A HLA-based Multi-Resolution Approach to Simulating Electric Vehicles in Simulink and SUMO

Jose Macedo, Zafeiris Kokkinogenis, Guilherme Soares, Deborah Perrotta
and Rosaldo J. F. Rossetti, *Member, IEEE*

***Abstract*— Traffic congestion affects not only the economic activity of cities but is also responsible for air quality and global warming problems. The employment of electric buses in metropolitan transportation as an alternative to internal combustion engine buses appears to be a promising eco-sustainable solution. However, an important aspect in evaluating the performance and adequateness of such vehicles is the fact they are immersed in an urban context. This work follows a HLA-based distributed architecture for electric bus powertrain simulation within a realistic urban mobility environment. In that sense, it discusses on coupling the SUMO (Simulation of Urban MObility) microscopic traffic simulator with a model of an electric bus powertrain subsystem designed in the MatLab/Simulink environment. Both the electric bus traction subsystems and the integration have been validated using field test experimental data. The proposed simulation framework is multi-faceted. As a matter of fact it can be used, not only as electric vehicle evaluation tool under different dynamics of traffic conditions and driving behavior, but also as a planner for charging strategies and optimizations.**

I. INTRODUCTION

The last decades have witnessed a large increase in traffic and transport demand that has created and aggravated capacity problems in the infrastructure causing traffic congestions and delays. Problems in the traffic system have a large impact on almost all areas of economic activities since the flow of people and goods between cities is directly related to the road network [1].

Furthermore, traffic congestion affects not only the welfare of citizens from the economic point of view but is also related to their health status both psychologically, due to stress accumulated during their travels, and physically due to high air pollution levels. In fact, a problem associated with the increasing use of personal vehicles is the emissions. According to the 2009 Urban Mobility Report [10] the congestion led urban Americans to travel 4.2 billion hours more, which resulted in 2.8 billion gallons of extra fuel, an increase of more than 50% over the previous decade. The green house effect, also known as global warming, is a serious issue that we have to face. There have been increased tensions in part of the world due to the energy crisis. Government agencies and organizations try to develop more stringent standards for the fuel consumption and gas emissions through reduction of the congestion on network infrastructures [11].

There is, of course, a diversity of different solutions trying to tackle congestion problems. As congestion begins to occur when the amount of traffic on a road network is approaching its maximum capacity, and the most obvious solution is to increase the network capacity [13]. This can be done in several ways, such as building new roads, extending the existing ones and adjusting the speed limit of roads to increase their capacity. However, creating new roads or adding additional capacity to the existing ones can be expensive, time consuming, can cause environmental and social impacts and is not guaranteed that it solves the problem, as demonstrated by the Braess' paradox [3]. Thus, the increase in traffic volumes combined with often short distances between intersections requires the adoption of a systems analysis approach to properly address traffic congestion. Often, traffic congestion is not the result of excessive traffic, but the result of overlapping bottleneck locations. The spillover effect of traffic congestion from one location to another makes conventional engineering methods inefficient [8].

In this sense, incentives and investments on public transports as well as research on more eco-sustainable solutions have been performed as attempts to minimize both the air pollution and congestion problems. One of the approaches currently investigated and implemented to provide an eco-sustainable solution for the public transport is related to the employment of electric bus powertrains in metropolitan transportation as an alternative to internal combustion engine buses [12]. However, there are still open issues related to the consumption of energy and other performance measures for considering the adoption of electric buses in urban scenarios as a cost-effective solution.

An important aspect of evaluating the performance and adequateness of such vehicles is the fact they are to be immersed in an urban environment context. That is, a route having many positive elevations or a traffic congestion situation will directly affect the autonomy and performance of the vehicle. Albeit there are different tools and models to assess the behaviour of electric buses, such evaluations often lack the aforementioned integration with the traffic dynamics. In [7] a first step towards the integration of microscopic traffic simulation with electric vehicle operations has been presented. In this context, authors have extended the "class vehicle" in the SUMO simulator by a build-in implementation of an electric vehicle model.

J. Macedo, Z. Kokkinogenis, G. Soares and R.J.F. Rossetti are with the Artificial Intelligence and Computer Science Lab, Department of Informatics Engineering, Faculty of Engineering, University of Porto, Portugal.

