PBL3: VANET SIMULATOR

1. **Giới thiệu về VANET**

**VANET (Vehicular Ad Hoc Network) là gì?**

VANET (Mạng Ad Hoc phương tiện giao thông) là một loại mạng không dây đặc biệt, trong đó các phương tiện (vehicles) đóng vai trò là các nút mạng có khả năng tự tổ chức để giao tiếp trực tiếp với nhau (V2V - Vehicle-to-Vehicle) hoặc với các cơ sở hạ tầng cố định (V2I - Vehicle-to-Infrastructure).

**Đặc điểm chính của VANET:**

1. **Cấu trúc động (Dynamic Topology)**:
   * Các phương tiện trong VANET di chuyển liên tục với tốc độ cao, làm cho cấu trúc mạng thay đổi nhanh chóng.
2. **Phạm vi giao tiếp ngắn**:
   * Các nút (phương tiện) thường giao tiếp trong một phạm vi giới hạn (thường từ 100 đến 300 mét).
3. **Tính di động cao**:
   * Các phương tiện di chuyển theo lộ trình, vận tốc, và hướng khác nhau, dẫn đến sự phức tạp trong việc duy trì kết nối mạng.
4. **Nguồn năng lượng không bị hạn chế**:
   * Không giống như mạng cảm biến (sensor networks), phương tiện có thể sử dụng năng lượng từ động cơ, do đó không bị giới hạn về nguồn năng lượng.
5. **Bảo mật và quyền riêng tư**:
   * Do sử dụng trong các ứng dụng quan trọng như an toàn giao thông, bảo mật thông tin và quyền riêng tư là yếu tố quan trọng.

**Kiến trúc của VANET:**

1. **V2V (Vehicle-to-Vehicle)**:
   * Giao tiếp trực tiếp giữa các phương tiện, thường để chia sẻ thông tin thời gian thực như tình trạng giao thông, tai nạn, hoặc điều kiện đường xá.
2. **V2I (Vehicle-to-Infrastructure)**:
   * Giao tiếp giữa phương tiện và các trạm cơ sở (RSU - Road Side Unit) hoặc các điểm truy cập hạ tầng cố định.
3. **Hybrid (Kết hợp V2V và V2I)**:
   * Sử dụng cả hai mô hình trên để tăng hiệu quả mạng lưới.
4. **Mô tả chương trình**

* **Chức năng**: Chương trình này cho phép mô phỏng các kịch bản mạng VANET (Vehicular Ad Hoc Network) trong **ns-3** để đánh giá hiệu năng, thông qua việc kiểm tra:
  + Đặc điểm của các lớp MAC/PHY trong giao thức 802.11p.
  + Các mô hình suy hao sóng truyền (ví dụ: Friss, Two-Ray Ground, ITU R-P.1411).
  + Các loại lưu lượng ứng dụng (ví dụ: thông điệp an toàn cơ bản - Basic Safety Message) hoặc lưu lượng định tuyến (ví dụ: DSDV, AODV, OLSR, DSR).
* **Kịch bản mô phỏng**:
  + **Kịch bản tổng hợp (Scenario 1)**:
    - Chạy trong 10 giây với 40 nút (tương ứng với 40 phương tiện) di chuyển theo **RandomWaypointMobilityModel**.
    - Tốc độ: 20 m/s, không có thời gian dừng trong vùng 300x1500 m.
    - Sử dụng WiFi 802.11p trên kênh điều khiển (Control Channel) 10 MHz liên tục.
    - Các nút gửi thông điệp an toàn (200 byte) với tần suất 10 lần/giây ở tốc độ 6 Mbps.
    - Tất cả các nút cũng có thể định tuyến liên tục các gói tin 64 byte đến 1 trong 10 nút đích.
    - Giao thức định tuyến mặc định: **AODV**.
    - Mô hình suy hao: **Two-Ray Ground**.
    - Công suất phát: 20 dBm, phạm vi truyền tải thông điệp an toàn: 145 m.
  + **Kịch bản mô phỏng từ vết di chuyển thực tế (Scenario 2)**:
    - Dựa trên các tệp dấu vết di chuyển của ns-2 được lấy từ trang web nghiên cứu về ad hoc.
    - Mô phỏng 300 giây với 99 phương tiện tại khu vực Unterstrass, Zurich, Thụy Sĩ, sử dụng dữ liệu từ giao thông thực tế.

**Đầu ra chương trình**

* **Các đầu ra có thể tạo ra**:
  + Tệp CSV ghi nhận số liệu thống kê về việc nhận dữ liệu mỗi giây.
  + Tệp CSV chứa số liệu thống kê cuối cùng.
  + Bảng định tuyến được ghi lại ở giây thứ 5 trong quá trình mô phỏng.
  + Tệp dấu vết ASCII.
  + Tệp PCAP (Packet Capture) cho từng nút.

**Lưu và tải cấu hình kịch bản**

* **Cách lưu kịch bản**:
  + Cấu hình mô phỏng có thể được lưu dưới dạng tệp văn bản (dùng **config-store**) và tái sử dụng sau này.
  + Ví dụ:
    - Để lưu cấu hình dưới tên scenario1.txt:

bash

Copy code

./waf --run "vanet-routing-compare --scenario=1 --saveconfig=scenario1.txt"

* + - Để tái sử dụng cấu hình:

bash

Copy code

./waf --run "vanet-routing-compare --loadconfig=scenario1.txt"

**Sơ đồ lớp**

* Chương trình bao gồm các thành phần chính:
  + **main()**:
    - Sử dụng VanetRoutingExperiment.
    - Thành phần chính:
      * WifiApp
      * ConfigStoreHelper
      * WaveBsmHelper (chứa WaveBsmStats)
      * RoutingHelper (chứa RoutingStats)
      * WifiPhyStats

1. **Thư viện và các hàm chức năng sử dụng**

**Các thư viện chính được sử dụng**

1. **#include <fstream> và #include <iostream>**:
   * Được sử dụng để làm việc với tệp và thực hiện các thao tác nhập/xuất dữ liệu trong chương trình.

**Các module của NS-3**

1. **#include "ns3/core-module.h"**:
   * Module lõi của NS-3, cung cấp các lớp và công cụ cơ bản như bộ lập lịch, trình giả lập thời gian, và hỗ trợ tạo đối tượng.
2. **#include "ns3/network-module.h"**:
   * Module mạng, cung cấp các công cụ để xây dựng và mô phỏng giao diện mạng, gói tin, và liên kết mạng.
3. **#include "ns3/internet-module.h"**:
   * Module internet, hỗ trợ giao thức IP và các giao thức liên quan (TCP, UDP).
4. **#include "ns3/mobility-module.h"**:
   * Module hỗ trợ mô phỏng mô hình di chuyển của các nút, cần thiết cho các kịch bản VANET.
5. **#include "ns3/aodv-module.h"**, **ns3/olsr-module.h**, **ns3/dsdv-module.h**, **ns3/dsr-module.h**:
   * Các module định tuyến:
     + **AODV (Ad-hoc On-Demand Distance Vector)**: Giao thức định tuyến theo yêu cầu.
     + **OLSR (Optimized Link State Routing)**: Giao thức định tuyến dựa trên trạng thái liên kết.
     + **DSDV (Destination-Sequenced Distance Vector)**: Giao thức định tuyến vector khoảng cách theo đích.
     + **DSR (Dynamic Source Routing)**: Giao thức định tuyến dựa trên nguồn.
6. **#include "ns3/applications-module.h"**:
   * Module ứng dụng, cung cấp các công cụ để mô phỏng các ứng dụng mạng như gửi và nhận dữ liệu.
7. **#include "ns3/itu-r-1411-los-propagation-loss-model.h"**:
   * Mô hình suy hao sóng dựa trên tiêu chuẩn ITU R-P.1411, phù hợp cho các kịch bản truyền sóng trực tiếp (LOS - Line Of Sight).
8. **#include "ns3/ocb-wifi-mac.h"** và **#include "ns3/wifi-80211p-helper.h"**:
   * Hỗ trợ chuẩn **802.11p**, giao thức truyền thông không dây dành riêng cho mạng VANET.
9. **#include "ns3/wave-mac-helper.h"** và **#include "ns3/wave-helper.h"**:
   * Hỗ trợ giao thức WAVE (Wireless Access in Vehicular Environments), một phần mở rộng của 802.11p.
10. **#include "ns3/flow-monitor-module.h"**:
    * Module giám sát lưu lượng, giúp thu thập và phân tích dữ liệu về hiệu năng mạng.
11. **#include "ns3/config-store-module.h"**:
    * Cho phép lưu và tải lại cấu hình mô phỏng từ tệp cấu hình.
12. **#include "ns3/integer.h"**:
    * Hỗ trợ các tham số số nguyên trong cấu hình NS-3.
13. **#include "ns3/wave-bsm-helper.h"**:
    * Trình trợ giúp để tạo và quản lý Basic Safety Message (BSM), một loại tin nhắn trong giao tiếp V2V.
14. **#include "ns3/yans-wifi-helper.h"**:
    * Hỗ trợ mô phỏng lớp vật lý và MAC của WiFi dựa trên mô hình YANS (Yet Another Network Simulator).
15. **#include "ns3/netanim-module.h"**:
    * Module hỗ trợ tạo hoạt hình và hình ảnh hóa mạng bằng công cụ NetAnim.

**Các nhóm chức năng**

1. **Cài đặt mô phỏng**:
   * **Simulate**: Hàm chính để chạy mô phỏng, phân tích các tham số dòng lệnh và gọi các hàm cấu hình cần thiết.
   * **SetDefaultAttributeValues**: Thiết lập các giá trị mặc định cho các thuộc tính mô phỏng.
   * **ParseCommandLineArguments**: Phân tích các tham số đầu vào để cấu hình mô phỏng.
2. **Cấu hình nút và mạng**:
   * **ConfigureNodes**, **ConfigureChannels**, **ConfigureDevices**, **ConfigureMobility**: Cấu hình các khía cạnh khác nhau của môi trường mô phỏng, bao gồm nút, kênh, thiết bị, và mô hình di động.
   * **AssignIpAddresses**: Gán địa chỉ IP cho các giao diện mạng.
3. **Định tuyến và ứng dụng**:
   * **Install**: Cài đặt giao thức định tuyến hoặc ứng dụng trên các nút.
   * **SetupRoutingProtocol**, **SetupRoutingMessages**: Cấu hình các giao thức và thông điệp định tuyến.
   * **ReceiveRoutingPacket**, **PrintReceivedRoutingPacket**: Xử lý và ghi lại các gói tin định tuyến được nhận.
4. **Ghi log và thống kê**:
   * **SetLogging**, **SetupLogging**, **PhyTxTrace**, **PhyRxDrop**, **PhyTxDrop**: Quản lý ghi log và các sự kiện liên quan đến tầng vật lý.
   * **CheckThroughput**: Đánh giá thông lượng trong quá trình mô phỏng.
5. **Thực thi và xuất kết quả**:
   * **RunSimulation**: Thực thi mô phỏng.
   * **ProcessOutputs**: Xử lý và xuất kết quả mô phỏng, có thể sử dụng **WriteCsvHeader** để tạo tệp CSV.
6. **Thiết lập di động và kịch bản**:
   * **SetupAdhocMobilityNodes**, **SetupAdhocDevices**: Cấu hình các nút di động và thiết bị cho mạng ad hoc.
   * **SetupScenario**: Chuẩn bị một kịch bản mô phỏng cụ thể.
7. **Xử lý lệnh**:
   * **CommandSetup**: Cấu hình các tùy chọn dòng lệnh cho mô phỏng.
8. **Theo dõi và sự kiện**:
   * **OnOffTrace**: Xử lý sự kiện theo dõi cho các ứng dụng On-Off.
   * **CourseChange**: Ghi lại các sự kiện liên quan đến di chuyển.
9. **Điểm bắt đầu chính**:
   * **main**: Điểm khởi đầu của mô phỏng, điều phối việc thiết lập, thực thi và dọn dẹp.
10. **Cài đặt công cụ hỗ trợ và các bước thực hiện**
11. Hướng dẫn cài đặt ns3.30.1

* Tải xuống thư viện venv

sudo apt install python3.8-venv

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

python3.8 -m venv ns3

source ns3/bin/activate

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/thanhhuy

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. Cài đặt 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.

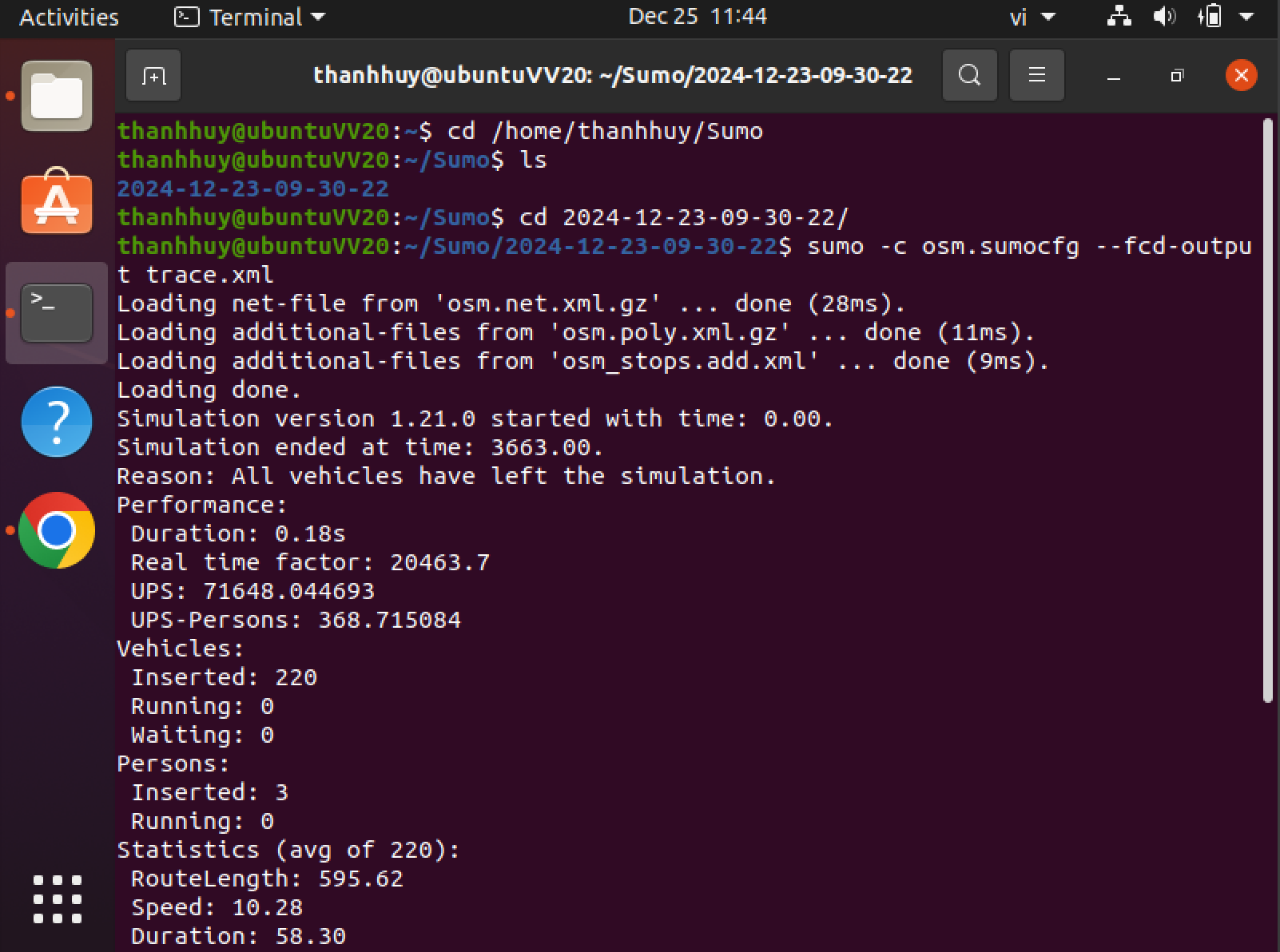
* Copy file ns2-mobility-trace.cc từ source/mobility/examples sang file scratch của ns3

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

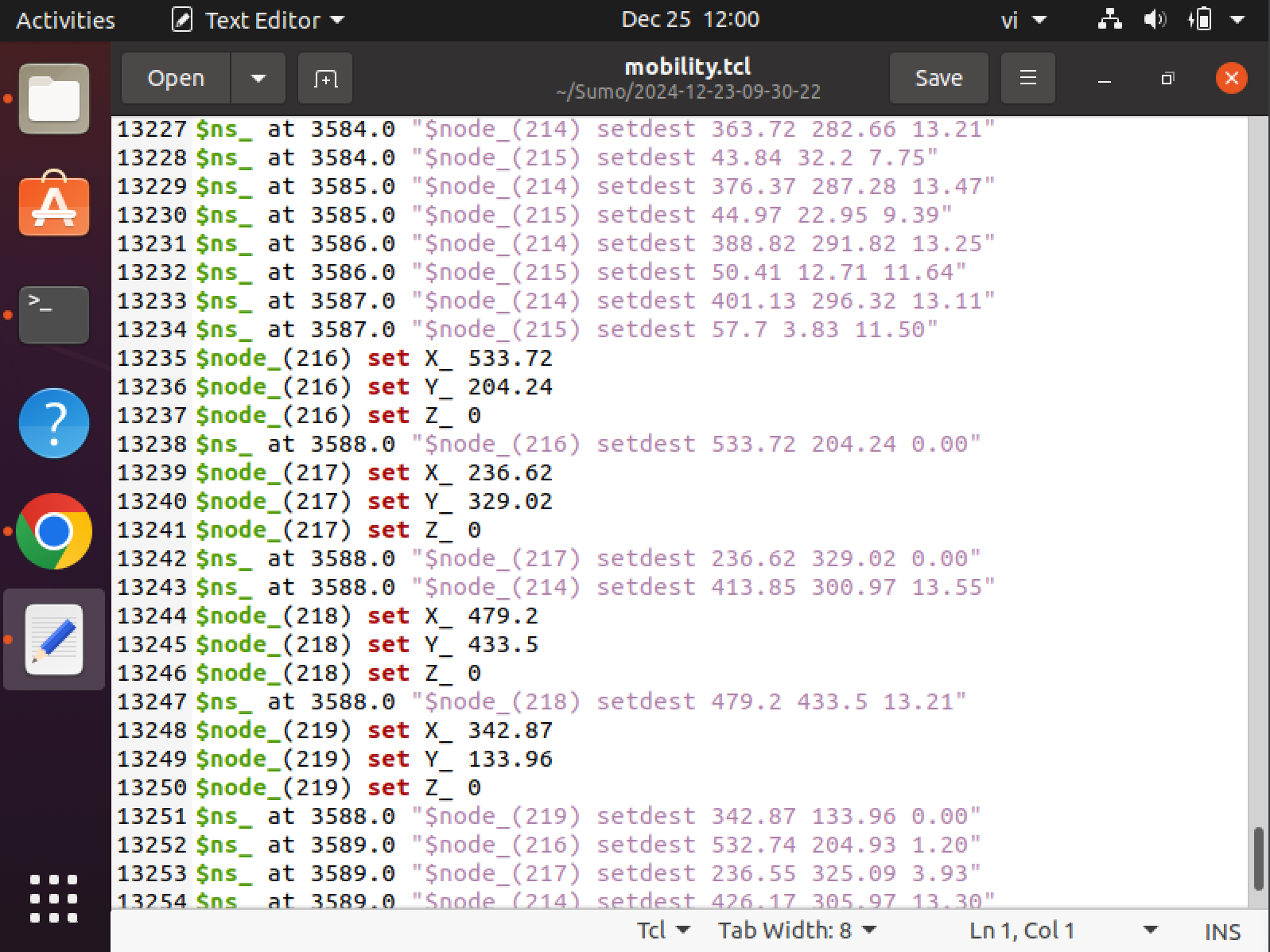
* Truy cập vào Sumo: cd /home/thanhhuy/Sumo
* Hiển thị các bản đồ đã tải xuống: ls
* Truy cập vào bản đồ vừa tải xuống: cd 2024-12-23-09-30-22
* Khởi tạo file trace.xml: sumo -c osm.sumocfg --fcd-output trace.xml



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

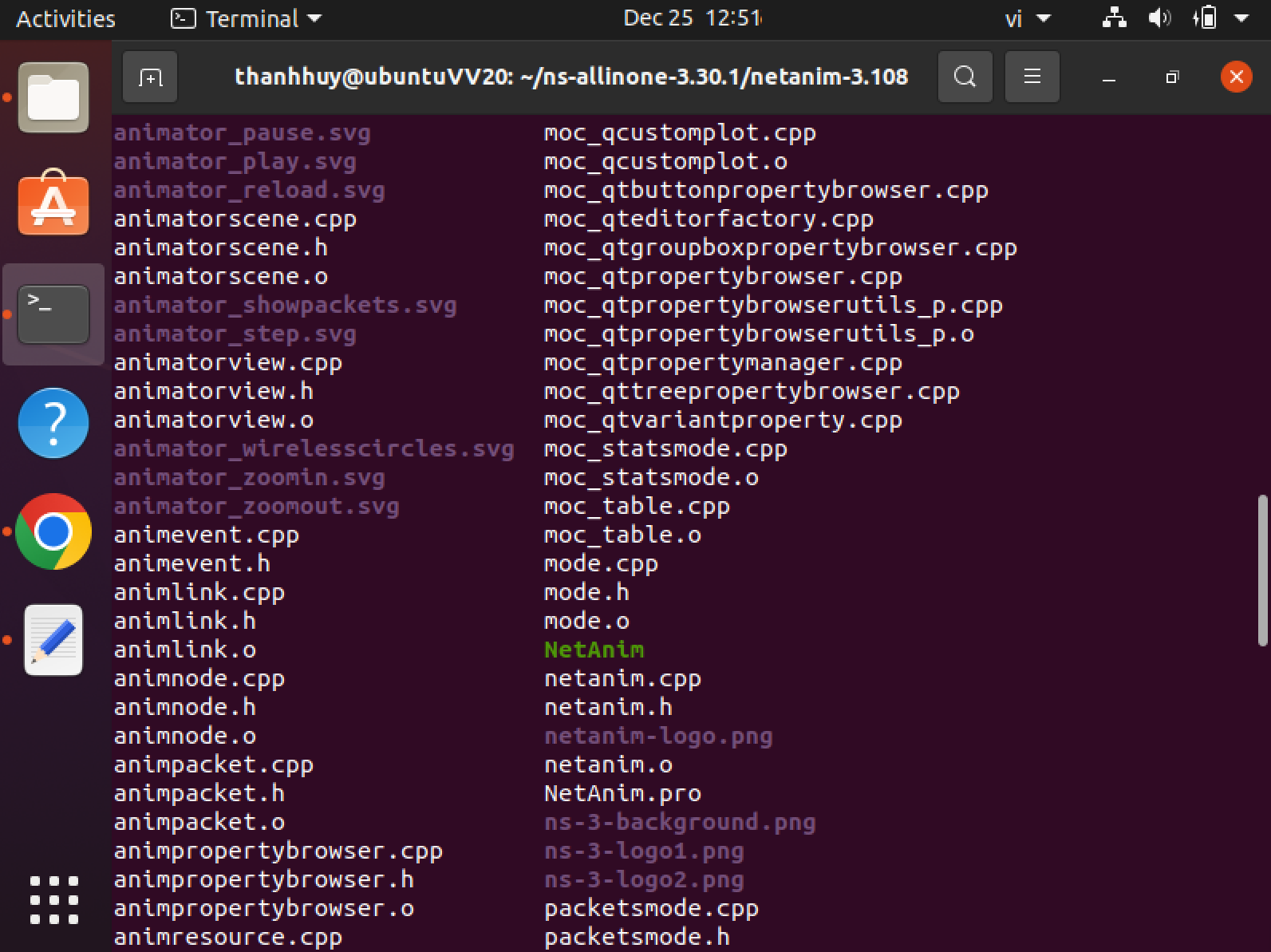
1. Mô phỏng trên NetAnim

* 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;

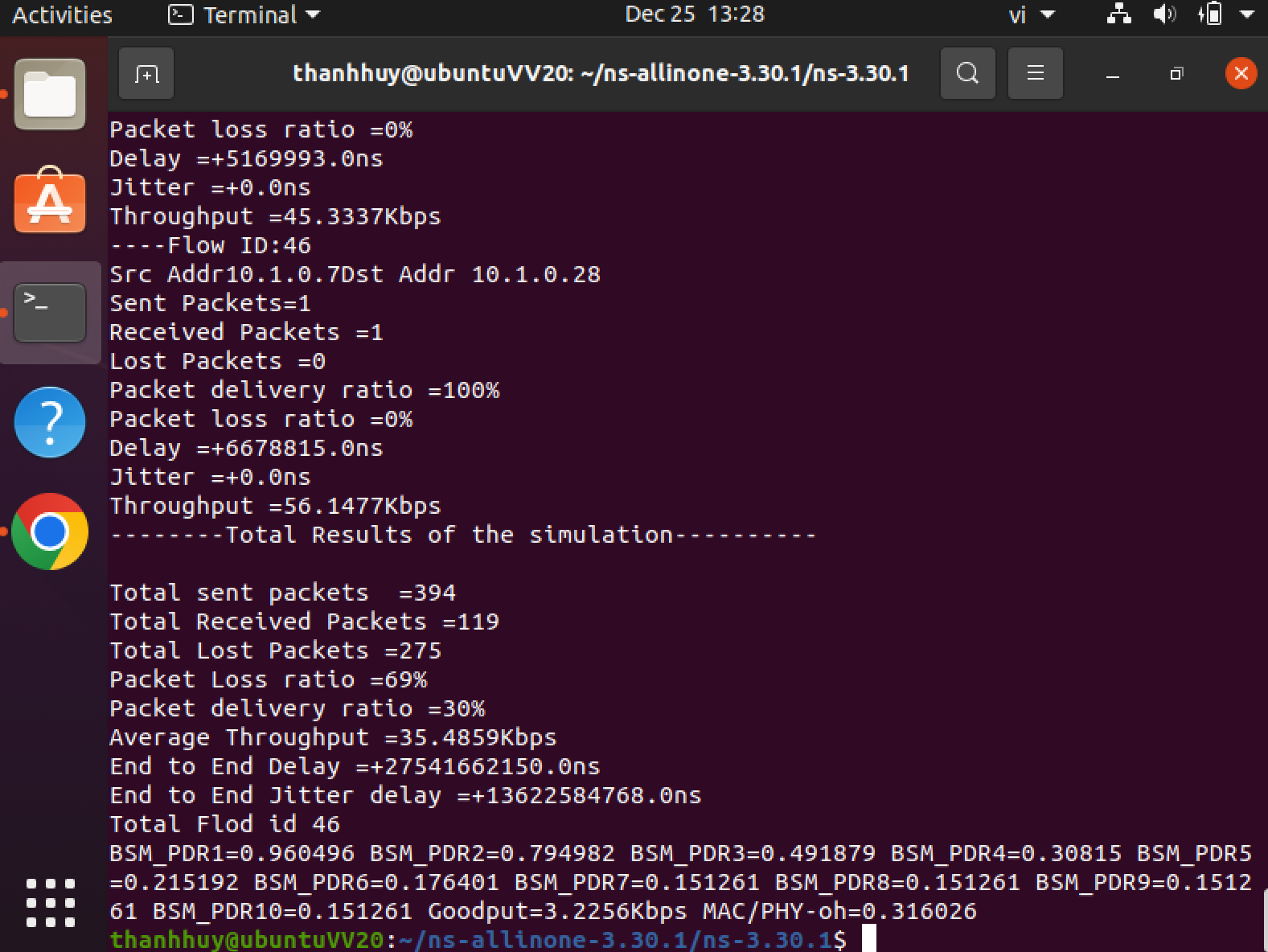
m\_CSVfileName = "vanet.csv";

m\_CSVfileName = "vanet2.csv";

}

* Chạy chương trình:

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



Hình : Chạy file code vanet-routing-compare.cc

* 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: