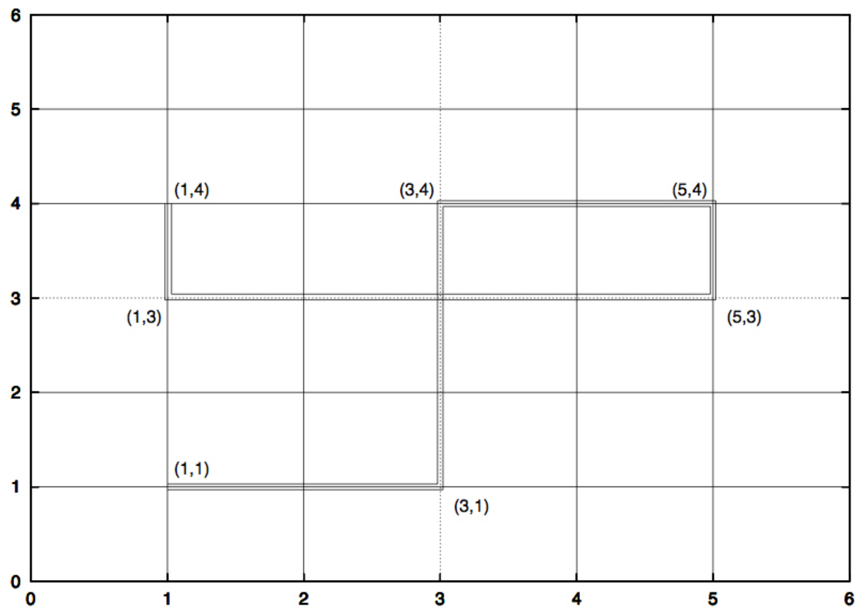


Figure 3.8: City Section Mobility Model. Traveling pattern of a MN using the CSMM.

Figure 3.8 show the movements of a MN using an example city section according to the CSMM. Within this example, the center-most vertical and horizontal streets are designated as mid-speed roads (i.e., $x = 3$ and $y = 5$), similar to main thoroughfares within a city; all other roads are considered to be slow residential roads. A MN starts the simulation at $(1, 1)$, moves to $(5, 4)$ and then moves to $(1, 4)$. The dashed lines in fig. 3.8 indicate the mid-speed roads and the double lines represent streets traveled by the MN in our example. As shown, both moves from $(1, 1)$ to $(5, 4)$ and from $(5, 4)$ to $(1, 4)$ use mid-speed roads.

The CSMM provides realistic movements for a section of a city since it severly restricts the traveling behavior of MNs. In other words, all MNs must follow predefind paths and behavior guidelines (e.g. traffic laws). In the real world, MNs do not have the ability to roam freely without regard to obstacles and traffic regulations. In addition, people typically tend to travel in similar patterns when driving across town or walking across campus. Enforcing that all MNs follow predefined paths will increase the average hop count in the simulations compared to other mobility models. Improvements of the CSMM include pause times at certain intersections and destinations, incorporate acceleration and deceleration, and account for higher/lower concentrations of MNs depending on the time of day. In addition, the model should be expanded to include a larger simulation area, an increased number of streets, a high-speed road along the border of the simulation area, and other novel path-finding algorithms.

3.2.5 Rice University Model (RUM)

RUM(84) is very similar to CSM but, on the contrary, it uses real maps obtained from the TIGER/Lines database(15). For each route segment 1 , the coordinates are extracted and converted using the Mercator projection(84). The extracted points are then presented by a graph, where the crossroads are presented by vertices, and routes by weighted arcs. The weight of each arc is dynamically calculated in such a way to mimic the estimated time required for a vehicle to move over the corresponding segment, which is proportional to its maximum authorized speed, its distance, and the number of vehicles it currently contains. Therefore, the lower the weight, more the vehicles move freely in the segment. Note that the maximum authorized speed of a route segment depends on its type. RUM has been compared to RWP, in the evaluation of DSR(44) in two different regions(84). The first one contains a lot of roads of the same type, with a maximum speed limit set to 56km/h, while the second one has fewer roads, but of different types (which ranges the maximum speed from 56km/h to 120km/h). The two models engender almost the same results in the first region simulation. However, in the second region simulation (where the mobility is more constrained) the results were clearly different. Moreover, like all the previous models, RUM denes no control mechanisms at crossroads.

3.2.6 Stop Sign Model (SSM)

SSM(77) is the first model that integrates a traffic control mechanism. In every crossroads, a stop signal is put, which obliges vehicles to slow down and make a pause there. This model is based on real maps of the TIGER/Lines database, but all roads are assigned a single lane in each direction. A vehicle should never overtake its successor (like in all the models presented before), and should tune its speed to keep the security distance. If many vehicles arrive at an intersection at the same time, they make a queue and then each one waits for its successor to traverse the crossroads. This results in gathering of nodes, and hugely affects the network connectivity as well as the vehicle mobility (average speeds). According to the authors(77), the problem with this model is the unrealistic disposition of the stop signals, since it is impossible to find a region with stop signals at each intersection. Therefore, they proposed TSM(77), described hereafter.

3.2.7 Traffic Sign Model (TSM)

In this model, stop signals are replaced by traffic lights. A vehicle stops at a crossroads when it encounters a red stoplight, otherwise it continues its movement. When the first vehicle reaches the intersection, the light is randomly turned red with probability p (thus turned green with probability $1 - p$). If it turns red it remains so for a random delay (pause-time), forcing the vehicle to stop as well as the ones behind it. After the delay, it turns green then the nodes traverse the crossroads one after the other until the queue is empty. When the next vehicle arrives at the crossroads the process is repeated. TSM and SSM has been evaluated by simulation with ns2(3). The results show that the two models are not signicantly influenced by the speed of nodes (maximum speeds). This is due to the traffic control models, which slow down the nodes and give more stability to the network(77). When increasing the pause-time at the intersections, the authors remarked that the performances improved for both models, and that SSM gives better results than TSM when using the same pause-time. The authors argue this by the fact that in SSM nodes always stop at the intersections, unlike TSM. Nevertheless, in reality the pause-time of stop signals is shorter than the one of traffic lights, which makes TSM more stable indeed(77).

3.2.8 STRAW

STRAW(21) is also a model relying on real maps of TIGER/line. Like the other models (except freeway), the roads include one lane in each direction, and is divided into segments. The model is basically composed of three modules: intra-segment mobility manager, inter-segment mobility manager, and finally the route management and execution module. At the beginning of the simulation, the nodes are placed randomly one behind the other. Then they move using the car following model(21) such that they try to accelerate until reaching the maximum speed of the segment. The first module manages this movement until reaching an intersection. The security distance is maintained, and the overtaking is not allowed. At crossroads the vehicles always slow down, even when they change a segment and turn without a full stop, which is realistic. The second module defines the traffic control mechanism including both stop signals and traffic lights, put on crossroads according to the class of the intersected routes. In addition to this usual control form, the module makes sure that the next segment to take contains enough available space before moving the vehicle toward it. If it is fully busy, the vehicle waits at the crossroads (at the end of the first segment). The last module selects the routes to be taken by each vehicle during the simulation. It implements two approaches: simple straw and straw OD. In the first one, the direction is randomly selected at each intersection. That is, when reaching an intersection, the vehicle randomly decides whether to continue straightforward, or to turn and change the route. On the other hand, in the second approach a destination is selected toward which the vehicle moves using the shortest path. The simulation study made by the authors (21) shows that when using STRAW the reception ratio decreases from 43% up to 53% compared to movements in an open area. The results of this simulation also illustrate that the routes arrangement has an impact; scenarios with a high number of crossroads slow down the average speeds of nodes, which improves the reception ratio.