D. Perrotta is with the Artificial Intelligence and Computer Science Lab, University of Porto, and Centro Algoritmi, University of Minho, Portugal.

E-mails: {jose.macedo, pro08017, guilherme.soares, dli10003, rossetti}@fe.up.pt.

This paper wants to propose a flexible approach to analyse the real system by integrating different aspects of it, namely a microscopic traffic model with a nanoscopic representation of an electric vehicle. Thus the paper presents and discusses on a distributed architecture for the multi-resolution simulation of electric bus powertrain system within a realistic urban mobility context. More specifically, it explains the concepts and the potential of using the High Level Architecture (HLA) to interconnect different simulation systems: the Matlab/Simulink model of an electric bus subsystem (to simulate the consumptions of an electric vehicle) and SUMO [2], a microscopic traffic simulator (to simulate the urban traffic conditions under which the electric vehicle should be evaluated).

II. HLA Preliminaries

The High Level Architecture addresses the reuse and interoperation of legacy model simulations. The HLA concept is based on the idea of the distributed simulation approach that no single simulation model can satisfy the requirements of all usages and users. In order to facilitate interoperability and reusability, HLA differentiates between the simulation functionality provided by the members of the distributed simulation and a set of basic services for data exchange, communication and synchronization.

A. Architecture and Components

In HLA, every participating application is called a *federate*, and these entities can interact with each other within a *federation*. A federation can be seen as a set of federates acting together in a distributed simulation to achieve a certain objective. There are three main components that comprise HLA:

- Federate Interface Specification
- Framework and Rules
- Object Model Template Specification

The HLA Framework defines a set of rules that must be obeyed to ensure the proper interaction of federates within a federation. These rules must be unchanged across all the simulation units as they define the overall architecture. They also define the responsibilities of federates and federation. There are five rules for federates and other five for federations. The definition and description of each rule is available in [5].

The HLA Federate Interface Specification describes the services which federates have to use for communicating with others. This communication is always made through a middle-ware structure, known as Run-Time Infrastructure, which provides the essential building ground for the software developers. The interface specification describes which services a federate can use and which services it has to provide [4]. In order to establish the interaction between federates and the Run-Time Infrastructure (RTI), the concept of ambassador is used. Ambassadors are objects that have the methods needed by the participants for performing communication. So, federates communicate with the RTI using its ambassador as an interface. Figure 1 illustrates the described concepts.

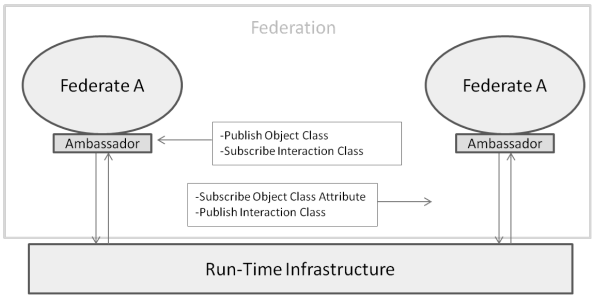

Fig. 1. HLA's Functional Architecture.

The HLA Object Model Template Specification describes the format and syntax of the data transferred between federates. This data exchange is represented in the form of an object class and the two types of object exchange are Object Class and Interaction Class. The first one contains the shared information within federation that persists during the run time. The second one, contains the sent and received information between federates. This component defines the object template data that all simulation unit needs to use in order to exchange data with each other [6].

III. Simulink Model Overview

The Electric Bus Powertrain Subsystem (EBPS) is a mathematical model of an electric bus engine model implemented in MATLAB Simulink [9]. The model is constituted by several modules that are used to compute specific parameters. One of these subsystems represents the vehicle's powertrain, taking into account the forces that work against its movement and the gear ratios involved. An output of this subsystem computes the amount of required energy for a driving cycle to be completed. A second subsystem is implemented that calculates the amount of energy that may be possibly recovered from the regenerative braking, taking into account the kinetic energy of the vehicle. Two other subsystems are related to the batteries and the super-capacitors, evaluating whether they are capable of absorbing the energy from the braking [9]. Figure 2 illustrates the main subsystem of the model.

