# DdoS Attack using NS3


## CEP Report

By


| NAME | RegistrationNumber |
|---|---|
| Syed Muhammad Hashir | CIIT/FA21-BCE-032/ATD |
| Muhammad Hassan | CIIT/FA21-BCE-051/ATD |
| Hafiz Rehmatullah | CIIT/FA21-BCE-061/ATD |


For the course

## Data Communication and Computer Networks

## Semester Spring 2024

Supervised by:

Dr. Mohsin Fayyaz


**Department of Electrical & Computer Engineering**

**COMSATS University Islamabad – Abbottabad Campus**

# DECLARATION

We Syed Muhammad Hashir (CIIT/FA21-BCE-032/ATD), Muhammad Hassan (CIIT/FA21-BCE-051/ATD), Hafiz Rehmatullah (CIIT/FA21-BCE-061/ATD) hereby declare that we have produced the work presented in this report, during the scheduled period of study. We also declare that we have not taken any material from any source except referred to wherever due. If a violation of rules has occurred in this report, we shall be liable to punishable action.

Date: _____

_____
Syed Muhammad Hashir
(CIIT/FA21-BCE-032/ATD)

_____
Muhammad Hassan
(CIIT/FA21-BCE-051/ATD)

_____
Hafiz Rehmatullah
(CIIT/FA21-BCE-061 /ATD)

# ABSTRACT

In this project, we simulate a Distributed Denial of Service (DDoS) attack using the NS-3 network simulator. The objective is to demonstrate the impact of a DDoS attack on a simple network topology and analyze the resulting network behavior. The simulation involves three primary nodes: a client node, an intermediary node, and a server node. The client node initiates legitimate requests to the server node, which are routed through the intermediary node. To simulate a DDoS attack, multiple bot nodes flood the intermediary node with fake requests, overwhelming its capacity and disrupting the legitimate communication. This setup provides a controlled environment to study the effects of DDoS attacks and the performance of the network under such conditions. The simulation is configured with detailed network parameters and visualized using NS-3's NetAnim tool, providing insights into the attack dynamics and potential mitigation strategies.

# TABLE OF CONTENTS



# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| DDoS | Denial distribution of services |
| Ns3 | Network simulator 3 |
| NetAnim | Network Animator |

# 1 Introduction

This project aims to address this crucial issue by simulating a DDoS attack using the NS-3 network simulator, a versatile tool widely adopted for network research and simulation. By leveraging NS-3, we can construct a realistic and controlled network environment to model and analyze the behavior and impact of DDoS attacks. The primary goal is to create a simulation that provides deep insights into how these attacks affect network performance and to evaluate potential strategies for mitigating their impact. DDoS attacks are a significant threat to network security, where multiple compromised systems flood the targeted system with traffic, overwhelming its resources and causing a denial of service to legitimate users. By simulating a DDoS attack in a controlled environment, we can better understand its impact on network performance and explore potential mitigation strategies.

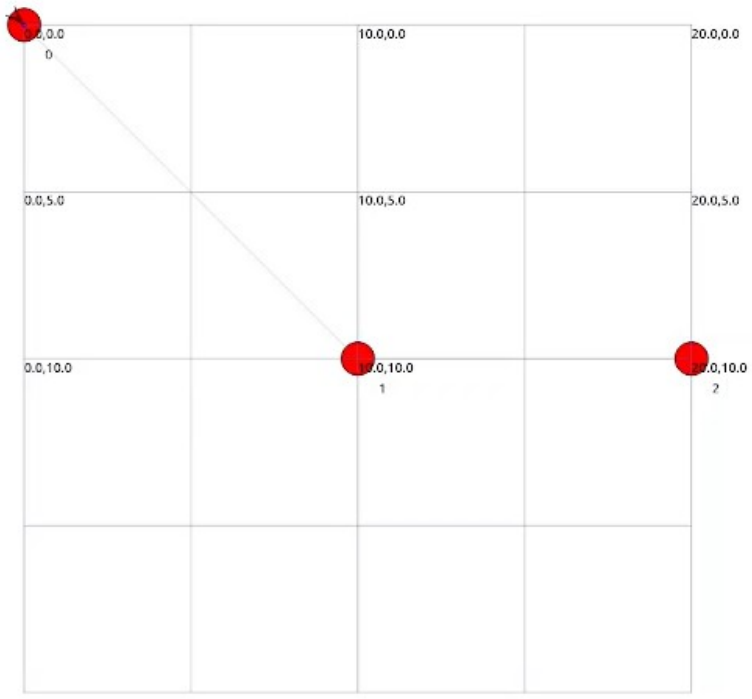



**Fig: 1.1 Network for simulation**

## 1.1 Objectives

Aims and objectives of our project are:

**Simulate a DDoS Attack:**

Create a network topology in NS-3 and simulate a DDoS attack to study its effects.

**Analyze Network Performance:**

Measure and analyze the impact of the DDoS attack on network performance metrics such as latency, packet loss, and throughput.

**Visualize the Attack:**

Use NS-3's NetAnim tool to visualize the network activity and better understand the dynamics of the DDoS attack.

## 1.2 Features

**Client Node (n0)**:

Represents a legitimate user generating normal traffic to communicate with the server.

**Intermediate Node (n1):**

Acts as a router or gateway through which the client's traffic is routed to the server.

**Server Node (n2):**

Hosts the server application that the client is trying to access.

Additionally, there are multiple bot nodes that are used to simulate the DDoS attack:

**Bot Nodes (B0-Bn):**

Represent compromised systems that generate fake traffic to overwhelm the intermediate node.

**Visualization and Analysis:**

The simulation generates an XML file that can be visualized using NS-3's NetAnim tool. NetAnim provides a graphical representation of the network topology and the flow of traffic between nodes. By visualizing the simulation, we can observe:

- The normal flow of legitimate traffic from the client to the server.
- The onset and progression of the DDoS attack by the bot nodes.
- The resulting congestion and its impact on legitimate traffic.

# 2 Literature Survey

## Related Works

**Simulation of DDoS Attacks:**

Numerous studies have been conducted to simulate DDoS attacks and analyze their impact on network performance. Researchers have used various network simulators, including NS-2, NS-3, OMNeT++, and others, to create controlled environments for studying these attacks. These simulations help in understanding the attack dynamics, the extent of damage, and potential mitigation strategies.

**Mitigation Strategies**

Several mitigation strategies have been proposed and tested in simulated environments to counter DDoS attacks. These strategies include rate limiting, traffic filtering, anomaly detection, and the use of machine learning algorithms for detecting and mitigating attacks in real-time.

# 3 Proposed Methodology

## 3.1 System Design / Block diagram

**Network Topology:**

The network topology for this simulation consists of three primary nodes

**Client Node (n0):**

Generates legitimate traffic directed towards the server.

**Intermediate Node (n1):**

Routes traffic between the client and the server. This node is the primary target of the DDoS attack.

**Server Node (n2):**

Receives and processes traffic from the client.

**Bot Nodes (B0-Bn):**

A set of nodes generating malicious traffic directed towards the intermediate node.

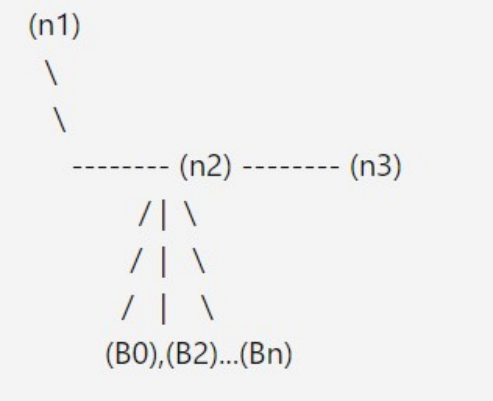
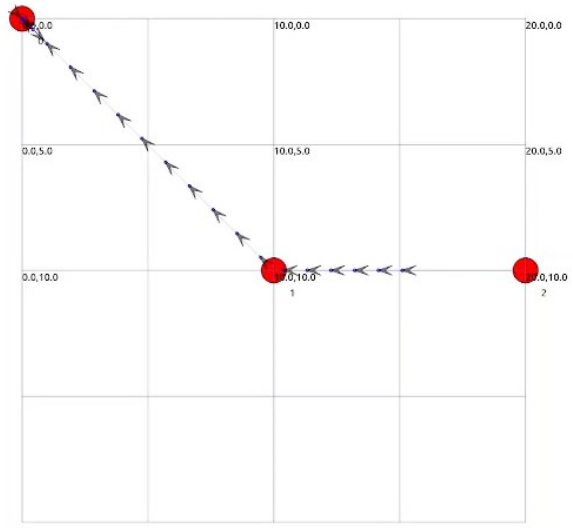


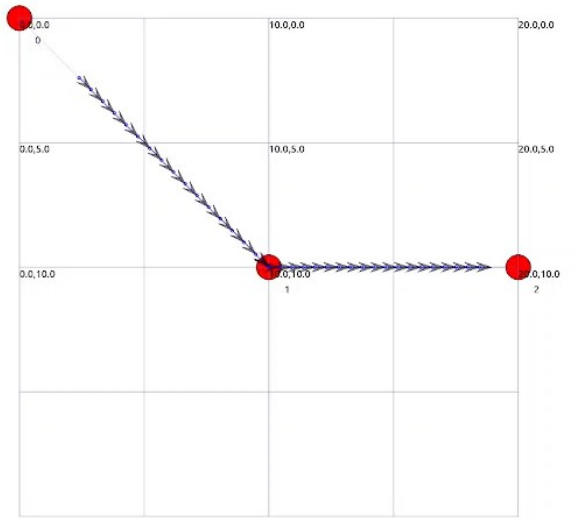

**Fig: 3.1 Block Diagram**

## 3.2Simulation

Below simulation is for simple network when bots are still not connected.

As we can see traffic is normal and client (n0) request is being fulfilled by server application (n2) and this is passed through an intermediate node (n1).



## 3.3simulation diagram

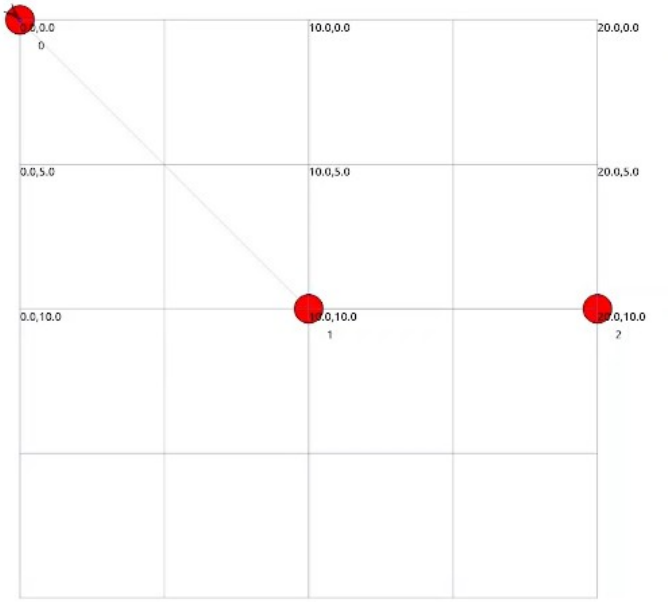



**Fig: 3.2 Network Diagram**

## 3.4Algorithm:

The provided program simulates a Denial-of-Service (DoS) attack using ns-3. Here's the algorithm breakdown:

1. **Setup:**

    - Define simulation parameters like data rate, simulation time, and number of bots.
    - Create nodes: 3 legitimate nodes (n0, n1, n2) and a number of bot nodes for the attack.
    - Define Point-to-Point links between nodes with specified data rates and delays.
    - Install the links to connect the nodes.

2. **IP Address Assignment:**

    - Install the internet stack on all nodes (legitimate and bots).
    - Assign unique IP addresses from different networks to bot nodes.

- Assign IP addresses from separate subnets to legitimate nodes (n0, n1, n2).

3. **DDoS Application Configuration:**

    - Create an OnOffHelper application to generate UDP traffic on bots.
    - Set the constant rate of traffic for the DDoS attack (DDOS_RATE).
    - Configure the application to send traffic continuously for 30 seconds (OnTime) with no off time.
    - Install this application on all bot nodes to initiate the attack.

4. **Legitimate Traffic Configuration:**

    - Create a BulkSendHelper application to generate TCP traffic on the legitimate user node (n0).
    - Set the maximum data size (MaxBytes) for the legitimate traffic.
    - Install this application on n0 to send data to the server (n2).
    - Configure the application to start at 0 seconds and stop 10 seconds before the simulation ends.

5. **Sink Applications:**

    - Create PacketSinkHelper applications for both UDP and TCP traffic on the server node (n2).
    - These applications will capture the incoming traffic on the respective ports.
    - Install both UDP and TCP sink applications on n2 to receive traffic.
    - Configure both applications to start at 0 seconds and stop when the simulation ends.

6. **Routing and Network Animation:**

    - Populate routing tables on all nodes to enable communication.
    - Configure the network animation using GridPositionAllocator for node placement.
    - Set constant positions for all nodes (legitimate and bots) in the animation.

7. **Simulation Execution:**

    - Run the network simulator to simulate the DDoS attack scenario.
    - Destroy the simulator instance after the simulation finishes.

This algorithm outlines the key steps involved in setting up and running the DoS attack simulation using the provided ns-3 program. It highlights the configuration of applications on different nodes to generate and capture traffic.

# 4 Simulation Results

## 4.1 Software simulation results

In the below simulation, we have connected 5 bots. Requests from the client are properly going to the server application (first image) but response from the server side is slow which can be seen in the second image

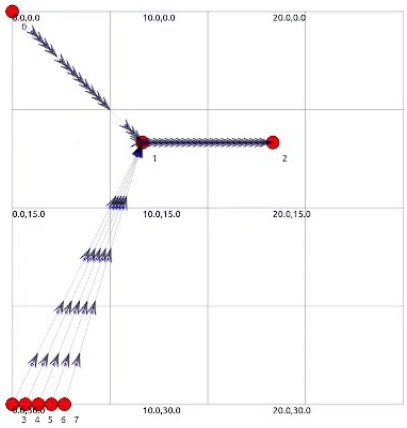



**Fig: 4.1 Image 1 showing client request**

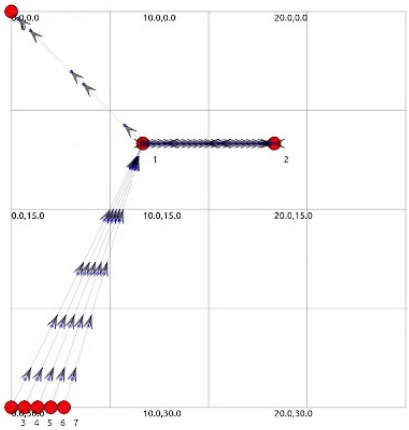



**Fig: 4.2 Image 2 showing server side response**

**Fig: 4.3 simulation results 1**

In the below simulation, we have used 20 bots. Requests from the client are properly going to the server application (first image) but response from the server side is very slow because now we have used more number of bots which can be seen in the second image.

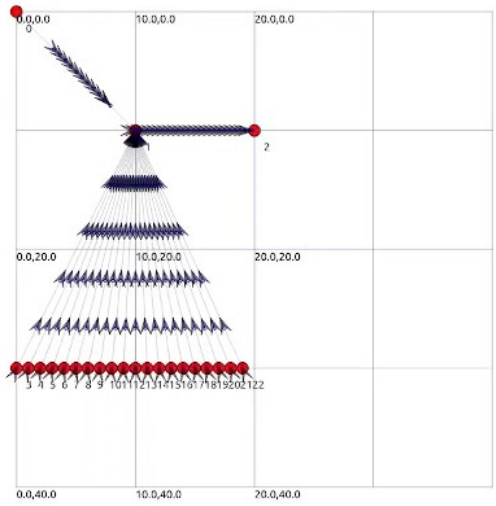



**Fig: 4.5 Image 1 showing client request**

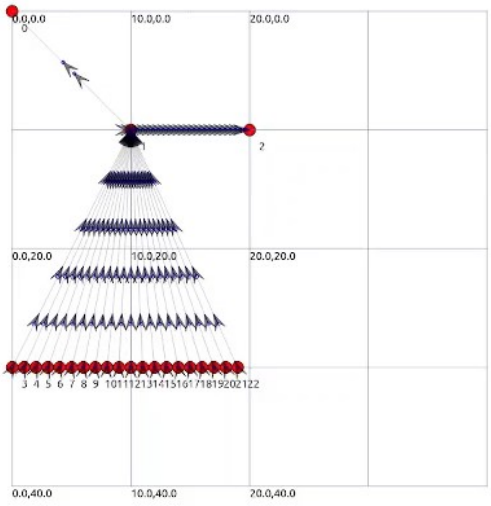



**Fig: 4.6 Image 2 showing client request**

**Fig: 4.1 simulation results 2**

## 4.2Design/simulation parameters

1. **Network Topology:**

    - The simulation utilizes a simple topology consisting of three nodes: a legitimate client node (N0), an intermediate node (N1), and a server application node (N2).
    - Additional nodes (bots) are introduced to simulate the DDoS attack, connected to the intermediate node (N1).

2. **Network Configuration:**

    - **Point-to-Point Links:** Utilized to establish communication channels between nodes. Configured with specific attributes like data rate (100 Mbps) and delay (1 ms) to mimic typical network conditions.
    - **IP Addressing:** IP addresses are assigned to nodes and bots using the IPv4 addressing scheme (e.g., 10.0.0.0/30 for legitimate nodes and 10.1.1.0/24 for internal communication).

3. **Attack Parameters:**

    - **Number of Bots:** Set to 20, representing the compromised devices participating in the DDoS attack.
    - **Traffic Generation:** Bots generate UDP traffic with a constant rate of 20480 kb/s directed towards the server application node (N2).
    - **Attack Duration:** Simulated over a maximum time span of 10 seconds to observe the immediate impact and response of the network under attack.

4. **Application Behavior:**

    - **On-Off Application:** Configured on bots to generate UDP traffic intermittently. The "OnTime" and "OffTime" attributes determine the active and inactive periods of traffic generation.

5. **Simulation Environment:**

    - **Mobility Model:** Nodes and bots are assigned constant position mobility models within a grid layout to visualize their movement and placement in the NetAnim simulation tool.
    - **Visualization:** NetAnim XML file ("DDoSim.xml") captures the simulated network topology and node movements for graphical representation and analysis.

### 4.3 Discussions

The simulation results and observations yield valuable insights into the behavior and impact of DDoS attacks on network performance and security measures:

1. **Effectiveness of DDoS Attack:**

    - The DDoS attack successfully disrupts normal network operations by overwhelming the intermediate node (N1) with a high volume of UDP traffic from bots.
    - Observations indicate a significant increase in network latency, packet loss, and potential denial of service to legitimate client-server communication.

2. **Impact on Server Application:**

    - During the attack, the server application node (N2) experiences degraded performance, evidenced by delayed or incomplete responses to client requests.
    - Resource exhaustion or bandwidth saturation on N1 may contribute to the improper handling of traffic, affecting the server's ability to process incoming requests.

3. **Simulation Limitations and Future Enhancements:**

    - Acknowledge limitations in the simulation setup, such as simplified network models or idealized attack behaviors, which may not fully represent the complexities of real-world DDoS incidents.
    - Propose enhancements like incorporating more realistic traffic patterns, scaling up the network size, or integrating advanced attack vectors to broaden the simulation's applicability and accuracy.

# 5 Conclusions

In conclusion, this project simulated a Distributed Denial of Service (DDoS) attack using NS-3, focusing on understanding its impact and exploring mitigation strategies in network security. The simulation employed a network topology with nodes representing legitimate clients, an intermediate node, and a server application, along with bots simulating the attackers. Through general parameterization of network configurations, attack characteristics, and simulation environments, the study observed significant disruptions caused by the DDoS attack. Discussions highlighted the effectiveness of DDoS attacks in compromising network performance and the

importance of implementing robust defense mechanisms such as rate limiting and traffic filtering to mitigate their impact. Future research could enhance simulation realism and explore advanced defense strategies to bolster network resilience against evolving cyber threats. This project underscores the critical role of simulations in informing network security practices and advancing cybersecurity preparedness in the face of persistent DDoS challenges.

# References

[1]. Upadhyay, S. (2020). DDoS simulation in NS-3. InfoSec Write-ups [online] Available at:https://infosecwriteups.com/ddos-simulation-in-ns-3-c-12f031a7b38c Accessed on 02-07-2024

# Appendix

```
 * In this we follow the following setup / node placement
 *
 *    (n1)
 *      \
 *       \
 *         -------- (n2) -------- (n3)
 *                  / |  \
 *                 /  |   \
 *                /   |    \
 *             (B0),(B2)...(Bn)
 *
 *  N0 is legitimate user, communicating with server N2 (data server)
via node N1 (maybe website server interface )
 *  B0-Bn are bots DDoS-ing the network.
 *
 * NetAnim XML is saved as -> DDoSim.xml
 *
 */
```

```
#include <ns3/csma-helper.h>
#include "ns3/mobility-module.h"
#include "ns3/nstime.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

#define TCP_SINK_PORT 9000
#define UDP_SINK_PORT 9001

//experimental parameters
#define MAX_BULK_BYTES 100000
#define DDOS_RATE "20480kb/s"
#define MAX_SIMULATION_TIME 10.0

//Number of Bots for DDoS
#define NUMBER_OF_BOTS 20

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("DDoSAttack");

int main(int argc, char *argv[])
{
    CommandLine cmd;
    cmd.Parse(argc, argv);

    Time::SetResolution(Time::NS);
    LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
```

```
    LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);


    //Legitimate connection bots
    NodeContainer nodes;
    nodes.Create(3);

    //Nodes for attack bots
    NodeContainer botNodes;
    botNodes.Create(NUMBER_OF_BOTS);

    // Define the Point-To-Point Links and their Paramters
    PointToPointHelper pp1, pp2;
    pp1.SetDeviceAttribute("DataRate", StringValue("100Mbps"));
    pp1.SetChannelAttribute("Delay", StringValue("1ms"));

    pp2.SetDeviceAttribute("DataRate", StringValue("100Mbps"));
    pp2.SetChannelAttribute("Delay", StringValue("1ms"));

    // Install the Point-To-Point Connections between Nodes
    NetDeviceContainer d02, d12, botDeviceContainer[NUMBER_OF_BOTS];
    d02 = pp1.Install(nodes.Get(0), nodes.Get(1));
    d12 = pp1.Install(nodes.Get(1), nodes.Get(2));

    for (int i = 0; i < NUMBER_OF_BOTS; ++i)
    {
        botDeviceContainer[i]       =       pp2.Install(botNodes.Get(i),
nodes.Get(1));
    }

    //Assign IP to bots
    InternetStackHelper stack;
    stack.Install(nodes);
```

```
    stack.Install(botNodes);
    Ipv4AddressHelper ipv4_n;
    ipv4_n.SetBase("10.0.0.0", "255.255.255.252");

    Ipv4AddressHelper a02, a12, a23, a34;
    a02.SetBase("10.1.1.0", "255.255.255.0");
    a12.SetBase("10.1.2.0", "255.255.255.0");

    for (int j = 0; j < NUMBER_OF_BOTS; ++j)
    {
        ipv4_n.Assign(botDeviceContainer[j]);
        ipv4_n.NewNetwork();
    }

    //Assign IP to legitimate nodes
    Ipv4InterfaceContainer i02, i12;
    i02 = a02.Assign(d02);
    i12 = a12.Assign(d12);

    // DDoS Application Behaviour
    OnOffHelper                          onoff("ns3::UdpSocketFactory",
Address(InetSocketAddress(i12.GetAddress(1), UDP_SINK_PORT)));
    onoff.SetConstantRate(DataRate(DDOS_RATE));
    onoff.SetAttribute("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=30]"));
    onoff.SetAttribute("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));
    ApplicationContainer onOffApp[NUMBER_OF_BOTS];

    //Install application in all bots
    for (int k = 0; k < NUMBER_OF_BOTS; ++k)
    {
        onOffApp[k] = onoff.Install(botNodes.Get(k));
```

```
        onOffApp[k].Start(Seconds(0.0));
        onOffApp[k].Stop(Seconds(MAX_SIMULATION_TIME));
    }

    // Sender Application (Packets generated by this application are
throttled)
    BulkSendHelper                    bulkSend("ns3::TcpSocketFactory",
InetSocketAddress(i12.GetAddress(1), TCP_SINK_PORT));
    bulkSend.SetAttribute("MaxBytes", UintegerValue(MAX_BULK_BYTES));
    ApplicationContainer bulkSendApp = bulkSend.Install(nodes.Get(0));
    bulkSendApp.Start(Seconds(0.0));
    bulkSendApp.Stop(Seconds(MAX_SIMULATION_TIME - 10));

    // UDPSink on receiver side
    PacketSinkHelper UDPsink("ns3::UdpSocketFactory",

Address(InetSocketAddress(Ipv4Address::GetAny(), UDP_SINK_PORT)));
    ApplicationContainer UDPSinkApp = UDPsink.Install(nodes.Get(2));
    UDPSinkApp.Start(Seconds(0.0));
    UDPSinkApp.Stop(Seconds(MAX_SIMULATION_TIME));

    // TCP Sink Application on server side
    PacketSinkHelper TCPsink("ns3::TcpSocketFactory",
                             InetSocketAddress(Ipv4Address::GetAny(),
TCP_SINK_PORT));
    ApplicationContainer TCPSinkApp = TCPsink.Install(nodes.Get(2));
    TCPSinkApp.Start(Seconds(0.0));
    TCPSinkApp.Stop(Seconds(MAX_SIMULATION_TIME));

    Ipv4GlobalRoutingHelper::PopulateRoutingTables();

    //Simulation NetAnim configuration and node placement
    MobilityHelper mobility;
```

```
    mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                                  "MinX",   DoubleValue(0.0),   "MinY",
DoubleValue(0.0),      "DeltaX",      DoubleValue(5.0),      "DeltaY",
DoubleValue(10.0),
                                  "GridWidth",        UintegerValue(5),
"LayoutType", StringValue("RowFirst"));

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

    mobility.Install(nodes);
    mobility.Install(botNodes);

    AnimationInterface anim("DDoSim.xml");

    ns3::AnimationInterface::SetConstantPosition(nodes.Get(0), 0, 0);
    ns3::AnimationInterface::SetConstantPosition(nodes.Get(1), 10, 10);
    ns3::AnimationInterface::SetConstantPosition(nodes.Get(2), 20, 10);

    uint32_t x_pos = 0;
    for (int l = 0; l < NUMBER_OF_BOTS; ++l)
    {
        ns3::AnimationInterface::SetConstantPosition(botNodes.Get(l),
x_pos++, 30);
    }

    //Run the Simulation
    Simulator::Run();
    Simulator::Destroy();
    return 0;
}
```

Teachers should assess CLO2, CLO3 and CLO4 based on the given rubrics

(overall weightage 20%)

## Recommended Percentage Breakdown

| **CLO** | **Percentage** |
|---|---|
| CLO2 (Investigation) | 10% |
| CLO3 (Referencing/Citations) *(Turnitin report should be generated.)* | 5% |
| CLO4 (Communication) | 5% |