3.2.9 Gorgorin et al. model

In addition to all the realistic parameters of the previous models, this one(18) implements an overtaking mechanism within multi-lane segments. A vehicle always tries to move on the most right lane (the lowest rang), except in case of overtaking during which it moves left, and intersections in urban environment where it chooses the lane according to the next direction.
This model belongs to the *psycho-physical models*(87), which divide the bi-dimensional space of distance Δx and speed difference Δv with respect to front vehicle into several areas corresponding to different drivers behaviors. According to this, we can have the following four driving modes:

- *no reaction:* no influence is exerted from the ahead vehicle, which is too distant or traveling at higher speed than the considered vehicle. Under this circumstance, cars are free to reach and keep the desired speed;
- *reaction:* the vehicle is approching the car ahead and, as a consequence, it has to reduce its speed to keep at safety distance;

- *unconscious reaction:* the vehicle is in short distance with respect to the car in front, the speed difference is small and the two vehicles build up a kind of *cluster*;
- *deceleration:* in this case, the distance is too small if compared to the current speed difference, and the vehicle under consideration is forced to brake.

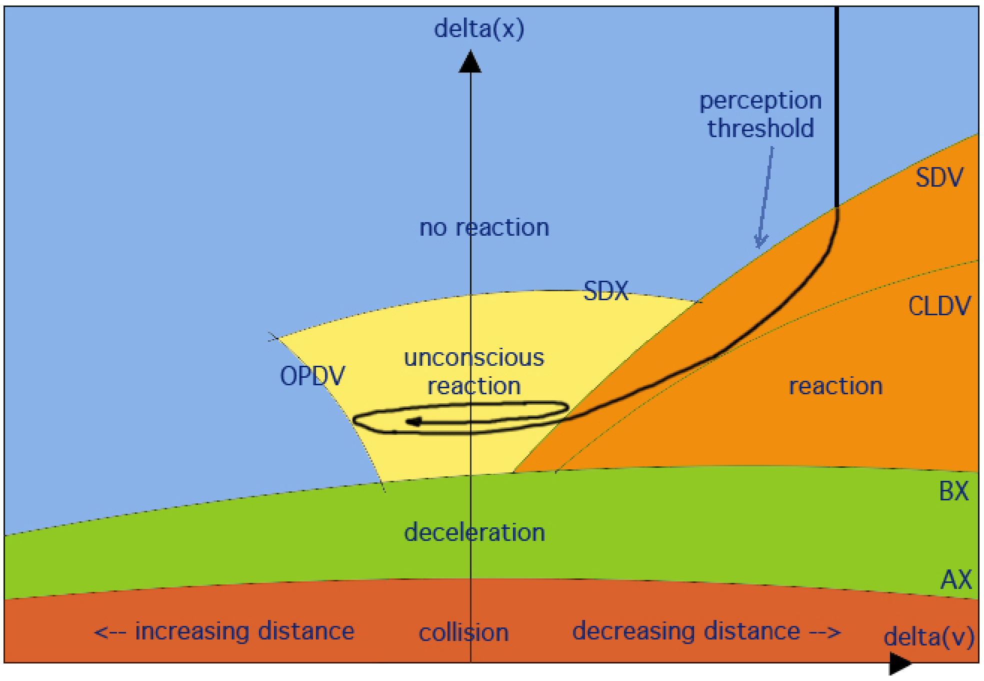


Figure 3.9: Psycho-physical vehicular interaction. The Δx axis divides positive (vehicle approaching front car) and negative (front vehicle increasing its distance) speed difference Δv. A possible trajectory is shown by the narrow: a vehicle, initially in free flow condition (i.e in the no reaction zone), starts approaching a vehicle in front. As long as the distance is high enough, the driver ignores the obstacles ahead, keeping his/her pace. However, when the distance becomes too small with respect to the current speed (i.e. the reaction zone is entered) the driver starts slowing down. The braking process brings the driver to match the front car speed ($\Delta v = 0$). Any increase in the speed if the ahead vehicle is then matched by the back vehicle with some delay, due to the finite reaction time of drivers ($\Delta v < 0$). For the same reason, any reduction in front car's speed is matched by the back vehicle with delay as well (the last segment of the movement curve, in the $\Delta v > 0$, unconscious area).

These behavioral areas are separated by thresholds, which trigger consistent acceleration changes. In the example of fig.3.9, AX and BX are the minimum and desired

safety distances, SDV and $CLDV$ are the distances at which the driver perceives he is approaching a slower vehicles and ends the approaching procedure, $OPDV$ and SDX model the distances beyond which the driver understands he is free to accelerate. All these distances are obviously functions of the speed difference. The main challange of this kind of mobility models lies in the characterization of such tresholds.
Finally, note that the model includes both traffic lights and stop signals at intersections. One of these two different control mechanisms is put at each intersections according to the types of the intersecting segments. The most important parameter added in this model is the overtaking mechanism. However, no study investigating this issue has been done yet.

3.2.10 Metrics for mobility models

In this chapter is often highlighted the importance of choosing the right mobility model for a simulation of a VANET protocol, because it can heavily affect the results [1]. Different mobility models lead to different mobility patterns but models themselves do not give clear images how mobility patterns are different each other, that is why some mobility metrics are required to describe these mobility patterns. In this scenario, the definition of precise performance metrics becomes essential but it is not trivial as it can seem.
The best contribute on this problem is given by "Important", a framework thought to systematically analyze the Impact of mobility on routing protocols for ad-hoc networks(25) (intuitively, the results can be easily extended to any other application deployed in a VANET). The basic idea was simulating and then analyzing three main routing protocols, largely used in mobile networks, such as DSR, DSDV and AODV, over different mobility patterns.
Very important are the so called *protocol independent metrics (or direct metrics)*, which are measurements with clear physics meaning used to extract the characteristics of mobility and the connectivity graph between the mobile nodes (i.e. host speed or relative

[1]i.e., in 2006 V. Naumov, R. Baumann and T. Gross compared AODV and GPSR in ns-2 with Random Waypoint model and real traces collected in the city of Zurich. The results were impressive: with RWP, both the protocols reported a delivery ratio of 99%, value that decreased with real traces up to 34% and 13% respectively(69)

speed); in addition, we can find the *derived mobility metrics* (such as graph connectivity), measurements derived from physical observations through mathematical modeling. Before formally defining the metrics, it is mandatory to introduce the following terminology:

1. $\vec{V_i}(t)$: velocity vector of node i at instant t;
2. $\left|\vec{V_i}(t)\right|$: speed of node i at instant t;
3. $\theta_i(t)$: angle made by $\vec{V_i}(t)$ at time t with the x-axis;
4. $\vec{a_i}(t)$: acceleration vector of node i at time t;
5. $x_i(t)$: x-coordinate of node i at instant t;
6. $y_i(t)$: y-coordinate of node i at instant t;
7. $D_{i,j}(t)$: euclidean distance between nodes i and j at time t;
8. $RD(\vec{a}(t), \vec{b}(t'))$: relative direction (or cosine of the angle) between the two vectors;
9. $SR(\vec{a}(t), \vec{b}(t'))$: speed ratio between the two vectors;
10. R: transmission range of a mobile node;
11. N: number of mobile nodes;
12. T: simulation time;
13. $random()$: returns a value uniformly distributed in the interval $[-1, 1]$.

Given the terminology, we can describe the metrics used to differentiate mobility patterns starting from *direct metrics*:

- *degree of spatial dependence*
  This metric can be defined as the extent of similarity of the velocities of two nodes that are not too far apart. Formally, we have:

$$D_{spatial}\left(i, j, t\right) = RD\left(\vec{v_i}(t), \vec{v_j}(t)\right) * SR\left(\vec{v_i}(t), \vec{v_j}(t)\right) \tag{3.1}$$

The value is higher when the two vehicles travel at almost the same speed and direction. However, it decreases if relative direction or speed ratio decreases. It is possible to define also the *average degree of spatial dependence* as the $\bar{D}_{spatial}\left(i,j,t\right)$ averaged over node pairs and time instants satisfying certain condition. Its value is small when vehicles move independently, while it increases when the node's movement is somehow coordinated and they have almost the same directions and speeds:

$$\bar{D}_{spatial} = \frac{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=i+1}^{N} D_{spatial}\left(i,j,t\right)}{P} \tag{3.2}$$

where P is the number of tuples $\left(i,j,t\right)$ such that $D_{spatial}\left(i,j,t\right) \neq 0$.

- *degree of temporal dependence*
  This metric can be defined as the extent of similarity of the velocities of a node at two time slots that are not too far apart. Formally, we have:

$$D_{temporal}\left(i,t,t'\right) = RD\left(\vec{v_i}(t), \vec{v_i}(t')\right) * SR\left(\vec{v_i}(t), \vec{v_i}(t')\right) \tag{3.3}$$

The value is high when the vehicle travels at almost the same speed and direction over a certain interval of time that can be defined; it decreases, instead, if relative direction or speed ratio decreases. It is possible to define also the *average degree of temporal dependence* as the $\bar{D}_{temporal}\left(i,t,t'\right)$ averaged over node pairs and time instants satisfying certain condition. This value is small when the node is completely independent of its velocity, while it increases when the node's movement is strongly dependent on the velocity at some previous step:

$$\bar{D}_{temporal} = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{t'=1}^{T} D_{temporal}\left(i,t,t'\right)}{P} \tag{3.4}$$

where P is the number of tuples $\left(i,t,t'\right)$ such that $D_{temporal}\left(i,t,t'\right) \neq 0$.

- *relative speed (RS) and average speed (AS)*
  This metric can be defined with the standard definition from physics:

$$RS\left(i,j,t\right) = \left|\vec{V_i}(t) - \vec{V_j}(t)\right| \tag{3.5}$$

and in case of $D_{spatial}\left(i,j,t\right)$, it is possible to add the following condition:

$$D_{i,j}(t) > k * R \Rightarrow RS(i,j,t) = 0 \tag{3.6}$$

where k is a constant tha can be determined through simulations.[1]
At the same time, it is possible to define the *average relative speed* as the value $RS(i, j, t)$ averaged over node pairs and time instans satisfying certain condition. Formally:

$$\bar{RS} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t=1}^{T} RS\,(i, j, t)}{P} \tag{3.7}$$

where P is the number of tuples (i, j, t) such that $RS\,(i, j, t) \neq 0$.
Another way to define this metric can be the following: if $P(m, t)$ and $P(n, t)$ are the positions of hosts m and n at time t, respectively, the *relative velocity* between m and n can be defined as follows:

$$V(m, n, t) = \frac{d(P(m, t) - P(n, t))}{dt} \tag{3.8}$$

Averaging the absolute value of relative velocity over time, we have:

$$M_{m,n} = \frac{1}{T} \int_{t_0}^{t_0+T} |V(m, n, t)|\; dt \tag{3.9}$$

that is the *average relative speed* $M_{m,n}$ of vehicles m and n. At this point, it is also possible to introduce the average relative speed M, defined as $M_{m,n}$ averaged over all host pairs:

$$M = \frac{1}{N\,(N-1)\,/2} \sum_{m,n} M_{m,n} = \frac{1}{N\,(N-1)\,/2} \sum_{m=1}^{N} \sum_{n=m+1}^{N} M_{m,n} \tag{3.10}$$

where N is the number of vehicles.

- *geographic restrictions*
  This metric is very important and, intuitively, can be referred to the notion of *degree of freedom of points in a map*, the number of directions a vehicle can go after reaching that point. Formally defining this metric is not as trivial as it may seem, and authors (Bai et al.) cite this as their future work in (25).

As it concerns the *derived mobility metrics*, they are derived from graph theoretic models as well as other mathematical models and become very important because

[1]In their paper, Bai et al. proved that the best value for c_3 is 2, which allows $\bar{RS}$ to differentiate between the different mobility patterns very clearly.

mobility models usually impact the connectivity graph which, in turn, influence the protocol performance. The most important metrics that fall into this category are the *link change rate*, the *link duration* and the *path duration*.
In (39) and (40), authors suggest the *link change rate* as an indicator of topology change: if a link between two hosts is established /severed due to host movement, we consider the state of the link between them up/down. Clearly, the link change rate is the total number of up/down in unit time. The *average link duration* metric(25) is defined as the average link durations over the host pairs that are within each other's transmission range, where the link duration is the interval during which two hosts are within each other's transmission range. The *average path duration*(82), on the other side, can be defined as the interval between the time when the path is set up and the time when the path is broken.

Chapter 4

Vehicular Network Simulators

In the previous chapter we have underlined the relevance of network simulation in VANETs research, which became a very important and mandatory step in software design before any further implementation. With network simulators it is possible to easily design new protocols and analyze their impact on a large number of network nodes under several traffic and environmental conditions that may reflect in specific mobility and propagation models (cfr. previous chapter).

The most important feature which characterizes simulation is, obviously, the repeatability, which allows to study the overall performances of a given protocol by varying simply the simulation parameters and, obtaining the results in a shorter amount of time with respect to the real testbeds.

Both in academic and industry research, researchers almost always must resort to simulation as the expense of actual deployment would be too high. Unfortunately, there is no standard vehicular networks simulator and the common practice is to generate a mobility trace using a vehicular mobility simulator and then input this trace to the network simulator. Further, the right choice of the mobility simulator is important as performance in vehicular networks highly depends on the connectivity which, in turn, is affected from the nodes' movements.

In this chapter we presents an overview of the most popular vehicular network simulators and mobility simulators, according to their integration component.

4.1 Introduction to VANET simulation

In order to simulate a VANET we need both the networking and the mobility component. As we already said, in most cases, these two functionalities are provided by two independent simulators so that researchers build a topology and produce a trace of vehicle movements using a mobility simulator. This trace file, which contains all the coordinates of the vehicles at each step of the simulation (which can range from tenths of milliseconds up to seconds) is then fed to the network simulator.
Recently, several integrated simulators - that contain both networking and vehicular mobility components - have been developed, thus allowing feedback between the two components (e.g., TraNS(76) and ASH(7)). However, the common practice in this research field seems to be still the use of two separate simulators.

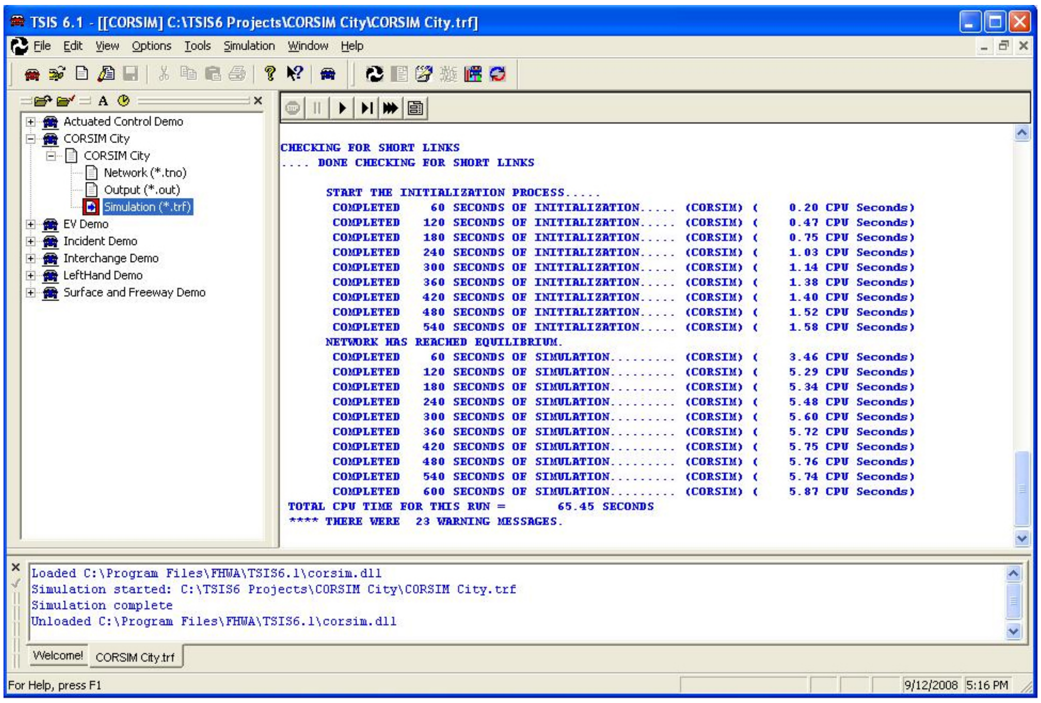
VANET simulators can be roughly classified as either loosely integrated or tightly integrated. *Loosely-integrated* simulators use separate mobility simulators and network simulators. The mobility simulator generates the mobility of vehicles and records the vehicular movements into trace files. The network simulator imports these trace files, but there is no direct interaction between the two simulators. *Tightly integrated simulators* do not use trace files but rather embed the mobility simulator and network simulator into a single vehicular network simulator. In some cases, the mobility model and network model can communicate, providing feedback from the network simulator to adjust parameters of vehicular movement. For example, in traffic congestion notification systems, the receipt of certain network messages may cause a vehicle to change its path (i.e., take an early exit). In collision avoidance systems, network messages may cause a vehicle to slow down to avoid an accident. This type of feedback is not supported by loose integration of mobility and network simulators. As loosely integrated simulation is currently more prevalent, separate mobility and network simulators will be discussed first, followed by a description of several tightly integrated simulators.

4.2 Mobility simulators

VANET mobility simulators are used to generate traces of the vehicles' motion that can be usually saved and subsequently imported into a network simulator in order to study the performances of the protocol/application. As mentioned earlier, it is important to generate realistic movement traces in order to rigorously evaluate VANET protocols because the overall performances depend on the connectivity which, in turn, relies on the movement traces.

4.2.1 TSIS-CORSIM

TSIS-CORSIM (Traffic Software Integrated SystemCorridor Simulation(5) is a powerful commercial traffic simulation package, developed at the University of Florida and funded by the Federal Highway Administration (FHWA). CORSIM is a microscopic simulation model that is especially designed to simulate highways and surface streets and, thus, includes traffic signals and stop signs.
This simulator is made up of two main components, NETSIM for simulating surface streets, and FRESIM for simulating freeways. Unfortunately, TSIS-CORSIM simulator requires Microsoft Windows and Internet Explorer to run. CORSIM can simulate very



```
CHECKING FOR SHORT LINKS
.... DONE CHECKING FOR SHORT LINKS

      START THE INITIALIZATION PROCESS.....
       COMPLETED    60 SECONDS OF INITIALIZATION..... (CORSIM) (    0.20 CPU Seconds)
       COMPLETED   120 SECONDS OF INITIALIZATION..... (CORSIM) (    0.47 CPU Seconds)
       COMPLETED   180 SECONDS OF INITIALIZATION..... (CORSIM) (    0.75 CPU Seconds)
       COMPLETED   240 SECONDS OF INITIALIZATION..... (CORSIM) (    1.03 CPU Seconds)
       COMPLETED   300 SECONDS OF INITIALIZATION..... (CORSIM) (    1.14 CPU Seconds)
       COMPLETED   360 SECONDS OF INITIALIZATION..... (CORSIM) (    1.38 CPU Seconds)
       COMPLETED   420 SECONDS OF INITIALIZATION..... (CORSIM) (    1.40 CPU Seconds)
       COMPLETED   480 SECONDS OF INITIALIZATION..... (CORSIM) (    1.52 CPU Seconds)
       COMPLETED   540 SECONDS OF INITIALIZATION..... (CORSIM) (    1.58 CPU Seconds)
      NETWORK HAS REACHED EQUILIBRIUM.
       COMPLETED    60 SECONDS OF SIMULATION......... (CORSIM) (    3.46 CPU Seconds)
       COMPLETED   120 SECONDS OF SIMULATION......... (CORSIM) (    5.29 CPU Seconds)
       COMPLETED   180 SECONDS OF SIMULATION......... (CORSIM) (    5.34 CPU Seconds)
       COMPLETED   240 SECONDS OF SIMULATION......... (CORSIM) (    5.48 CPU Seconds)
       COMPLETED   300 SECONDS OF SIMULATION......... (CORSIM) (    5.60 CPU Seconds)
       COMPLETED   360 SECONDS OF SIMULATION......... (CORSIM) (    5.72 CPU Seconds)
       COMPLETED   420 SECONDS OF SIMULATION......... (CORSIM) (    5.75 CPU Seconds)
       COMPLETED   480 SECONDS OF SIMULATION......... (CORSIM) (    5.76 CPU Seconds)
       COMPLETED   540 SECONDS OF SIMULATION......... (CORSIM) (    5.74 CPU Seconds)
       COMPLETED   600 SECONDS OF SIMULATION......... (CORSIM) (    5.87 CPU Seconds)
 TOTAL CPU TIME FOR THIS RUN =       65.45 SECONDS
 **** THERE WERE  23 WARNING MESSAGES.
```

```
Loaded C:\Program Files\FHWA\TSIS6.1\corsim.dll
Simulation started: C:\TSIS6 Projects\CORSIM City\CORSIM City.trf
Simulation complete
Unloaded C:\Program Files\FHWA\TSIS6.1\corsim.dll
```

Figure 4.1: CORSIM. A screenshot of CORSIM simulator..

complex traffic scenarios and thus it can take a large amount of information as input. A

graphical editor, called TRAFED, is included with the package to allow the traffic scenario to be described. The description of highways, or freeways, includes intersections, road segment lengths, number of lanes, merge lane lengths, adding or removing lanes, free flow speeds, roadway curvature, and roadway grade. The description of surface streets, or arterial roadways, includes road segment lengths, lane utilization, free flow speeds, and signal timings. CORSIM also allows the traffic volume to be specified.
The output of this simulator is a MOE (Measure Of Effectiveness) file which is processed into tables that summarize the information. For freeway models, the key metrics produced are throughput, speed, and density. The information for arterial models includes throughput, control delay, and maximum queues. The tables also highlight problem areas that affect arterial and freeway performance, such as ramp intersections. The full TSIS-CORSIM package also includes TRAFVU, which uses CORSIM input and output files to produce animations of the traffic network that has been simulated.

4.2.2 VisSim

VisSim[1] is a commercial simulator developed by PTV Planung Transport Verkehr AG in Karlsruhe, Germany, which allows users to simulate large and complex traffic scenarios, including many different objects like cars, trucks, pedestrians, cyclists, etc.
In detail, VisSim is a *microscopic multi-modal traffic flow simulation software*, which means that each entity of reality that is to be simulated is simulated individually, i.e. it is represented by a corresponding entity in the simulation, thereby considering all relevant properties. The same holds for the interactions between the entities. The opposite would be a macroscopic simulation, in which the description of reality is shifted from individuals to averaged variables like flow and density.
It is also possible to perform to generate 3D animations by using a sophisticated graphical user interface (GUI).
VisSim is only avaliable for Microsoft Windows platforms.

4.2.3 PARAMICS

PARAMICS is a software suite, developed by Quadstone Ltd, which adopts the microsimulation paradigm and runs only on Microsoft Windows based platforms. With

[1] http://www.vissim.com

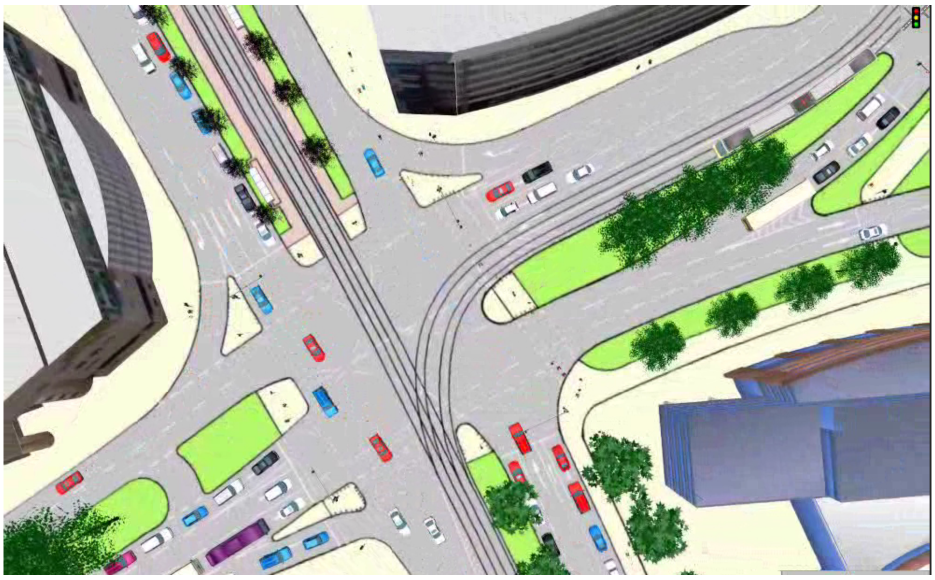


Figure 4.2: VisSim example (A). Simulation of intersections with regard to design alternatives (roundabouts, unsignalized and signal-controlled, grade separated interchanges) and design, test and analysis of vehicle-actuated signal controls.

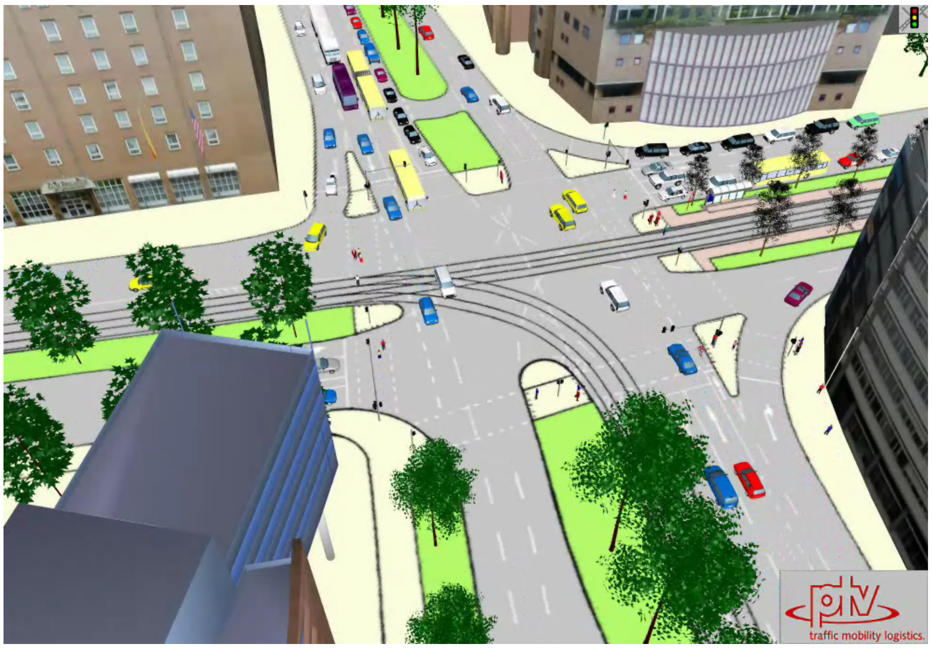


Figure 4.3: VisSim example (B). The same intersection under another perspective.

PARAMICS it is possible to simulate intersections, highways, urban zones and, like VisSim, it includes a 3D visualization tool.

As already said, PARAMICS is a software suite, which means that it is made up of several modules: modeller, analyser, processor, estimator, designer, converter, programmer and monitor.

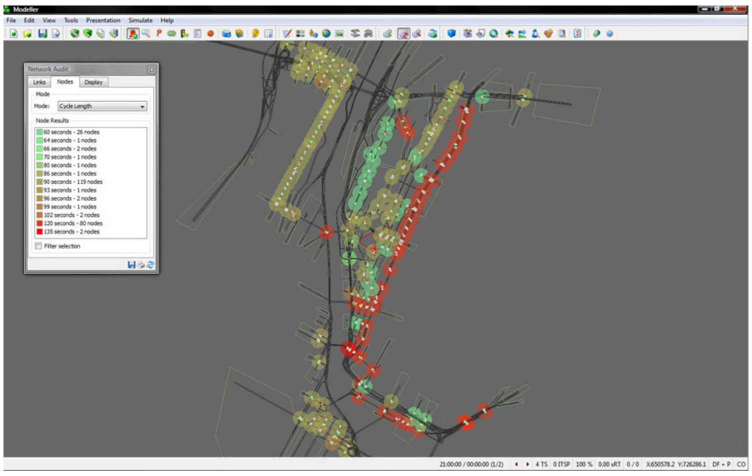


Figure 4.4: Paramics example (A).

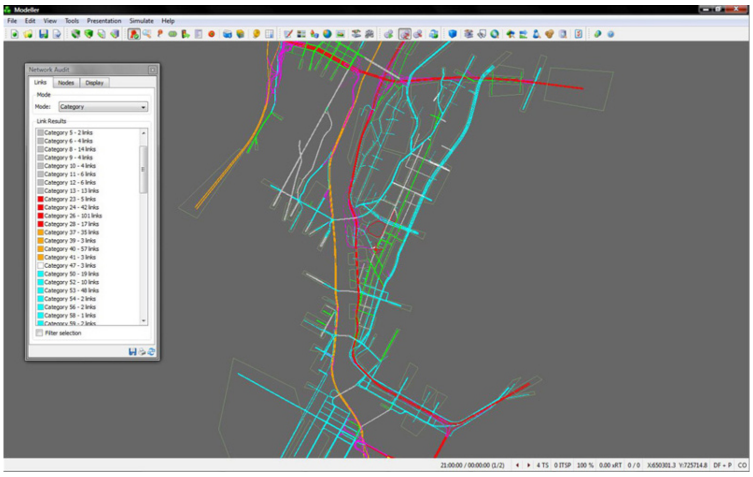


Figure 4.5: Paramics example (B).

4.2.4 Canumobisim and Vanetmobisim

CanuMobiSim(1) is a Java-based simulator designed for mobility simulation by the CANU research group at the University of Stuttgart. It implements several mobility models and can generate mobility traces suitable for use in various network simulators, including ns-2 and GloMoSim. CanuMobiSim allows maps formatted according to the GDF (Geographical Data Files) standard[1] to be imported. Since CanuMobiSim was originally designed for MANET, the mobility models are more suitable for MANETs than VANETs. The mobility models implemented in the CanuMobiSim simulator include the following: *brownian motion(60)*, *GaussMarkov random motion*, *incrementally changed random motion*, *random waypoint(44)*, *graph-based mobility model*, *constant speed motion*, *fluid traffic model*, *intelligent driver model (98)* and *smooth motion.*
Vanetmobisim(35) is an opensource user mobility simulator developed as an extension to Canumobisim(1). It is completely platform independent as it is coded in Java and it is capable to produce mobility traces for different simulator such as ns-2(3), Glomosim(2) and Qualnet(4).
The aim of Vanetmobisim is to provide a high degree of realism in mobility simulations, thus taking into account both macro and micro mobility features. When considering macro-mobility, it is considered not only the road topology but also the road structure (unidirectional, bidirectional, single/multi-lane), the road characteristic (speed limits, cehicle class based restrictions) and the presence of traffic signs (stop signs, traffic lights, etc.). Moreover, in the macro-mobility it is also considered the effects of the presence of point of interests, which can influence movements patterns. As the road topology is akey factor to obtain realistic results when simulating vehicular movements, Vanetmobisim allows to define the road topology in the following ways (the first two were imported from Canumobisim):

- *GDF map:* the road topology is imported from Geographical Data File. Unfortunately, most of GDF libraries are not freely accessibile.
- *TIGER map:* in this case, topology is extracted from a map obtained from the TIGER database(6). The map's details are not as high as GDF but it is open

[1]International Standard Organization (ISO), Intelligent transport systems - Geographic Data Files (GDF), overall specification, ISO 14825:2004, 2004.

and contains digital descriptions of wide urban and rural areas of all districts of United States.

- *Clustered Voronoi graph:* the road topology is randomly generated by creating a Voronoi tessellation[1] on a set of non-uniformly distributed points. This creates fast and configurable random graphs, yet reflecting the non-uniform distribution of obstacles in a urban area.

- *User-defined maps:* the road topology is specified by listing the vertices of the graph and their interconnecting edges.

- *OpenStreetMap:* this feature is included in the unofficial version 2.0.9 of Vanet-mobisim and it allows to import real maps from Openstreetmap website, a free editable map of the whole world. It is also possible to import a selected map area

[1]In mathematics, a Voronoi diagram is a special kind of decomposition of a metric space determined by distances to a specified discrete set of objects in the space, e.g., by a discrete set of points. It is named after Georgy Voronoi, also called a Voronoi tessellation, a Voronoi decomposition, or a Dirichlet tessellation (after Lejeune Dirichlet). In the simplest case, we are given a set of points S in the plane, which are the Voronoi sites. Each site s has a Voronoi cell, also called a Dirichlet cell, V(s) consisting of all points closer to s than to any other site. The segments of the Voronoi diagram are all the points in the plane that are equidistant to the two nearest sites. The Voronoi nodes are the points equidistant to three (or more) sites.
A point location data structure can be built on top of the Voronoi diagram in order to answer nearest neighbor queries, where one wants to find the object that is closest to a given query point. Nearest neighbor queries have numerous applications. For example, when one wants to find the nearest hospital, or the most similar object in a database. A large application is vector quantization, commonly used in data compression. With a given Voronoi diagram, one can also find the largest empty circle amongst a set of points, and in an enclosing polygon; e.g. to build a new supermarket as far as possible from all the existing ones, lying in a certain city. The Voronoi diagram is useful in polymer physics. It can be used to represent free volume of the polymer. It is also used in derivations of the capacity of a wireless network. In climatology, Voronoi diagrams are used to calculate the rainfall of an area, based on a series of point measurements. In this usage, they are generally referred to as Thiessen polygons. Voronoi diagrams are used to study the growth patterns of forests and forest canopies, and may also be helpful in developing predictive models for forest fires. Voronoi diagrams are also used in computer graphics to procedurally generate some kinds of organic looking textures. In autonomous robot navigation, Voronoi diagrams are used to find clear routes. If the points are obstacles, then the edges of the graph will be the routes furthest from obstacles (and theoretically any collisions). In computational chemistry, Voronoi cells defined by the positions of the nuclei in a molecule are used to compute atomic charges. This is done using the Voronoi deformation density method (source $http:$ $www.wikipedia.org$).

directly into the simulator GUI. This is the option we used to simulate our data dissemination protocol, CORP.

Anyways, the road topology is implemented as a graph over whose edges the vehicle movements are constrained.

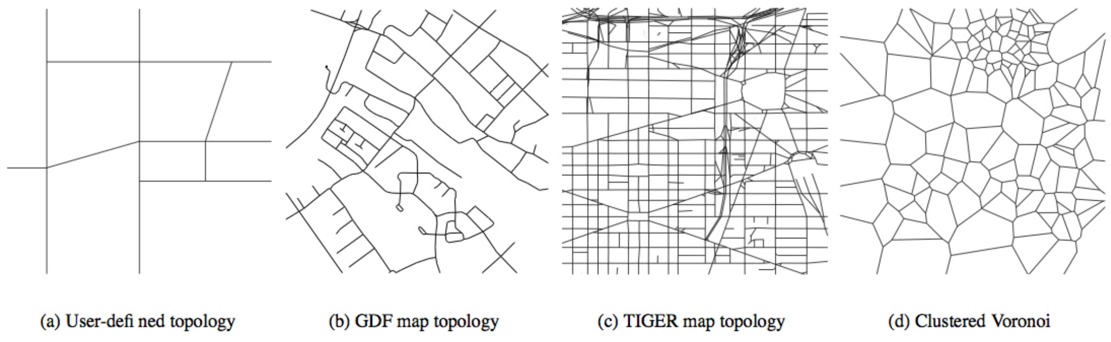


Figure 4.6: Vanetmobisim road topology example.

It is important to recall that Vanetmobisim introduces new feature on road characterization like:

- introduction of roads with multiple lanes in each direction;
- physical separation of opposite traffic flows on each road;
- definition of independent speed limits on each road of the topology;
- implementation of traffic signs at each road intersection. By default, traffic intersections are regulated by stop signs, forcing vehicles to stop and wait for free road before crosing. In addition (but it is still experimental), it is possible to use traffic lights, whose temporization is customizable.

The vehicle movements in Vanetmobisim are very realistic as it exploits the capability of building up movements patterns from the cooperation of a trip generation module, which defines the sets of points of interests, and a path computation module, whose task is to compute the best path between those points.

The trip generation module is mainly based on two options: the first is a *random trip*, where the start and stop points of movement patterns are randomly selected

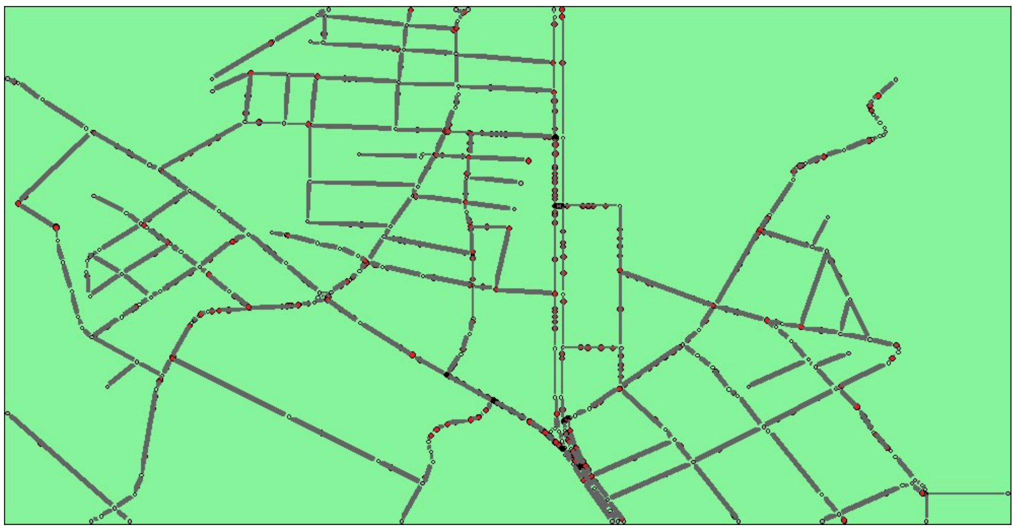


Figure 4.7: Vanetmobisim road topology example imported from OpenStreetMap database. The map showed in this picture (”Quarto Miglio” district in Rome) was exported in xml-format from OpenStreetMap and then imported into Vanetmobisim simulator. The red dots represent the constrained vehicles moving to across roadmap during the simulation.

among the vertices of the graph representing the road topology; the second is an *activity sequences generation*, as a set of start and stop points are explicitly provided in the road topology description, and cars are forced to move among them. Independently from the trip generation method employed, the path computation, i.e. the selection of the best sequence of edges to reach the selected destination, can be performed in three ways:

1. *Dijkstra's algorithm:* the shortest path to destination is evaluated with the Dijkstra's algorithm, with edges' cost inversely proportional to their length;
2. *traffic congestion level:* method does not only considers the length of the path, but also the traffic congestion level, by weighting the cost of traversing an edge also on the number of cars traveling on it, thus modeling the real world tendency of drivers to avoid crowded paths;
3. *road speed limit:* The last method, which is not present in the original CanuMobiSim, extends the other two, by also accounting for the road speed limit when calculating the cost of an edge, in a way that fastest routes are preferred.

Of course, when the definition of vehicular movement paths is a factor of interest in the mobility simulation, it is possible to use a combination of trip generation and path computation methods offers a wide range of possibilities.

4.2.4.1 Microbility features in Vanetmobisim

At this point, it is important to come again on the micro-mobility features included in vanetmobisim, as they play the main role in the generation of realistic vehicular movements and they can, thus, making the difference in the simulation results. It is possible to distinguish three classes of micro-mobility models, featuring an increasing degree of detail, mainly based on whether the individual speed of vehicles is computed: in a *deterministic way*, *as a function of nearby vehicles in a single lane scenario*, *as a function of nearby vehicles' behavior in a multi-flow interaction scenario* (i.e. like the urban one). The first three models we can find in Vanetmobisim (inherited from CanuMobiSim) are the *Graph-Based Mobility Model (GBMM)(96)*, the *Constant Speed Motion (CSM)(1)* and the *Smooth Motion Model (SMM)(14)*. All these models fall into the first class, as the speed of each vehicle is determined on the basis of the local state of each car while any external effect is ignored. Both models constrain a random movement of nodes on a graph, possibily including pauses at intersections (CSM) or smooth speed changes when reaching or leaving a destination (SSM). The movements is random, as the vehicles select a destination and move towards it along a shortest length path (ignoring other vehicles and overlapping). Obviously, these models fail to reproduce realistic movements of vehicles' clusters.
In the second class, instead, there are two main models: the *Fluid Traffic Model (FTM)(88)* and the *Intelligent Driver Model(98)*. Both models fall this time into the second class, as they take into account the presence of nearby vehicles when calculating the relative speeds (but do not consider, however, the case in which multiple vehicular flows have to interact in presence of intersections). In particular, the FTM describes the speed as a monotonically decreasing functionof the vehicular density, forcing a lower bound on speed when traffic congestion reach a critical state according to the following equation:

$$s = max\left[s_{min}, s_{max}\left(1 - \frac{k}{k_{jam}}\right)\right] \tag{4.1}$$

where s is the output speed, s_{min} and s_{max} are respectively the minimum and the maximum speed, k_{jam} is the vehicular density for which a traffic jam is detected, and k is the current vehicular density of the road the node is moving on. The parameter k is calculated as n/l, where n is the number of cars on the road and l is the road segment itself.
The IDM model, instead, relies mainly on the following car (that is why this model falls into the so called *"car following model"*) and the istantaneous acceleration of a vehicle is computed according to the formula:

$$\frac{dv}{dt} = a\left[1 - \left(\frac{v}{v_0}\right)^4 - \left(\frac{s^*}{s}\right)^2\right] \tag{4.2}$$

$$s^* = s_0 + \left(vT + \frac{v\Delta v}{2\sqrt{ab}}\right) \tag{4.3}$$

In the equation (4.2), v is the vehicle's current speed, v_0 the desired velocity, s is the distance from the preceding car and s_0 is the so called *desired dynamical distance*, computed in (4.3) as a function of the minimum bumper-to-bumper distance s_0, the minimum safe time headway T, the speed difference with respect to front vehicle's velocity Δv, and the maximum acceleration values a and b.
Vanetmobisim, as said before, inherit alle these models from CanuMobiSim but, in addition, we can find two IDM variants: the *Intelligent Driver Model with Intersection Management (IDM-IM)* and the *Intelligent Driver Model with Lane Changes (IDM-LC)*. According to the first, vehicles moving in the scenario can handle, basically, two kinds of intersections: crossroads regulated by stop signs and road junction ruled by traffic lights. Clearly, even if the intersection modeling is applied only to the first car on each road, the overall effect of handling such a scenario is reflected to the other vehicles as IDM automatically adapts the behavior of car following the leading one. Everytime the vehicle doesn't find any other vehicle between itself and the intersection regulated by stop signs, the following parameters are used:

$$\begin{cases} s = \sigma - S \\ \Delta v = v \end{cases} \tag{4.4}$$

where σ represents the current distance to the intersection and S is a safety margin, which considers the gap between the center of the intersection and the point where the car would actually stop. Once a car is halted at a stop sign, it is informed by the macroscopic level description of the number of cars already waiting to cross the intersection from any of the incoming roads. If there are no other cars, the vehicle may pass. Otherwise, it has to wait for its turn in a first-arrived-first-passed and right hand rule policy. When a vehicle is heading towards a traffic light intersection, it is informed by the macroscopic description about the state of the semaphore. If the color is green, passage is granted and the car maintains its current speed through the intersection. If the color is red, crossing is denied and the car is forced to decelerate and stop at the road junction by using the modied IDM parameters similarly to a stop sign. It may also be stressed out that vehicles behavior can dynamically vary in presence of traffic lights, according to red-to-green and green-to-red switches. In the former case, a car currently decelerating to stop at a red light will accelerate again if the semaphore turns green before it has completely halted. In the latter case, a vehicle keeping its pace towards a green light will try to stop if the light becomes red before it has passed through the intersection. A minimum breaking distance $\bar{s}$ is evaluated by means of simple kinematic formulae as the following:

$$\bar{s} = vt - \frac{kb}{2}t^2 = v\left(\frac{v}{kb}\right) - \frac{kb}{2}\left(\frac{v}{kb}\right)^2 = \frac{v^2}{2kb} \tag{4.5}$$

which describes the space needed to come to a full stop as a function of the current speed pf the vehicle, v, the time t and the deceleration value, kb. The last parameter is the maximum safe deceleration, evaluated as the IDM comfortable braking value b scaled by a factor $k \geq 1$.

Upon computation of $\bar{s}$, if the vehicle finds that it is not possible to stop before the intersection, even braking as hard as possible (i.e., if $\bar{s} > \sigma - S$), then it crosses the intersection at its current speed. Otherwise, it stops by applying a strong enough deceleration. This reproduces a real world simulation, since when a traffic light switches to red, drivers only stop if safety braking conditions can be respected.

The second variant, the IDM-LC, extends the original IDM model introduced in CanuMobiSim with the possibility for vehicles to change lane and overtake each others, taking advantage of the multi-lane capability. The multi-lane handling is a trivial task especially when approaching an intersection; in that case, a vehicle can decide to keep traveling on the same lane in the next road (if the same lane is present on the new segment) or try to merge to its right (if the current lane is not present in the new road). In the latter situation, if the lane on the right is congestioned, the vehicle has to stop and wait until a spot becomes avaliable. As it concerns the overtaking mechanism, the IDM-LC inherits the MOBIL model(97), based on a game theoretical approach to address the lane changing problem; basically, it allows a vehicle to change lane if this shift minimize the overall braking. Mathematically speaking, this is achieved when the following two conditions are satisfied:

$$a^l - a \pm a_{bias} > p\left(a_{cur} + a_{new} - a^l_{cur} - a^l_{new}\right) + a_{thr} \tag{4.6}$$

$$a^l_{new} > -a_{safe} \tag{4.7}$$

This means that a vehicle is allowed to move to lane l if the disadvantage of the driver who is doing the change is greater than the disadvantages of the following cars (to be more precise, when the first inequality is satisfied) in the current ($a_{cur} - a^l_{cur}$) and in the candidate ($a_{new} - a^l_{new}$) lanes. The p factor is the *driver's politness*, ranging from above 1 (very polite driver) to 0 (selfish or even malicious driver); the a_{thr} is the threshold acceleration and represents a the minimum acceleration advantage to allow lane change (this is in order to avoid lane hopping in border cases). The last to parameters, a_{bias} and a_{safe}, represent respectively the real tendency to stay on the right most lane and the condition that the driver can change lane only if the back vehicle does not to brake too hard (its deceleration has to be over the value a_{safe}).

4.2.5 SUMO and MOVE

SUMO is a microscopic, space continuous and time discrete traffic simulator written in C++ capable to provide accurate and realistic mobility patterns. The project started as an open source project in 2001 with the goal to support the traffic research community with a common platform to test and compare models of vehicle behaviour, traffic light optimisation, routing etc.

In SUMO, each vehicle has an own route and is simulated individually according to a fast but still accurate model developed in 1998 by Stefan Krauss (51) (extended by some further assumptions), which uses discrete time steps of 1s. SUMO is very fast, around 100.000-200.000 vehicles can be simulated in real time on a desktop PC, including: