ENSC 427: COMMUNICATION NETWORKS

Spring 2025

Simulation and impact of a DDoS attack in a 5G network

https://www.sfu.ca/~sss78/ENSC427_webpage.html

Sandhu, Sajan

301475886

sss78@sfu.ca

Team 5

Table of Contents

Abstract

5G networks are being evermore utilized as time goes on, with the high speed and low latency they bring allowing users to connect with each other faster than they've ever been able to before. Though this increasing popularity brings more threats to 5G networks, such as DDoS attacks which can impede a user's ability to properly utilize the network. The aim of this project is to simulate a 5G network and a DDoS attack on that network while varying the number of attackers to observe how it affects network performance.

Introduction

Communications technology is constantly advancing at a rapid pace, at that quick pace is easily seen in the various generations of mobile network standards. In the span of less than a single lifetime, mobile network standards have advanced several generations, from the first 1G networks utilizing analog communication to new 5G networks which have high speeds to keep up with the ever increasing demand for quick communication in this digital era.

Though one key aspect has stayed the same throughout those many years, and it is the presence of malicious attackers. For every generation prior and every generation to yet be made, the need to account for malicious attackers will always exist. To help minimize the damage of attacks that will surely occur, it is crucial to first understand how those attacks affect networks before one can implement any defenses against them.

The goal of this project is to simulate a DDoS attack on a 5G network using NS-3. This will be accomplished by utilizing the 5G-LENA module for NS-3[1], and creating an end-to-end 5G New Radio (5G NR) network which will then be attacked using a UDP flood. The resulting simulation will be examined, and statistics such as how the throughput of a user varied and the latency of user packets will be collected and discussed.

Background

DDoS Attacks

A denial of service (DoS) attack is simply an attack conducted by a malicious entity or entities which denies a legitimate user or users access to a service or other network resource such as bandwidth[2]. A distributed denial of service (DDoS) attack is a more specific DoS, and is implemented by having various distributed attackers go through with the denial of service attack rather than a single centralized attacker. The various attacker agents are activated from a single master and all work together to try to deny service to a shared target[3]. DDoS attacks take many approaches to deny service, such as using a volume based approach by flooding the target server with UDP or ICMP packets, as well as attacks that take advantage of protocols such as a SYN flood[3]. DDoS attacks are very common approaches to attacking a network[4] and have to be accounted for when attempting to improve security for any network.

5G Cellular Networks

5th Generation Technology or simply 5G technology is the next generation of cellular network technology which succeeds the prior generation of 4G LTE technology. It is characterized by low latency and very high bandwidth to help speed up future wireless communication[5]. It uses millimetre wave technology for transmitting data though due to the lower transmission range compared to 4G LTE, 5G deployment is focused on dense urban centres rather than more widespread population centres. 5G New Radio (5G NR) is the global standard for 5G technology, and is developed by 3rd Generation Partnership Program (3GPP)[6].

Literature Review

As 5G is a relatively new technology, there has been an increasing amount of research done into 5G networks as a whole, and particularly 5G network security is actively being extensively researched. The need for this type of research is also ever growing, since DDoS attacks are shown to be increasing in frequency and intensity as launching large scale DDoS attacks has become easier over time[4]. This has led to a number of simulations of attacks such as DDoS attacks on 5G networks to examine their effects[2][7][8]. Additionally, the purpose of this type of research is to eventually develop countermeasures against network attacks, and there has been research conducted into the progress and effectiveness of these countermeasures[9][10].

A variety of simulations with varying topologies and parameters have been conducted into seeing how DDoS attacks affect 5G network performance. Work done in literature [2] replicated a 5G network using the NetSim software, using non-MIMO and MIMO topologies. They examined a decrease in throughput for authentic users when conducting their simulated attacks, and saw that MIMO topologies were more resistant to DDoS attacks compared to non-MIMO topologies. Authors in [7] conducted their simulation using a physical Samsung phone as an end user over a 5G network, and saw that some attacks managed to almost completely deny any bandwidth to that user. In [8], authors examined broad security issues in 5G networks and conducted a simple 5G network simulation showing network resources being almost entirely denied due to the attack.

With regards to countermeasures for these attacks, research is also underway in examining possible solutions. In literature [10] the authors discuss how Software Defined Networks (SDNs) are commonly proposed solutions for helping with attacks on networks in general, but also discuss the limitations of SDNs and security issues that arise when integrating them into 5G networks. Researchers in [9] examine machine learning models to help detect any DDoS attacks that are occurring in a slice of a 5G network, and find that they can reliably detect when DDoS attacks occur.

Simulation

Overview

As mentioned, this simulation will use the NS-3 simulator and the 5G-LENA module[1] for NS-3 to create an end-to-end simulation of a 5G network. The 5G-LENA simulates many aspects of a 5G network such as end user equipment, the 5G base stations named Next Generation Node B, as well as a gateway to the core services of a 5G network. The various parameters of the 5G network will be based on the IMT-2020 standard[11] as these standards define the minimum specifications a 5G network implementation must meet to be considered correctly functioning.

A control scenario will be established with only the legitimate user sending data to a target server. Then, the succeeding scenarios will add on varying numbers of malicious attackers as they attempt a UDP flood based DDoS attack. The number of packets sent by the user and the attackers will be tracked, along with the number of the packets able to make it to the server by the legitimate user and attackers. This will allow us to see the amount of packet loss occurring during an attack, as well as the throughput for the user. Additionally parameters such as latency and jitter will be observed as well. To help track these parameters, the NS-3 module named Flow Monitor will be used to collect data from the network.

Topology & Parameters

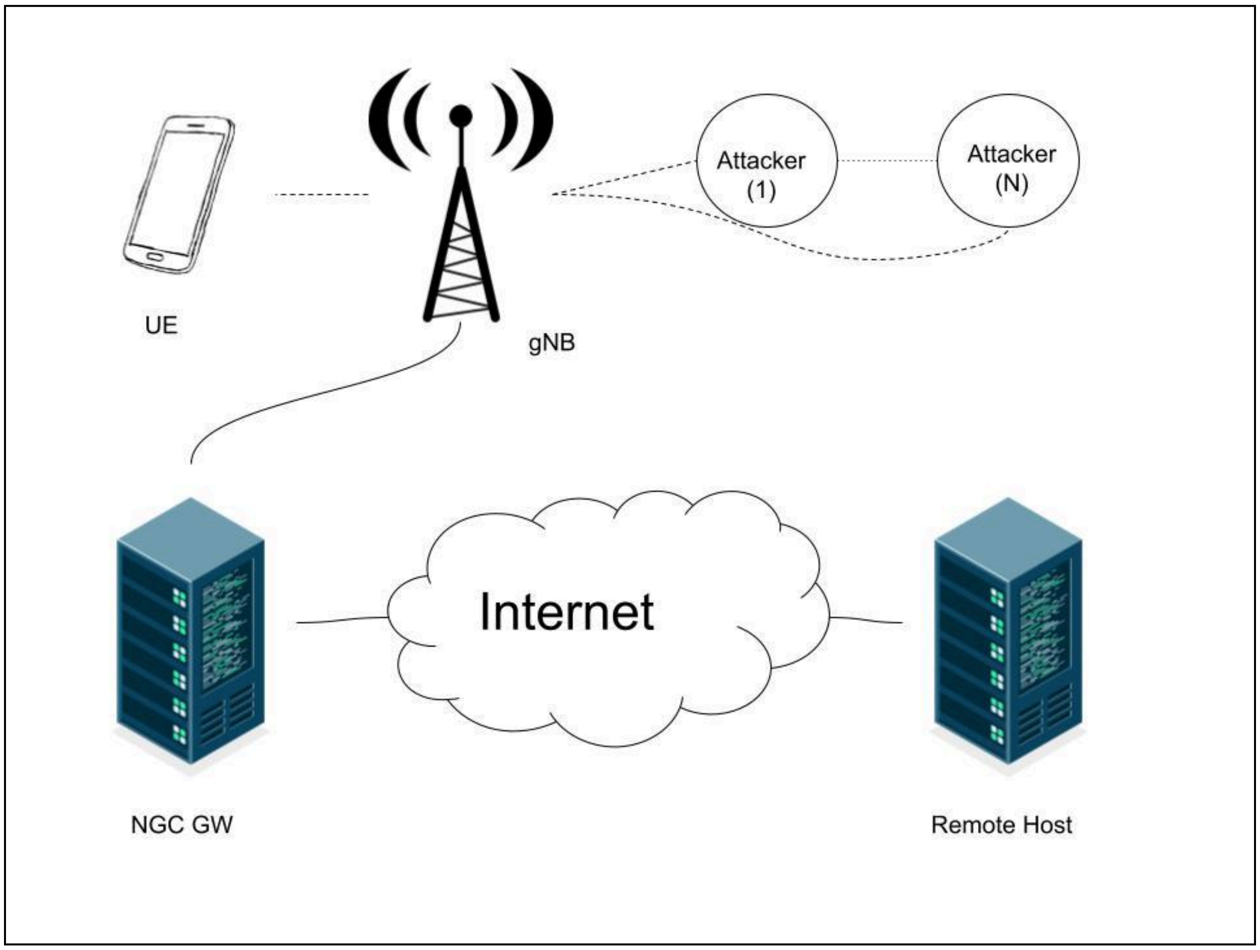

Fig. 1

As can be seen in figure 1, an authentic user is represented by the user equipment image which is associated with a 5G base station. Additionally, multiple attackers are also associated with the same base station. This base station then connects to the next generation core gateway (NGC GW) of the 5G network. The NGC GW represents the core services provided by a 5G network. This NGW GW is then connected to the internet, where a remote host is also connected. This remote host is the end destination for packets sent by the user and attackers.

With regards to specific simulation parameters, the total simulation time will be 5 seconds with the user starting to send packets at 0.5 seconds, and at 1.5 seconds the attackers will start their DDoS attack. The number of attackers will vary from 0 in the control scenario, to 1,5,10, or 15 in scenarios where an attack is being conducted. UDP packets will be used for both the user and the attackers, and the packet size will be 1000 bytes for all packets. The user will send at a rate of 1 MB/s, while attackers will send at a rate of 2 MB/s.

Simulation Results & Discussion

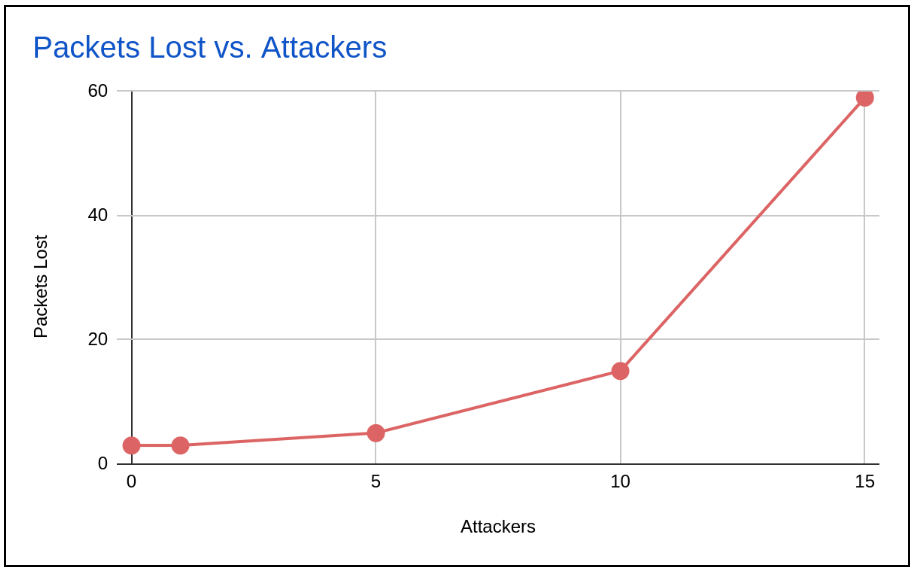

Fig. 2

Figure 2 above shows the total numbers of packets lost for the user as the number of attackers in the simulation varies. This amount was measured by subtracting the number of packets that arrived to the remote server from the total packets sent by the user, and this difference is the packet loss. Initially, the number of packets lost is minimal with almost zero percent packet loss. Loss still occurs even with zero attackers due to the fact that the 5G network portion of simulation naturally assumes some loss due to factors such as path loss and noise which affects transmission reliability. With the addition of one attacker, the packet loss does not increase due to the minimal effect a single attacker has on the network. Though as more attackers are contributing to the DDoS attack, the packet loss increases rapidly. By the max 15 attackers test, almost 60 packets are lost due to the attack. A total number of 4500 packets were sent by the user during the simulation time, which results in the percent packet loss being roughly 1.3% which is an acceptable amount. Although the relative loss is still low, the increasing amount of packet loss as more attackers are added does show the negative effect that the DDoS attack has on the end user experience.

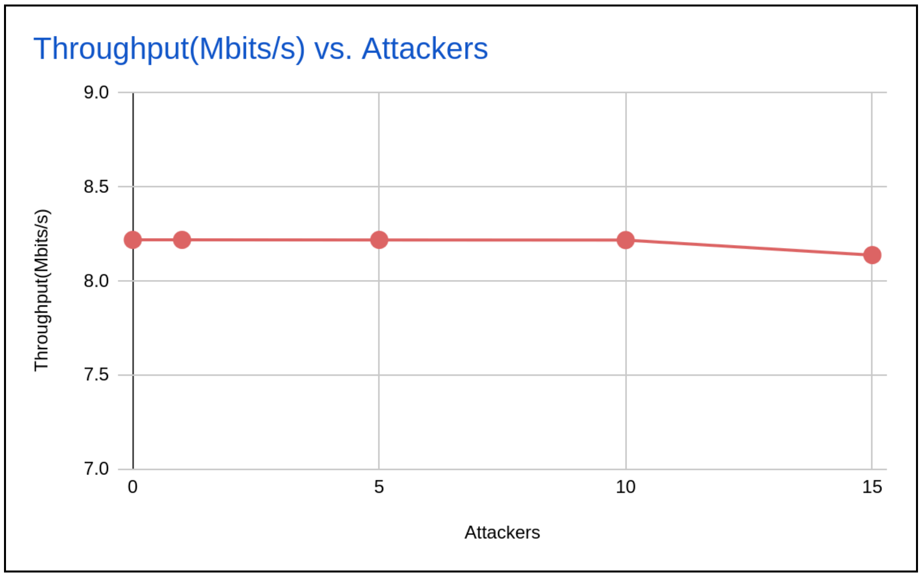

Fig. 3

Continuing, it can be observed in figure 3 above that throughput slightly decreases as more attackers are added. This is an expected result, as it results from packet loss increasing while the rate of sent packets remains the same. This results in lower throughput from the perspective of the server. Additionally, no throughput issues occur due to lack of total bandwidth from the 5G network, as the bandwidth of the 5G network is in the range of gigabits per second. As the total data rate of the users and attackers is not around that gigabit per second range, the 5G network bandwidth is not saturated and does not become a constraint for the user.

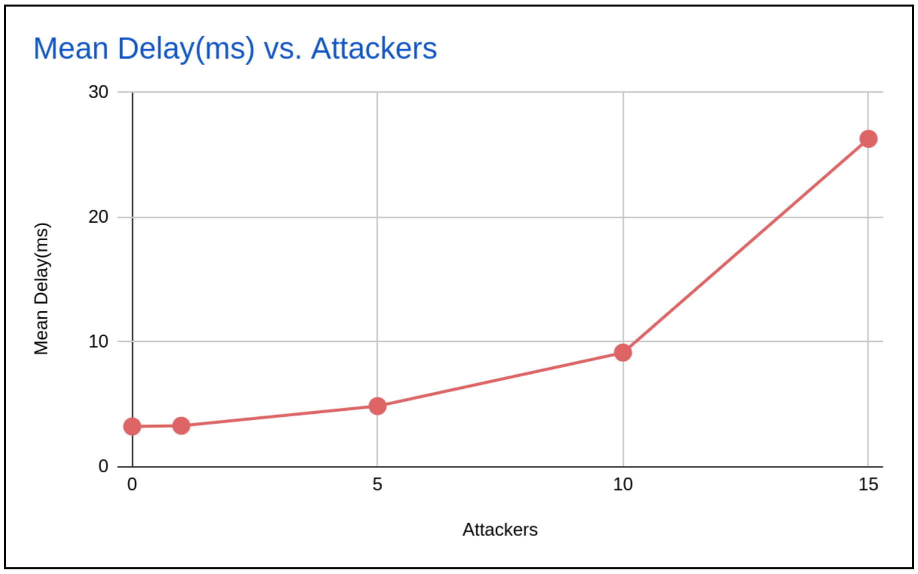

Fig. 4

Thirdly, it can be seen in figure 4 that as more attackers are added to the network, the latency increases for the user. Again, this is to be expected as the 5G network has to process the many additional packers generated due to the UDP flood attack from the attackers. This delays processing for the user's packets, which therefore increases the delay of the packets' arrival to the remote server. The latency itself increases from roughly around 5 milliseconds with no attackers to almost 30 milliseconds with 15 attackers. This indicates the latency is increasing at an accelerating rate as more attackers are added, and that each additional attacker has a larger effect on the delay compared to the previously added attacker.

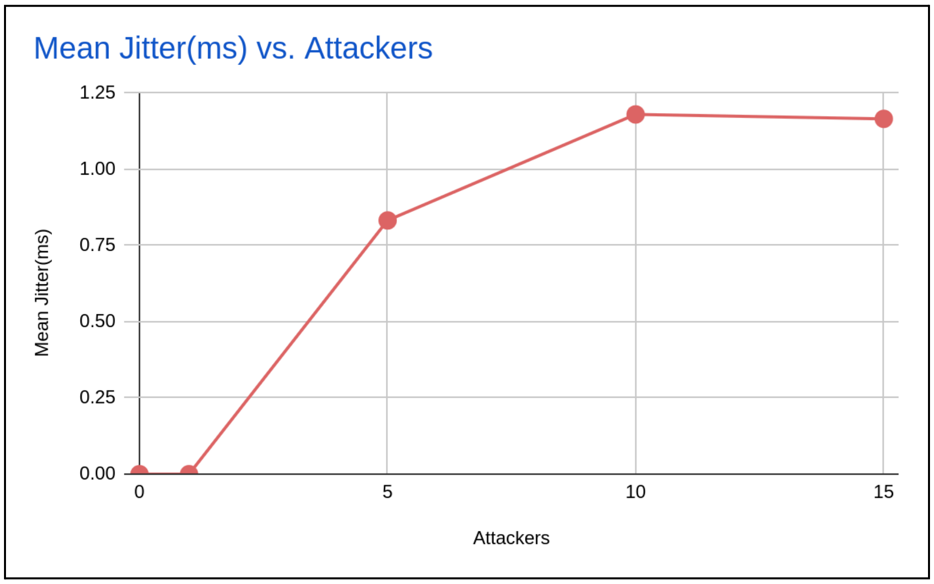

Fig. 5

Lastly, it can be seen that the packet jitter is initially 0 milliseconds, and increases to roughly 1.25 milliseconds with additional attackers and then remains relatively stable as even more attackers are increased. An increase in packet jitter is expected, as with more attackers added, the user's packets will not be processed as consistently which results in varying arrival times from the perspective of the remote host. The stable behaviour of packet jitter from 10 to 15 attackers is also somewhat expected, as even though the latency increases and delays arrival time of the packets, the variation in that arrival time should not vary much even as more attackers are added. This is to say that even though the delay is larger with more attackers, that delay will stay consistently large and not vary resulting in a stable packet jitter.

Limitations & Future Work

There are a number of limitations to this study which limit the effectiveness into the study of the effect of DDoS attacks in 5G networks. Firstly, the scale of the simulated attacks were relatively quite small compared to real world attacks. As mentioned in the literature review above, DDoS attacks are getting larger by the year and that scale makes the attacks more successful at overwhelming networks. While in the conducted simulation, a fairly small number of attackers were used due to constraints in the execution time of the simulation, and therefore the rate at which data was sent from the attackers was not entirely reflective of real world scenarios.

Additionally, the general model was quite simplified and not completely representative of real world scenarios. In most 5G networks, there would be a far larger amount of user equipment associated with a larger number of base stations. The more complex topology in the real world would better reflect real traffic flows, and could change the results on how a DDoS attack affects the end user experience.

Future work could expand on the general complexity of the simulations, and match real world situations more closely for more accurate results. Additionally, adding more attacks and attempting several types of DDoS attacks at once which are more common in attacks on real networks would also improve the efficacy of the simulation and results.

Conclusion

With the ever increasing adoption of 5G technologies it is becoming very common for the average person to interact daily with 5G networks for various purposes. With this increasing adoption does come increasing security concerns. Particularly, DDoS attacks remain ever present as a way for malicious actors to attack digital networks and hamper communications. As it was seen, DDoS attacks have negative effects for the end user experience when those users are interacting with 5G networks and generally worsen 5G network performance. Although due to the high bandwidth of 5G networks, small scale DDoS attacks do not have massive negative effects on the networks.

Acknowledgments

The author of this report would like to acknowledge the 5G LENA team for their implementation of 5G networks in NS-3. Additionally, they would like to acknowledge the 5G LENA team for providing example code for setting up a 5G network and connecting it to the internet, as that example code was used as a base for the DDoS simulation code used for this analysis.

References

[1] 5G-LENA team. “5G-LENA: The 5G NR module for the ns-3 simulator.” 5g-lena.cttc.es. Accessed: April. 17, 2025. [Online.] Available: https://5g-lena.cttc.es

[2] A. Pagadala and G. Ahmed, "Analysis of DDoS Attacks in 5G Networks," *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.10307311.
[3] “What is a DDoS attack?” Cloudflare.com. Accessed: April. 17, 2025. [Online.] Available: https://www.cloudflare.com/en-ca/learning/ddos/what-is-a-ddos-attack/
[4] H. Huang, J. Chu and X. Cheng, "Trend Analysis and Countermeasure Research of DDoS Attack Under 5G Network," *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, Zhuhai, China, 2021, pp. 153-160, doi: 10.1109/CSP51677.2021.9357499
[5] “What is 5G?” Qualcomm.com. Accessed: April. 17, 2025. [Online.] Available: https://www.qualcomm.com/5g/what-is-5g
[6] A. Sultan. “5G System Overview.” 3gpp.org. Accessed: April. 17, 2025. [Online.] Available: https://www.3gpp.org/technologies/5g-system-overview
[7] H. Djuitcheu, T. Shah, M. Tammen and H. D. Schotten, "DDoS impact assessment on 5G system performance," *2023 IEEE Future Networks World Forum (FNWF)*, Baltimore, MD, USA, 2023, pp. 1-6, doi: 10.1109/FNWF58287.2023.10520536.
[8] B. Kaur and K. Tony Joseph, "Security Challenges and Solutions in 5G Networks," 2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 2024, pp. 1-5, doi: 10.1109/IATMSI60426.2024.10502490.
[9] M. S. Khan, B. Farzaneh, N. Shahriar, N. Saha and R. Boutaba, "SliceSecure: Impact and Detection of DoS/DDoS Attacks on 5G Network Slices," *2022 IEEE Future Networks World Forum (FNWF)*, Montreal, QC, Canada, 2022, pp. 639-642, doi: 10.1109/FNWF55208.2022.00117
[10] S. M. Vidhani and A. V. Vidhate, "Security Challenges in 5G Network: A technical features survey and analysis," *2022 5th International Conference on Advances in Science and Technology (ICAST)*, Mumbai, India, 2022, pp. 592-597, doi: 10.1109/ICAST55766.2022.10039654
[11] *Minimum requirements related to technical performance for IMT-2020 radio interface(s) M Series Mobile, radiodetermination, amateur and related satellite services*, IMT-2020, ITU, 2017. [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2410-2017-PDF-E.pdf

Appendix

DDoS_427.cc contents:

```
#include "ns3/antenna-module.h"
#include "ns3/applications-module.h"
#include "ns3/config-store-module.h"
#include "ns3/config-store.h"
#include "ns3/core-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/nr-module.h"
#include "ns3/point-to-point-module.h"
/**
 *  uE --- gNb --- NGC GW --- Internet --- Remotehost
 *
 * ./ns3 run DDoS_427 in ns3-dev to run file
 *
 */

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("3gppChannel");

int
main(int argc, char* argv[])
{
        // enable logging or not
        bool logging = false;
        if (logging)
        {
        LogComponentEnable("UdpClient", LOG_LEVEL_INFO);
        LogComponentEnable("UdpServer", LOG_LEVEL_INFO);
        LogComponentEnable("NrPdcp", LOG_LEVEL_INFO);
        }

        // set simulation time and mobility
        double simTime = 5;          // seconds
        double udpAppStartTime = 0.5; // seconds
        double attackStartTime = 1.5; //seconds

        // other simulation parameters default values
        uint16_t numerology = 0;

        uint16_t gNbNum = 1;
        uint16_t ueNumPergNb = 6;
        uint16_t numUserUe = 1;
        uint16_t numAttackerUe = 5;
        uint16_t totalUe = numUserUe + numAttackerUe;

        double centralFrequency = 7e9;
        double bandwidth = 100e6;
        double txPower = 14;
        double lambda = 1000;
        double lambdaAttacker = 2000;
        uint32_t udpPacketSize = 1000;
        bool udpFullBuffer = false;
        uint8_t fixedMcs = 28;
        bool useFixedMcs = true;
        bool singleUeTopology = false;
        // Where we will store the output files.
        std::string simTag = "default";
        std::string outputDir = "./";

        CommandLine cmd(__FILE__);


        cmd.Parse(argc, argv);

        NS_ASSERT(totalUe > 0);

        // setup the nr simulation
        Ptr<NrHelper> nrHelper = CreateObject<NrHelper>();

        /*
        * Spectrum division. We create one operation band with one component carrier
        * (CC) which occupies the whole operation band bandwidth. The CC contains a
        * single Bandwidth Part (BWP). This BWP occupies the whole CC band.
        * Both operational bands will use the StreetCanyon channel modeling.
        */
        CcBwpCreator ccBwpCreator;
        const uint8_t numCcPerBand = 1; // in this example, both bands have a single CC
        BandwidthPartInfo::Scenario scenario = BandwidthPartInfo::RMa_LoS;
        if (totalUe > 1)
        {
        scenario = BandwidthPartInfo::InH_OfficeOpen;
        }

        // Create the configuration for the CcBwpHelper. SimpleOperationBandConf creates
        // a single BWP per CC
        CcBwpCreator::SimpleOperationBandConf bandConf(centralFrequency,
                bandwidth,
                numCcPerBand,
                scenario);

        // By using the configuration created, it is time to make the operation bands
        OperationBandInfo band = ccBwpCreator.CreateOperationBandContiguousCc(bandConf);
```

```
/*
* Initialize channel and pathloss, plus other things inside band1. If needed,
* the band configuration can be done manually, but we leave it for more
* sophisticated examples. For the moment, this method will take care
* of all the spectrum initialization needs.
*/
nrHelper->InitializeOperationBand(&band);

BandwidthPartInfoPtrVector allBwps = CcBwpCreator::GetAllBwps({band});

/*
* Continue setting the parameters which are common to all the nodes, like the
* gNB transmit power or numerology.
*/
nrHelper->SetGnbPhyAttribute("TxPower", DoubleValue(txPower));
nrHelper->SetGnbPhyAttribute("Numerology", UintegerValue(numerology));

// Scheduler
nrHelper->SetSchedulerTypeId(TypeId::LookupByName("ns3::NrMacSchedulerTdmaRR"));
nrHelper->SetSchedulerAttribute("FixedMcsDl", BooleanValue(useFixedMcs));
nrHelper->SetSchedulerAttribute("FixedMcsUl", BooleanValue(useFixedMcs));

if (useFixedMcs)
{
nrHelper->SetSchedulerAttribute("StartingMcsDl", UintegerValue(fixedMcs));
nrHelper->SetSchedulerAttribute("StartingMcsUl", UintegerValue(fixedMcs));
}

Config::SetDefault("ns3::NrRlcUm::MaxTxBufferSize", UintegerValue(999999999));

// Antennas for all the UEs
nrHelper->SetUeAntennaAttribute("NumRows", UintegerValue(2));
nrHelper->SetUeAntennaAttribute("NumColumns", UintegerValue(4));
nrHelper->SetUeAntennaAttribute("AntennaElement",
                    PointerValue(CreateObject<IsotropicAntennaModel>()));

// Antennas for all the gNbs
nrHelper->SetGnbAntennaAttribute("NumRows", UintegerValue(4));
nrHelper->SetGnbAntennaAttribute("NumColumns", UintegerValue(8));
nrHelper->SetGnbAntennaAttribute("AntennaElement",
                    PointerValue(CreateObject<ThreeGppAntennaModel>()));

// Beamforming method
Ptr<IdealBeamformingHelper> idealBeamformingHelper = CreateObject<IdealBeamformingHelper>();
idealBeamformingHelper->SetAttribute("BeamformingMethod",
                    TypeIdValue(DirectPathBeamforming::GetTypeId()));
nrHelper->SetBeamformingHelper(idealBeamformingHelper);

Config::SetDefault("ns3::ThreeGppChannelModel::UpdatePeriod", TimeValue(MilliSeconds(0)));
//  nrHelper->SetChannelConditionModelAttribute ("UpdatePeriod", TimeValue (MilliSeconds (0)));
nrHelper->SetPathlossAttribute("ShadowingEnabled", BooleanValue(false));

// Error Model: UE and GNB with same spectrum error model.
nrHelper->SetUlErrorModel("ns3::NrEesmIrT1");
nrHelper->SetDlErrorModel("ns3::NrEesmIrT1");

// Both DL and UL AMC will have the same model behind.
nrHelper->SetGnbDlAmcAttribute(
"AmcModel",
EnumValue(NrAmc::ErrorModel)); // NrAmc::ShannonModel or NrAmc::ErrorModel
nrHelper->SetGnbUlAmcAttribute(
"AmcModel",
EnumValue(NrAmc::ErrorModel)); // NrAmc::ShannonModel or NrAmc::ErrorModel

// Create EPC helper
Ptr<NrPointToPointEpcHelper> nrEpcHelper = CreateObject<NrPointToPointEpcHelper>();
nrHelper->SetEpcHelper(nrEpcHelper);
// Core latency
nrEpcHelper->SetAttribute("S1uLinkDelay", TimeValue(MilliSeconds(0)));

// gNb routing between Bearer and bandwidh part
uint32_t bwpIdForBearer = 0;
nrHelper->SetGnbBwpManagerAlgorithmAttribute("GBR_CONV_VOICE", UintegerValue(bwpIdForBearer));

// Initialize nrHelper
nrHelper->Initialize();

/*
*  Create the gNB and UE nodes according to the network topology
*/
NodeContainer gNbNodes;
NodeContainer ueNodes;
NodeContainer attackNodes;
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
Ptr<ListPositionAllocator> bsPositionAlloc = CreateObject<ListPositionAllocator>();
Ptr<ListPositionAllocator> utPositionAlloc = CreateObject<ListPositionAllocator>();
Ptr<ListPositionAllocator> attackPositionAlloc = CreateObject<ListPositionAllocator>();


const double gNbHeight = 10;
const double ueHeight = 1.5;

//topology of base station and ues, single if statement for default case
if (singleUeTopology)
{
gNbNodes.Create(1);
ueNodes.Create(1);
gNbNum = 1;
ueNumPergNb = 1;

mobility.Install(gNbNodes);
mobility.Install(ueNodes);
bsPositionAlloc->Add(Vector(0.0, 0.0, gNbHeight));
utPositionAlloc->Add(Vector(0.0, 30.0, ueHeight));
}
else
{
gNbNodes.Create(gNbNum);
ueNodes.Create(numUserUe);
attackNodes.Create(numAttackerUe);

int32_t yValue = 0.0;
for (uint32_t i = 1; i <= gNbNodes.GetN(); ++i)
{
// 2.0, -2.0, 6.0, -6.0, 10.0, -10.0, ....
if (i % 2 != 0)
{
yValue = static_cast<int>(i) * 30;
}
else
{
yValue = -yValue;
}

bsPositionAlloc->Add(Vector(0.0, yValue, gNbHeight));

// 1.0, -1.0, 3.0, -3.0, 5.0, -5.0, ...
double xValue = 0.0;
for (uint16_t j = 1; j <= numUserUe; ++j)
{
if (j % 2 != 0)
{
xValue = j;
}
else
{
xValue = -xValue;
}

if (yValue > 0)
{
utPositionAlloc->Add(Vector(xValue, 1, ueHeight));
}
else
{
utPositionAlloc->Add(Vector(xValue, -1, ueHeight));
}
}


//1.5, -1.5, 3.5, -3.5, ...
xValue = 1.0;
for (uint16_t j = 1; j <= numAttackerUe; ++j)
{
if (j % 2 != 0)
{
xValue = j + 0.5;
}
else
{
xValue = -xValue;
}

if (yValue > 0)
{
attackPositionAlloc->Add(Vector(xValue, 1, ueHeight));
}
else
{
attackPositionAlloc->Add(Vector(xValue, -1, ueHeight));
}
}

}
}
mobility.SetPositionAllocator(bsPositionAlloc);
mobility.Install(gNbNodes);

mobility.SetPositionAllocator(utPositionAlloc);
mobility.Install(ueNodes);

mobility.SetPositionAllocator(attackPositionAlloc);
mobility.Install(attackNodes);

// Install nr net devices
NetDeviceContainer gNbNetDev = nrHelper->InstallGnbDevice(gNbNodes, allBwps);

NetDeviceContainer ueNetDev = nrHelper->InstallUeDevice(ueNodes, allBwps);

NetDeviceContainer attackNetDev = nrHelper->InstallUeDevice(attackNodes, allBwps);

int64_t randomStream = 1;
randomStream += nrHelper->AssignStreams(gNbNetDev, randomStream);
randomStream += nrHelper->AssignStreams(ueNetDev, randomStream);
randomStream += nrHelper->AssignStreams(attackNetDev, randomStream);

// When all the configuration is done, explicitly call UpdateConfig ()

for (auto it = gNbNetDev.Begin(); it != gNbNetDev.End(); ++it)
{
DynamicCast<NrGnbNetDevice>(*it)->UpdateConfig();
}

for (auto it = ueNetDev.Begin(); it != ueNetDev.End(); ++it)
{
DynamicCast<NrUeNetDevice>(*it)->UpdateConfig();
}

for (auto it = attackNetDev.Begin(); it != attackNetDev.End(); ++it)
{
DynamicCast<NrUeNetDevice>(*it)->UpdateConfig();
}

// create the internet and install the IP stack on the UEs
// get SGW/PGW and create a single RemoteHost
Ptr<Node> pgw = nrEpcHelper->GetPgwNode();
NodeContainer remoteHostContainer;
remoteHostContainer.Create(1);
Ptr<Node> remoteHost = remoteHostContainer.Get(0);
InternetStackHelper internet;
internet.Install(remoteHostContainer);

// connect a remoteHost to pgw. Setup routing too
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute("DataRate", DataRateValue(DataRate("10GBps")));
p2ph.SetDeviceAttribute("Mtu", UintegerValue(2500));
p2ph.SetChannelAttribute("Delay", TimeValue(Seconds(0.000)));
NetDeviceContainer internetDevices = p2ph.Install(pgw, remoteHost);
Ipv4AddressHelper ipv4h;
ipv4h.SetBase("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign(internetDevices);
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting =
ipv4RoutingHelper.GetStaticRouting(remoteHost->GetObject<Ipv4>());
remoteHostStaticRouting->AddNetworkRouteTo(Ipv4Address("7.0.0.0"), Ipv4Mask("255.0.0.0"), 1);
internet.Install(ueNodes);
internet.Install(attackNodes);

Ipv4InterfaceContainer ueIpIface =
nrEpcHelper->AssignUeIpv4Address(NetDeviceContainer(ueNetDev));

Ipv4InterfaceContainer attackIpIface =
nrEpcHelper->AssignUeIpv4Address(NetDeviceContainer(attackNetDev));

// Set the default gateway for the UEs
for (uint32_t j = 0; j < ueNodes.GetN(); ++j)
{
Ptr<Ipv4StaticRouting> ueStaticRouting =
ipv4RoutingHelper.GetStaticRouting(ueNodes.Get(j)->GetObject<Ipv4>());
ueStaticRouting->SetDefaultRoute(nrEpcHelper->GetUeDefaultGatewayAddress(), 1);
}

for (uint32_t j = 0; j < attackNodes.GetN(); ++j)
{
Ptr<Ipv4StaticRouting> attackStaticRouting =
ipv4RoutingHelper.GetStaticRouting(attackNodes.Get(j)->GetObject<Ipv4>());
attackStaticRouting->SetDefaultRoute(nrEpcHelper->GetUeDefaultGatewayAddress(), 1);
}

// attach UEs to the closest gNB
nrHelper->AttachToClosestGnb(ueNetDev, gNbNetDev);
nrHelper->AttachToClosestGnb(attackNetDev, gNbNetDev);

// assign IP address to UEs, and install UDP downlink applications
uint16_t dlPort = 1234;

ApplicationContainer serverApps;

// The sink will always listen to the specified ports
UdpServerHelper dlPacketSinkHelper(dlPort);
serverApps.Add(dlPacketSinkHelper.Install(ueNodes.Get(0)));
serverApps.Add(dlPacketSinkHelper.Install(attackNodes.Get(0)));

//printf("\nTest\n");

//attributes for client and attacker
UdpClientHelper dlClient;
UdpClientHelper dlAttack;
dlClient.SetAttribute("RemotePort", UintegerValue(dlPort));
dlClient.SetAttribute("PacketSize", UintegerValue(udpPacketSize));
dlClient.SetAttribute("MaxPackets", UintegerValue(0xFFFFFFFF));
dlAttack.SetAttribute("RemotePort", UintegerValue(dlPort));
dlAttack.SetAttribute("PacketSize", UintegerValue(udpPacketSize));
dlAttack.SetAttribute("MaxPackets", UintegerValue(0xFFFFFFFF));
if (udpFullBuffer)
{
double bitRate = 75000000; // 75 Mbps will saturate the NR system of 20 MHz with the
                       // NrEesmIrT1 error model
bitRate /= totalUe;                  // Divide the cell capacity among UEs
if (bandwidth > 20e6)
{
bitRate *= bandwidth / 20e6;
}
lambda = bitRate / static_cast<double>(udpPacketSize * 8);
}
dlClient.SetAttribute("Interval", TimeValue(Seconds(1.0 / lambda)));
dlAttack.SetAttribute("Interval", TimeValue(Seconds(1.0 / lambdaAttacker)));

// The bearer that will carry low latency traffic
NrEpsBearer bearer(NrEpsBearer::GBR_CONV_VOICE);

Ptr<NrEpcTft> tft = Create<NrEpcTft>();
NrEpcTft::PacketFilter dlpf;
dlpf.localPortStart = dlPort;
dlpf.localPortEnd = dlPort;
tft->Add(dlpf);

ApplicationContainer clientApps;

for (uint32_t i = 0; i < ueNodes.GetN(); ++i)
{
Ptr<Node> ue = ueNodes.Get(i);
Ptr<NetDevice> ueDevice = ueNetDev.Get(i);
Address ueAddress = ueIpIface.GetAddress(i);

// The client, who is transmitting, is installed in the remote host,
// with destination address set to the address of the UE
dlClient.SetAttribute("RemoteAddress", AddressValue(ueAddress));
clientApps.Add(dlClient.Install(remoteHost));

// Activate a dedicated bearer for the traffic type
nrHelper->ActivateDedicatedEpsBearer(ueDevice, bearer, tft);
}
```

```
ApplicationContainer attackerApps;

for (uint32_t i = 0; i < attackNodes.GetN(); ++i)
{
Ptr<Node> ue = attackNodes.Get(i);
Ptr<NetDevice> ueDevice = attackNetDev.Get(i);
Address ueAddress = attackIpIface.GetAddress(i);

// The client, who is transmitting, is installed in the remote host,
// with destination address set to the address of the UE
dlAttack.SetAttribute("RemoteAddress", AddressValue(ueAddress));
attackerApps.Add(dlAttack.Install(remoteHost));

// Activate a dedicated bearer for the traffic type
nrHelper->ActivateDedicatedEpsBearer(ueDevice, bearer, tft);
}

// start server and client apps
serverApps.Start(Seconds(udpAppStartTime));
clientApps.Start(Seconds(udpAppStartTime));
attackerApps.Start(Seconds(attackStartTime));
serverApps.Stop(Seconds(simTime));
clientApps.Stop(Seconds(simTime));
attackerApps.Stop(Seconds(simTime));

// enable the traces provided by the nr module
// nrHelper->EnableTraces();

FlowMonitorHelper flowmonHelper;
NodeContainer endpointNodes;
endpointNodes.Add(remoteHost);
endpointNodes.Add(ueNodes);
endpointNodes.Add(attackNodes);

//printf("Lambda is %f\n", lambda);

Ptr<ns3::FlowMonitor> monitor = flowmonHelper.Install(endpointNodes);
monitor->SetAttribute("DelayBinWidth", DoubleValue(0.001));
monitor->SetAttribute("JitterBinWidth", DoubleValue(0.001));
monitor->SetAttribute("PacketSizeBinWidth", DoubleValue(20));

Simulator::Stop(Seconds(simTime));
Simulator::Run();

// Print per-flow statistics
monitor->CheckForLostPackets();
Ptr<Ipv4FlowClassifier> classifier =
DynamicCast<Ipv4FlowClassifier>(flowmonHelper.GetClassifier());
FlowMonitor::FlowStatsContainer stats = monitor->GetFlowStats();

double averageFlowThroughput = 0.0;
double averageFlowDelay = 0.0;

std::ofstream outFile;
std::string filename = outputDir + "/" + simTag;
outFile.open(filename.c_str(), std::ofstream::out | std::ofstream::trunc);
if (!outFile.is_open())
{
NS_LOG_ERROR("Can't open file " << filename);
return 1;
}
outFile.setf(std::ios_base::fixed);

//calculate stats based on flowmonitor data
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin();
i != stats.end();
++i)
{
Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow(i->first);
std::stringstream protoStream;
protoStream << (uint16_t)t.protocol;
if (t.protocol == 6)
{
protoStream.str("TCP");
}
if (t.protocol == 17)
{
protoStream.str("UDP");
}
outFile << "Flow " << i->first << " (" << t.sourceAddress << ":" << t.sourcePort << " -> "
<< t.destinationAddress << ":" << t.destinationPort << ") proto "
<< protoStream.str() << "\n";
outFile << "  Tx Packets: " << i->second.txPackets << "\n";
outFile << "  Tx Bytes:   " << i->second.txBytes << "\n";
outFile << "  TxOffered:  "
<< i->second.txBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000 << " Mbps\n";
outFile << "  Rx Bytes:   " << i->second.rxBytes << "\n";
if (i->second.rxPackets > 0)
{
// Measure the duration of the flow from receiver's perspective
double rxDuration =
i->second.timeLastRxPacket.GetSeconds() - i->second.timeFirstTxPacket.GetSeconds();

averageFlowThroughput += i->second.rxBytes * 8.0 / rxDuration / 1000 / 1000;
averageFlowDelay += 1000 * i->second.delaySum.GetSeconds() / i->second.rxPackets;

outFile << "  Throughput: " << i->second.rxBytes * 8.0 / rxDuration / 1000 / 1000
<< " Mbps\n";
outFile << "  Mean delay:  "
<< 1000 * i->second.delaySum.GetSeconds() / i->second.rxPackets << " ms\n";
// outFile << "  Mean upt:  " << i->second.uptSum / i->second.rxPackets / 1000/1000 << "
// Mbps \n";
outFile << "  Mean jitter:  "
<< 1000 * i->second.jitterSum.GetSeconds() / i->second.rxPackets << " ms\n";
}
else
{
outFile << "  Throughput:  0 Mbps\n";
outFile << "  Mean delay:  0 ms\n";
outFile << "  Mean upt:  0  Mbps \n";
outFile << "  Mean jitter: 0 ms\n";
}
outFile << "  Rx Packets: " << i->second.rxPackets << "\n";
}

double meanFlowThroughput = averageFlowThroughput / stats.size();
double meanFlowDelay = averageFlowDelay / stats.size();
Ptr<UdpServer> serverApp = serverApps.Get(0)->GetObject<UdpServer>();
double totalUdpThroughput =
((serverApp->GetReceived() * udpPacketSize * 8) / (simTime - udpAppStartTime)) * 1e-6;

outFile << "\n\n  Mean flow throughput: " << meanFlowThroughput << "\n";
outFile << "  Mean flow delay: " << meanFlowDelay << "\n";
outFile << "\n UDP throughput (bps) for UE with node ID 0:" << totalUdpThroughput << std::endl;

outFile.close();

std::ifstream f(filename.c_str());

if (f.is_open())
{
std::cout << f.rdbuf();
}

Simulator::Destroy();

double toleranceMeanFlowThroughput = 383.557857 * 0.0001;
double toleranceMeanFlowDelay = 3.533664 * 0.0001;
double toleranceUdpThroughput = 372.5066667 * 0.0001;

//testing code
// called from examples-to-run.py with all default parameters
if (argc == 0 && (meanFlowThroughput < 383.557857 - toleranceMeanFlowThroughput ||
meanFlowThroughput > 383.557857 + toleranceMeanFlowThroughput ||
meanFlowDelay < 3.533664 - toleranceMeanFlowDelay ||
meanFlowDelay > 3.533664 + toleranceMeanFlowDelay ||
totalUdpThroughput < 372.5066667 - toleranceUdpThroughput ||
totalUdpThroughput > 372.5066667 + toleranceUdpThroughput))
{
return EXIT_FAILURE;
}
else
{
return EXIT_SUCCESS;
}
}
```