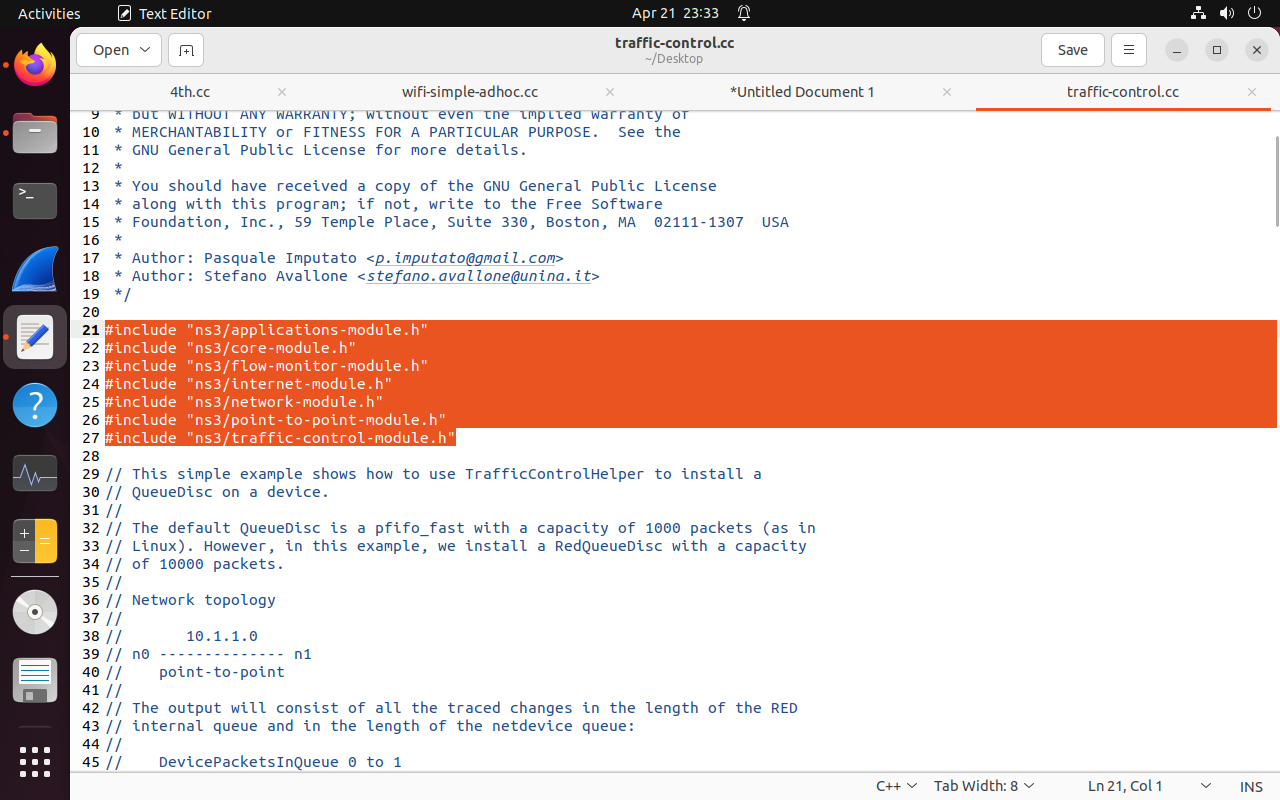
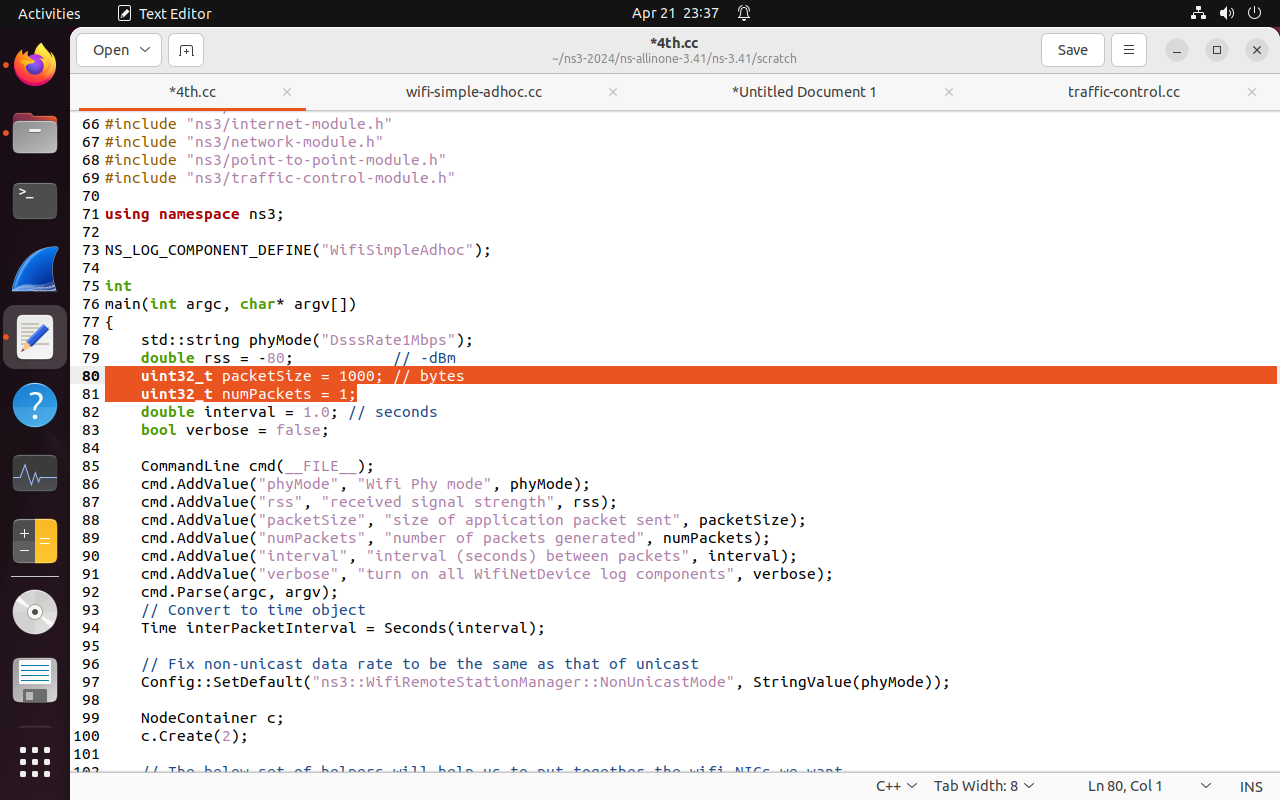
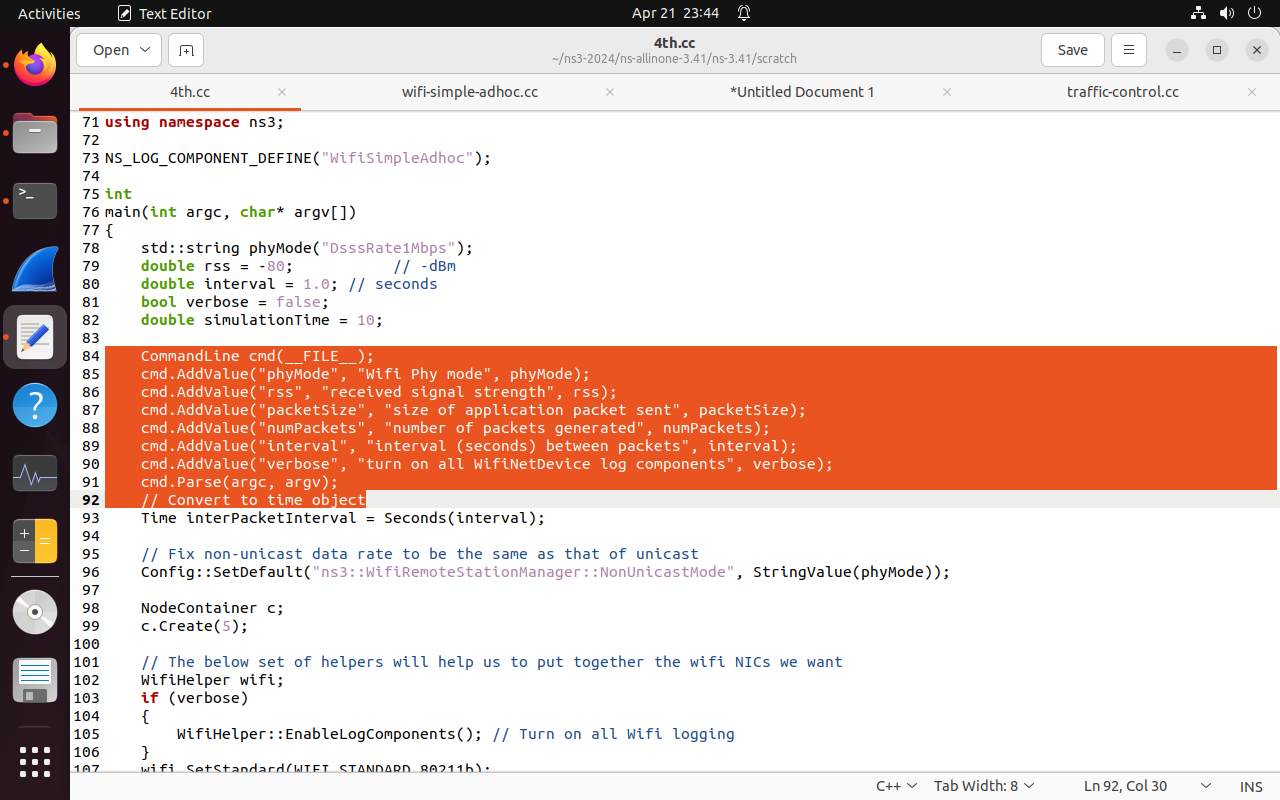
**DELETE CODE**

**Copy these Header files from traffic-contril.cc into wifi-simple-adhoc.cc**

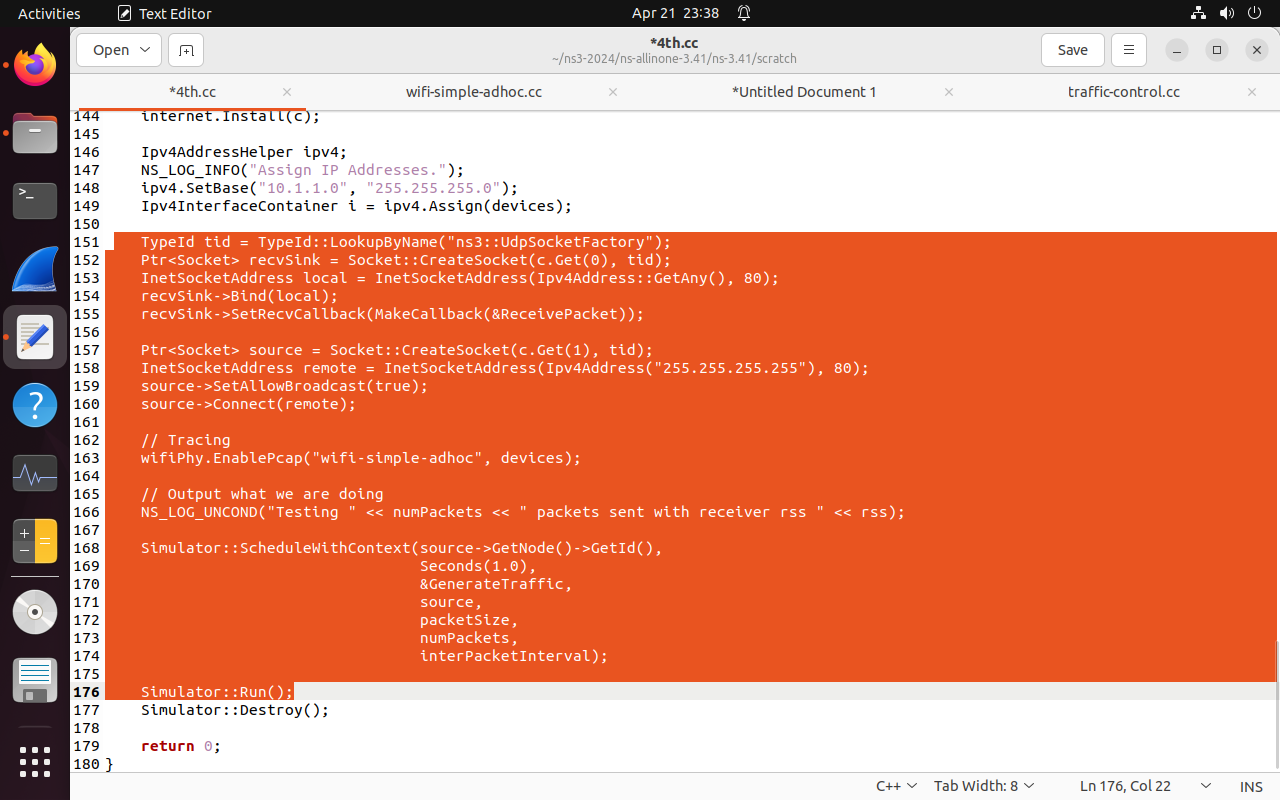


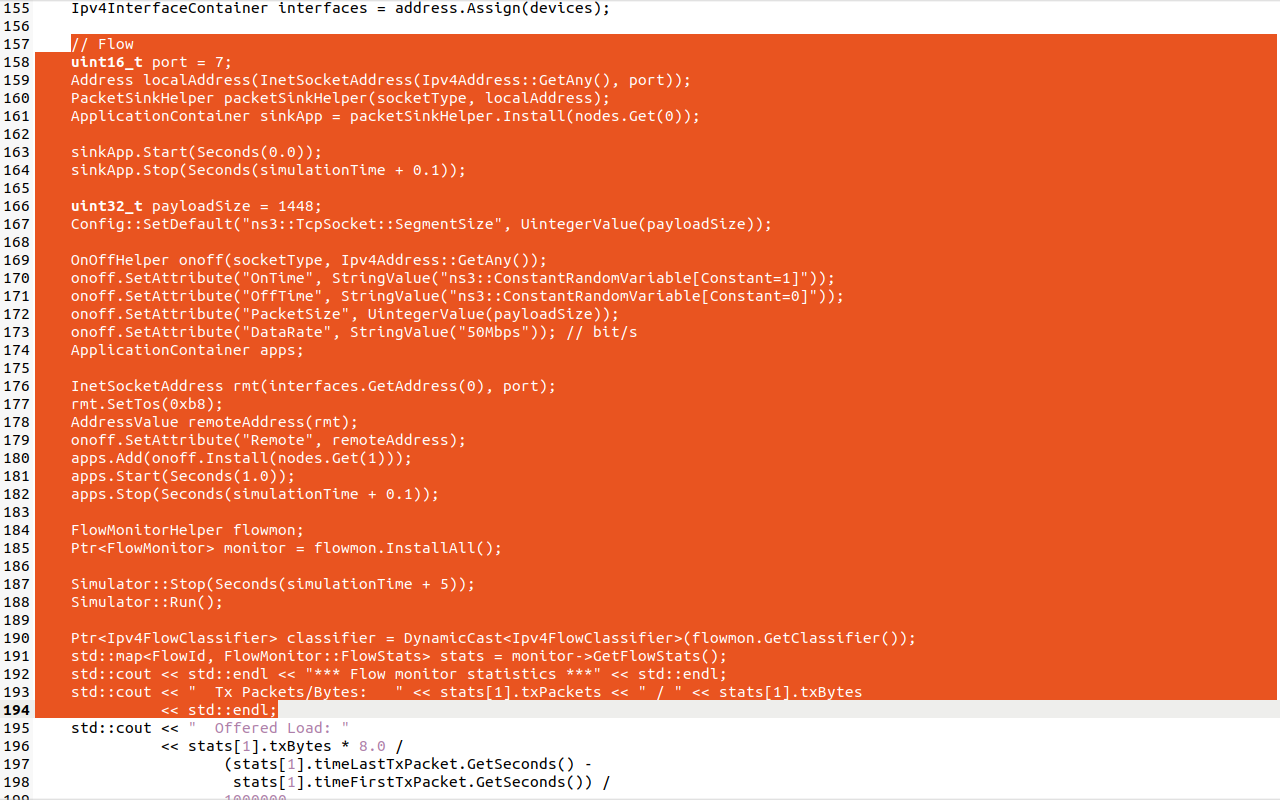






Delete even Simulator::Run()



 **Copy these Header files from traffic-contril.cc into wifi-simple-adhoc.cc**

**CHANGES IN CODE**

double simulationTime = 10; // seconds //add these

c.Create(5);

//positionAlloc->Add(Vector(0.0, 0.0, 0.0));

//positionAlloc->Add(Vector(5.0, 0.0, 0.0));

positionAlloc->Add(Vector(5.0, 5.0, 0.0));

positionAlloc->Add(Vector(5.0, 10.0, 0.0)); // add these lines

positionAlloc->Add(Vector(15.0, 0.0, 0.0));

//copy from traffic-control.cc and paste here

// Flow

uint16\_t port = 7;

Address localAddress(InetSocketAddress(Ipv4Address::GetAny(), port));

PacketSinkHelper packetSinkHelper**("ns3::UdpSocketFactory"**, localAddress);

ApplicationContainer sinkApp = packetSinkHelper.Install(**c.Get(4)**);

sinkApp.Start(Seconds(0.0));

sinkApp.Stop(Seconds(simulationTime + 0.1));

uint32\_t payloadSize = 1448;

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

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", UintegerValue(payloadSize));

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

ApplicationContainer apps;

InetSocketAddress rmt(**i**.GetAddress(**4**), port);

rmt.SetTos(0xb8);

AddressValue remoteAddress(rmt);

onoff.SetAttribute("Remote", remoteAddress);

apps.Add(onoff.Install(**c**.Get(1)));

apps.Start(Seconds(1.0));

apps.Stop(Seconds(simulationTime + 0.1));

FlowMonitorHelper flowmon;

Ptr<FlowMonitor> monitor = flowmon.InstallAll();

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;