Proceedings of the 9th International Conference "Reliability and Statistics in Transportation and Communication" (RelStat'09), 21-24 October 2009, Riga, Latvia, p. 384-393. ISBN 978-9984-818-21-4
Transport and Telecommunication Institute, Lomonosova 1, LV-1019, Riga, Latvia

MICROSCOPIC DISCRETE EVENT URBAN TRAFFIC MODEL VALIDATION USING SIMULATION

Emmanuel López-Neri
Universidad del Valle de México, Campus Guadalajara Sur
Periférico Sur No. 8100 C.P. 45601,
Guadalajara, Jal. México.
emmanuel.lopez@guadalajara.uvmnet.edu

In this paper a microscopic discrete event urban traffic model validation using simulation is presented. In a previous study a hierarchical microscopic urban traffic system (UTS) model was developed [**1**]. That model integrates the event oriented and agent-based approach. The UTS is described using the multi-level Petri net based formalism, named n-LNS. The first level describes the traffic network; the second level models the behavior of diverse road network users considered as agents, and the third level specifies detailed procedures performed by the agents, namely travel plans, tasks, etc.. Usually simulators are designed using time step approach and are validated using real data and is verified that the flow/density relationship (fundamental diagram) are conserved and then state the simulator generates a valid behavior. However, the model used in this paper uses the event oriented approach, doing more complex the process to obtain these validation graphs and their corresponding analysis. In order to validate it, was developed a library known as *CiudadelaSim* [1].

Keywords: discrete event, traffic simulation, microscopic validation, n-LNS, Petri nets

Introduction

An urban traffic system (UTS) is composed of vehicles, pedestrians, traffic lights, and a traffic network structure. The large number of the vehicles provokes well known problems such as traffic jams, air and noise pollution, fuel consumption, stress, etc. These problems may be reduced by the efficient use of current urban resources through the performance analysis under different traffic light control policies along the day. Model-based simulation is often used for evaluating UTS yielding statistics about travel times, fuel consumption, and road density; such information is useful to study traffic control strategies, urban transport routes, etc. Simulation has been increasingly adopted by the engineers and personnel charged to plan the signaling policies of the traffic network, in the literature there exist different approaches to model UTS [**2**].

Within the urban area, micro-simulation is better adapted for the analysis in detail of the vehicles behavior, the performance of streets and intersections, and the effectiveness of traffic lights control strategies [**3**][**2**]. Under the micro-simulation approach an UTS can be considered as a discrete event system in which the simulation time advance is handled using the next event technique [**4**][**5**].

In this paper a previous study of hierarchical microscopic urban traffic system (UTS) model is used [**1**]. That model integrates the event oriented and agent-based approach. The UTS is described using the multi-level Petri net based formalism, named n-LNS. The first level describes the traffic network; the second level models the behavior of diverse road network users considered as agents, and the third level specifies detailed procedures performed by the agents, namely travel plans, tasks, etc..

Usually simulators are designed using time step approach and are validated using real data and is verified that the flow/density relationship (fundamental diagram) are conserved and then state the simulator generates a valid behavior. However, the model used in this paper uses the event oriented approach, doing more complex the process to obtain these validation graphs and their corresponding analysis. In order to validate it, was developed a library known as *CiudadelaSim* [1]. The system is open-source and free. *CiudadelaSim* may be downloaded at http://sites.google.com/site/ciudadelasimsite/. CiudadelaSim is not derived from any other toolkit, but rather was built from scratch using multi-agent event oriented principles. Our design philosophy was to build a fast, orthogonal, minimal model library to which an experienced Java programmer can easily add features, rather than one with many domain-specific, intertwined features which are difficult to remove or modify.

Urban Traffic System Components

The UTS entities or components are: network streets and intersections, road users (vehicle, pedestrians, cyclists, etc.), traffic signs (dynamic: traffic light, and static: speed limit sign) and individual and emergent behavior (see fig. 3).are classed into static and dynamic entities. Static entities cannot change

their state, for instance traffic signals (speed limit, priority flow, etc.) or the street network. Dynamical entities or road users are objects that can move through the road network and/or change their own state, i.e., they have their own behavior (cars, pedestrians, traffic lights, variable messages signs, etc.).

The road user behavior is defined as a discrete event system. For instance, the relevant events for the entity named "vehicle" are advance, stop, accelerate, decelerate, change lane, and the states are stopped and advancing. Since in actual UTS the car drivers see other cars in their neighborhood or field of view (FOV), then road users perceive the events of other dynamic entities in their neighborhood.

