PBL3: VANET SIMULATOR

1. Hướng dẫn cài đặt ns3.30.1

* Khởi tạo thư mục ảo và cài đặt tải phiên bản Python tương thích

|  |  |
| --- | --- |
| sudo apt install python3.8-venv | Tải xuống thư viện venv |
| python3.8 -m venv ns3 | Tạo thư mục ảo có tên ns3 |
| source ns3/bin/activate | Kích hoạt môi trường ảo |

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình 1: Khởi tạo thư mục ảo với phiên bản Python 3.8

* Cài đặt package cho g++ và python3  
  sudo apt-get install gcc g++ python

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình 2: Cài đặt package g++

* Tải xuống các dependencies cần thiết

sudo apt install python3-setuptools git

sudo apt install qt5-default

sudo apt install gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 3: Tải xuống các dependencies cần thiết

* Tải xuống ns-3

Truy cập trang chủ ns3

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình 4: Chọn tải về phiên bản 3.30.1

* Giải nén file đã tải đến home/user

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động  
Hình 5: Giải nén file ns3

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 6: Dùng lệnh “sudo apt update” để cập nhật các gói

* Tải các packages cho ns-3: g++ clang++ và cmake

sudo apt install cmake libc6-dev libc6-dev-i386 libclang-dev llvm-dev automake python3-pip

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình 7: Cài các packages

* Chuyển sang đường dẫn thư mục ns- allinone-3.30.1/ns-3.30.1

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 8: cd ns- allinone-3.30.1/ns-3.30.1

Thực thi các lệnh sau:  
./waf configure --enable-examples --enable-tests  
./waf build

./test.py

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình 8: Sau khi thực thi lệnh kiếm tra ./test.py

1. Tải SUMO

* Chạy các lệnh thực thi sau để tải phiên bản mới nhất của SUMO

sudo add-apt-repository ppa:sumo/stable

sudo apt-get update

sudo apt-get install sumo sumo-tools sumo-doc

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình 11: Tải SUMO sau khi thực thi các lệnh

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình 12: Thực thi lệnh sumo –version để kiểm tra phiên bản SUMO đã được cài đặt

* Truy cập vào địa chỉ tools của SUMO và mở giao diện OSM web

cd /usr/share/sumo/tools/

python osmWebWizard.py

* Chọn khu vực, loại phương tiện, số lượng.

Ảnh có chứa văn bản, ảnh chụp màn hình, bản đồ, phần mềm

Mô tả được tạo tự động

Hình 12: Giao diện OSM, và lựa chọn phương tiện, nhấn Generate Scenario để download map xuống.

