Dynamic Traffic Light Control in NetSim VANETs

Software: NetSim Standard v13.0 (32/64 bit), SUMO 1.2.0, Visual Studio 2019

Project Download Link:

https://github.com/NetSim-TETCOS/Dynamic_Traffic_Light_Control_in_NetSim_VANETs_v13.0/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and setup the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

Vehicular ad-hoc networks (VANETs)

VANETs are created by applying the principles of mobile ad hoc networks (MANETs), vehicle-to-vehicle and vehicle-to-roadside communication architectures co-exist in VANETs to provide road safety, navigation, and other roadside services. VANETs are a key part of the intelligent transportation systems (ITS) framework. In VANETs, Vehicles and roadside units (RSUs) are the communicating nodes, providing each other with information, such as safety warnings and traffic information. Both types of nodes are dedicated short-range communications (DSRC) devices. Roadside unit (RSU). The RSU is a WAVE device usually fixed along the roadside or in dedicated locations such as at junctions or near parking spaces.

Vehicles and RSUs have communication capabilities which allow them to send and receive network packets. They periodically broadcast traffic safety messages called "Basic Safety Messages" (BSMs) to all the other vehicles in its communication range In NetSim, users can model network traffic between vehicles V2V and between vehicle to infrastructure V2I.The BSM Application class sends and receives the IEEE 1609 WAVE (Wireless Access in Vehicular Environments) Basic Safety Messages (BSMs). The BSM is a 20-byte packet that is generally broadcast from every vehicle at a nominal rate of 10 Hz. In NetSim this can be configured as a broadcast or as a unicast application.

Dynamic Traffic control

In the urban areas, the traffic light systems are designed in such a way that the waiting time of the vehicles in the traffic signal is independent of the traffic density in that road. In VANETs the traffic signal can be modelled to dynamically change the traffic light based on the traffic congestion in the respective roads. In this example the emergency vehicles are prioritized over other vehicles and traffic signals are controlled dynamically.

In the VANET example shown below, there are two lanes namely East-West (EW) lane and North-South (NS) lane. At the intersection of roads there is a traffic signal that is programmed to allow only the emergency vehicles in the NS lane and has regular traffic in the EW lane. The vehicles in the scenario are allowed to have two movements which are going straight in the lane and taking a U-turn at the end of the either of the lanes.

In this scenario, the vehicle movements are detected by the Roadside Unit (RSU). The Emergency vehicles in the NS lane communicates with the RSU throughout the simulation time.

The vehicles in the EW lane and NS lane have green and red light respectively, except when RSU triggers the change of traffic light in NS lane from red to green when the emergency vehicle approaches the intersection of lanes where the traffic signal is present so that it is allowed to pass

through the traffic signal as soon as it arrives .Here the control signals are passed from RSU in NetSim to the traffic light controller in SUMO through the Traffic Control Interface (TraCI).

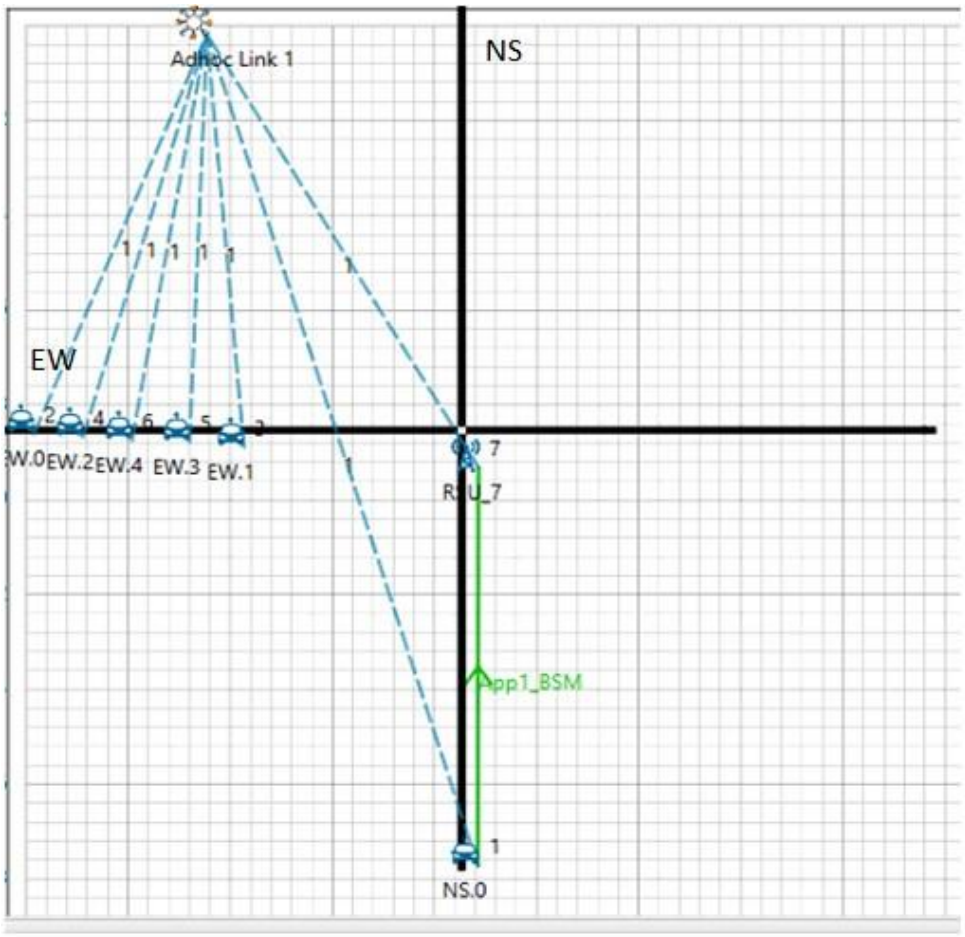


Figure 1: The Scenario shown two lanes namely East-West (EW) lane and North-South (NS) lane

NetSim sumo interface

NetSim's VANET module allows users to interface with SUMO which is an open-source road traffic simulation package designed to handle vehicular & road networks. The road traffic simulation is done by SUMO while NetSim does the network simulation along with RF propagation modelling in the physical layer. While SUMO Simulates the road traffic conditions and movements, NetSim Simulates the communication occurring between the Vehicles.

NetSim and SUMO are interfaced using 'pipes. A pipe is a section of shared memory that processes use for communication. SUMO process writes information to pipe, then NetSim process reads the information from pipe. On running the Simulation, SUMO determines the positions of vehicles with respect to time as per the road conditions. NetSim reads the coordinates of vehicles from SUMO (through pipes) in runtime and uses it as input for vehicles mobility.

Traffic Signal phases

For this example, four phases of traffic signal are defined in sumo as shown below. Here green light indicates that vehicles can cross the traffic signal. Red light indicates that vehicles must stop at the traffic signal. Yellow light indicates that the vehicles must decelerate and finally stop when traffic signal changes to red.

Phase 0: This phase indicates that NS lane has green light and EW lane has red light for a duration of 10s.
Phase 1: This phase indicates that NS lane has yellow light and EW lane has red light for a duration of 6s.
Phase 2: This phase indicates that NS lane has red light and EW lane has green light for a duration of 31s.
Phase 3: This phase indicates that NS lane has red light and EW lane has yellow light for a duration of 6s.
In this scenario traffic signal is always in phase 2 until an emergency vehicle in NS lane triggers the signal change from phase 2 to phase 3. When the duration of phase 3 is completed, the signal goes to phase 0.

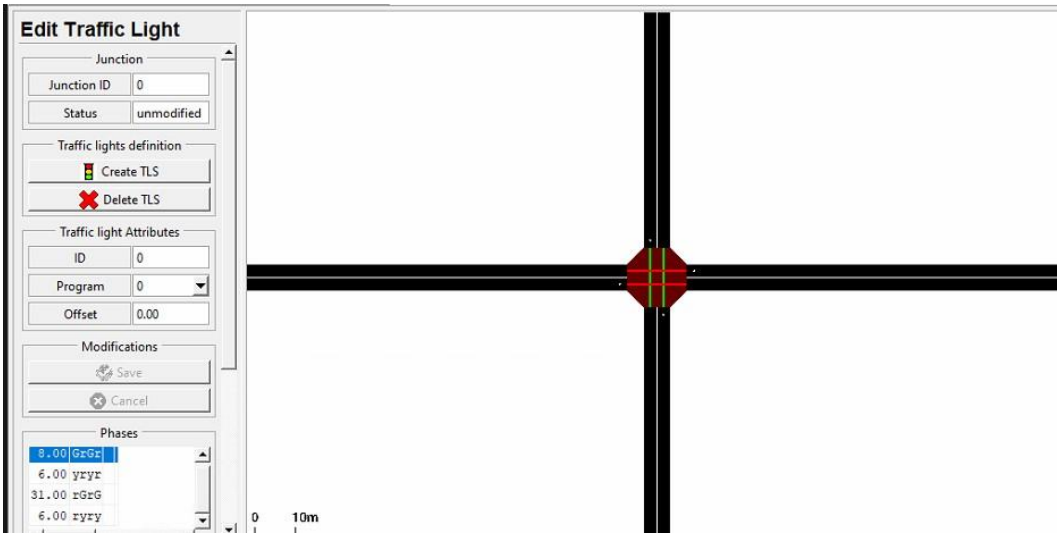


Figure 2: Four phases of traffic signal defined in sumo

Steps to simulate

1. Open the Source codes in Visual Studio by going to Your work-> Workspace Options and Clicking on Open code button in NetSim Home Screen window.

2. Under the Application project in the solution explorer, you will be able to see that Application.c file which contains the source codes related to interactions with Sumo and controlling the traffic signal in NetSim respectively.

3. Based on whether you are using NetSim 32 bit or 64-bit setup you can configure Visual studio to build 32 bit or 64-bit Dll files respectively as shown below

4. Right click on the Application in the solution explorer and select Rebuild.

5. After rebuilding the Application project if an error occurs.

6. Right click on Mobility project and Rebuild the Mobility project first.

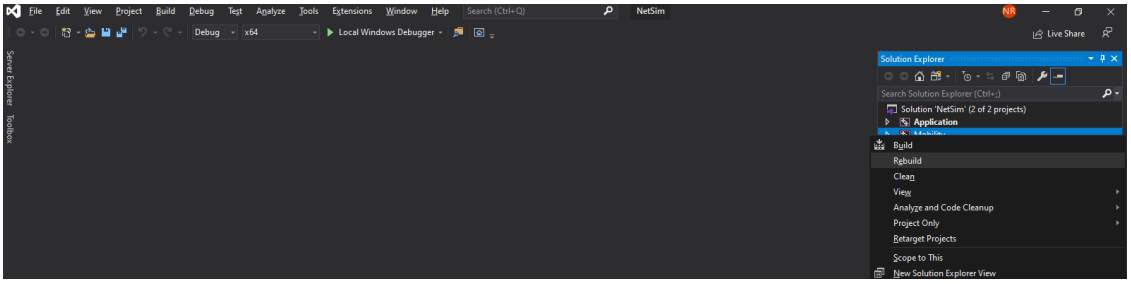


Figure 3: Screen shot of NetSim project source code in Visual Studio

7. Goto Dynamic_Traffic_Workspace directory, open bin_x64 and select Mobility.lib file as shown below:

8. Copy the Mobility.lib file and paste it in lib_x64 folder present in src folder of Dynamic Traffic_Workspace and rename the Mobility.lib file as libMobility.lib as shown below:

9. Now right click on Application project and click on Rebuild.

Example

1. The Dynamic_Traffic_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the Dynamic_Traffic_Control from the list of experiments.
2. The saved network scenario shown below consists of 6 vehicles, 5 in the EW lane and one emergency vehicle in the NS lane. The emergency vehicle transmits BSM Application packets periodically to RSU.

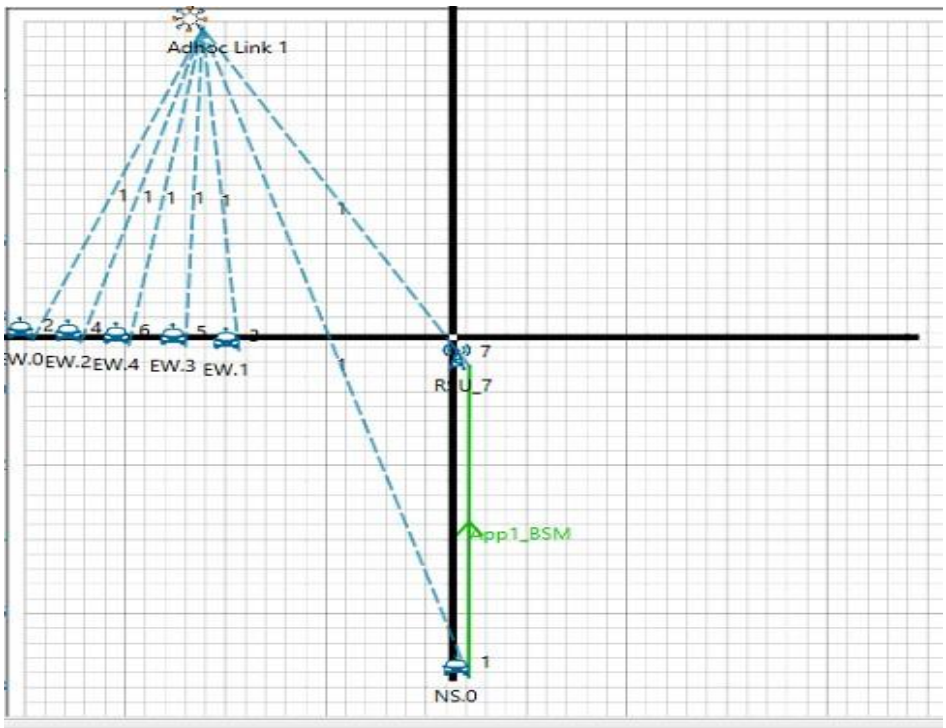


Figure 4: Network topology in this project

3. The downloaded folder consists of a folder named **sumo,** this folder contains a file**SumoRun.py** a python script. This file must be replaced with the original SumoRun.py file present in the bin folder of NetSim install directory.

4. Run the Scenario by selecting the **play and record animation** option as shown below and click OK. You will observe that as the simulation starts in NetSim, SUMO gets initialized and there are three windows open during the runtime. NetSim Console shows the Control messages being sent to sumo, Sumo simulation is seen in other console where the received control messages are displayed, simultaneously in the Sumo GUI window the vehicle movement and traffic signal can be seen when zoomed.

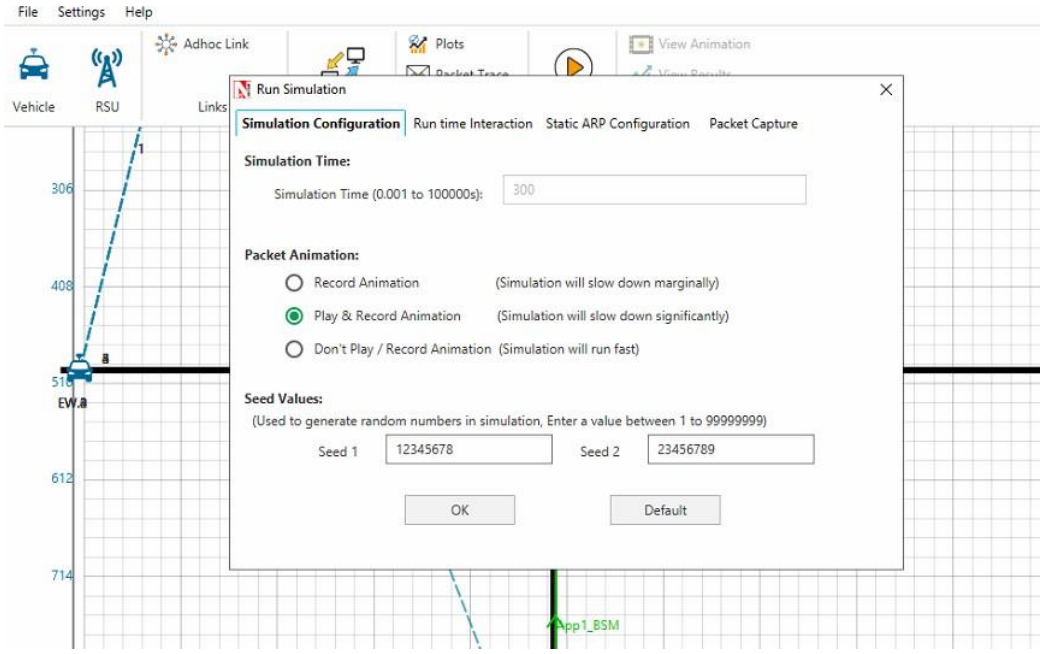


Figure 5: Select play and record animation in simulation configuration window

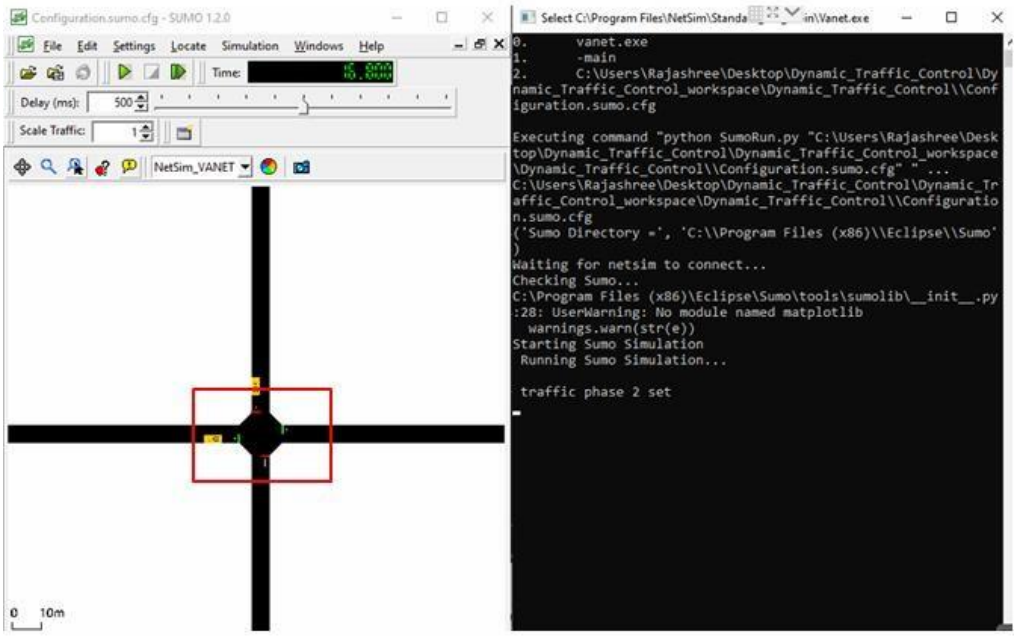
Results and discussion

It can be observed that when the control signal is sent from NetSim to Sumo (can be seen in NetSim console) the Traffic signal changes from phase 2 to phase 3 as soon as the emergency vehicle arrives in the NS lane (can be seen in Sumo GUI).

1. The Sumo GUI and Sumo console shown below is traffic signal being in phase 2.

2. The image shown below is when the lane is zoomed and the traffic signal phase 2 can be observed.

3. The control message being sent from NetSim to Sumo can be seen in the image below.



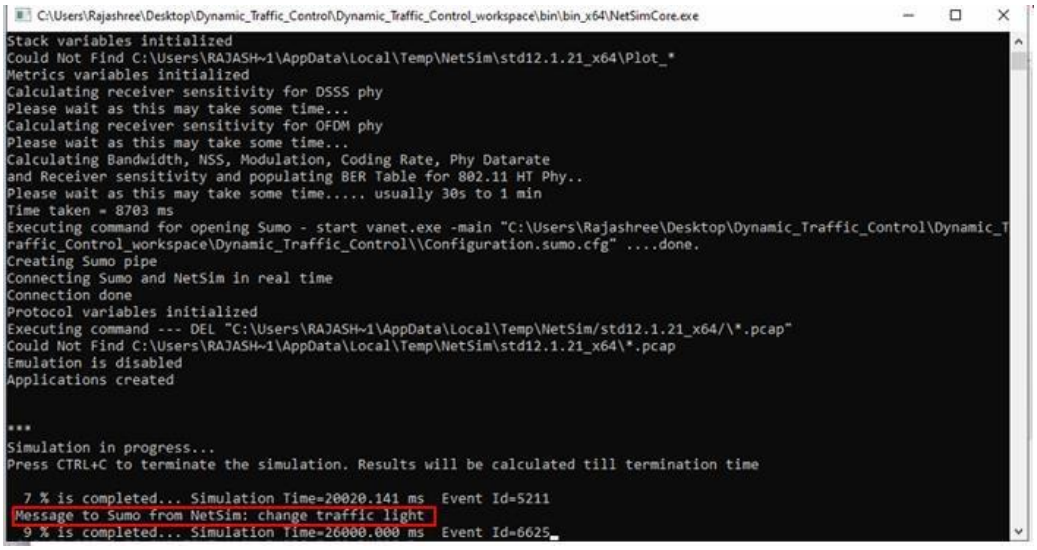


Figure 6: Control message being sent from NetSim to Sumo

4. Traffic signal changes from phase 2 to phase 3 indicating that vehicles in the EW lane have to slow down due to the control message received by NetSim in the sumo console is shown below.

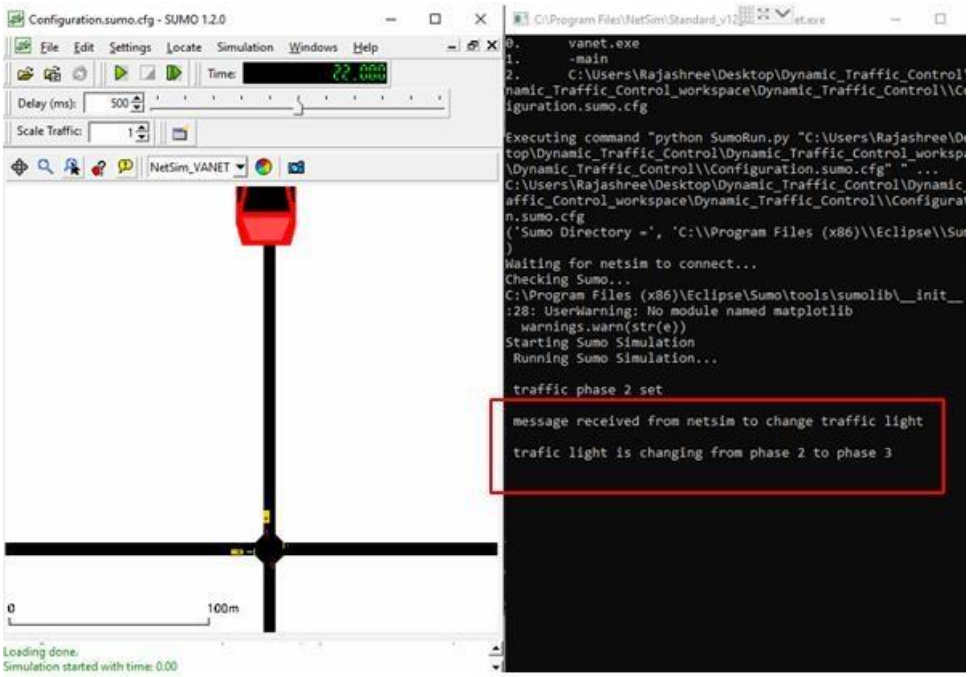


Figure 7: Traffic signal changes from phase 2 to phase 3

5. The traffic signal changes from red to green in the NS lane which can be observed in the traffic signal shown below.

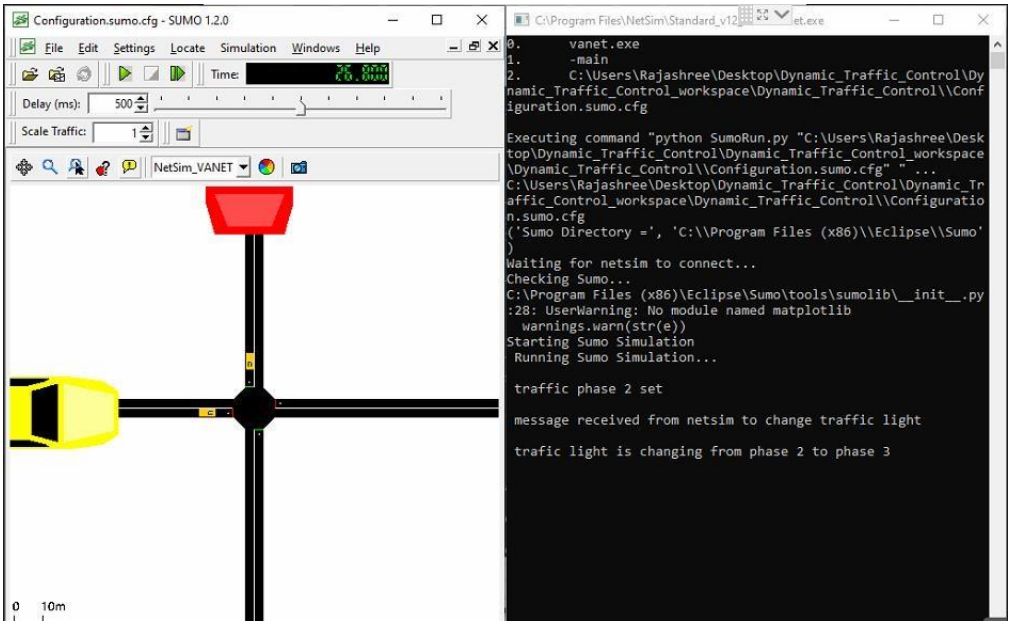


Figure 8: Traffic signal changes from red to green in the NS lane

Inference:

In this example, a simple scenario with a single emergency vehicle in one of the lanes shows that the RSU in VANETs can be used to control the traffic signal dynamically by communication between vehicles and RSUs. Further the real time scenarios can be created based on this example.

Appendix: NetSim source code modifications

Changes to int fn_NetSim_Application_Run(), in Application.c file, within Application project

```
/*Trafic light cases Case 0: no change in traffic light Case 1: change traffic light*/
NETSIM_ID vehicle, rsu;
vehicle = pstruEventDetails->pPacket->nSourceId;
NETSIM_Name vehicle_name = DEVICE_NAME(pstruEventDetails->pPacket->nSourceId);
rsu = pstruEventDetails->nDeviceId;
double time = pstruEventDetails->dEventTime;
double y_rsu = DEVICE(rsu)->pstruDevicePosition->Y;
double y_vehicle = DEVICE(vehicle)->pstruDevicePosition->Y;
NETSIM_ID nApplicationId= pstruEventDetails->nApplicationId; //Unique id of application
int c;
if (y_vehicle <(y_rsu + 210) && y_vehicle>(y_rsu - 210))
{
c = 2;

          }
else if (y_vehicle > (y_rsu + 210) || y_vehicle < (y_rsu -210))
  c = 1;
if ((count_message == 2) && (c==1))
   count_message = 1;

  char temp[BUFSIZ] = "no change in traffic light";

  char temp1[BUFSIZ] = "change traffic light";

     if (c == 1)
          {
   for (int i = 0; i <strlen(temp); i++)
          message[i] = temp[i];
  }
  else
          {
   if (count_message == 1)
  {
   for (int i = 0; i < strlen(temp); i++)
          {
           message[i] = '\0';
           }
  for (int i = 0; i < strlen(temp1); i++)
          message[i] = temp1[i];
          count_message++;
         fprintf(stderr, "\n Message to Sumo from NetSim: %s \n", message);
}
          else
          {
          for (int i = 0; i < strlen(temp1); i++)
          message[i] = '\0';
          for (int i = 0; i < strlen(temp); i++)
          message[i] = temp[i];
          break;
}
double coordinates = corr(message);


process_saej2735_packet(pstruPacket);}
```

fn_NetSim_Application_Run() The code mentioned above is part of the this function. This function is called only when there is APP-IN event at the RSU when an RSU receives a message from the emergency vehicle. The location of the emergency vehicle in the NS lane is compared with the location of the RSU. The threshold value set for comparison takes time required to change the traffic signal from phase 2 to phase 3 into consideration. When the vehicles are near the RSU, the signal change message is sent from NetSim to sumo via the pipe that is created for the communication between NetSim and sumo process.

_declspec(dllexport) double *corr(char* id) This function sends the traffic signal message after comparison, to sumo via the pipe and returns null values. When there is no change in signal the function returns co-ordinates of the vehicles in the scenario.

Sumo API: getPhase() Returns the index of the current phase in the current program.

setPhase() Sets the phase of the traffic light to the given. The given index must be valid for the current program of the traffic light, this means it must be between 0 and the number of phases known to the current program of the tls - 1.

getIDList() Returns a list of ids of all vehicles currently running within the scenario

traci.vehicle.getPosition() Returns the position (two doubles) of the named vehicle (center of the front bumper) within the last step [m,m]; error value: [-2^30, -2^30].

Changes code to SumoRun.py

```
# Import Sumo Libraries
import traci, string
import traci.constants as tc

traci.init(PORT)

print(" Running Sumo Simulation...")

step = 0
#traci.trafficlight.setPhaseDuration("0", 50)
traci.trafficlight.setPhase("0", 2)
print("\n traffic phase 2 set")
while step < 100000:
garbage= "hello".encode() #Send Garbage
win32file.WriteFile(p1, garbage)
jk =win32file.ReadFile(p1, 4096)#Read vehicle from Netsim



vehicle_from_Netsim=jk[1].lower()[:-1]  #convert to lower case
vehicle_from_Netsim = vehicle_from_Netsim.decode("utf-8")


if ((vehicle_from_Netsim == "change traffic light")):
        print("\n message received from netsim to change traffic light")
        traci.trafficlight.setPhase("0", 2)



        if traci.trafficlight.getPhase("0") == 2:
                traci.trafficlight.setPhase("0", 3)
```

```
            print("\n trafic light is changing from phase 2 to phase 3")
            traci.simulationStep()
    else:
            traci.trafficlight.setPhase("0", 2)
            print("\n trafic light is changing from phase 3 to phase 2")
            traci.simulationStep()
    win32file.WriteFile(p1,
    'c'.encode())  # The vehicle was found, being sent to Netsim for Connection('c' for confirmation)
    position_x = str(0)
    position_y = str(0)
    win32file.WriteFile(p1, position_x.encode())
    win32file.WriteFile(p1, position_y.encode())
    step += 1
```

The traffic signal is initially in phase 2. This is done by using the setPhase() API. getPhase() is used to get the current traffic signal phase. getIDList() is used to get the vehicles in the scenario and traci.vehicle.getPosition() is used to get their instantaneous location in order to send the location of the vehicles to the NetSim using win32file.WriteFile().The win32file.ReadFile() is used to read the message sent from NetSim. When the signal has to be changed, the traffic signal is changed from phase 2 to phase 3 using the setPhase() API.