.3.2.1 / 49153 --> 10.2.2.1 / 8080)
  Tx Bytes: 618404
  Rx Bytes: 617816
  Tx Packets: 1074
  Rx Packets: 1073
  Time LastRxPacket: 11.3122s
  Lost Packets: 1
  Pkt Lost Ratio: 0.000931099
  Throughput: 0.50617 Mbps
  Mean{Delay}: 0.221981
  Mean{Jitter}: 0.00332768
FlowID: 4 (UDP 10.1.2.1 / 49153 --> 10.2.3.1 / 8084)
```

```
FlowID: 4 (UDP 10.1.2.1 / 49153 --> 10.2.3.1 / 8084)
  Tx Bytes: 4746060
  Rx Bytes: 3298320
  Tx Packets: 8789
  Rx Packets: 6108
  Time LastRxPacket: 13.0641s
  Lost Packets: 2681
  Pkt Lost Ratio: 0.30504
  Throughput: 2.27462 Mbps
  Mean{Delay}: 1.70639
  Mean{Jitter}: 0.00075395
FlowID: 5 (TCP 10.2.1.1 / 8082 --> 10.1.1.1 / 49153)
  Tx Bytes: 82440
  Rx Bytes: 82440
  Tx Packets: 1491
  Rx Packets: 1491
  Time LastRxPacket: 11.6309s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 0.0657182 Mbps
  Mean{Delay}: 0.0602063
  Mean{Jitter}: 2.8008e-07
FlowID: 6 (TCP 10.3.1.1 / 8081 --> 10.1.3.1 / 49153)
  Tx Bytes: 55124
  Rx Bytes: 55124
  Tx Packets: 1060
  Rx Packets: 1060
  Time LastRxPacket: 11.1204s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 0.0464188 Mbps
  Mean{Delay}: 0.0602321
  Mean{Jitter}: 4.73011e-05
FlowID: 7 (TCP 10.2.2.1 / 8080 --> 10.3.2.1 / 49153)
```