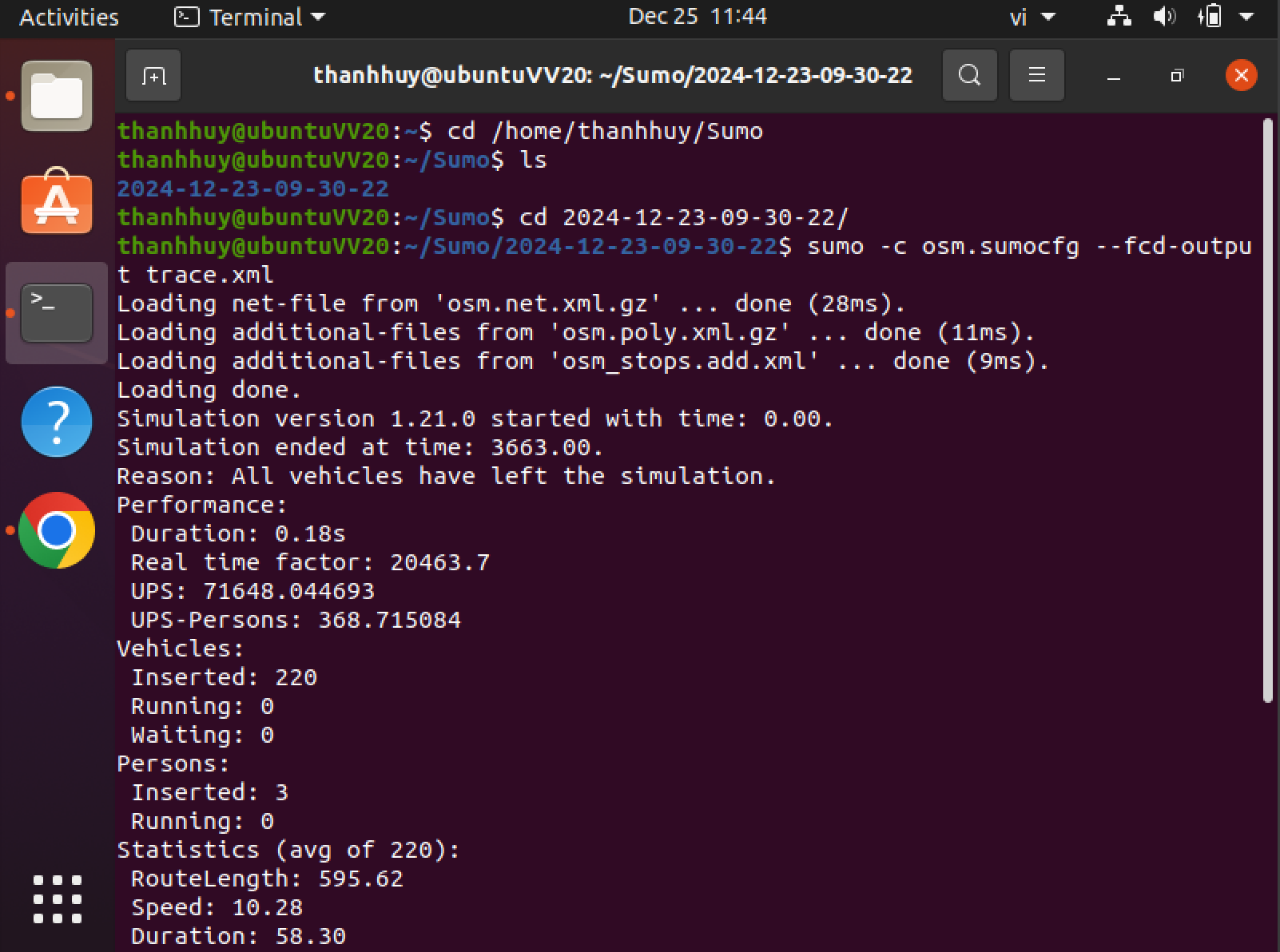
* Copy file ns2-mobility-trace.cc từ source/mobility/examples sang file scratch của ns3

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Hình 9: Copy file ns2-mobility-trace.cc