This system is modelled in continuous time. It receives a vectorial structure as input, where velocities are related to time instants. As outputs, the model produces a structure for each metric, with values associated with an instance of time. The most significant parameters calculated by this model are:

- **Power:** Expresses the power required for each instant of the driving cycle.
- **Total Energy Cycle:** The energy required to complete a whole cycle.
- **Kinetic energy:** It is known that the effect of the vehicle mass when accelerating and stopping the vehicle in urban conditions has considerable influence on vehicle performance. So, the kinetic energy is used in this model for calculating the amount of energy dissipated

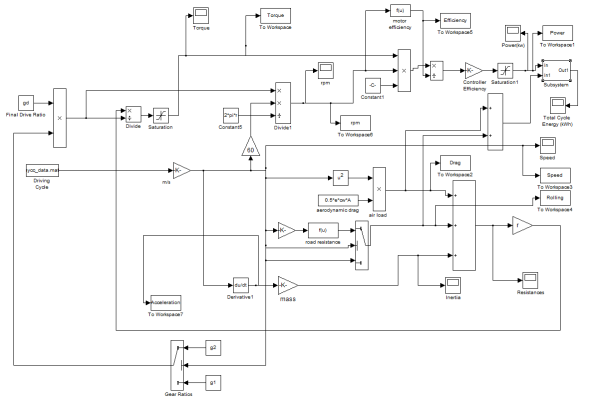

Fig. 2. Main subsystem of the EBPS model.

in braking, considering the tires and air resistance.

In order to be used in the intended integration, this model was modified to a discrete model. In this way, it could receive a unique variable and produce output values for that specific instance.

IV. SUMO-SIMULINK: An HLA Integration

In HLA, each federate application is an independent application. In this sense, each federate is carefully designed and developed. Fig. 3 illustrates the main federate components and interactions.

For the development of the federate applications, two aspects had to be taken into account: one is the communication module with the simulators, the other is the federate ambassador module for communications with the RTI.

The *federate ambassador* is the module through which all the communication with the RTI is performed. This module must comprise the standard methods required for the communication with the RTI. Some of this methods are used by the RTI to perform call-backs to the federate, others are the methods used by the federate to perform calls to the RTI. The latter use a local RTI component in order to perform the calls which is a local library comprising the RTI methods.

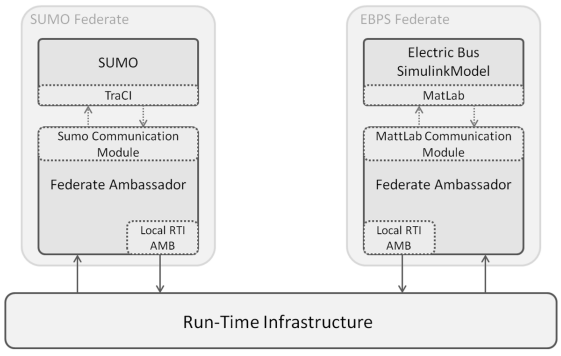

Fig. 3. HLA Implementation Architecture.

In order to establish communication with the RTI the federate must need to perform a connection to it. To do that, it is used an object called RTI ambassador that is created using an *RTIambassadorFactory* provided by the local RTI library. This object will be used to perform all the calls to the RTI.

After the connection, the federates needs to join or create a specific federation. These methods as well as *resignFederationExecution*, *destroyFederationExecution* and *disconnect*, comprise the main activity for connection to and disconnection from a federation. The methods for the data exchange within federation needed to be implemented too. The HLA standards provide an extensive number of methods for different purposes. However, implementing them all would be very time consuming. Thus, only the necessary methods for the intended integration were developed.

There are two different groups of methods that are related to the type of data exchange. The first group is directed to the interaction classes and the other, to the object classes. But before exchanging data between federates, the Federation Object Model needed to be specified and the communication module with the simulators needed to be created.

A. Federation Object Model (FOM) Specification

A coherent and persistent integration among different models is obtained using a well-defined protocol where the data exchange and attribute ownership among the federates are declared.

For this purpose HLA standards define the FOM entity where a description of the data exchange in the federation (i.e. the objects and interactions that will be exchanged) is described. This can be seen as the language of the federation. Each federation defines its own FOM, since a federation is defined for a given purpose each time.

During the creation of the FOM three of the most important issues to be considered are the Object classes, the Interaction classes, and the Data types. These entities have to be defined well in order to describe the information exchanged between the federates of a federation.

For the intended purpose of the proposed integration between SUMO and EPBS, all data necessary to be exchanged are related to the electric bus attributes. In this sense, the object class that needs to be described in FOM is the class Bus declared for the federation. Moreover, to define the class object, all the attributes and its variable types have to be defined too.

This class has twelve attributes each one with a specific type. The attribute *Name* was included to identify the bus instance in case there is more than one bus. The velocity attribute is the variable that the traffic simulator will update and the other attributes are the variables for the outputs of the EBPS model.

Regarding the attributes types, the *HLAunicodeString* is already a default HLA type. However, other types are not and need to be described in FOM.

To describe a new type, it is necessary to define the default HLA type that it comprises, the units that it represents, the

resolution that it accepts and a simple description about its semantic.

To conclude the FOM description, the interaction classes must be described. An interaction is something that does not persist over time, it is used to perform some immediate reaction. The interaction classes created are the Start and Stop interactions. These interactions are used to start and stop the simulation execution on both simulators. Thus, when federates receive one of these interactions, they start or stop the simulation respectively.

Also, an interaction could have some parameters as arguments, and if it is the case then they need to be specified too. In this case, only the Start interaction comprises an argument. This parameter is used to specify the frame rate at which the simulation should be processed.

B. The MatLab Communication module

MatLab could be seen as an API for Simulink since Simulink models can be controlled by MatLab methods. Simulink models are standalone models and exist in mutual symbiosis with the MatLab environment. The only way to access them externally is through the MatLab methods and calls. For this reason a control using the MatLab interface to Simulink was applied. In this sense, this module is used by the federate ambassador to communicate with Simulink model through MatLab.

In order to establish communication with MatLab, the matlabcontrol API was used. The matlabcontrol is a Java API to interact with MatLab allowing its methods invoke behaviour of Java objects [14]. Using this, it was only necessary to create a proxy to work as an image of the MatLab application. After that, all the calls to MatLab are performed through this proxy.

Establishing the communication with MatLab was the first part in the development of this module. The second part was to control the Simulink models step-by-step simulation through Matlab commands. To do so, the set_param() function was chosen. This function allows starting, pausing, stopping and resuming the simulation, as well as advancing one simulation step at a time.

C. The Sumo Communication Module

Unlike Matlab/Simulink, SUMO already comes with an API that provides a communication protocol. In fact, the TraCI interface provides a set of methods that allow an easy interaction with the simulator's state variables.

This module is composed of a set of functions, implemented in Java that comprises the necessary commands that would be used by the SUMO federate ambassador to interact with SUMO. TraCI has an extensive number of methods, each one associated with an entity in the simulation. For the scope of this project, the only TraCI's methods that were used are relative to the vehicle entity and its speed.

D. Specification of Federates

There are two different groups of methods that are related to the type of data exchange. The first group is directed to the interaction classes and the other, to the object classes.

Related to the interaction classes, a set of methods were used in order to allow one of the federates to interact with the other. The first methods that needed to be used are the *PublishInteractionClass* and *SubscribeInteractionClass*. These methods acknowledge what kind of interaction the federates are able to publish or receive. For example, if a federate will be responsible to instruct the simulation to start, this federate could publish an interaction class named "start" while the others subscribe it. In this case, the SUMO federate will be the responsible to initiate the simulation and to publish an interaction class named "Start". The EBPS federate will need to subscribe that interaction in order to receive it.

The other methods required for interaction are the *ReceiveInteraction* and *SendInteraction*. The SUMO federate needs to call the method *SendInteration* from RTI, whenever it wants to start the simulation. The EBPS federate needs to implement the *ReceiveInteraction* method in order to receive a callback from the RTI and starts the simulation.

In a similar way each federate needs to call "Publish" or "Subscribe" for the *ObjectClass* and *ObjectClassAttributes* that they want to send or receive. Both, SUMO and EBPS federates need to perform "Publish" and "Subscribe" to the Bus class. However, the EBPS only subscribes the attribute velocity and publish all the others. The SUMO federate performs the publish and subscribe attributes of the Bus class in the opposite way. Table I presents the *InteractionClasses*, *ObjectClasses* and *ObjectClasseAttributes* Published and Subscribed by each federate.

TABLE I
PUBLIC AND SUBSCRIBE ENTITIES BY SUMO AND EBPS FEDERATES

Class Type	Identifier	SUMO	EPBS
InteractionClass	Start	P	S
ObjectClass	Bus	PS	PS
ObjectClassAttribute	Velocity	P	S
ObjectClassAttribute	Acceleration	S	P
ObjectClassAttribute	Power	S	P
ObjectClassAttribute	Torque	S	P
ObjectClassAttribute	TotalCycleEnergy	S	P
ObjectClassAttribute	BrakingKinectEnergy	S	P
ObjectClassAttribute	BrakingResistanceEnergy	S	P
ObjectClassAttribute	SuperCapacitorsChargingEnergy	S	P
ObjectClassAttribute	SuperCapacitorsDischargingEnergy	S	P
ObjectClassAttribute	BatteriesChargingEnergy	S	P

Now that the federates informed the RTI of what they Publish and Subscribe, they have to implement the necessary methods for exchanging data during federation execution. These methods are the following:

- registerObjectInstance
- discoverObjectInstance
- updateAttributeValues
- reflectAttributeValues
- attributeOwnershipAcquisition

The *registerObjectInstance* service registers a new object instance of the specified type. It returns a handle to the new instance. This handle is saved so that the federate could update this object later on. In this specific case, the only federate that uses this service is the SUMO federate, being responsible for registering the class Bus in the federation. When a new object is registered by one federate, the RTI

will make sure that it is discovered by other federates that subscribe to the specified class. This is made by the *discoverObjectInstance* method. If a federate joins a federation where there are already a number of objects, the RTI will also make sure the new federate discovers existing instances of a class that it subscribes to.

The *updateAttributeValues* service sends an attribute update for a particular object instance. The update contains a number of attribute/value pairs where the attribute is described by its handle. There are two cases to consider when the federation send updates for attributes: one is when a federate requests a value, the other is when a federate have a new value to update. In this implantation the only way to updates happen is when a federate receives a new value. The *reflectAttributeValues* is called-back by the RTI to the federate that subscribes the attributes that was updated by the other federate.

Finally, the *attributeOwnershipAcquisition* is used by the EBPS federate to request ownership for some attributes. Since it was the SUMO federate to register the class Bus, by default all the attributes were owned by it and the EBPS could not update any. Calling the *attributeOwnershipAcquisition* service to RTI, it can request the attributes that it want to update and gain their ownership.

With all methods implemented, their execution sequence will be performed as illustrated in Figure 4.

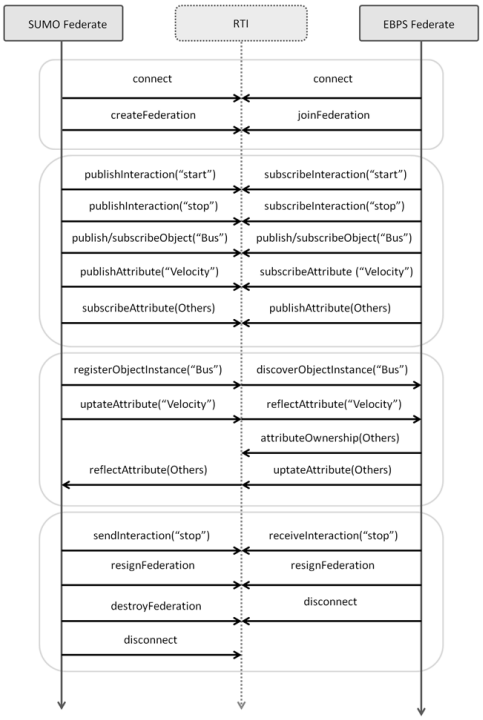

Fig. 4. Interaction diagram of federation executions.

V. SIMULATION SCENARIO

As it has been mentioned previously, the devised integrated platform intends to offer a valid tool to traffic managers and practitioners for analysing how traffic flow and its dynamics affect the performance of electric buses when there are obstructions or intense traffic conditions. In this section we present a scenario where two different traffic flows were specified, so that the EBPS could be exposed to different solicitations due to daily traffic conditions. This illustration serves only as a demonstration of the possible type of analysis one can conduct using the proposed tool.

For this purpose a new road network has been designed and created for SUMO. The network is a model of the central area of Porto down-town (Fig. 5) and it has been extracted from OpenStreetMaps[1] database. The choice of this urban traffic area is motivated by the fact that it is an area of the centre of the city with high traffic densities and a large number of traffic lights. In order to provide an example of what kind of study the framework can perform for the EBPS analysis, the Acceleration and TotalEnergyCycle metrics have been chosen among others. Other metrics are defined in [9].

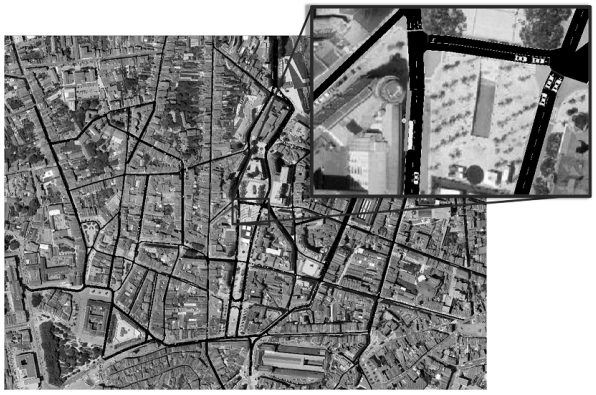

Fig. 5. Aliados network for test-bed experiments.

In order to create the traffic demand, the *randomTrips* application embedded to SUMO package was used to automatically generate the trips for the vehicles. So, to customize the traffic flow and obtain the two different set-ups of traffic volume, the repetition rate of the generated trips for each one was different. A repetition rate of one second means that each trip is repeated by a new vehicle at every simulation step. Considering this, a repetition of five seconds has been used for first set-up experiment while for the second one it was used a repetition trip rate of one second. In the second one, the network rapidly begins to become overloaded, representing rush-hour periods. In each experiment, the bus travelled through the same route. In this way, it was possible to compare the results of the EBPS for the different traffic situations.

The first analyzed parameter was the acceleration profile of the electric bus under both traffic conditions. It can be

[1] http://www.openstreetmap.org/

observed that under intense traffic, there were performed tougher accelerations and decelerations (represented by the peaks in the plot of Fig. 6). Besides that, the episodes were characterized by being performed in smaller periods of time when compared to low traffic. This type of behaviour have direct negative implications on the maintenance costs and on the efficiency of the bus, once the higher values of acceleration impact on the necessary energy for the bus to perform a certain route.

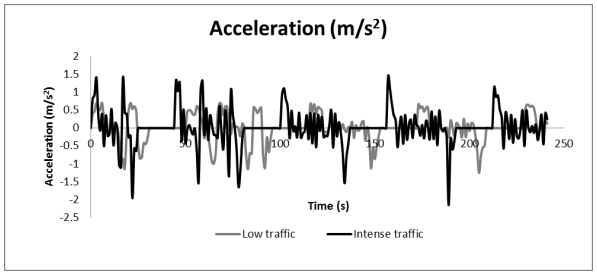

Fig. 6. EBPS acceleration under different traffic conditions.

After observing the acceleration profile, it was decided to analyse the amount of energy that the electric bus spent under the proposed conditions, as a way to corroborate the assumptions made previously. Not surprisingly, under intense traffic, the electric bus spent 36% more energy (0.125 kWh) when compared to the low traffic scenario (0.092 kWh), as it can be observed in Fig. 7. This has immediate consequences on the costs associated with recharging bus batteries. It could be concluded that intense traffic conditions affect negatively the performance of an electric bus, making it spend more energy to perform similar routes and also demanding more power from the bus motor, which implies higher maintenance costs.

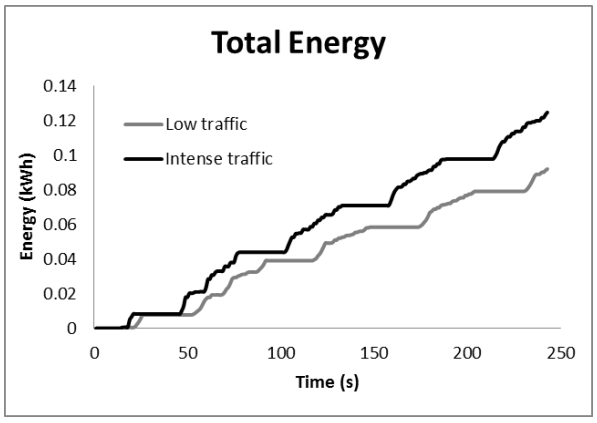

Fig. 7. EBPS energy consumption under different traffic conditions.

VI. Conclusion

This paper discusses on the implementation of a HLA-based distributed architecture for electric bus powertrain simulation in urban mobility settings using two types of simulation models, namely the SUMO (Simulation of Urban Mobility) microscopic traffic simulator and the MatLab/Simulink environment. The proposed architecture envisions a flexible approach for coupling two simulators, one from the transportation area, and another from the automotive area. Our motivation is to offer a valid tool for traffic managers and practitioners so they can analyse how traffic flow and its dynamics affect the performance of electric buses (as well as other type of electric vehicles) when there are obstructions or intense traffic conditions. To illustrate the usefulness of the platform architecture a driving scenario was presented, as well as were discussed preliminary results obtained by the simulation. The advent of promising technologies will increase the need for flexible R&D tools for such complex systems to be evaluated. Furthermore, there is an encouraging potential on the proposed approach to foster synergies and to bring together the automotive and transportation research communities to work together on the development of the Future Urban Transport Systems.

ACKNOWLEDGMENT

This project has been partially supported by FCT (Fundação para a Ciência e a Tecnologia), the Portuguese Agency for R&D, under PhD Scholarship grants SFRH/BD/67202/2009 and SFRH/BD/51256/2010.

References

[1] R Arnott, T Rave, and R Schöb. Alleviating urban traffic congestion. *MIT Press Books*, 1, 2007.

[2] M Behrisch, L Bieker, J Erdmann, and D Krajzewicz. SUMO - Simulation of Urban MObility: An Overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain, 2011.

[3] D Braess. Über ein Paradoxon aus der Verkehrsplanung. *Mathematical Methods of Operations Research*, 12(1):258–268, 1968.

[4] IEEE Std 1, editor. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federati Interface Specification*. IEEE Computer Society, 2010.

[5] IEEE std 2, editor. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules*. IEEE Computer Society, 2010.

[6] IEEE std 3, editor. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template Specification*. IEEE Computer Society, 2010.

[7] R Maia, M Silva, R Araujo, and U Nunes. Electric vehicle simulator for energy consumption studies in electric mobility systems. In *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, pages 227–232. IEEE, 2011.

[8] M D Meyer and E J Miller. *Urban transportation planning: A decision-oriented approach*. 2001.

[9] Deborah Perrotta, Bernardo Ribeiro, Rosaldo J F Rossetti, and João L. Afonso. On the potential of regenerative braking of electric buses as a function of their itinerary. *Euro Working Group On Transportation*, 2012.

[10] David Schrank and Tim Lomax. The 2009 URBAN MOBILITY REPORT. Technical report, Texas Transportation Institute, The Texas A&M University System, 2009.

[11] Benjamin K Sovacool. A transition to plug-in hybrid electric vehicles (PHEVs): why public health professionals must care. *Journal of epidemiology and community health*, 64(3):185–7, March 2010.

[12] N Unger, T C Bond, J S Wang, D M Koch, S Menon, D T Shindell, and S Bauer. Attribution of climate forcing to economic sectors. *Proceedings of the National Academy of Sciences*, 107(8):3382–3387, 2010.

[13] X Zhang and H Yang. The optimal cordon-based network congestion pricing problem. *Transportation Research Part B: Methodological*, 38(6):517–537, 2004.