Besides the description of the behavior of dynamic entities, the evolving rules must be also specified. These rules govern the joint behavior of entities. For instance, an evolving rule could be "two or more entities cannot be in the same space at the same time". The evolving rules are axioms that the UTS entities cannot violate. The interaction of one road user with other road users, static components, and traffic signs leads to more complex behaviors known as emergent behaviors, for example: queues, traffic jams, gridlock, green wave, etc. [6]. This emergent behavior is not explicitly captured in the model, but it will be appear when the UTS model evolves, for instance, when a micro-simulator is used. The knowledge about queues, traffic jams, etc., allow to road users making better decisions during their execution.

The road network is a set *S* of interconnected streets and intersections called segments; it contains the traveling road users the dynamic and static traffic signals. These interconnections are defined by the following two relations:

The segment is the UTS environment basic modeling unit. Each segment represents a network structure road where the entity could displace in a sequential way. The entity may find physical obstacles during their displacement or caused by obey certain traffic policies; among other informative objects. Only one direction at time is allowed in each intersection. In this way, each intersection segment can contains only one vehicle at time. Also, the use of some segments is restricted to a certain kinds of entities.

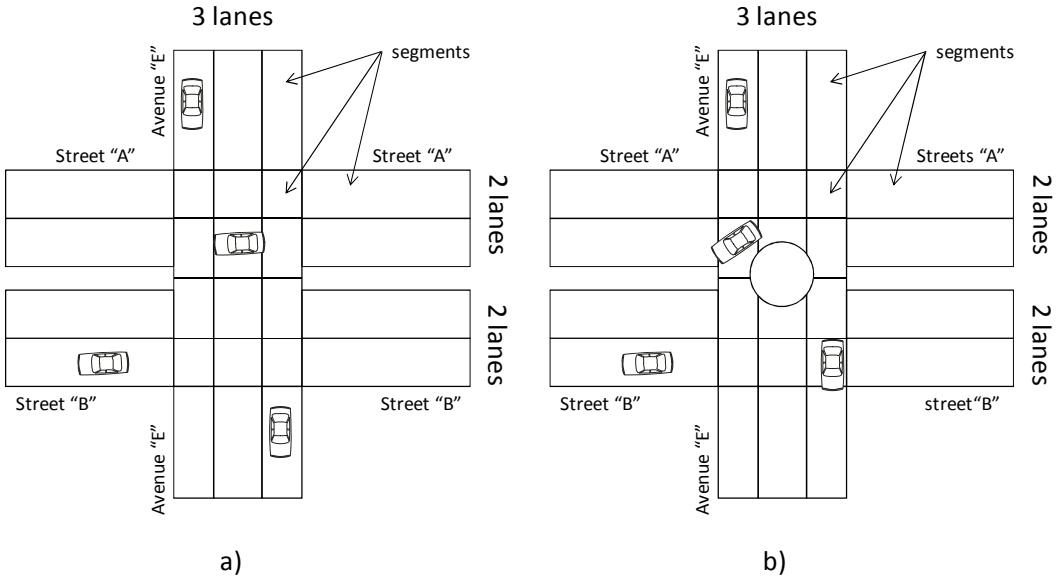

Figure 1. Urban traffic network description using segments a) Intersection b) Roundabout

Definition An object $\boldsymbol{obj_i}$ is a 3-tuple defined by $\boldsymbol{obj_i} = (type_i, value_i, w_i)$ where:

- $type_i \in \{staticSignal, variableMessageSign, bump, trafficLight, stopSignal\}$.
- $value_i$ is the information provided to the entity, such that $value_i \in \aleph$.
- w_i is the object relative position at segment, such that $w_i \in \Re^+$

Definition 1. A segment s_i is a 5-tuple defined by $s_i = (O_i, typeS_i, a_i, b_i)$ where:

- O_i={obj_j } is the set of objects in the segment *i*.
- $\text{typeS}_i \in \{\text{use}_{\text{vehicular}}, \text{cross}_{\text{pedestrian}}, \text{exclusive}_{\text{bus}}, \text{exclusive}_{\text{train}}\}$ represents the segment restrictions use.
- $a_i, b_i = (lat_i, long_i)$ y $lat_i, long_i \in \Re$, which are the geometric coordinates (latitude y longitude) that describe the segment endpoints on a map.

The next relationships allow to establish the connections between segments:

Relation 1. Sequential Neighborhood. $NS = \{(s_i, s_j) \mid s_i s_j \in S$, the entities can displace sequentially from segment s_i to segment s_j, adding it to the tail end of s_j }. If $\exists(s_i, s_j) \epsilon NS \rightarrow \exists(s_j, s_i) \epsilon NS$, since the relationship NS describes the physical connection between segments.

If the entity's ability to make a change lane is modeled then is added the following relation:

Relation 2: Contiguous Neighborhood $NC = \{(s_i, s_j) \mid s_i s_j \in S$, the entities can displace in a parallel way from s_i to s_j and be added in any segment position}. If $\exists(s_i, s_j) \epsilon NC \rightarrow \exists(s_j, s_i) \epsilon NC$, since the relationship NC describes the physical connection between segments.

Urban Traffic System Model based on n-LNS

The UTS model is expressed with n-LNS using three levels. In the first level the road network is described, the general behavior of the road users is specified by level 2 nets; then the tasks or procedures needed to implement specific behaviors of the road user are represented by nets of level 3. Figure 2 shows the hierarchical UTS description using n-LNS.

The formalism follows the approach of nets within nets introduced by R. Valk [24], in which a two level nested net scheme called EOS (Elementary Object System) is proposed. An extension to the Valk's technique, called n-LNS, has been proposed [23]; in this section we present an overview of n-LNS. A more accurate definition of the formalism is detailed in [23]. In the next section is presented the UTS model using n-LNS, for a detailed information refer to [**1**].

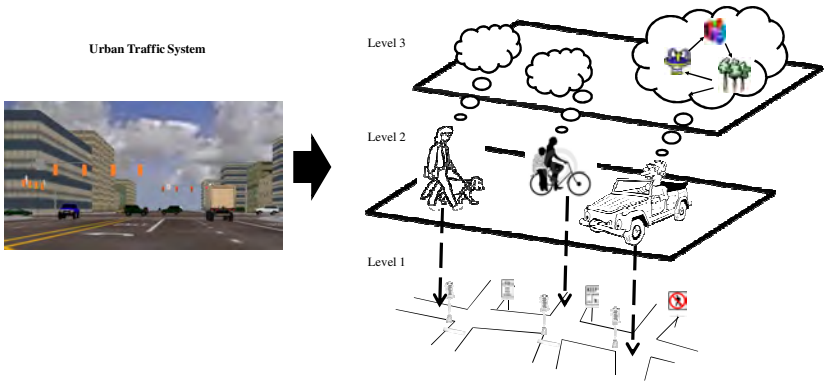

Figure 2. Hierarchical urban traffic system abstraction levels using n-LNS

First Level: The road network

The road network model can be straightforward obtained. For every segment $s_i \in S$, a place p_i is assigned. Then one transition t_{ij} is added for every (s_i, s_j) in *NC* or *NS*, together with arcs (p_i, t_{ij}) and (t_{ij}, p_j). Furthermore some transitions t_i must be added for every segment s_i source or sink; arcs (t_i, p_i) or (p_i, t_i) are added accordingly. Using this strategy the resulting model $typeNet_{1,1}$ ($EnvironmentNET_1$) for the traffic network showed in figure 3, the static traffic signals for instance speed limit, bumps position, segment size, are information sent to the agent when a *leavSeg* transition is fired (*t06, t17*,etc.).

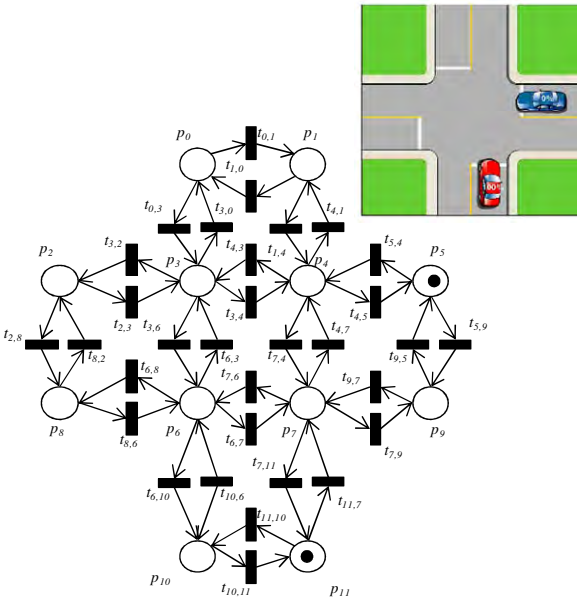

Figure 3. Road Network described with type net *EnvironmentNET*$_1$

Second Level: Agents

The decision making mechanism (DMM) of an agent is described by the net *typeNet*$_{2,1}$ showed in figure 4. During the reasoning process, the evolving rules and traffic policies are taken into account.

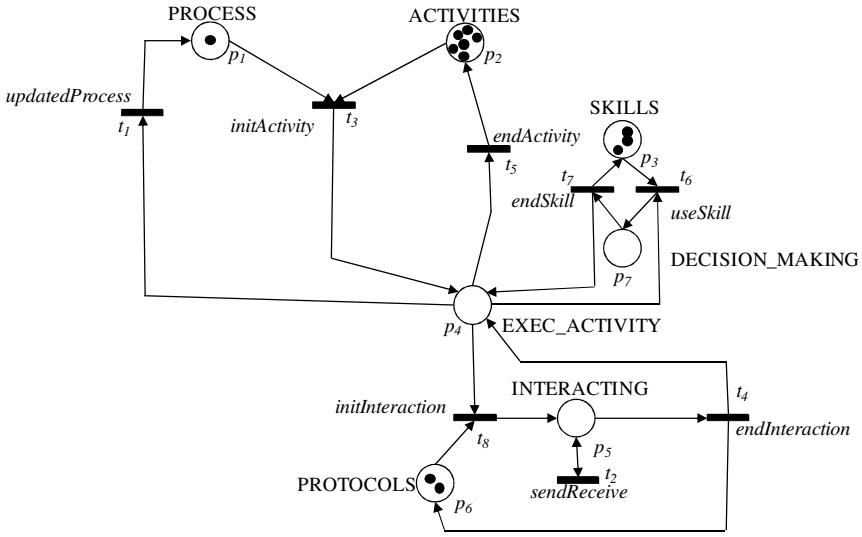

Figure 4. Decision Making Mechanism Described by typeNet$_{2,1}$

Third Level: Objects

Agent activities can be described by a third level net. In figure 5 shows the *typeNet*$_{3,1}$ that describe the vehicle driver activities and its possible states. If a in *typeNet*$_{3,1}$ transition is fired, then a fact is modified. Each transition (agent event) modifies some of the agent facts; for instance the *endChLn* transition modifies the position fact. These events start a DMM cycle. For other dynamical entities in the UTS, the behavior can be also represented by level 3 nets.

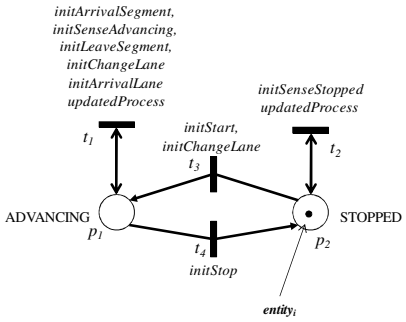

Figure 5. Vehicle Activities Described by typeNet$_{3,1}$

Simulator Implementation

CiudadelaSim library is the UTS n-LNS model implementation using Java™. This library provides to the computer programmer the necessary classes to implement a specific traffic model using the multi-agent paradigm. This library can be substituted, partially changed or increased to test new techniques or paradigms in an easy way. The *CiudadelaSim* library provides all the classes needed to implement different types of driver's behaviors as well as the models for the main driving task: car following, gap acceptance, and lane change. We have used a modular approach allowing each component to be easily redefined and extended.

A simulation is an instantiation of the classes from the *CiudadelaSim* library with the corresponding parameters to a specific experiment. The simulation can provide output statistics of each segment, as well as global network statistics. The results of the simulation and the links occupation ratios are obtain easily. Also a XML interface is provided to describe the UTS models using UTYiL language for the data model [**6**]. In figure 1 the correspondent library architecture is shown.

The library consists of five distributed modules: the car generator, traffic light control, simulation kernel, visualization and statics analyzer (see figure 6). These modules are distributed along the network. In order to distribute the simulation is used a connectivity software *ProActive* known as *middleware* [**7**]. The *middleware* allows a clear communication between different computers connected into a network (Internet, Intranet, etc.).

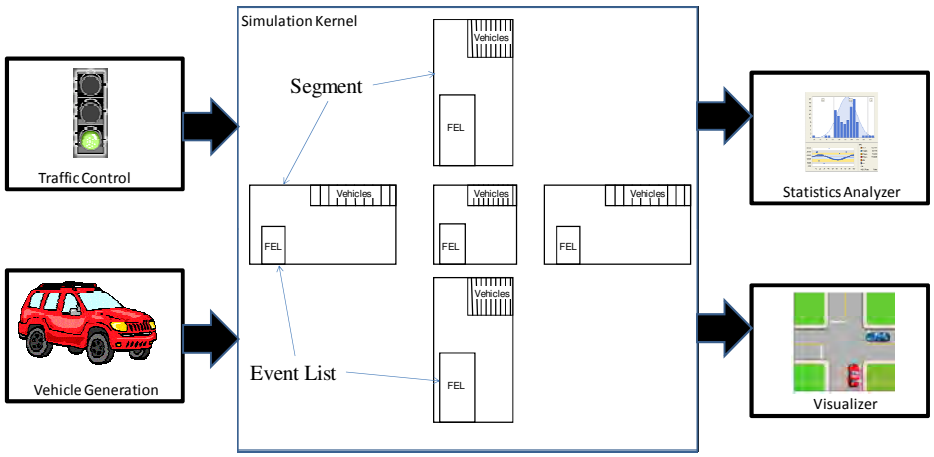

Figure 6. CiudadelaSim Architecture

In figure 7 the simulator class diagram is depicted. The *carGenerator*, *ligthControl*, *Analyzer*, *Visualisation*, *SimulatorControler* and *StreetController* classes implement the Proactive interface *RunActive*, this allow each module be distributed along the network.

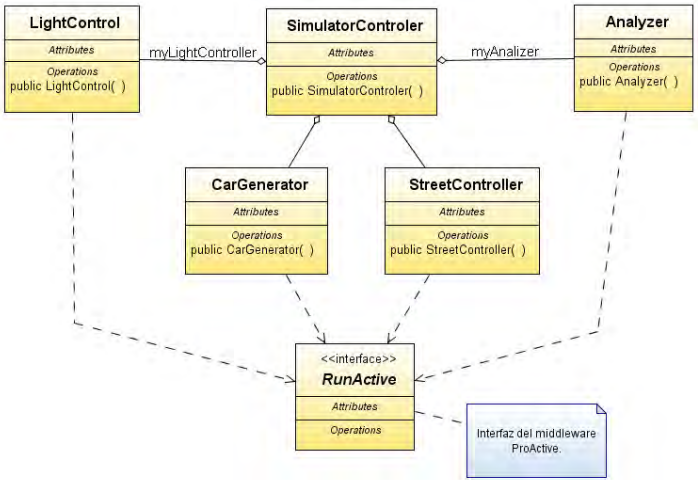

Figure 7. *CiudadelaSim* UML class diagram

The *SimulatorController* class creates instances of all other classes, read the UTS description model and generates the required data structures. The class *carGenerator* use a *Poisson* distribution to generate new vehicle events. This class sends the new events to the correspondent *StreetController* FEL structure. The *ligthControl* class generates new change light events for each tf_i in the traffic network structure. The *Visualization* class reads output from each simulation and graphically show the entity movement. The *Analyzer* class also reads output from each simulation but use them to obtain results statics for density, flow and travel times for each segment.

The *StreetController* class executes the events of each segment concurrently, but taking care of the causality rules. In this class the *runActivity()* method could be modified to evaluate distinct execution strategies. The space is subdivided by sequential sets of events (SQS) assigned to each segment. The potential event (event with minor timestamp) of each segment is ordered in a potential event list so could be executed in a distributed and concurrent approach. In figure 8 is shown the future event lists and potential event lists data structures.

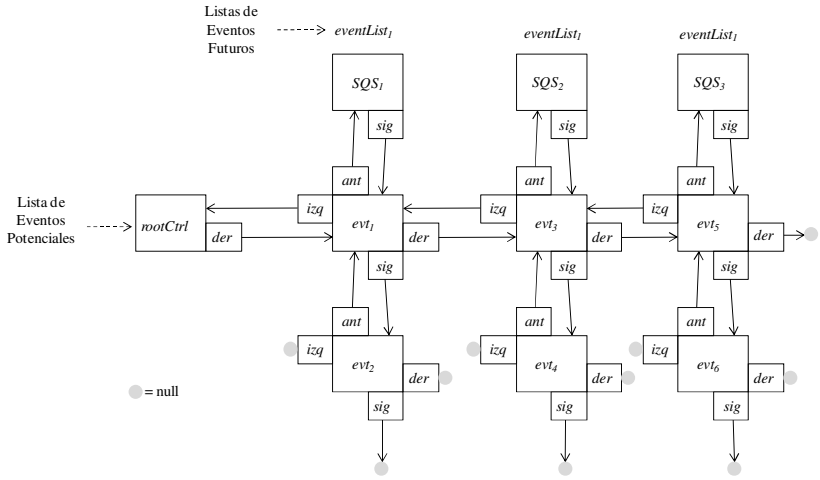

Figure 8. Future event list and potential event list data structure implementations.

Case Study Description

Using the *CiudadelaSim* library a microscopic urban traffic simulation is run. The modeled area consists of 38 streets (one lane) of a vehicular traffic network section (see figure 9). The traffic is regulated by

traffic lights (tf_i) at each intersection. Each tf_i control the vehicle flow from street s_i. The study site is located at the downtown of Guadalajara city, so it experiences heavy congestion even during non-peak periods.

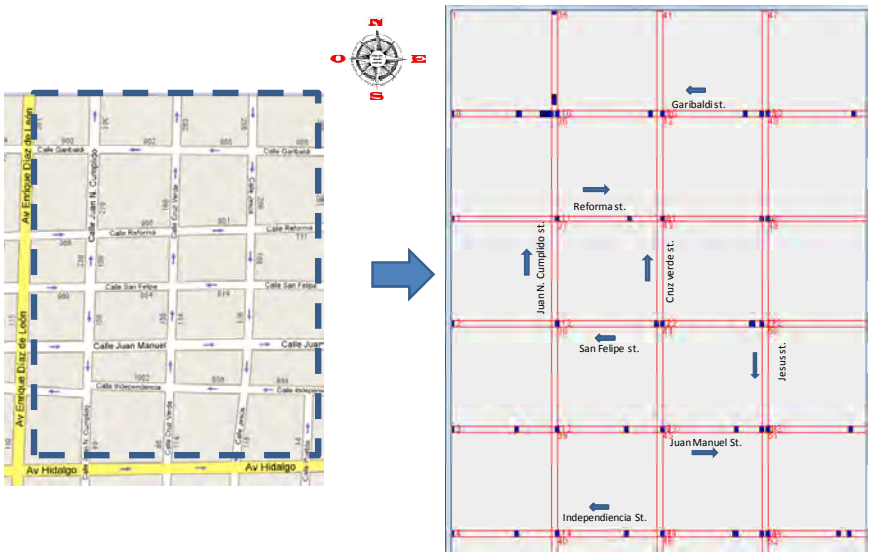

Figure 9. Segment road system of an italian town[]

Traffic Control

The possible vehicle paths at the intersections, subdivided by the phases of the traffic light control system, are shown in figure 10. In the first phase, the traffic lights are green for vehicular flows incoming from sides A and C and red for flows incoming from sides B and D. In the second phase, the lights are green for sides B and D and red otherwise.

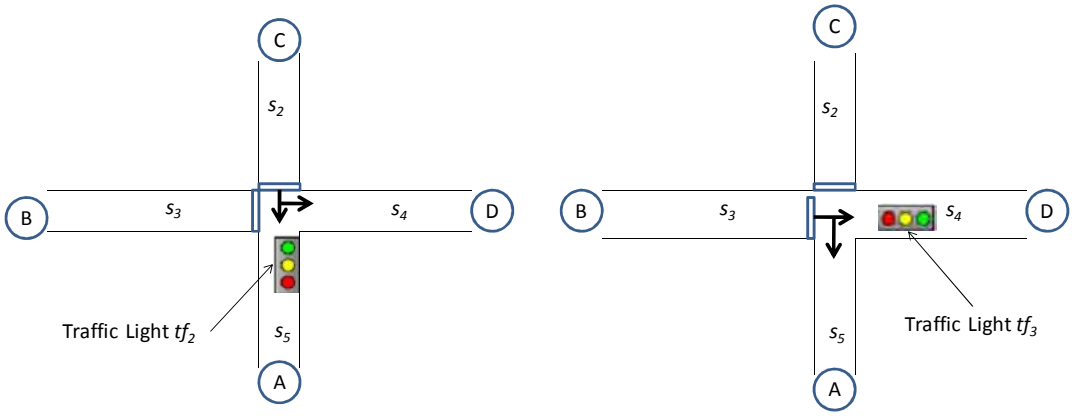

Figure 10. Scheme of vehicular paths of the traffic lights phase one

The figure 11 depicts the four different control strategies used during the simulation run. In the first 600 seconds the strategy A is used; after 600 seconds the strategy B is used; finally strategy C is used.

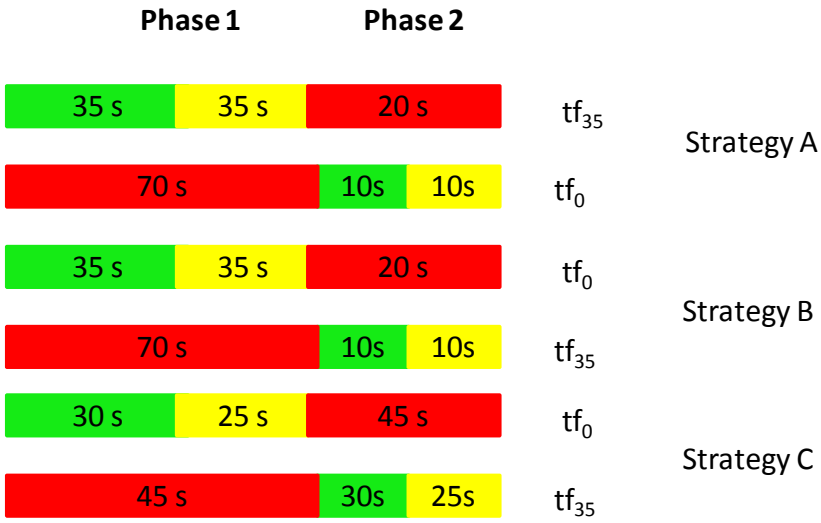

Figure 11. Traffic Light control strategies used in simulation

Vehicle Generation

In order to create new entities as the road users, a list of source segments are specified in the population generation system. In the case study the segments 0, 1, 3, 35, 32, 34, 46 y 47 are source segments. The model used in this research is a *Poisson* distribution, but could be substituted by real demographic data. The Poisson distribution is noted by the formula:

$$P(n)=((\lambda t)^n e^{-\lambda t}) / n!$$

where:

$P(n)$ is the probability of exactly n vehicles arrive at time t
λ is the average arrival rate (*veh/ min*)
t is the duration of time over which the vehicles are counted

Output data model

CiudadelaSim generates the file *log.dat*. This file contains the event execution of the simulation. In Table 1 is shown the format used. The first field (*objectID*) contains the unique number identification for each vehicle; *evtime* field contains the event execution time; the *segment* field contains the segment id where the event was executed; the *evPos* field contains the segment position where the event was executed and finally the *evtype* field contains the event type that was executed using the next symbology 1=**SE** (stop Event),2=**CE** (Cross Event), 3 = **LCE** (Light Change Event), 5 = **CLE** (Change Lane Event), 6 = **CLEE** (Change Lane End Event), 7 = **ALE** (Arrival Link Event), 8 = **LLE** (Leave Link Event), 9 = **BE** (Begin Event), 10 = **AE** (Arrival Event), 11 = **WSE** (Warning Event for stop), 12 = **WBE** (warning Event for start). Figure 1 depicts the example of use.

Results Analysis

The Analyzer class generates the field flowdensity.dat as shown in figure 1. This file is generated reading the file generated by the simulation, each time an event type equal to 7 is read then increments the *numberOutVehicles* variable and when the event type is equal to 8 then the *numberInputVehicles* variable is incremented. Then is calculated the density (space of a segment equal to 100 mts used on an instant time) and flow in each segment using the next equations:

flow = numberOutputVehicles / numberInputVehicles

density = numberInputVehicles – numberOutputVehicles

The calculated values for each second are stored in the file *flowdensity*.dat as shown in table 1.

Table 1: FlowDensity.dat sample file format.

Density Value	Flow Value
11.0	0.35294117647058826
11.0	0.35294117647058826
12.0	0.3333333333333333
11.0	0.3888888888888889
11.0	0.3888888888888889
11.0	0.3888888888888889
11.0	0.3888888888888889

Using the calculated values of table 1, then is obtained a plot with axis X the density values and Y axis the flow values.

Control strategies change

The simulator allows to define different control strategies (see figure 11). In figure 12 is shown the segments st_0 and st_{35} density-flow relationship. The st_{35} maintains high flow levels (and respectively low density) than st_0. This behavior is generated when the strategy A is used, since the stop time is greater than green time for st_{35}. That increases the saturation of st_0. Although there is a change strategy, 600 seconds after , there is no enough time to reduce density of st_0, then continue with high density values. The change of phases provokes the instability of the diagram. The *x* axis shows the density and *y* axis the flow. Observe how the fundamental diagram of flow-density is conserved.

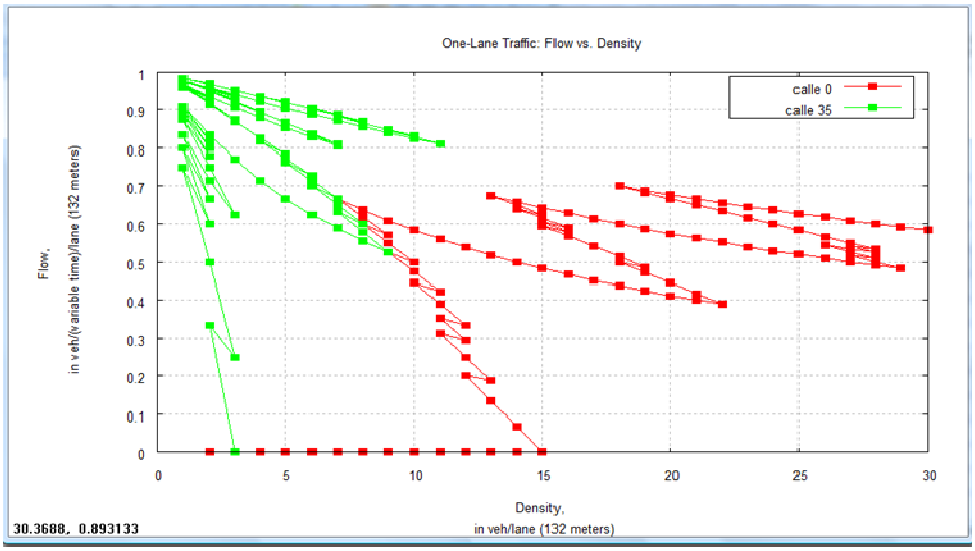

Figure 12. Flow-Density relationship diagram for st_0 and st_{35} segments

Individual parameters change

The *carGenerator* class allows to generate vehicles with different attributes or parameters such as: preferred velocity, safe distance, perception-reaction time, etc. In figure 13 is shown the density of segment st_2 using distinct perception-reaction parameters. In figure 1a the perception-reaction time parameter is equal to 0.2 min. The observed density increment is caused by the queue of vehicles when the traffic light turns red. When the traffic light turns green the density decrease and only some fluctuations appear in the density value. In figure 1b is used a perception-reaction time equal to 8 mins.

Although the traffic light changes to green, the vehicle will delay 8 min. before start. This will convert the vehicle in an obstacle. Then the segment will increase exponentially their density.

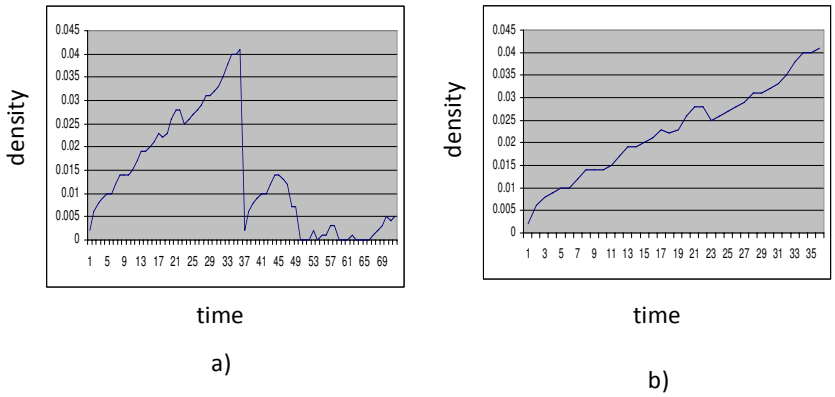

Figure 13. Density graphics for different perception-reaction times a) 0.2 minutes b) 8 min.

Conclusions

In this paper a hierarchical modeling framework for the simulation of urban traffic systems validation is presented. Simulation is a powerful tool for traffic managers that allow them to study and evaluate many traffic control strategies in order to implement the best one. Thus, is proposed a modeling framework that allows capturing systematically both the urban traffic network and the users' behavior. The system model is a modular specification that provides the knowledge used by a micro-simulation engine based on a multi agent approach in which the vehicles are represented individually by mobile agents. A UTS description contains several formal models expressed in a three level Petri net formalism allowing selecting the microscopic desired level of road user behavior and verifying the correct functioning of the desired behavior before the implementation. The model is validated implementing *CiudadelaSim* library. *CiudadelaSim* is a java application that implements the n-LNS UTS model components. Future research includes the distribution of the simulation kernel.

References

[1] E. Lopez-Neri, E. Lopez-Mellado, and A. Ramirez-Treviño, "Microscopic Modeling Framework for Urban Traffic Systems Simulation," in *the 7th International conference on system simulation and scientific computing*, Beijing, China, 2008.

[2] J. Barceló, "Microscopic traffic simulation: A tool for the analysis and assessment of ITS systems," 2001.

[3] J. Barceló, J. L. Ferrer, and D. García, "Microscopic traffic simulation for ATT system analysis. A parallel computing version," University of Montreal Contribution to the 25th Aniversary of CRT, 1998.

[4] W. J. Barclay, "Point-to-Point microscopic simulation: a discussion of issues," in *Proceedings of the First Western Pacific and Third Australia-Japan Workshop on stochastic models*, Christchurch, New Zealand, 1999.

[5] K. S. Perumalla, "A Systems Approach to Scalable Transportation Network Modeling," in *proceedings of the 2006 Winter Simulation Conference*, vol. 0, 2006, pp. 1500-1507.

[6] E. Lopez-Neri, E. Lopez-Mellado, and A. Ramirez-Treviño, "Un lenguaje para la descripción de la información geográfica de sistemas de tráfico urbano," in *IV Semana Nacional de Ingeniería Electrónica* , Aguascalientes,Ags., 2008.

[7] F. Baude, et al., "Grid Computing: Software Environments and Tools," in *Programming , Deploying, Composing, for the Grid*, J. C. Cunha and O. F. Rana, Eds. Springer-Verlag, 2006.