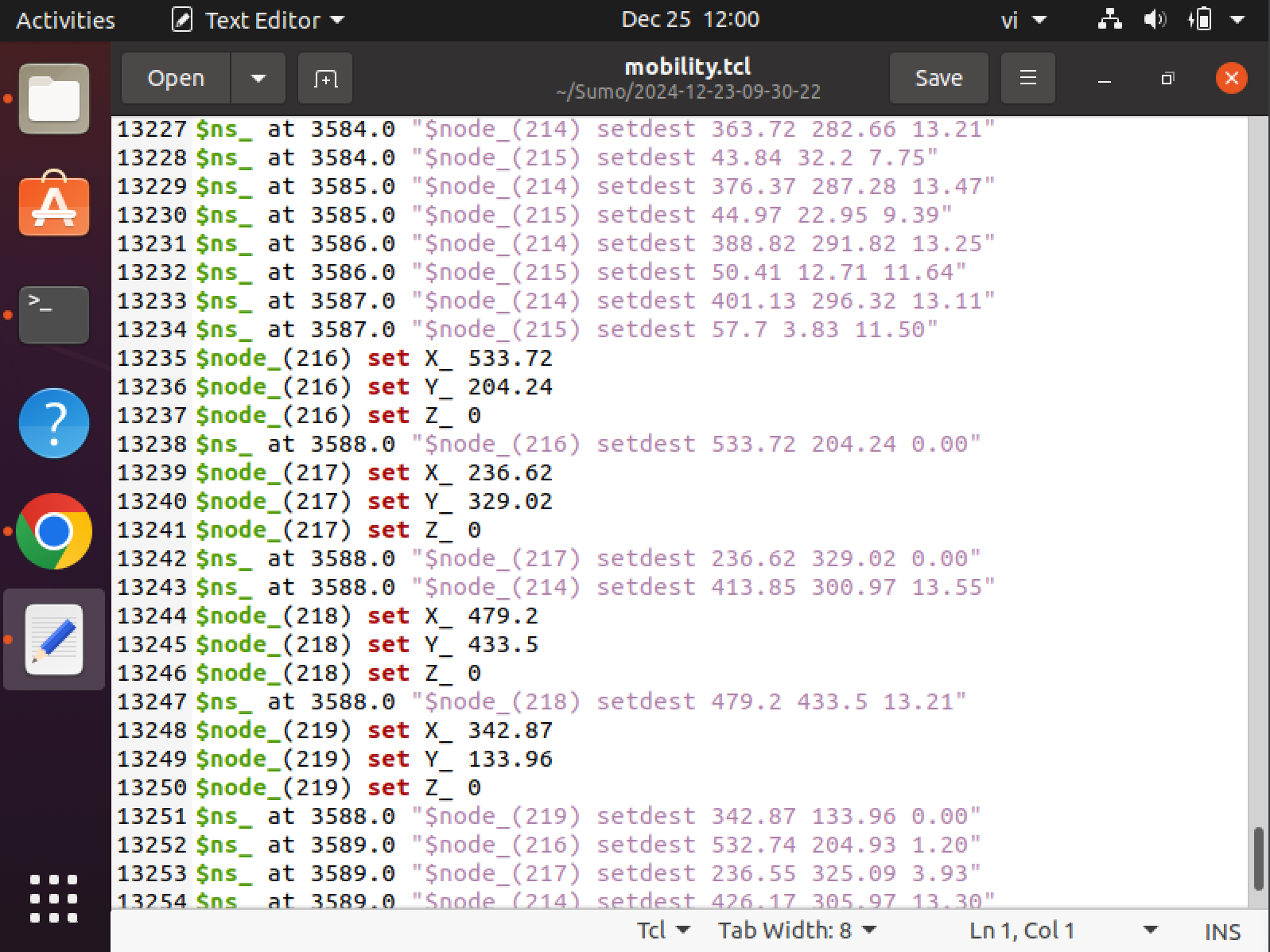
* Truy cập vào Sumo: cd /home/thanhhuy/Sumo
* Hiển thị các bản đồ đã tải xuống: ls
* Truy cập vào bản đồ vừa tải xuống: cd 2024-12-23-09-30-22
* Khởi tạo file trace.xml: sumo -c osm.sumocfg --fcd-output trace.xml



Hình 13: Sau khi thực thi lệnh khởi tạo file trace.xml

* Bây giờ, hãy chạy traceExporter.py có trong /usr/share/sumo/tools để xuất mô phỏng sang định dạng mà ns-3 hiểu:

python /usr/share/sumo/tools/traceExporter.py -i trace.xml --ns2mobility-output=/home/thanhhuy/Sumo/2024-12-23-09-30-22/mobility.tcl



Hình 14: File mobility.tcl

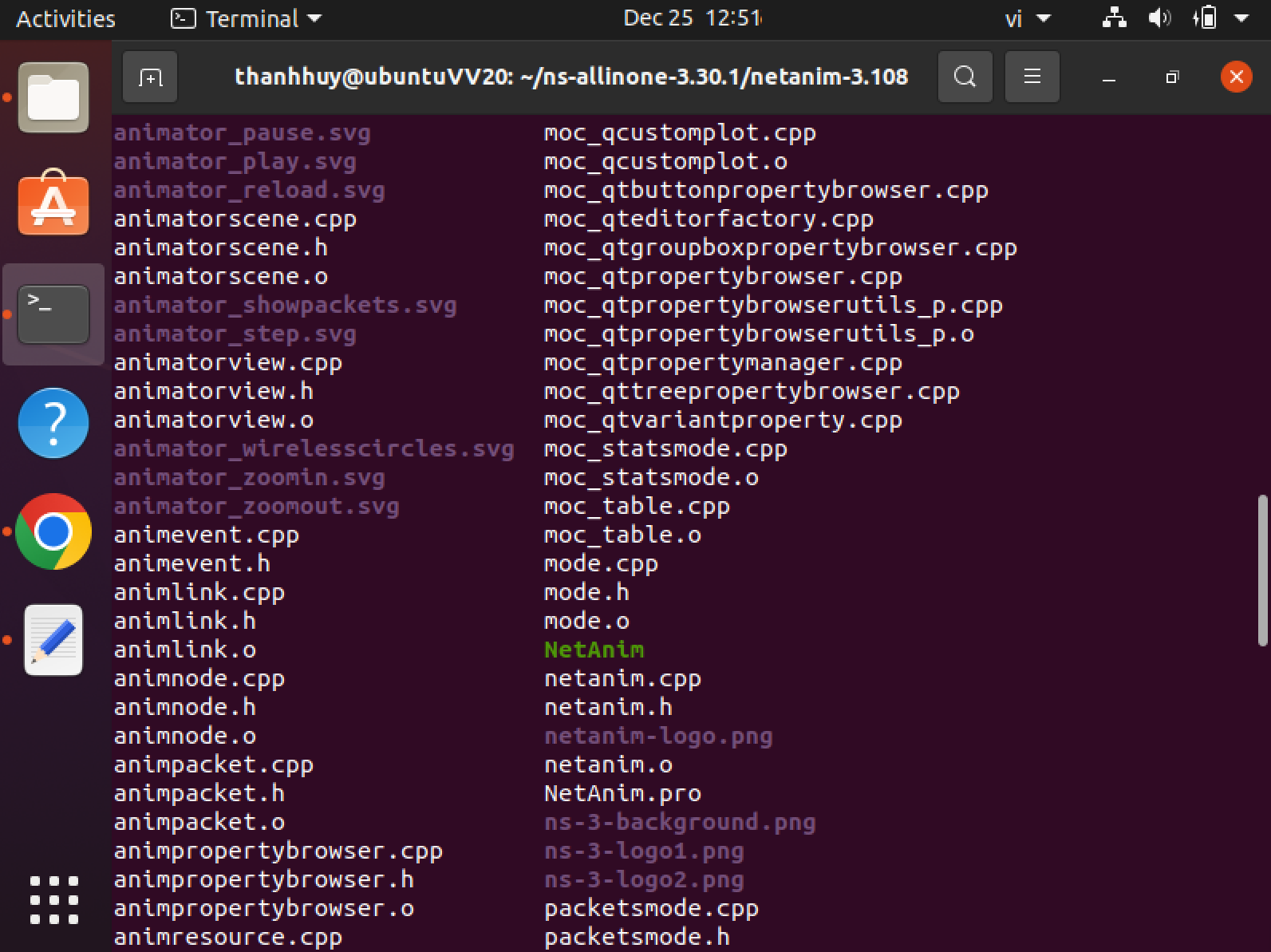
1. Mô phỏng trên NetAnim

* Biên dịch netAnim:

cd ~/netanim

qmake NetAnim.pro

make



Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Hình 15,16: Sau khi biên dịch thành công, xuất hiện file NetAnim, chạy lệnh ./NetAnim để kiểm tra

* Cập nhật thư viện netAnim vào file vanet-routing-compare.cc, và file Vanetanim.xml

#include "ns3/netanim-module.h"

AnimationInterface anim("Vanetanim.xml"); // Thêm vào trước Simulator::Run();

* Bổ sung code phân tích hiệu suất

// Network Performance Calculation

// Add performance analysis code to the vanet program

///////////////////////// Network Perfomance Calculation ///////////////////////

uint32\_t SentPackets = 0;

uint32\_t ReceivedPackets = 0;

uint32\_t LostPackets = 0;

int j=0;

float AvgThroughput = 0;

Time Jitter;

Time Delay;

Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());

std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

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

{

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

NS\_LOG\_UNCOND("----Flow ID:" <<iter->first);

NS\_LOG\_UNCOND("Src Addr" <<t.sourceAddress << "Dst Addr "<< t.destinationAddress);

NS\_LOG\_UNCOND("Sent Packets=" <<iter->second.txPackets);

NS\_LOG\_UNCOND("Received Packets =" <<iter->second.rxPackets);

NS\_LOG\_UNCOND("Lost Packets =" <<iter->second.txPackets-iter->second.rxPackets);

NS\_LOG\_UNCOND("Packet delivery ratio =" <<iter->second.rxPackets\*100/iter->second.txPackets << "%");

NS\_LOG\_UNCOND("Packet loss ratio =" << (iter->second.txPackets-iter->second.rxPackets)\*100/iter->second.txPackets << "%");

NS\_LOG\_UNCOND("Delay =" <<iter->second.delaySum);

NS\_LOG\_UNCOND("Jitter =" <<iter->second.jitterSum);

NS\_LOG\_UNCOND("Throughput =" <<iter->second.rxBytes \* 8.0/(iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds())/1024<<"Kbps");

SentPackets = SentPackets +(iter->second.txPackets);

ReceivedPackets = ReceivedPackets + (iter->second.rxPackets);

LostPackets = LostPackets + (iter->second.txPackets-iter->second.rxPackets);

AvgThroughput = AvgThroughput + (iter->second.rxBytes \* 8.0/(iter->second.timeLastRxPacket.GetSeconds()-iter->second.timeFirstTxPacket.GetSeconds())/1024);

Delay = Delay + (iter->second.delaySum);

Jitter = Jitter + (iter->second.jitterSum);

j = j + 1;

}

AvgThroughput = AvgThroughput/j;

NS\_LOG\_UNCOND("--------Total Results of the simulation----------"<<std::endl);

NS\_LOG\_UNCOND("Total sent packets =" << SentPackets);

NS\_LOG\_UNCOND("Total Received Packets =" << ReceivedPackets);

NS\_LOG\_UNCOND("Total Lost Packets =" << LostPackets);

NS\_LOG\_UNCOND("Packet Loss ratio =" << ((LostPackets\*100)/SentPackets)<< "%");

NS\_LOG\_UNCOND("Packet delivery ratio =" << ((ReceivedPackets\*100)/SentPackets)<< "%");

NS\_LOG\_UNCOND("Average Throughput =" << AvgThroughput<< "Kbps");

NS\_LOG\_UNCOND("End to End Delay =" << Delay);

NS\_LOG\_UNCOND("End to End Jitter delay =" << Jitter);

NS\_LOG\_UNCOND("Total Flod id " << j);

monitor->SerializeToXmlFile("manet-routing.flowmon", true, true);

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

* Cấu hình lại các thông số trong bản đồ

// Cấu hình kịch bản 2

else if (m\_scenario == 2)

{

// Theo dõi phương tiện thực tế trong KCT

// "mật độ trung bình, tổng cộng 40 phương tiện"

m\_traceFile = "/home/thanhhuy/Sumo/2024-12-23-09-30-22/vanetmobility.tcl"; //đường dẫn đến tệp "vanetmobility.tcl"

m\_logFile = "vanet.log";

m\_mobility = 1;

m\_nNodes = 40;

m\_TotalSimTime = 150.01;

m\_nodeSpeed ​​= 0;

m\_nodePause = 0;

m\_CSVfileName = "vanet.csv";

m\_CSVfileName = "vanet2.csv";

}

* Chạy chương trình:

./waf --run "scratch/vanet-routing-compare --protocol=1 --scenario=2"

* Mở mô phỏng netAnim

cd ns-allinone-3.30.1/netanim

./NetAnim

Khởi chạy file Vanetanim.xml để tiến hành mô phỏng

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

Hình 17:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Hình 18:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

Hình 19: