# Vehicle Speed and Acceleration Control in a Vanet Simulation (Four Legs Intersection Case Study)

**Albina TOCILLA**

*POLIS University*

**Tamara LUARASI**

*POLIS University*

**Abstract**

*Vehicular Ad-Hoc Networks (VANETs) are a specialized form of ad-hoc networks that enable communication between automobiles and also between vehicles and roadside infrastructure. The speed parameter is significant in multiple aspects of VANET design and applications and it denotes the rate at which vehicles move inside the network. The goal of this paper is to achieve different levels of the speed and study their influence on different parameters of vehicles' network especially in beacon message transmissions. Another aspect is how the speed and the load channel are related to other. In other words, the main aim consists on adapting the traffic simulation to the reality as close as possible, specifically to find how can we adapt the simulation speed to real vehicle speed. An important element in this effort was the control on the speed and acceleration of the vehicles. Besides this, we highlighted the influence of the speed on the communication between vehicles in a VANET simulation, considering specifically the BSM messages and beacon interval. According to the result of these simulations, there is a relation between the speed and the network features, and a relation between the combination of speed and channel load to network properties. The goal of the simulations after all, was to envision how this network would be implemented in real situations. These vehicle networks have a promising future*

**Keywords**

Speed control, OMNET++, SUMO, TraCi, Traffic simulation, Veins

## INTRODUCTION

In recent years, the advancements in wireless communications have created new areas of research in computer networking. These areas focus on expanding the connectivity of data networks to settings where wired solutions are not feasible. Of all these factors, vehicular traffic is receiving increasing attention from both academia and industry. This is due to the significant number and relevance of the applications associated with it, which range from ensuring road safety to managing traffic and even providing mobile entertainment. The speed parameter is an essential one while creating communication protocols, routing algorithms and applications in VANETs to ensure reliable and timely communication among the vehicles.

The vehicle speed and acceleration control are managed from two sides: from SUMO (Simulation of Urban Mobility) side and OMNET++ &Veins side.

OMNET++ is an extensible , modular, component-based C++ simulation library and framework, which make possible the building of network simulators. These networks, called VANET, are networks between vehicles and units on the street and between vehicles themselves. These networks make possible the communication among vehicles and among vehicles and outside infrastructure, providing in this way a big help to drivers to regulate the traffic and avoid accidents.

Veins is a framework that joins the OMNET++ network simulator and SUMO. It provides an interface between these two components which allows us to customize different elements of the simulated traffic and the communication.

The vehicle speed and acceleration control are managed from two sides: From the SUMO side and OMNET++&Veins side. In Sumo there are a wide range of influences on vehicle speed. Each of these influences sets an upper bound on the vehicle speed (dlr.de, n.d). Urquiza-Aguiar et al., (2019) analyzed different alternatives that we can use in SUMO to generate a traffic. First, the model chosen in Sumo has its own influence and interpretation of different parameters used in different commands. By default, within SUMO, the microscopic model developed by Stefan Krauß (Lopez et al., 2018) is used and defines the car speed in relation to the vehicle ahead.

The research question of this paper is how does the speed influence on other VANET parameters, specifically channel load, BSM messages and beacon interval. The main aim consists on achieving different levels of speed and evaluate this relation.

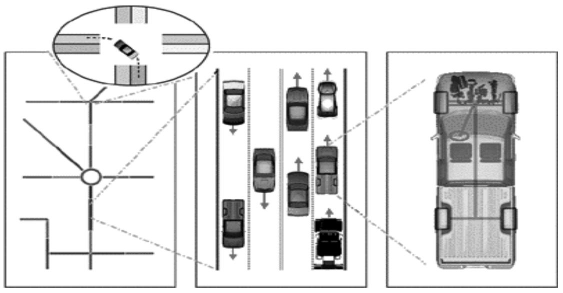


**Figure 1.** The different simulation granularities; from left to right: macroscopic, microscopic, sub-microscopic (within the circle: mesoscopic) (dlr.de, nd)

## LITERATURE REVIEW

There are a lot of papers and materials whose subject is the simulation of the vehicle traffic, and a variation of analysis related to this traffic. Regarding the main purpose of this paper, the studies about the relation among BSM, the beacon interval, the channel load and their influence on speed, are considered.

Bouk et al., (2015) proposed an examination of the latest hybrid adaptive beaconing techniques suggested for VANETs. Their paper provided a detailed discussion on the parameters that these systems optimize, which include beacon power, rate, and/or CW. It also explained the operating principles of these schemes. Ultimately, the assessment and simulation parameters are comprehensively presented to facilitate a clear understanding of the differences among all the schemes discussed in the literature. Furthermore, they offered a compilation of current obstacles and potential areas for further exploration. Their intention was to inspire deeper investigation into the limitations of beaconing in VANETs.

Furthermore, Sathyapriya et al (2020) studied the regulation of vehicles' speed base on the speed limit of a certain area. A single transmission of data from a central control site is sufficient to initiate vehicle-to-vehicle communication. A vehicle initially receiving the signal transmits it to the adjacent vehicles, creating a continuous chain of transmission. VANET mitigates the drawback of signal loss caused by the velocity of vehicles. This technique has the capability to effortlessly regulate the speed of the vehicle in an automated manner. Vehicle-to-vehicle (V2V) transmission is more efficient compared to previous technologies such as GPS transceiver systems, which require transceivers to be installed at regular intervals along the route. This signal may experience signal degradation if the vehicle is traveling at a high velocity. VANET effectively mitigates these limitations.

Amour & Jaekel, (2023) proposed a decentralized congestion control algorithm where each factor adjusts the data rate (bitrate) used to transmit its wireless packet congestion based on the current load on the channel.

In Tomar et al (2022) one aspect of the work analyses the impact of beaconing in the network.

Considering these researches and many other ones related to VANET simulation, this paper aims to contribute to the studies related to the interplay of speed, channel load and other VANET parameters.

## METHODS

This work is a study about how to control the vehicle speed and acceleration of traffic and about the relations between the speed and some VANET parameters, specifically channel load, BSM messages, and other network parameters.

The methodology was based on traffic simulation and the establishment of a network among the vehicles involved in this traffic, specifically VANET. Many experiments are done, changing different parameters, until some conclusions were presented.

Software SUMO is used to generate the vehicle traffic. Two files are needed as inputs by SUMO to generate the traffic: one that represents the road network of the zone that we study, and a file that describes the vehicles, their characteristics, and the routes they take.

The real information that we have is the number of cars during 24 hours per one week, collected by cameras in some intersections of Tirana.

Then the first input, that is, the road network, is imported from the map around one of these intersections – specifically, the Don Bosco intersection, by OpenStreetMap. Then, this file is converted into a format accepted by SUMO and is called file.net.xml.

For the second input - the file that describe vehicles, their characteristics, and the routes that they take, there are different approaches in SUMO software to generate the vehicle traffic depending on traffic demand generation tools (Luis F. Urquiza-Aguiar, Pablo Barbecho Bautista, William Coloma Gómez, Xavier Calderón, Comparison of Traffic Demand Generation Tools in SUMO, Case Study).

We argue that the combination of duarouter (one of traffic demand generation tools) and randomTrips.py command is the best way to judge and analyze the real vehicular traffic. This choice is dictated by the limited information that we have.

As a result we get the file named file.rou.xml.

Providing two files file.net.xml (road network generated from maps imported from OpenStreetMap) and file.rou.xml (generated routes), as well as a configuration named file.sumocfg, the vehicular traffic is generated independently by SUMO.

The traffic generated by SUMO is integrated into OMNET++. On OMNET++ side, we use programming to test the vehicle network features, especially the BSM messages in different situations of the speeds, by using different speed modes. We also combine the speed mode parameter with different channel load in message transmission.

Therefore, the set of software SUMO&OMNET++&Veins were used in this study. The simulated traffic by SUMO was integrated into OMNET++, where a VANET network was simulated. It is Veins that provides the join of SUMO and OMNET++.

## The work done

As we mentioned before the first phase of the work is the generation of the vehicle traffic and SUMO software was chosen to do it. There are some preparatory steps to prepare the information that SUMO will use to generate the traffic. Each of these steps has some input and output that are used by the following one. The following alternative of a sequence of commands was applied:

The command openmapstreet creates the file file.osm generated from maps

The command netconvert creates the file file.net.xml – routes network

The command randomTrips.py creates the file file.trips.xml –contains a set of random trips for the network

The command duarouter generates the file file.rou.xml – contains vehicle, their characteristic and the routes that they take

There is the command randomTrips.py where we have some parameters that influence the vehicle speed: maxSpeed, tau, minGap. Considering the default values of these parameters the command has the view:

```
…/randomTrips.py --vehicle-class passenger -n "file.net.xml" -b 0 -e 3600 -p 2.6 --route-file "file.trips.xml"--trip-attributes="length=\"5\" accel=\"2.6\" decel=\"4.5\" sigma=\"0.5\"
tau=\"1\" minGap=\"2.5\" maxSpeed=\"55.55\" emergencyDecel=\"9\"
carFollowModel=\"Krauss\"" –validate
```

Some of the parameters can be interpreted differently in different models. Considering the default model, which is represented by carFollowModel=\"Krauss\"", the following parameters are almost all the parameters of this model and their meaning is:

vehicle-class defines the type of vehicles, in this case we have considered passenger vehicles

-n is used to specify the network in this case file.net.xml

-e specifies the end time which is set to 3600 sec

-p represents the arrival rate. The arrival rate is calculated by the formula: (t2-t1)/ n

o specifies where resulting trips are stored

To define the parameter -p we have used the information taken during a week, 24 hours per day, but we have considered the time interval 7:00-to 20:00, when the number of cars has no big differences, as it is shown in the following chart. The calculation of the formula (t2-t1)/ n gives us the value 2.6.

-trip-attributes: by the Specification - SUMO Documentation,

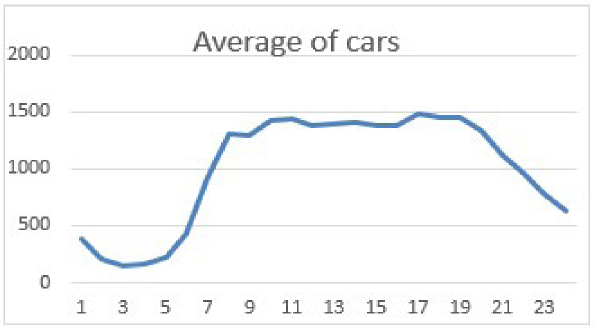
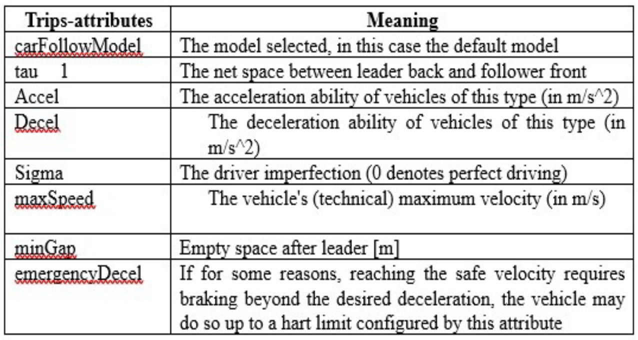


**Figure 2.** The average of vehicles' number in 24 hour, we have considered the time interval 7:00-to 20:00, when the number of cars has no big differences,



| Trips-attributes | Meaning |
|---|---|
| carFollowModel | The model selected, in this case the default model |
| tau 1 | The net space between leader back and follower front |
| Accel | The acceleration ability of vehicles of this type (in m/s^2) |
| Decel | The deceleration ability of vehicles of this type (in m/s^2) |
| Sigma | The driver imperfection (0 denotes perfect driving) |
| maxSpeed | The vehicle's (technical) maximum velocity (in m/s) |
| minGap | Empty space after leader [m] |
| emergencyDecel | If for some reasons, reaching the safe velocity requires braking beyond the desired deceleration, the vehicle may do so up to a hart limit configured by this attribute |

n.d. the meaning of the trip attributes are presented in the following table

the --trip-attributes parameter generates the following information in resulting file of the command roundTrips.py, which is file.trips.xml.

```
<vType id="passenger" length="5.00" minGap="2.50" maxSpeed="55.55" vClass="passenger" carFollowModel="Krauss" accel="2.6" decel="4.5" emergencyDecel="9" sigma="0.5" tau="1"/>
```

After that, the command duarouter is used,

```
duarouter file.net.xml --route-files file.trips.xml -o file.rou.xml,
```

where the file file.rou.xm is generated, and we can simulate a traffic in SUMO with the command:

```
sumo-gui file.sumocfg,
```

where file.sumocfg is the configuration file:

```
<configuration>
<input>
<net-file value="file.net.xml" />
<route-files value="file.rou.xml" />
</input>
. . .
<output>
. .
<tripinfo-output
      value="/. . ./tripinfos.xml"/>
<tripinfo-output.write-unfinished value="true"/>
</output>
    . . .
</configuration>
```

If we use the OUTPUT tag in the configuration file, we can get an output file, in this case, tripinfos.xml with information, which gives us the following results:

All these parameters, which are trip-attributes, influence the vehicle speed and acceleration, as we can see it in the maximum speed, because of the parameter maxSpeed=\"55.55\".

| Defaut parameters | *Speed Average* | *Speed Max* | *Waiting Time* | *Time Loss* |
|---|---|---|---|---|
| | 18.4km/hour | 42,3km/hour | 21.9 | 28.8 |

**Table 1.** Some statistics generated by SUMO

We can improve the vehicle behavior using the attribute “speedFactor,” which makes the vehicles drive with that factor of the current speed limit. Dhe default value of this parameter is 1.

Having a distribution of speed factors (and hence of desired speeds) is beneficial to the realism of a simulation (Specification - SUMO Documentation, n.d.). The truncated normal distribution[1] is chosen for this parameter, which can be represented by:

speedFactor="normc (mean, deviation, lowerCutOff, upperCutOff)"

and if we use it in the shorter way, speedFactor="normc(mean,deviation)", the two last parameters have the values [0.2, 2]

Adding this parameter, the command randomtrips.py has now the form:

```
…/randomTrips.py --vehicle-class passenger -n "file.net.xml" -b 0 -e 3600 -p 2.6 --route-file "file.trips.xml" --trip-attributes="length=\"5\" accel=\"2.6\" decel=\"4.5\" sigma=\"0.5\" tau=\"1\"minGap=\"2.5\"maxSpeed=\"55.55\" speedFactor="normc(1.2,0.3)" emergencyDecel=\"9\" carFollowModel=\"Krauss\"" –validate
```

The values 1.2 and 0.3 imply a speed between 80% and 120% of the legal speed limit.

The following table shows an important improvement related to speed and other traffic indicators.

## Integration of SUMO simulation in OMNET++

| Using speedFactor distribution | *Speed Average* | *Speed Max* | *Waiting Time* | *Time Loss* |
|---|---|---|---|---|
| | 31.5km/ore | 76.9km/ore | 0.19 | 0.86 |

**Table 2.** Some statistics generated by SUMO considering speedFactor parameter

The vehicle traffic generated by SUMO can be integrated and controlled from OMNET++ side.

TraCI ("Traffic Control Interface") is an interface between SUMO and OMNET++, where SUMO plays the role of a server and the client is on OMNET++ side.

Some steps, then, are necessary on the client side.

As a first step, we create a project in OMNET++, and we include the Veins project on it, which is an open-source framework and provides a C++ client library for the TraCI API.

OMNET++, as an object-oriented modular discrete event network simulation framework, provides infrastructure and tools for the VANET project, and this one, using the OMNET++ modules and its own C++ library, makes possible the use of the running traffic by TraCi on the client side, that is, from OMNET++ side.

Some files need to be added in the project created.

File file.NED: as we have already mentioned, the role of OMNET++ is the creation of a VANET network between road side units and vehicles and between vehicles themselves. For this purpose, a topology description language is used in a file called file.NED, with a minimum content as follows:

```
network networkName extends Scenario{
   submodules:
rsu[1]: RSU {
    @display("p=150,140;i=veins/sign/yellowdiamond;is=vs");
      }
}
```

where a road unit side is added to the content of inherited .ned file, Scenario (part of the veins project).

Files file.net.xml, file.rou.xml, file.sumocfg, used in SUMO, have to also be included in the project.

File file.launchd.xml, with the following content is also created in project

```
<?xml version="1.0"?>
<!-- debug config -->
<launch>
            <copy file="file.net.xml" />
            <copy file="file.rou.xml" />
            <copy file="file.sumo.cfg" type="config" />
</launch>
```

and the name of it is represented in the configuration file OMNETpp.ini.

File OMNETpp.ini: to run the simulation, we need to create an OMNETpp.ini file. The file tells the simulation program which network to simulate and allows you to assign values to different parameters declared in the .ned files (your file or the inherited ones), and it explicitly specifies seeds for the random number generators (Team, n.d.). The initial content of this file can be copied from the file with the same name in the folder . . ./home/veins/src/modules/application/traci and modified. Many other parameters are included here, which can be grouped by some categories, and some of them are:

```
            [General]
            Simulation parameters
            Obstacle parameter (some updates here)
            TraCIScenarioManager parameters
            RSU Settings
            11p specific parameters (some updates here)
            App Layer (some updates here)
            Mobility
```

Files VehicleControlApp.h, VehicleControlApp.cc. These two files will replace the respective default files TraCIDemo11p, which are used by default in OMNETpp.ini. To take into consideration our code in simulation, we add the following lines on file.ned file with the following content,

```
simple VehicleControlApp extends DemoBaseApplLayer
{
   parameters:
   @class(veins:: VehicleControlApp);
   double allowedSpeed=default(55.55);
   string appName = default("My first Veins App!");
}
```

Some customizations are done in OMNETpp.ini file like:

```
            [General]
                        . . .
network = networkName
            #Obstacle parameter(some updates here)
#*.obstacles.obstacles = xmldoc("config.xml", "//AnalogueModel[@type='SimpleObstacleShadowing']/obstacles") (commented)
            #11p specific parameters (some updates here)
*.**.nic.mac1609_4.useServiceChannel = true
            #App Layer (some updates here)
*.node[*].applType = "VehicleControlApp "
. . .
*.node[*].appl.sendBeacons = true
*.node[*].appl.dataOnSch = true
*.node[*].appl.beaconInterval = 1s
```

TraCIDemo11p is an extension of the module DemoBaseApplLayer and this one inherits the class BaseApplLayer.

Some modifications are done in the module VehicleControlApp. There are some event based methods that can be customized as:

```
            void initialize(int stage)
            void handleSelfMsg(cMessage* msg)
            void onWSM(BaseFrame1609_4* wsm){};
            void onBSM(DemoSafetyMessage* bsm){};
            void onWSA(DemoServiceAdvertisment* wsa){};
            void handlePositionUpdate(Object* obj);
```

These methods are event methods, and they will run based on the events that happen in one moment. As we see, some of these methods accept as information messages of different types, and based on the information that they receive, they react in a specific way.

There are different types of messages that can be sent or received, and some are managed by OMNET++ classes, like cMessage, and others by Veins classes like DemoSafetyMessage, DemoServiceAdvertisment.

In this work we are interested on the basic safety messages (type DemoSafetyMessage), or beacon messages, because these are periodic messages, and they broadcast regular information like the position speed, status of vehicle, address, location, speed direction, etc., but we could allow other messages too. Other messages can be:

DemoServiceAdvertisment– is the type of wave service announcement (WSA) messages,

DemoServiceAdvertisment– is the type of wave service announcement (WSA) messages, and

BaseFrame1609_4–is the type of wave service message (WSM).

Event-driven messages are sent whenever certain events, such as traffic accidents or road hazards, are detected, while BSMs are sent at regular intervals by each vehicle in the network, regardless of the road conditions. This means that, as the vehicle density increases, the total number of BSMs being transmitted within the network increases proportionally (Bouk et al., 2015).

Because the beacon messages are the most frequently sent messages from one vehicle to the others and to the vehicle itself, and the function that is activated associating this event is on BMS (DemoSafetyMessage msg), we can write here some code that would cause changes on simulated traffic behavior.

The logic used in this function is the same as in Park (n.d), but we have changed some parameters. This code forces to adapt the vehicle speed to the real speed.

The pseudo code is:

```
onBSM: (DemoSafetyMessage msg)
  get the sender speed from msg object;
  get the position of preceding vehicle from msg object
```

```
 set SpeedMode to traciVehicle object
 define e desired distance between two vehicles
 define e coefficient as beaconInterval;
 find the distance between preceding vehicle and the current position
   find the acceleration = (distance – desired distance)/ coefficient^2
 if ( distance - desiredDistance> 1)
   set SpeedMode to traciVehicle object
   set acceleration to traciVehicle
   apply slowDown function with parameter allowedMaxSpeed
                  (defined in .ned file)
Otherwise   if ( distance - desiredDistance< -1)
set SpeedMode to traciVehicle object
setEmergencyDecel(-acceleration *5)
apply slowDown function with parameter allowedMinSpeed []
```

Some explanation here:

• SpeedMode is a TRACI command. As we have already mentioned, TRACL gives access to a running road traffic simulation generated by SUMO and can interfere in it due to a TCP based client/server architecture.

For simulation purposes, three types of parameters for speed mode command are considered here. This command retrieves the values set by TRACL commands speed (0x40) and slowdown (0x14) and regulates the behavior of the car. By default, the vehicle is restricted to driving at a speed lower than what is considered safe according to the following car's model. Additionally, it must not overcome the limits on acceleration and deceleration. In addition, the vehicles follow the right-of-way regulations when approaching an intersection and, if required, they brake hard to prevent crossing a red traffic signal. To regulate this behavior, one can utilize the speed mode command. The parameter is a bitset, where bit 0 represents the least significant bit. It contains the following fields (A., 2018):

bit0: Regard safe speed
bit1: Regard maximum acceleration
bit2: Regard maximum deceleration
bit3: Regard right of way at intersections
bit4: Brake hard to avoid passing a red light

Three different parameter values are used for the SpeedMode in our experiments.

The first is parameter value 0x1f or bitset 11111, for the speed mode, which means that all the bits with respective meaning are considered : bit0(Regard safe speed), bit1(Regard maximum acceleration), bit2(Regard maximum deceleration), bit3( Regard right of way at intersections), bit4(Brake hard to avoid passing a red light)

The second is the parameter value 0x07 for the speed Mode, which means that bit set 0011is considered. And the third parameter value is 0x06 or the bit set 000110.

To calculate the acceleration, we have used the formula.

Acceleration= (distance – desired distance)/ coefficient^2

where coefficient = beaconInterval

SUMO runs externally as a dedicated service and is not "built in" while compiling.

We have to launch sumo in parallel, so that it can wait for incoming connections on the port specified in the behavior of our application (generally 9999). Therefore, we need to start the TraCI server first by the command:

```
python /home/veins/src/veins/sump-launch.py -vv -c sumo-gui
```

## Results

The tests are done with the following simulation parameters:

Beacon interval 1s
Fixed transmission power 20 mW
BSM size 256
Minimum power level −110 dBm
Noise floor −98 dBm
Vehicle number 60

The results represented in the file.vec, which is the output in OMNET++ in result folder, are presented in the table below:

The graphic below shows vehicle speed-modes influence to different network indicators. The incrementing of the speed is associated with the decrementing of the other network properties.

| *Speed Mode* | *Speed Mean* | *Speed StdDev* | *Generated BSM* | *Received BSM* | *Received Broadcast* | *Sent Packets* | *Total Lost Packets* |
|---|---|---|---|---|---|---|---|
| **0x1f** | 28.8 | 7.9 | 3 | 51 | 143 | 8 | 64 |
| **0x07** | 29.8 | 4.02 | 6 | 50 | 126.6 | 7 | 69 |
| **0x06** | 118.8 | 15.3 | 3 | 11 | 11 | 3 | 19 |

**Table 3.** Some statistics generated by OMNET++ using different speed mode

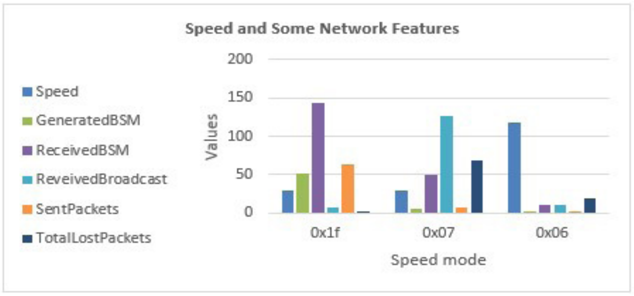


**Figure 3.** The influence of speed mode in other network parameters when beacon interval is 1 sec.

We can see that the incrementation of the speed is caused because in the 0x06 mode there are fewer limitations dictated by different bit fields of speed mode, while, the Received BSM and Received Broadcast are higher in the 0x1f speed mode.

Continuing the experiment with different channel load we have the following results. The load of the channel, discussed on Amour, B., & Jaekel, A. is considered:

Low channel load:

Beacon interval 0.1s
BSM size 256

Medium channel Load
Medium channel Load

| *Speed Mode* | *Speed Mean* | *Speed StdDev* | *Generated BSM* | *Received BSM* | *Received Broadcast* | *Sent Packets* | *Total LostPackets* |
|---|---|---|---|---|---|---|---|
| **0x1f** | 31.1 | 3.9 | 35 | 484 | 1278 | 74 | 154 |
| **0x07** | 31.3 | 3.9 | 15 | 458 | 1228 | 51 | 92 |
| **0x06** | 190.4 | 1.8 | 29 | 80 | 106 | 37 | 55 |

**Table 4.** The statistics for low load channel

| Speed Mode | Speed Mean | Speed StdDev | Generated BSM | Received BSM | Received Broadcast | Sent Packets | Total Lost Packets |
|---|---|---|---|---|---|---|---|
| **0x1f** | 31.1 | 3.9 | 160 | 2446 | 3242 | 200 | 111 |
| **0x07** | 31.1 | 3.8 | 152 | 2275 | 3061 | 192 | 124 |
| **0x06** | 190.4 | 53.8 | 36 | 59 | 398 | 483 | 158 |

**Table 5.** The statistics for medium load channel

Beacon interval 0.02s
BSM size 256
High channel Load:
- Beacon interval 0.05s
- BSM size 1024

The three tables in Fig. 4,5,6 show that the channel load does not have an influence on the speed average, but the standard deviation for the speed is better in the low channel load with the combination Bacon Interval = 0.1s and BSM size=256.

| Speed Mode | Speed Mean | Speed StdDev | Generated BSM | Received BSM | Received Broadcast | Sent Packets | Total LostPackets |
|---|---|---|---|---|---|---|---|
| **0x1f** | 31.1 | 3.9 | 64 | 969 | 1780 | 104 | 87 |
| **0x07** | 31.4 | 3.8 | 62 | 912 | 1694 | 101 | 120 |
| **0x06** | 190.4 | 55.5 | 58 | 162 | 187 | 66 | 81 |

**Table 6.** The statistics for high load channel

Regarding the network communication, that is, the BSM messages and Broadcast, the channel load is important.
If we compare the network data for different channel mode for each of speed modes, we have the following evidences:
In the 0x1f speed mode for the medium load channel we have the highest number of beacon messages, broadcasts and packets. Also in 0x07 speed mode for the medium load channel, we have the highest number of beacon messages, broadcasts and packets. In the 0x06 speed mode, the high load channel provides the highest number of beacon messages, broadcasts.

| Channel load **0x1f speed mode** | Generated BSM | Received BSM | Received Broadcast | Sent Packets | Total LostPackets |
|---|---|---|---|---|---|
| **Low load** | 35 | 484 | 1278 | 74 | 154 |
| **Medium Load** | 160 | 2446 | 3242 | 200 | 111 |
| **High load** | 64 | 969 | 1780 | 104 | 87 |

**Table 7.** Statistics for 0x1f speed mode

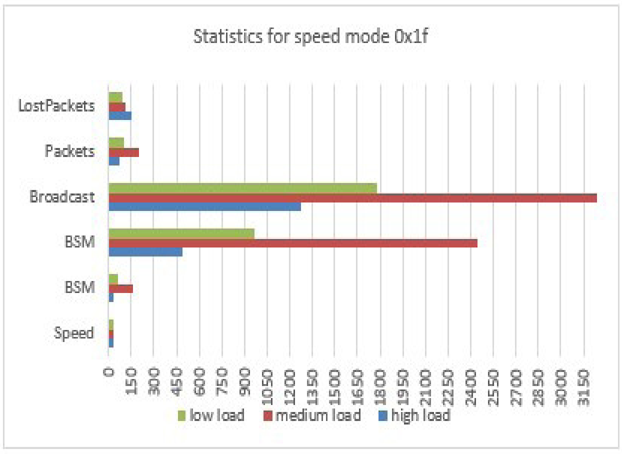


**Figure 4.** The graph for 0x1f speed mode,

| Channel load **0x07 speed mode** | Generated BSM | Received BSM | Received Broadcast | Sent Packets | Total LostPackets |
|---|---|---|---|---|---|
| **Low load** | 15 | 458 | 1228 | 51 | 92 |
| **Medium Load** | 152 | 2275 | 3061 | 192 | 124 |
| **High load** | 62 | 912 | 1694 | 101 | 120 |

**Table 8.** In 0x07 speed mode

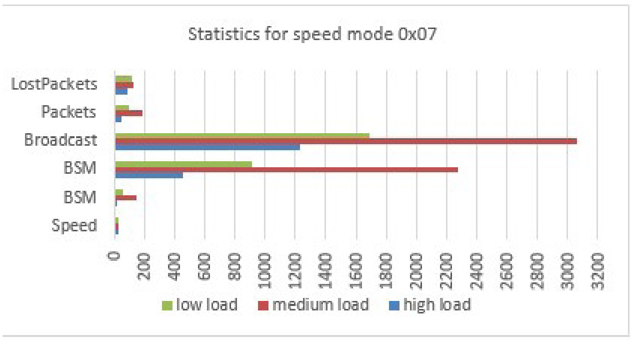


**Figure 5.** The graph for 0x07 speed mode

| Channel load **0x06 speed mode** | Generated BSM | Received BSM | Received Broadcast | Sent Packets | Total LostPackets |
|---|---|---|---|---|---|
| **Low load** | 29 | 80 | 106 | 37 | 55 |
| **Medium Load** | 36 | 59 | 398 | 483 | 158 |
| **High load** | 58 | 162 | 187 | 66 | 81 |

**Table 9.** In 0X06 speed mode

## Conclusion

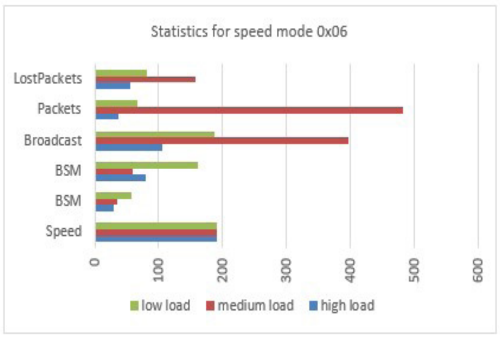


**Figure 6.** The graph for 0x06 speed mode

Monitoring car traffic today is possible through various technologies that enable the extraction of some important statistics regarding real-time traffic condition and its problematic points. However, the studies on traffic are necessary to make predictions to prevent difficult traffic situations, to establish control strategy regarding the direction of vehicles, and to find suitable topology of key points of the road network. These studies are practically possible only under traffic simulation conditions.

This paper is presented in the form of tutorials theat use some of the software such as SUMO, OMNET++, and Veins to carry out a simulation. In addition to the simulating traffic by SUMO, OMNET++ has established a network between vehicles, through which communication between them is carried out. The data comes as a result of traffic monitoring from the Municipality of Tirana at a point in Tirana, Don Bosko, where monitoring through cameras is accomplished

In our paper the goal is to achieve different levels of the speed, and how it influences different parameters of the vehicle network, especially in beacon message transmissions. Another aspect is how the speed mode is related to the load channel.

The tables and charts represent different levels of the speed, depending on command parameters used and the load channel and BSM messages related to different levels of the speed mode.

## REFERENCES

Luis F. Urquiza-Aguiar, Pablo Barbecho Bautista, William Coloma Gómez, Xavier Calderón

Comparison of Traffic Demand Generation Tools in SUMO, Case Study: Access Highways to Quito

PE-WASUN'19, November 25-29, 2019, Miami Beach, FL, USA

Dep. of Network Engineering. Universitat Politècnica de Catalunya (UPC) Barcelona, Spain pablo.barbecho@upc.edu Xavier Calderón Dept. Electrónica, Telecomunicaciones y Redes de Información, Escuela Politécnica Nacional Quito, Ecuador xavier.calderon@epn.edu.ec

Bouk, S. H., Kim, G., Ahmed, S. H., & Kim, D. (2015, May 1). Hybrid Adaptive Beaconing in Vehicular Ad Hoc Networks: A Survey. International Journal of Distributed Sensor Networks, 11(5), 390360. https://doi.org/10.1155/2015/390360Team, O. (n.d.).

Lopez, P. A., Wiessner, E., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y. P., Hilbrich, R., Lucken, L., Rummel, J., & Wagner, P. (2018, November). Microscopic Traffic Simulation using SUMO. 2018 21st International Conference on Intelligent Transportation Systems (ITSC). https://doi.org/10.1109/itsc.2018.8569938

Sathyapriya, A., Sathiya, K., Sneha, T. M., Rohit Raja, D., & Manikandan, T. (2020, August 30). Automatic Speed Control System in Vehicles Using VANET. Advances in Intelligent Systems and Computing, 719–726. https://doi.org/10.1007/978-981-15-5029-4_60

St. Amour, B., & Jaekel, A. (2023, September 7). Data Rate Selection Strategies for Periodic Transmission of Safety Messages in VANET. Electronics, 12(18), 3790. https://doi.org/10.3390/electronics12183790

Sangyoung Park Module 'Vehicle-2-X: Communication and Control'

https://cse.iitkgp.ac.in/~soumya/micro/t2-4.pdf

Tomar Ravi, Sastry G. Hanumat, and Prateek Manish, "Establishing Parameters for Comparative Analysis of V2V Communication in VANET," Journal of Scientific & Industrial Research, vol. 79, no. 1, pp. 26-29, 2022

Urquiza-Aguiar, L. F., Coloma Gómez, W., Barbecho Bautista, P., & Calderón, X. (2019, November 25). Comparison of Traffic Demand Generation Tools in SUMO. Proceedings of the 16th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks. https://doi.org/10.1145/3345860.3361521

A. (2018, October 5). TraCI SpeedMode. AVCOURT's Traffic Simulation Blog. https://avcourt.github.io/comp4560-blog/2018/10/speed_mode.html

Definition of Vehicles, Vehicle Types, and Routes - SUMO Documentation (dlr.de) https://sumo.dlr.de/docs/Definition_of_Vehicles%2C_Vehicle_Types%2C_and_Routes.html

Vehicle Type Parameter Defaults - SUMO Documentation (dlr.de)

https://sumo.dlr.de/docs/Vehicle_Type_Parameter_Defaults.html

Change Vehicle State - SUMO Documentation (dlr.de)
https://sumo.dlr.de/docs/TraCI/Change_Vehicle_State.html
VehicleSpeed - SUMO Documentation (dlr.de)
https://sumo.dlr.de/docs/Simulation/VehicleSpeed.html
Specification - SUMO Documentation (dlr.de). https://sumo.dlr.de/docs/Specification/index.html