Anomaly Detection in Network Traffic Using ML-Based Algorithms

EHB 415E

Study of Simulation Tools:

Network Traffic Simulations using OMNeT++ and Sionna

Prepared by:

Batu Burgu
Student ID: 0402010098

Javad Ibrahimli
Student ID: 040210932

Kerem Karadeniz
Student ID: 040210049

2024-2025 Academic Year

Supervised by:
İbrahim Hökelek PhD.

Contents

1 Simulation Tool Setup

1.1 System Requirements and Before Installation

For the simulation environment, we have used Ubuntu as our operating system due to its compatibility and wide support for development tools required by OMNeT++. We installed the latest stable version of Ubuntu (Ubuntu 22.04 LTS) to ensure compatibility with OMNeT++ and the necessary libraries.

Before installing OMNeT++, several essential development libraries and tools need to be installed. These include:

```
build-essential        gcc                              g++
bison                  flex                             perl
qt5-default            tcl-dev                          tk-dev
libxml2-dev            zlib1g-dev                       default-jre
doxygen                graphviz                         libwebkitgtk-3.0-0
libopenscenegraph-dev  openscenegraph-plugin-osgearth   libosgearth-dev
openmpi-bin            libopenmpi-dev
```

1.2 Selection of Simulation Tool

We have chosen OMNeT++ as the simulation tool to be used for our project "Anomaly Detection in Network Traffic Using ML-Based Algorithms". OMNeT++ is an ideal simulation tool for our topic due to its versatility, modularity, and open-source framework. Designed specifically for network simulations, OMNeT++ allows for detailed modeling of network protocols and complex communication scenarios, making it an excellent choice for capturing and analyzing network traffic data.

Also, it being a C++ based simulation environment suits our team since the members of our team are more experienced with this programming language than others. Its extensible architecture supports custom implementations, enabling the integration of machine learning models for real-time anomaly detection, which is the main purpose of our project.

Furthermore, there are different kinds of router protocols natively supported by the INET framework such as BGP, OSPF, and RIP available to simulate different scenarios. This helps us to train and test our AI through various routing protocols and observe the effects of them on the network and our AI.

Figure 1: Omnet 6.6 is used for the Simulation

Additionally, OMNeT++'s compatibility with various libraries and frameworks, such as INET for Internet protocols, provides realistic traffic scenarios essential for developing and testing anomaly detection algorithms. This simulation environment thus offers a robust platform for investigating network behavior and testing ML-based solutions under different traffic conditions, enhancing the overall reliability and validity of our findings.

1.3 Installation of OMNeT++ and INET Framework

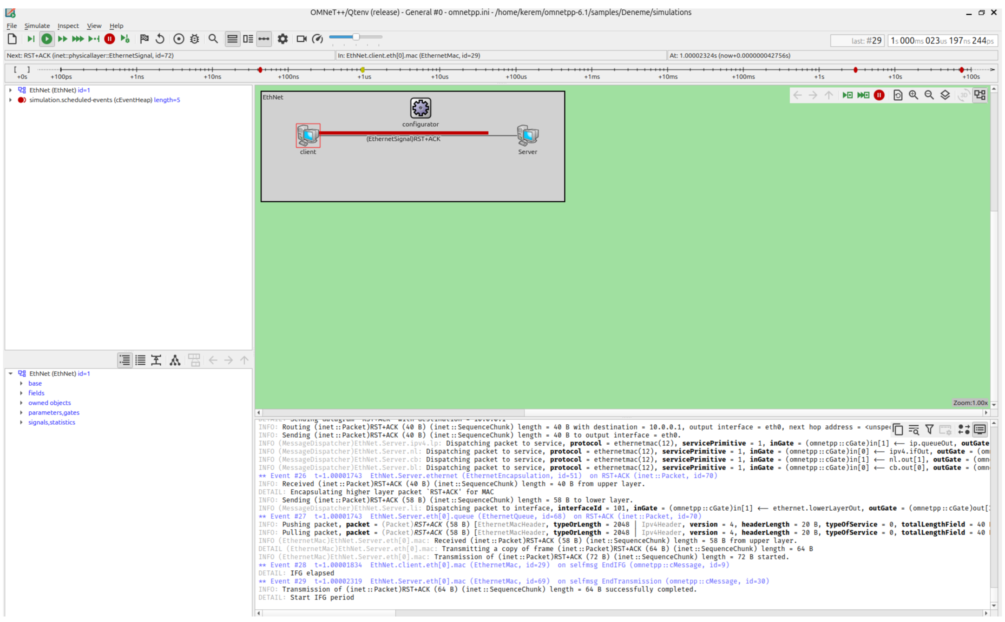
We installed the OMNeT++ software by following the instructions provided in sections 5 and 6 of the OMNeT++ Installation Guide (Version 6.0.3) [**1**].

After installing OMNeT++, we tried to install INET 4.5 from the "Install Simulation Models" menu located in the "Help" section of the application. Unfortunately, it did not work for a reason we were unable to determine. We kept getting the error "Installing Project has encountered a problem, cannot download archive".

Therefore, we decided to manually download INET 4.5 and add it to OMNeT++ as a project using the "Open Projects from File System" option in the File menu. We downloaded the latest stable version of INET (version 4.5.2) from the "Download" section on the INET website [**2**], and followed the instructions written in the installiation manual of INET [**3**]

After these steps, we encountered a reference error related to the file named `NodeStatus.cc` [**3**] when building the INET project. The cause of the error and the solution are explained below:

In the original file, an enum containing the `State` attribute of the object `NodeStatus` is registered on line 17. However, the creation of the enum variable is incorrect because it tries to assign the `State` information directly to the `NodeStatus` object instead of the attribute



```
OMNeT++/Qtenv (release) - General #0 - omnetpp.ini - /home/kerem/omnetpp-6.1/samples/Deneme/simulations
Next: RST+ACK (inet::physicallayer::EthernetSignal, id=72)   In: EthNet.client.eth[0].mac (EthernetMac, id=29)   At: 1.00002324s (now+0.000000042756s)
```

```
INFO: Routing (inet::Packet)RST+ACK (40 B) (inet::SequenceChunk) length = 40 B with destination = 10.0.0.1, output interface = eth0, next hop address = <unspe
INFO: Sending (inet::Packet)RST+ACK (40 B) (inet::SequenceChunk) length = 40 B to output interface = eth0.
INFO (MessageDispatcher)EthNet.Server.ipv4.lp: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGate)in[1] ⟵ ip.queueOut, outGate
INFO (MessageDispatcher)EthNet.Server.nl: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGate)in[0] ⟵ ipv4.ifOut, outGate = (om
INFO (MessageDispatcher)EthNet.Server.cb: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGate)in[1] ⟵ nl.out[1], outGate = (omn
INFO (MessageDispatcher)EthNet.Server.bl: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGate)in[0] ⟵ cb.out[0], outGate = (omn
** Event #26  t=1.00001743  EthNet.Server.ethernet (EthernetEncapsulation, id=51)  on RST+ACK (inet::Packet, id=70)
INFO: Received (inet::Packet)RST+ACK (40 B) (inet::SequenceChunk) length = 40 B from upper layer.
DETAIL: Encapsulating higher layer packet `RST+ACK' for MAC
INFO: Sending (inet::Packet)RST+ACK (58 B) (inet::SequenceChunk) length = 58 B to lower layer.
INFO (MessageDispatcher)EthNet.Server.li: Dispatching packet to interface, interfaceId = 101, inGate = (omnetpp::cGate)in[1] ⟵ ethernet.lowerLayerOut, outGate = (omnetpp::cGate)out[
** Event #27  t=1.00001743  EthNet.Server.eth[0].queue (EthernetQueue, id=68)  on RST+ACK (inet::Packet, id=70)
INFO: Pushing packet, packet = (Packet)RST+ACK (58 B) [EthernetMacHeader, typeOrLength = 2048 | Ipv4Header, version = 4, headerLength = 20 B, typeOfService = 0, totalLengthField = 40
INFO: Pulling packet, packet = (Packet)RST+ACK (58 B) [EthernetMacHeader, typeOrLength = 2048 | Ipv4Header, version = 4, headerLength = 20 B, typeOfService = 0, totalLengthField = 40
INFO (EthernetMac)EthNet.Server.eth[0].mac: Received (inet::Packet)RST+ACK (58 B) (inet::SequenceChunk) length = 58 B from upper layer.
DETAIL (EthernetMac)EthNet.Server.eth[0].mac: Transmitting a copy of frame (inet::Packet)RST+ACK (64 B) (inet::SequenceChunk) length = 64 B
INFO (EthernetMac)EthNet.Server.eth[0].mac: Transmission of (inet::Packet)RST+ACK (72 B) (inet::SequenceChunk) length = 72 B started.
** Event #28  t=1.00001834  EthNet.client.eth[0].mac (EthernetMac, id=29)  on selfmsg EndIFG (omnetpp::cMessage, id=9)
DETAIL: IFG elapsed
** Event #29  t=1.00002319  EthNet.Server.eth[0].mac (EthernetMac, id=69)  on selfmsg EndTransmission (omnetpp::cMessage, id=30)
INFO: Transmission of (inet::Packet)RST+ACK (64 B) (inet::SequenceChunk) length = 64 B successfully completed.
DETAIL: Start IFG period
```

Figure 2: Simulation of Basic Client and Server

`NodeStatus::State`. We resolved the issue by adjusting the corresponding line as shown below, all on one line:

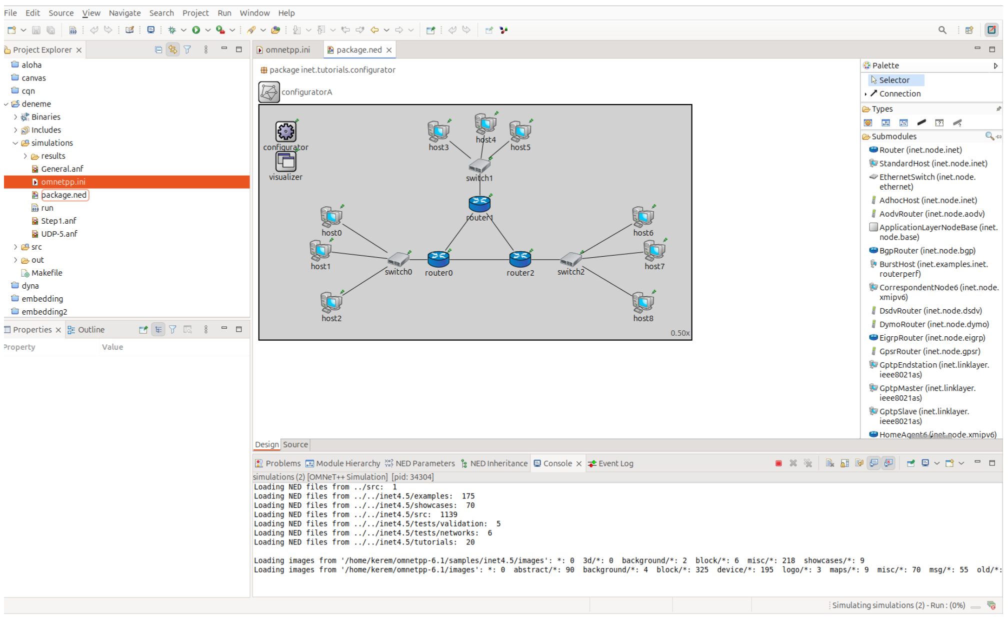
```
Register_Enum(inet::NodeStatus::State, (NodeStatus::State::UP,
                                        NodeStatus::State::DOWN,
                                        NodeStatus::State::GOING_UP,
                                        NodeStatus::State::GOING_DOWN));
```

After making these adjustments, we successfully built the simulation and proceeded to create the simulation environments.

2 Generating Network Topologies and Simulations

2.1 Deploying a Basic Client-Server Topology

A basic client-server direct link topology was deployed as an initial step to familiarize ourselves with the fundamentals of the software and simulation. Both server and client nodes were imported from INET libraries. The IPv4 configurator was also used; however, since the nodes were directly connected, no Layer 3 routing or IP addressing was required. The IPv4 configurator was later utilized to configure more complex topologies.



```
simulations (2) [OMNeT++ Simulation]  [pid: 34304]
Loading NED files from ../src:  1
Loading NED files from ../../inet4.5/examples:  175
Loading NED files from ../../inet4.5/showcases:  70
Loading NED files from ../../inet4.5/src:  1139
Loading NED files from ../../inet4.5/tests/validation:  5
Loading NED files from ../../inet4.5/tests/networks:  6
Loading NED files from ../../inet4.5/tutorials:  20

Loading images from '/home/kerem/omnetpp-6.1/samples/inet4.5/images': *: 0  3d/*: 0  background/*: 2  block/*: 6  misc/*: 218  showcases/*: 9
Loading images from '/home/kerem/omnetpp-6.1/images': *: 0  abstract/*: 90  background/*: 4  block/*: 325  device/*: 195  logo/*: 3  maps/*: 9  misc/*: 70  msg/*: 55  old/*:
```

Figure 3: Network Topology of Three Subnets

2.2 Deploying a Complex Network Topology

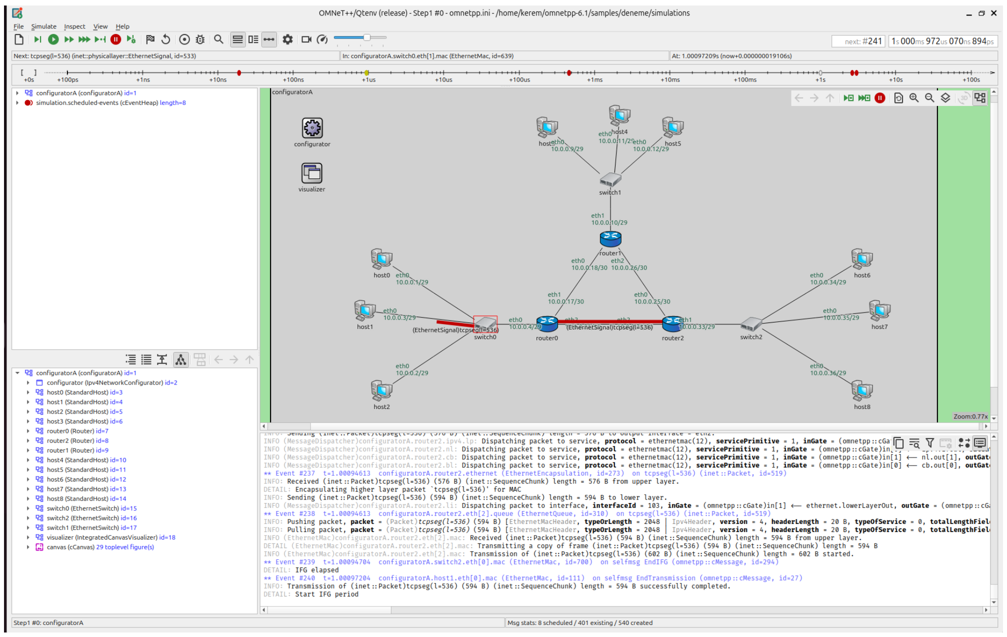
From the INET tutorials, we studied a network topology to better understand the setup of more complex environments. A network consisting of three routers and three subnets was deployed, with the IPv4 configurator used to apply different routing protocols. By default, the configurator assigns IP addresses and routing tables statically, but it is possible to configure the routing table to use protocols like OSPF, BGP, or any other supported protocol.

2.3 Resource Usage

OMNeT++ efficiently handles small to medium networks, but resource demands increase with complexity. Future simulations will require careful resource management as we scale and integrate machine learning models for anomaly detection.

3 Accessing Simulation Result Data

After the simulations were completed, the files containing the simulation results were examined in the "Result" folder. Statistical analyses were performed using the built-in analysis tools of OMNeT++, resulting in histogram representations of significant data. Additionally,



```
OMNeT++/Qtenv (release) - Step1 #0 - omnetpp.ini - /home/kerem/omnetpp-6.1/samples/deneme/simulations
Next: tcpseg(l=536) (inet::physicallayer::EthernetSignal, id=533)    In: configuratorA.switch0.eth[1].mac (EthernetMac, id=639)    At: 1.00097209s (now+0.000000019106s)

configuratorA (configuratorA) id=1
simulation.scheduled-events (cEventHeap) length=8

configuratorA (configuratorA) id=1
  configurator (Ipv4NetworkConfigurator) id=2
  host0 (StandardHost) id=3
  host1 (StandardHost) id=4
  host2 (StandardHost) id=5
  host3 (StandardHost) id=6
  router0 (Router) id=7
  router2 (Router) id=8
  router1 (Router) id=9
  host4 (StandardHost) id=10
  host5 (StandardHost) id=11
  host6 (StandardHost) id=12
  host7 (StandardHost) id=13
  host8 (StandardHost) id=14
  switch0 (EthernetSwitch) id=15
  switch2 (EthernetSwitch) id=16
  switch1 (EthernetSwitch) id=17
  visualizer (IntegratedCanvasVisualizer) id=18
  canvas (cCanvas) 29 toplevel figure(s)
```

```
INFO (MessageDispatcher)configuratorA.router2.ipv4.lp: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGa
INFO (MessageDispatcher)configuratorA.router2.nl: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGate)in
INFO (MessageDispatcher)configuratorA.router2.cb: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGate)in[1] ⟵ nl.out[1], outGat
INFO (MessageDispatcher)configuratorA.router2.bl: Dispatching packet to service, protocol = ethernetmac(12), servicePrimitive = 1, inGate = (omnetpp::cGate)in[0] ⟵ cb.out[0], outGat
** Event #237  t=1.00094613  configuratorA.router2.ethernet (EthernetEncapsulation, id=273)  on tcpseg(l=536) (inet::Packet, id=519)
INFO: Received (inet::Packet)tcpseg(l=536) (576 B) (inet::SequenceChunk) length = 576 B from upper layer.
DETAIL: Encapsulating higher layer packet `tcpseg(l=536)' for MAC
INFO: Sending (inet::Packet)tcpseg(l=536) (594 B) (inet::SequenceChunk) length = 594 B to lower layer.
INFO (MessageDispatcher)configuratorA.router2.li: Dispatching packet to interface, interfaceId = 103, inGate = (omnetpp::cGate)in[1] ⟵ ethernet.lowerLayerOut, outGate = (omnetpp::cGa
** Event #238  t=1.00094613  configuratorA.router2.eth[2].queue (EthernetQueue, id=310)  on tcpseg(l=536) (inet::Packet, id=519)
INFO: Pushing packet, packet = (Packet)tcpseg(l=536) (594 B) [EthernetMacHeader, typeOrLength = 2048 | Ipv4Header, version = 4, headerLength = 20 B, typeOfService = 0, totalLengthFiel
INFO: Pulling packet, packet = (Packet)tcpseg(l=536) (594 B) [EthernetMacHeader, typeOrLength = 2048 | Ipv4Header, version = 4, headerLength = 20 B, typeOfService = 0, totalLengthFiel
INFO (EthernetMac)configuratorA.router2.eth[2].mac: Received (inet::Packet)tcpseg(l=536) (594 B) (inet::SequenceChunk) length = 594 B from upper layer.
DETAIL (EthernetMac)configuratorA.router2.eth[2].mac: Transmitting a copy of frame (inet::Packet)tcpseg(l=536) (594 B) (inet::SequenceChunk) length = 594 B
INFO (EthernetMac)configuratorA.router2.eth[2].mac: Transmission of (inet::Packet)tcpseg(l=536) (602 B) (inet::SequenceChunk) length = 602 B started.
** Event #239  t=1.00094704  configuratorA.switch2.eth[0].mac (EthernetMac, id=700)  on selfmsg EndIFG (omnetpp::cMessage, id=294)
DETAIL: IFG elapsed
** Event #240  t=1.00097204  configuratorA.host1.eth[0].mac (EthernetMac, id=111)  on selfmsg EndTransmission (omnetpp::cMessage, id=27)
INFO: Transmission of (inet::Packet)tcpseg(l=536) (594 B) (inet::SequenceChunk) length = 594 B successfully completed.
DETAIL: Start IFG period
```

```
Step1 #0: configuratorA    Msg stats: 8 scheduled / 401 existing / 540 created
```

Figure 4: TCP Simulation of the Three Subnet Network

Table 1: Resource Usage Comparison Between Simple and Complex Simulations

Resource	**Simple Topology (Client-Server)**	**Complex Topology (Three-Subnet Network)**
CPU Usage	10-15% of CPU	60-70% of CPU due to intensive routing and traffic processing
Memory Usage	Approx. 500 MB	2-3 GB due to larger routing tables and packet queues
Disk Usage	Minimal, depending on result files	Approx. 200 MB for 30-minute simulation; larger for fine-grained data
Scalability	Handled by standard workstations	Optimization required for large networks, parallel execution recommended

the vector and scalar files were exported as CSV from within OMNeT++. The same processes can also be carried out using the Scave tool in Python. Some fundamental data points were collected as proof of concept, and visual representations of these basic data points are provided below.

4 Results Analysis

The successful setup and execution of both basic and advanced network simulations using OMNeT++ and the INET framework confirm the completion of the simulation phase of the project. These simulations provide a strong foundation for implementing machine learning-based anomaly detection in the next stages of the project, where we will utilize real-time network traffic data generated through these environments.

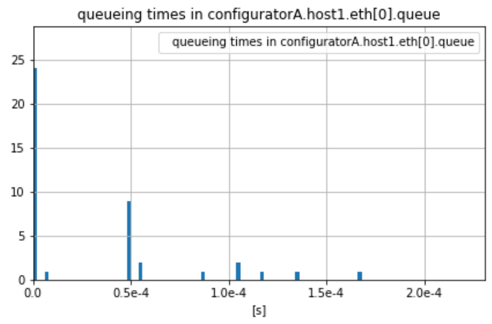


Figure 5: Queueing Times for Host1

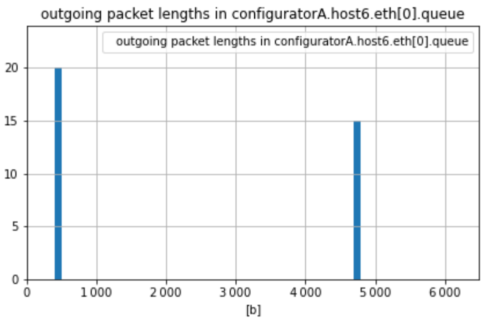


Figure 6: Outgoing Packet Lengths from Host6

Figure 7: Comparison of Queueing Times and Packet Lengths in the Three Subnet Network

5 References

1. OmNet++ Installation Guide: `https://doc.omnetpp.org/omnetpp/InstallGuide.pdf`
2. INET Download Section: `https://inet.omnetpp.org/Download.html`
3. INET Installation Guide: `https://github.com/inet-framework/inet/blob/master/INSTALL.md`
4. The Source File Related With Error: `https://github.com/inet-framework/inet/blob/master/src/inet/common/lifecycle/NodeStatus.cc`