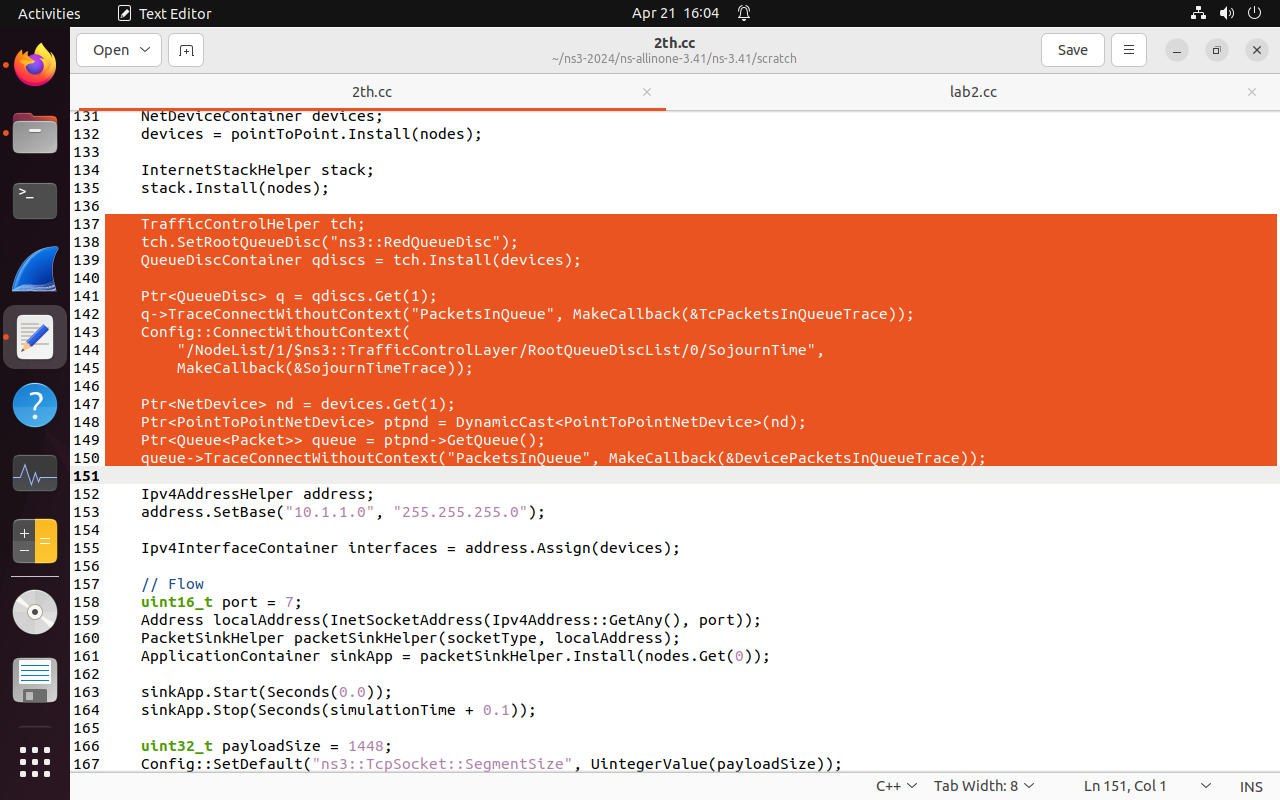
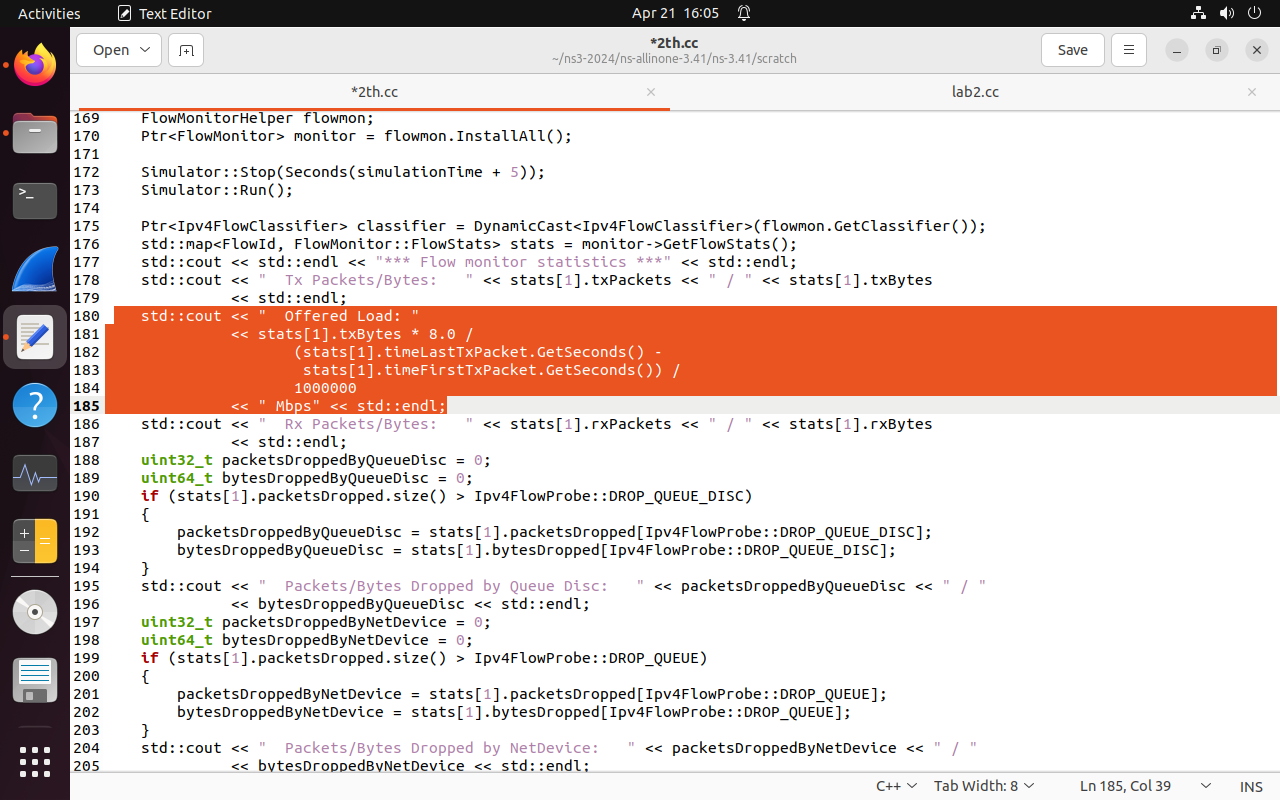
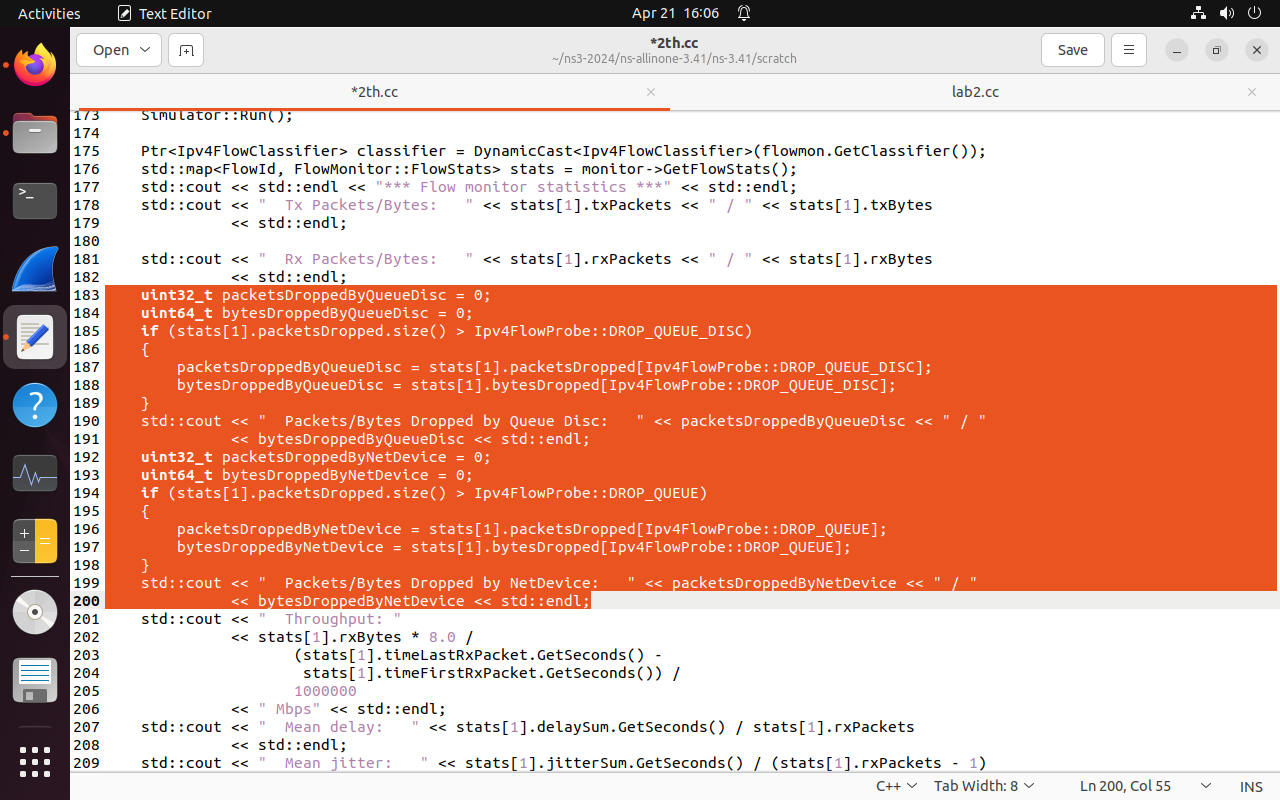
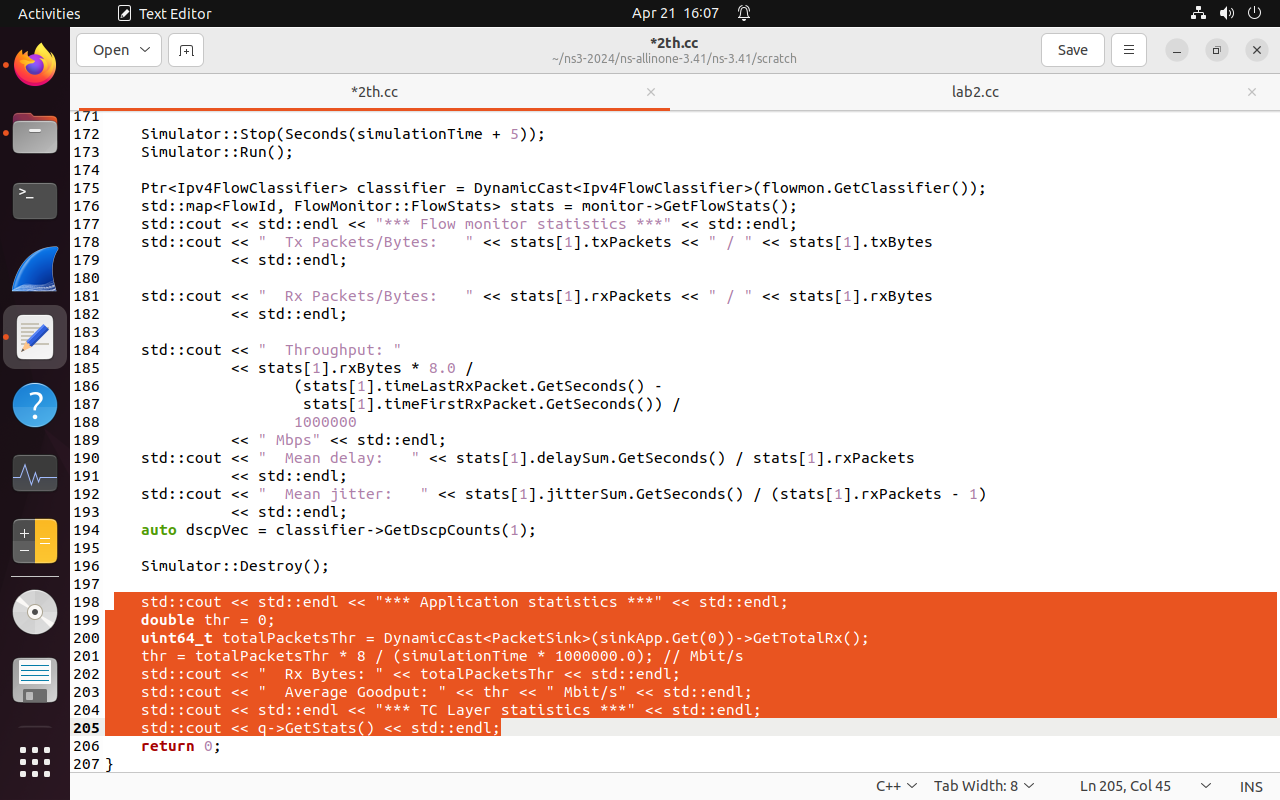
**DELETE CODE**









**CHANGES IN CODE**

std::string transportProt = "Udp";

nodes.Create(4);

NetDeviceContainer devices01;

devices01 = pointToPoint.Install(nodes.Get(0),nodes.Get(1));

NetDeviceContainer devices12;

devices12 = pointToPoint.Install(nodes.Get(1),nodes.Get(2));

NetDeviceContainer devices23;

devices23 = pointToPoint.Install(nodes.Get(2),nodes.Get(3));

//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);

address.SetBase("10.1.3.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces23 = address.Assign(devices23);

Ipv4GlobalRoutingHelper::PopulateRoutingTables(); // type these global routing

// Flow ---> rename as UDP flow, copy these same code and paste after apps.stop() and make changes

//UDP Flow

InetSocketAddress rmt(interfaces01.GetAddress(0), port); //change interfaces01

apps.Add(onoff.Install(nodes.Get(3))); //change .Get(3)

// TCP Flow

uint16\_t porttcp = **9**; //change to 9

socketType = "ns3::TcpSocketFactory"; //add these line

Address **localAddresstcp**(InetSocketAddress(Ipv4Address::GetAny(), **porttcp** ));

PacketSinkHelper **packetSinkHelpertcp** (socketType, **localAddresstcp**);

ApplicationContainer **sinkApptcp** = **packetSinkHelpertcp**.Install(nodes.Get(0));

**sinkApptcp**.Start(Seconds(**0.5**));

**sinkApptcp**.Stop(Seconds(simulationTime + 0.1));

**Delete 2 line code from here**

OnOffHelper **onofftcp**(socketType, Ipv4Address::GetAny());

**onofftcp**.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));

**onofftcp**.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));

**onofftcp**.SetAttribute("PacketSize", UintegerValue(payloadSize));

**onofftcp**.SetAttribute("DataRate", StringValue("50Mbps")); // bit/s

ApplicationContainer **appstcp**;

InetSocketAddress **rmttcp**(**interfaces01**.GetAddress(0), **porttcp**); //change to interfaces01

**rmttcp**.SetTos(0xb8);

AddressValue **remoteAddresstcp**(**rmttcp**);

**onofftcp**.SetAttribute("Remote", **remoteAddresstcp**);

**appstcp**.Add(**onofftcp**.Install(nodes.Get(**2**))); //change to .Get(2)

**appstcp**.Start(Seconds(**1.5**)); // change to Seconds(1.5)

**appstcp**.Stop(Seconds(simulationTime + 0.1));

// write these code in the flow monitor statistics section

std::cout << " Dropped Packets/Bytes: " << stats[1].lostPackets

<< " / " << stats[1].rxBytes << std::endl;

//std::cout << " Mean jitter: " << stats[1].jitterSum.GetSeconds() / (stats[1].rxPackets - 1)

// << std::endl;

// auto dscpVec = classifier->GetDscpCounts(1);

**//Add these for loop**

for (std::map<FlowId, FlowMonitor::FlowStats>::const\_iterator iter = stats.begin (); iter != stats.end (); ++iter)

{

Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);

std::cout << "Flow ID: " << iter->first << " Src Addr " << t.sourceAddress << " Dst Addr " << t.destinationAddress<< std::endl;

std::cout << "Tx Packets = " << iter->second.txPackets<< std::endl;

std::cout << "Rx Packets = " << iter->second.rxPackets<< std::endl;

std::cout << "Lost Packets = " << iter->second.lostPackets<< std::endl;

std::cout << "Throughput = " << iter->second.rxBytes \* 8.0 / (iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds()) / 1000000 << " Kbps"<< std::endl;

}