```
FlowID: 7 (TCP 10.2.2.1 / 8080 --> 10.3.2.1 / 49153)
  Tx Bytes: 31240
  Rx Bytes: 31240
  Tx Packets: 589
  Rx Packets: 589
  Time LastRxPacket: 11.3152s
  Lost Packets: 0
  Pkt Lost Ratio: 0
  Throughput: 0.0258718 Mbps
  Mean{Delay}: 0.100523
  Mean{Jitter}: 0.000365074
Total throughput of System: 0.780793 Mbps
Total packets transmitted: 17838
Total packets received: 15148
Total packets dropped: 2690
Packet Lost Ratio: 0.150802
```

Flow vs Throughput graph. I used Latex for plotting the graph. Attaching the latex code in file named as "graph_2a".

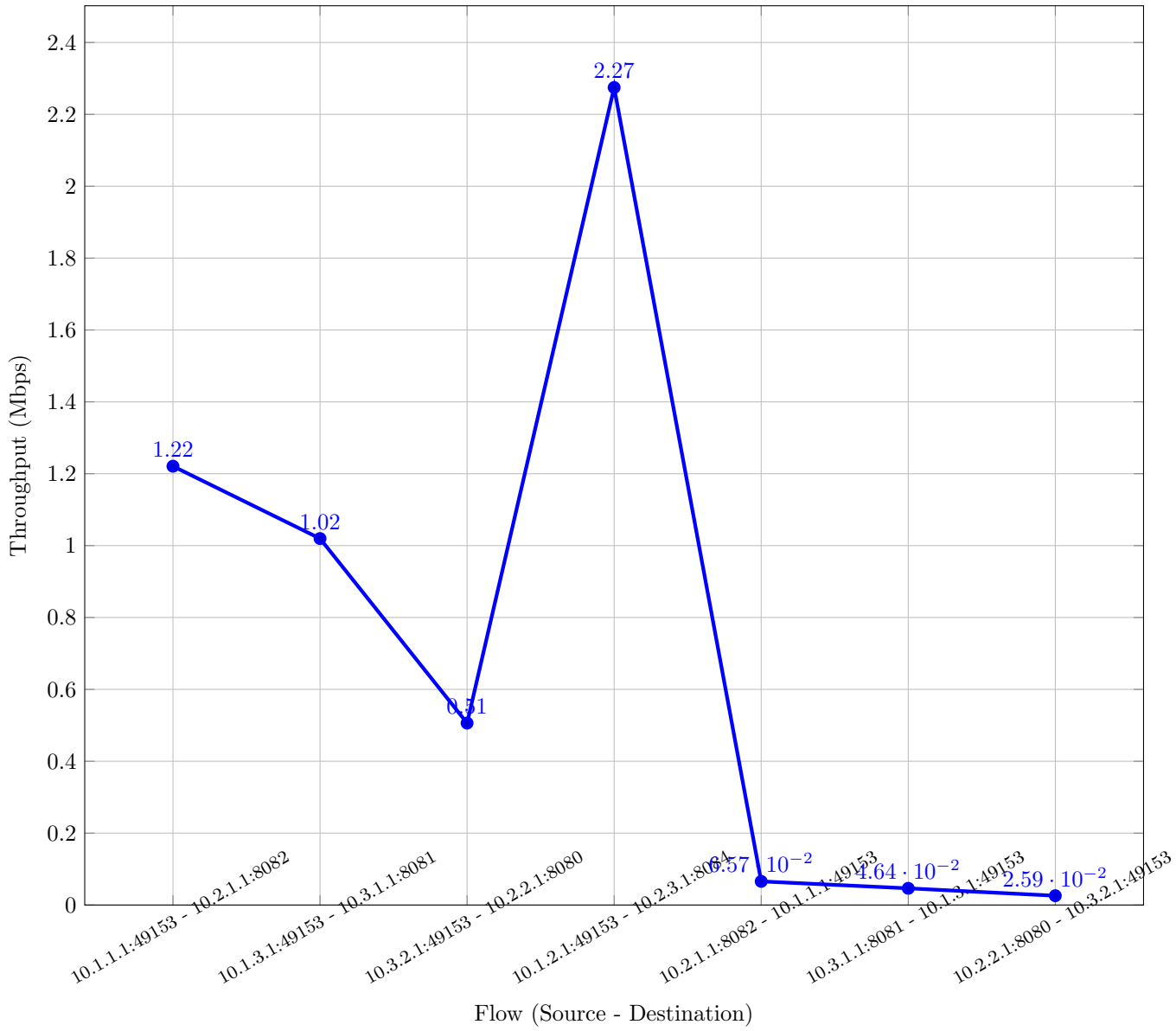


Figure : Flow vs Throughput (Mbps)

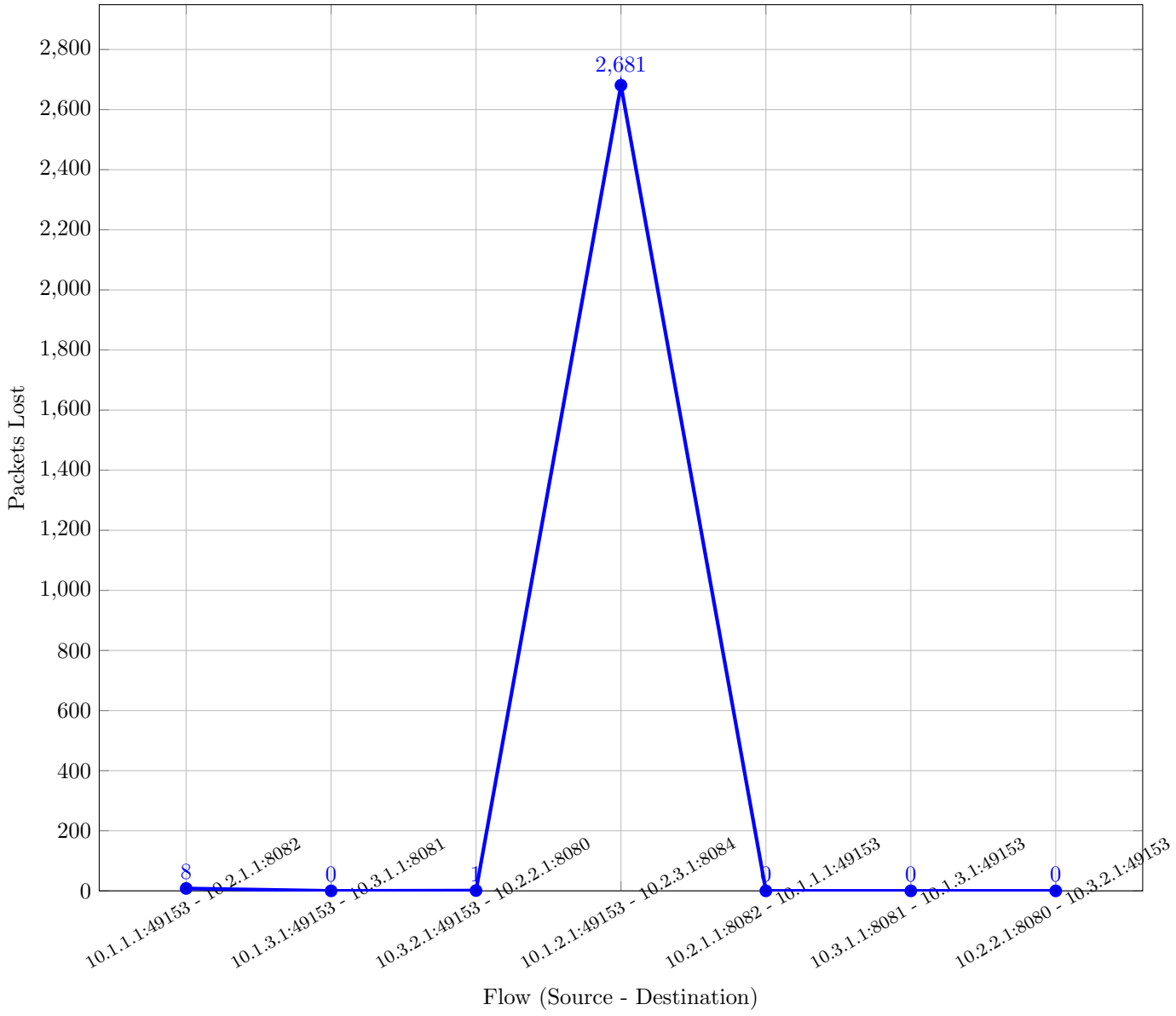


Figure : Flow vs Packets Lost

We can see we have lost 2681 packets for the flow 10.1.2.1:49153 - 10.2.3.1:8084. Corresponding packet losses are mentioned for each flow.

I have saved the routing tables in file names as "p2_routing_tables.txt" And the netanim file for this part is saved as p2.xml.

PART B

In this part, we are asked to introduce an error model. So we replace all the connections made in PART A with udpEchoClient and udpEchoServer flows. We show the changes for one of the connection flow, we repeat the same procedure with all the other flows.

```
int main(int argc, char *argv[]) {

    double error = 0.2;
    uint32_t rngSeed = 1;
    std::string errorUnit = "packet";
    ns3::RateErrorModel::ErrorUnit errorUnitEnum = ns3::RateErrorModel::ERROR_UNIT_PACKET;
    // Enable logging for this component
    LogComponentEnable("P2B", LOG_LEVEL_ERROR);

    CommandLine cmd;
    cmd.AddValue("RngRun", "Seed for the random number generator", rngSeed);
    cmd.AddValue("error", "Error rate of the error model", error);
    cmd.AddValue("errorUnit", "Error unit of the error model", errorUnit);
    cmd.Parse(argc, argv);

    if (errorUnit == "byte") {
        errorUnitEnum = ns3::RateErrorModel::ERROR_UNIT_BYTE;
    } else if (errorUnit == "bit") {
        errorUnitEnum = ns3::RateErrorModel::ERROR_UNIT_BIT;
    }else if (errorUnit == "packet") {
        errorUnitEnum = ns3::RateErrorModel::ERROR_UNIT_PACKET;
    }
    else {
        NS_FATAL_ERROR("Invalid error unit specified. Correct values are: byte, bit, packet");
    }

    // Setting the random number generator seed value
    RngSeedManager::SetSeed(rngSeed);

    // Setting the default time resolution
    Time::SetResolution(Time::NS);
```

```
    // Creating UdpEchoServer applications for n7 and n4
    UdpEchoServerHelper echoServer(9);
    ApplicationContainer serverApps = echoServer.Install(NodesP2.Get(1));
    serverApps.Start(Seconds(1.0));
    serverApps.Stop(Seconds(11.0));

    // Creating UdpEchoClient applications
    UdpEchoClientHelper echoClient(InterfacesP2.GetAddress(1), 9);
    echoClient.SetAttribute("MaxPackets", UintegerValue(10));
    echoClient.SetAttribute("Interval", TimeValue(MilliSeconds(500.0)));
    echoClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApps = echoClient.Install(NodesP3.Get(1));
    clientApps.Start(Seconds(2.0));
    clientApps.Stop(Seconds(11.0));
```

Now we install our error model on the links and devices and then run the simulator, monitor the stats with the help of flowmonitor and then destroy the simulator.

```
    // Defining the error model
    Ptr<RateErrorModel> errorModel = CreateObject<RateErrorModel>();
    errorModel->SetAttribute("ErrorRate",DoubleValue(error));
    errorModel->SetAttribute("ErrorUnit",EnumValue(errorUnitEnum));

    // Installing and applying error model on the links and devices

    for (uint32_t i = 0; i < DevicesP1.GetN(); ++i){
        DevicesP1.Get(i)->SetAttribute("ReceiveErrorModel", PointerValue(errorModel));
    }

    for (uint32_t i = 0; i < DevicesP2.GetN(); ++i){
        DevicesP2.Get(i)->SetAttribute("ReceiveErrorModel", PointerValue(errorModel));
    }

    for (uint32_t i = 0; i < DevicesP3.GetN(); ++i){
        DevicesP3.Get(i)->SetAttribute("ReceiveErrorModel", PointerValue(errorModel));
    }

    Simulator::Stop (Seconds (100.0));

    AnimationInterface anim("p2_b.xml");
    FlowMonitorHelper flowHelper;
    Ptr<FlowMonitor> monitor = flowHelper.InstallAll();

    // Running the simulation, printing the statistics and destroying the simulator.
    Simulator::Run();
    printStats (flowHelper, true);
    monitor->SerializeToXmlFile("p2_b_flow_mon.xml", true, true);
    Simulator::Destroy();
```

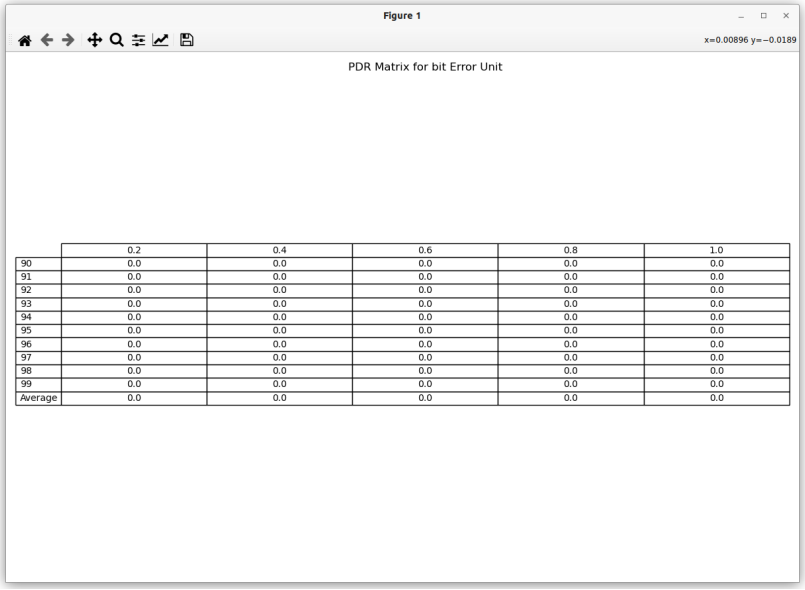
```
    // Creating UdpEchoServer applications for n7 and n4
    UdpEchoServerHelper echoServer(9);
    ApplicationContainer serverApps = echoServer.Install(NodesP2.Get(1));
    serverApps.Start(Seconds(1.0));
    serverApps.Stop(Seconds(11.0));

    // Creating UdpEchoClient applications
    UdpEchoClientHelper echoClient(InterfacesP2.GetAddress(1), 9);
    echoClient.SetAttribute("MaxPackets", UintegerValue(10));
    echoClient.SetAttribute("Interval", TimeValue(MilliSeconds(500.0)));
    echoClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApps = echoClient.Install(NodesP3.Get(1));
    clientApps.Start(Seconds(2.0));
    clientApps.Stop(Seconds(11.0));
```

To automate the tasks, I used python script named "p2_b_script.py". Simply run this script to generate all the tables and corresponding graphs. We print the table on the terminal as well. Attached all the corresponding screenshots.

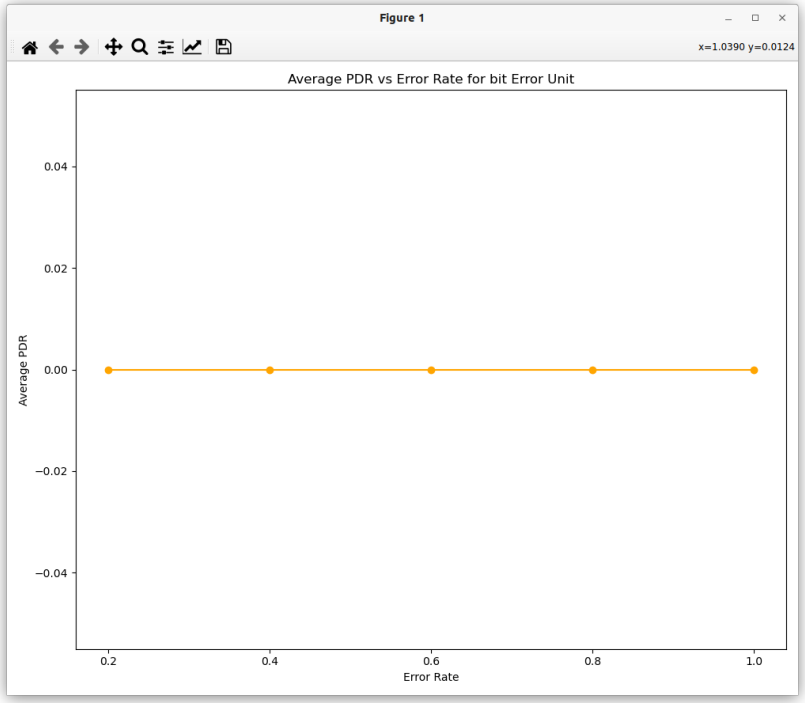
Now to select the RngRun values we take last 2 digits of my roll number. My id is 12241190. So RngRun values starts from 90 and goes upto 99.

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import subprocess

rng_runs = range(90, 100)
error_rates = [0.2, 0.4, 0.6, 0.8, 1.0]
errorUnits = ["bit", "byte", "packet"]

for errorUnit in errorUnits:

    pdr_values = []

    # Collecting PDR values for each combination of RngRun and ErrorRate
    for rng_run in rng_runs:
        for error_rate in error_rates:
```

Table for Bit error unit

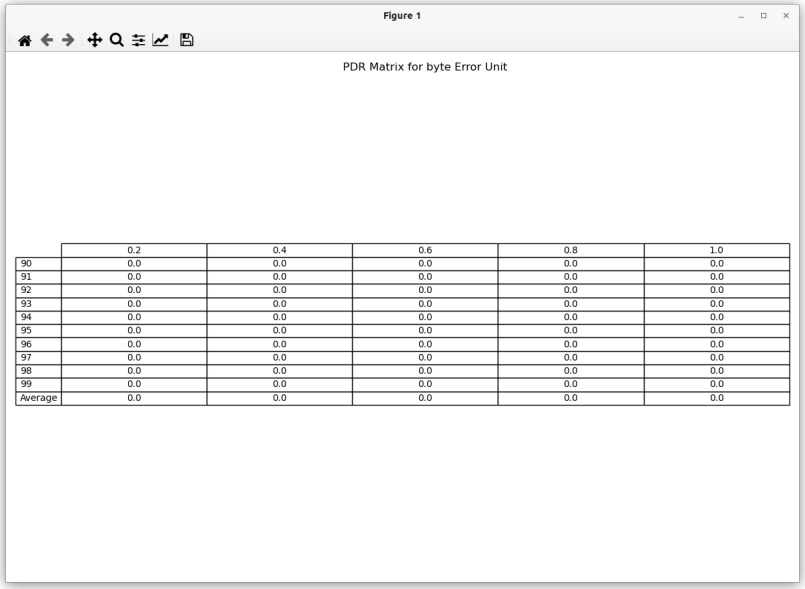


	0.2	0.4	0.6	0.8	1.0
90	0.0	0.0	0.0	0.0	0.0
91	0.0	0.0	0.0	0.0	0.0
92	0.0	0.0	0.0	0.0	0.0
93	0.0	0.0	0.0	0.0	0.0
94	0.0	0.0	0.0	0.0	0.0
95	0.0	0.0	0.0	0.0	0.0
96	0.0	0.0	0.0	0.0	0.0
97	0.0	0.0	0.0	0.0	0.0
98	0.0	0.0	0.0	0.0	0.0
99	0.0	0.0	0.0	0.0	0.0
Average	0.0	0.0	0.0	0.0	0.0



ErrorRate	0.2	0.4	0.6	0.8	1.0
RngRun					
90	0.0	0.0	0.0	0.0	0.0
91	0.0	0.0	0.0	0.0	0.0
92	0.0	0.0	0.0	0.0	0.0
93	0.0	0.0	0.0	0.0	0.0
94	0.0	0.0	0.0	0.0	0.0
95	0.0	0.0	0.0	0.0	0.0
96	0.0	0.0	0.0	0.0	0.0
97	0.0	0.0	0.0	0.0	0.0
98	0.0	0.0	0.0	0.0	0.0
99	0.0	0.0	0.0	0.0	0.0
Average	0.0	0.0	0.0	0.0	0.0

Table for Byte error unit



	0.2	0.4	0.6	0.8	1.0
90	0.0	0.0	0.0	0.0	0.0
91	0.0	0.0	0.0	0.0	0.0
92	0.0	0.0	0.0	0.0	0.0
93	0.0	0.0	0.0	0.0	0.0
94	0.0	0.0	0.0	0.0	0.0
95	0.0	0.0	0.0	0.0	0.0
96	0.0	0.0	0.0	0.0	0.0
97	0.0	0.0	0.0	0.0	0.0
98	0.0	0.0	0.0	0.0	0.0
99	0.0	0.0	0.0	0.0	0.0
Average	0.0	0.0	0.0	0.0	0.0

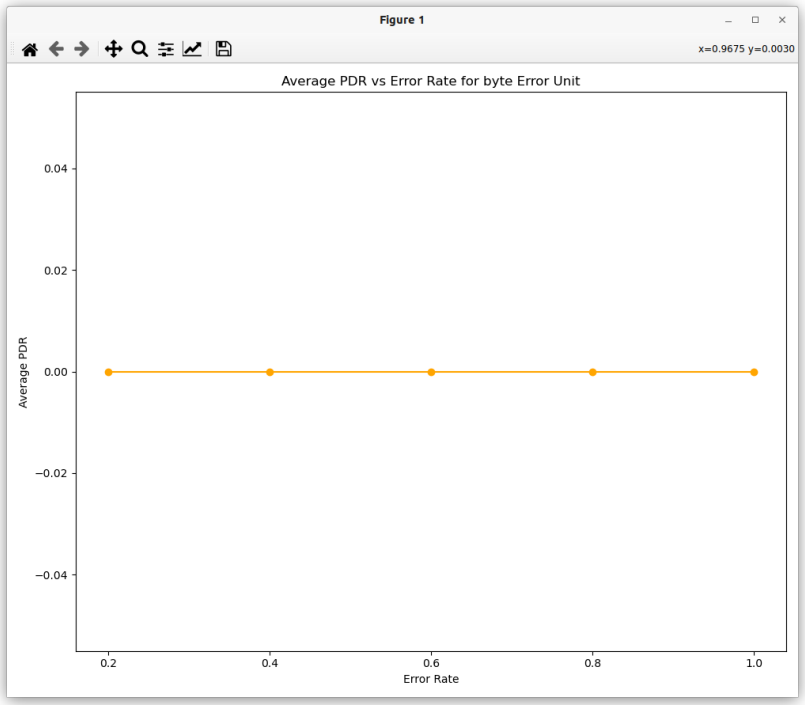
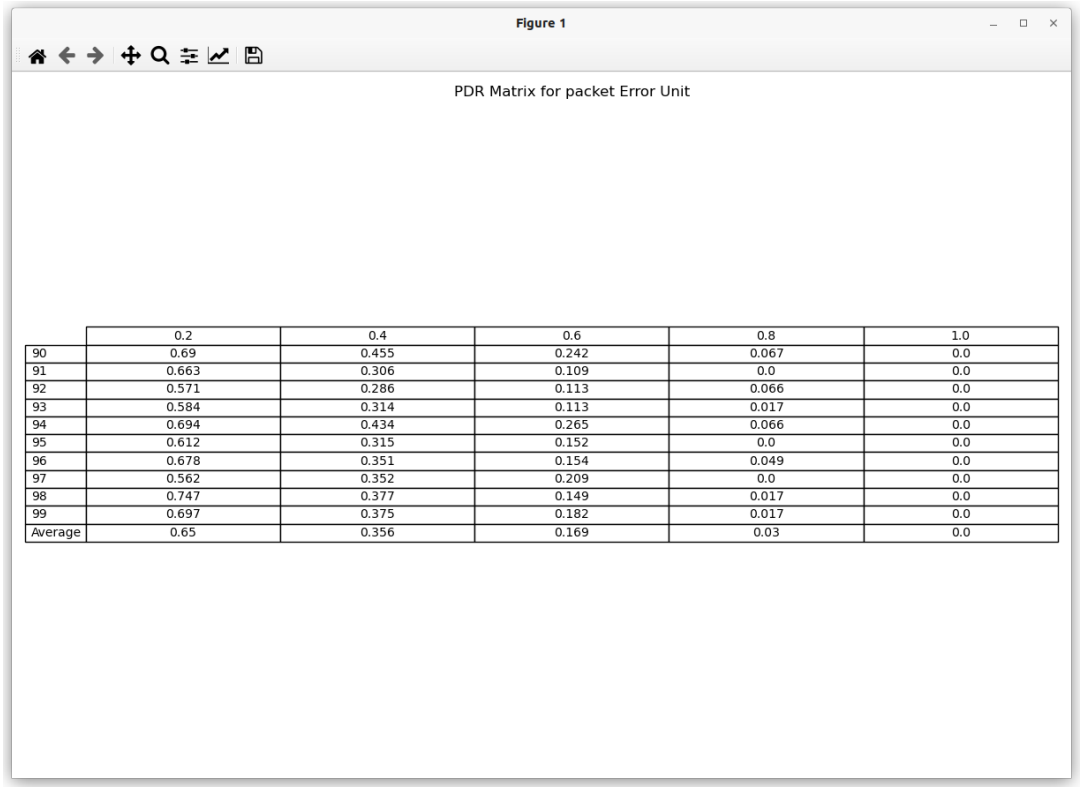
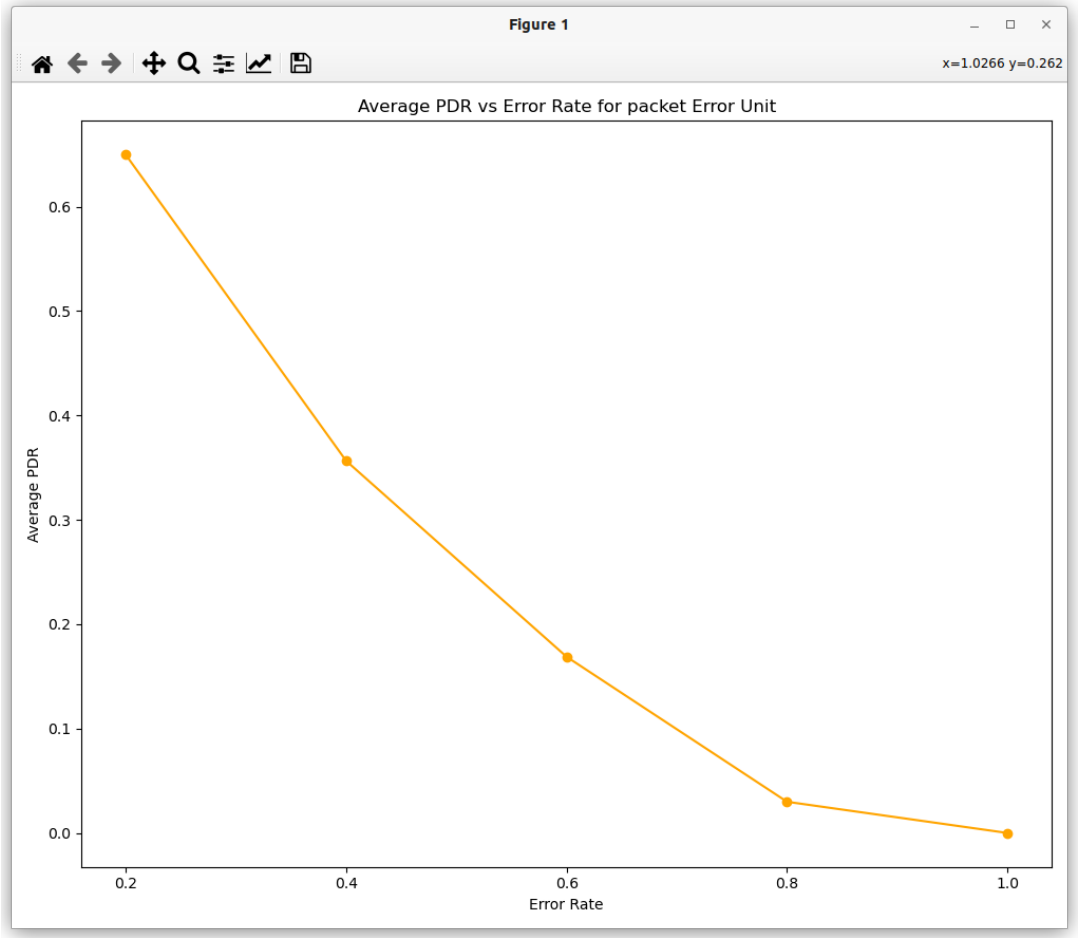


Table for Packet error unit



	0.2	0.4	0.6	0.8	1.0
90	0.69	0.455	0.242	0.067	0.0
91	0.663	0.306	0.109	0.0	0.0
92	0.571	0.286	0.113	0.066	0.0
93	0.584	0.314	0.113	0.017	0.0
94	0.694	0.434	0.265	0.066	0.0
95	0.612	0.315	0.152	0.0	0.0
96	0.678	0.351	0.154	0.049	0.0
97	0.562	0.352	0.209	0.0	0.0
98	0.747	0.377	0.149	0.017	0.0
99	0.697	0.375	0.182	0.017	0.0
Average	0.65	0.356	0.169	0.03	0.0

```
ErrorRate     0.2     0.4     0.6     0.8  1.0
RngRun
90         0.6900  0.4550  0.2420  0.0670  0.0
91         0.6630  0.3060  0.1090  0.0000  0.0
92         0.5710  0.2860  0.1130  0.0660  0.0
93         0.5840  0.3140  0.1130  0.0170  0.0
94         0.6940  0.4340  0.2650  0.0660  0.0
95         0.6120  0.3150  0.1520  0.0000  0.0
96         0.6780  0.3510  0.1540  0.0490  0.0
97         0.5620  0.3520  0.2090  0.0000  0.0
98         0.7470  0.3770  0.1490  0.0170  0.0
99         0.6970  0.3750  0.1820  0.0170  0.0
Average    0.6498  0.3565  0.1688  0.0299  0.0
```



From this graph it is clearly visible that increasing the error rate decreases the Average Packet Delivery Ratio (PDR). It is easy to understand that it is because on increasing the error rate, more packets are being dropeed, so PDR decreases. With error rate = 1, all the packets are dropped, so avg PDR = 0. Also for error units byte and bits, the error rates are very large so all the packets are lost and we have 0 value in all the fields of the table.