From Formal Modeling to Discrete Event Simulation: Application to the Design and Evaluation of a Routing Protocol for Vehicular Ad Hoc Networks

Emna Chebbi
Univ. Littoral Côte d'Opale
LISIC - EA 4491
F-62228 Calais, France
Email: chebbi@univ-littoral.fr

Patrick Sondi
Univ. Littoral Côte d'Opale
LISIC - EA 4491
F-62228 Calais, France
Email: sondi@univ-littoral.fr

Eric Ramat
Univ. Littoral Côte d'Opale
LISIC - EA 4491
F-62228 Calais, France
Email: ramat@univ-littoral.fr

***Abstract*—Simulation studies on ITS-dedicated routing protocols usually focus on their performance in specific scenarios. However, the evolution of transportation systems towards autonomous vehicles requires robust protocols with proven or at least guaranteed properties. Though formal approaches provide powerful tools for system design, they cannot be used for every types of ITS components. Our goal is to develop new tools combining formal tools such as Event-B with DEVS-based (Discrete Event System Specification) virtual laboratories in order to design the models of ITS components which simulation would allow proving and verifying their properties in large-scale scenarios. This work present a methodology to increase the amount of proven properties on ITS-components. In this paper we describe how the methodology can apply to the study of a routing protocol. We describe how both the Event-B and DEVS models of the routing protocol are implemented and validated.**

Keywords–Routing protocol; Vehicular Networks; Formal Modeling; Discrete Event Simulation; Intelligent Transport Systems.

I. INTRODUCTION

Wireless communication technology plays a key role in the development of Intelligent Transportation Systems (ITS). Early deployed in the European Rail Traffic Management System (ERTMS), the Global System for Mobile communications Railway (GSM-R) allows a continuous location and movement management of the trains. However, before GSM-R was adopted in ERTMS, it had to fulfill several specific requirements regarding notably the control-command processes, materialized through the European Train Control System (ETCS) applications, and the security mechanisms, achieved through the Euroradio protocol. While formal methods have been widely used in order to prove the correctness of ETCS applications, the evaluations regarding the GSM-R have been performed essentially by simulation and real-world testing based on key performance indicators.

The same trend is now observed in the evaluation of wireless technologies for vehicular networks (VANET), where the evaluations regarding the wireless technology are mostly conducted through simulation and testings, considering mainly performance issues instead of proven properties. The convergence of the main network architectures to the all-IP (Internet Protocol) is pushing both railway operators and car manufacturers to evolve from dedicated infrastructures to a global network connecting all the communicating objects in the smart city. The Internet protocols, initially designed for best-effort applications, are now confronted to the requirements of application domains that are traditionally more sensitive such as tactical units, e-health, and intelligent transport systems. Given the variety of the requirements that could be imposed by such applications, rapid and efficient tools for validating and evaluating custom domain-specific protocols are suitable at the earliest stages of their design.

Based on the formal models of the custom protocols designed on top of IP for managing the communications in a ITS, the research work presented in this paper aims at developing a methodology for obtaining through simulation, not only performance indicators, but also additional formal proofs of some properties attached to both the designed protocols and the entire transport system itself. To that end, a methodology combining formal methods and discrete event simulation has been introduced in [1]. Though the approach itself can be generalized to other applications, this research work will focus on the design and the evaluation of a routing protocol for ad hoc communications between the vehicles of an ITS.

In this paper, we present the formal models of a vehicular ad hoc network routing protocol realized with an Event-B based tool, namely Rodin, and the related DEVS-based models implemented in the Virtual Laboratory Environment (VLE). The rest of this paper is organized as follows. After a brief review of the literature in Section II, the proposed approach is described and discussed in Section III. The models currently developed are explained in Section IV. The validation of these models is then described in Section V.

II. RELATED WORK

Several alternative approaches can be adopted while designing and evaluating the protocols for ad hoc communications in vehicular networks. The following review focus on the main trends observed in the literature for the protocols that manage only vehicle-to-vehicle communications [2].

A. On the design of ad hoc routing protocols

Most of ad hoc routing protocols for VANET identified in the literature are inspired from those standardized for mobile ad hoc networks (MANET), and they can be grouped into a limited number of approaches as follows:

- The reactive protocols (DSR, AODV, etc.) that compute the routes on-demand [3]. They do not maintain periodically information on the network topology. As a

result, they generate less routing traffic, thus preserving the bandwidth for useful applications. However, since the route computation starts only when the traffic has to be sent, this often leads to higher delays, which is not suitable for real-time applications and high mobility environment such as vehicular ad hoc networks.

- Another family of protocols identified as proactive (OLSR, etc.) permanently maintains a structure in the network topology. These protocols compute periodically the routes ready to use when traffic arrives for one of the known destinations [4]. Though they are convenient for highly dynamic environments, these protocols are more prone to routing traffic overhead when the network is dense. Thus, they generally implement a clustering scheme in order to reduce the effects of broadcast transmissions, including that related to routing traffic, and avoid congestion.
- The last approach developed concerns the so called geographical protocols (GRP, etc.). These protocols use information on the geographic location of the vehicles in order to organize the routing of the data [5]. However, they assume the existence of a centralized location service which provides to each vehicle the positions of all the other vehicles. A such assumption do not cope with pure ad hoc networks where no infrastructure should be needed to ensure network management functionalities. They are more likely to serve in infrastructure-based or mixed vehicular networks.

B. On simulation based routing protocol evaluation

Most of the evaluations performed on ad hoc routing protocols focus on measuring the performance obtained based on a set of metrics in specific scenarios. The issues mainly addressed in these studies are the following:

- In particular conditions of density, mobility and given a communication technology, what performance profile can the protocol guarantee to the various vehicular network applications?
- For a set of possible configurations of the vehicular network, does the protocol allow all the time to satisfy a set of requirements according to a set of metrics (delay, bandwidth, loss rate, etc.)?
- By varying both network configurations and the application requirements, how both the protocol behavior and the application performance vary?

The results obtained through this type of evaluations generally lead to a qualitative and quantitative appreciation on the performance of the protocol. However, they are not performed to obtain the proofs on the protocol intrinsic properties. One of the studied properties concerning ad hoc network protocols is the convergence of the protocol, otherwise its ability to complete its tasks and deliver a stable network structure in a given time. Another property is the robustness which reflects its ability to rebuild its structure and maintain its functionalities despite the changes in the network. A third property studied is its scalability (in terms of vehicle density or traffic load). We are particularly interested by the properties related to quality of service [6] or security [7]. These latter are the most crucial for guaranteeing an increased interest for ad hoc communications in future and effective vehicular ad hoc networks.

C. On the contribution of formal approaches

The adoption of wireless technologies in transportation systems has been concerned with formal methods in their earlier stages, since most of these systems impose stringent safety requirements. Particularly, many formal approaches have been proposed for the analysis of the routing protocols for vehicular ad hoc networks. Other authors [8] propose a methodology for verifying the properties related to security in network protocols[9]. Singh et al. [10] present a formal model of AODV with Event-B. Another study[11] concerns a similar model of the DSR protocol in order to prove its properties related to security. Finally, Kamali et al. [12] describe a set of Event-B refinements of a formal model of the OLSR protocol. Two problems persist with formal approaches in network protocol design for intelligent transport systems, and suggest an approach combining formal methods with discrete event simulation:

- The first concerns the joint evaluation of components which are prone to formal modeling with the other components of the ITS which are not adapted to such approach. The tools such as Event-B can only evaluate formal models, while the simulation tools based on multi-modeling allow connecting heterogeneous models and devices in a single simulation-based evaluation process;
- The second problem is related to the scalability of the formal tools. Though they provide some extensions that can animate formal models, they do not support large scale animation of a great number of interacting objects, which would be mandatory in order to verify the behavior of systems such as ITS and VANETS.

A first solution comes from Yacoub et al. [13]. They propose to integrate the DEVS formalism mechanisms into a formal modeling tool. This approach partially solves the problems related to scalability of the formal tool by reducing the search space through simulation between two applications of the formal prover. However, it does not solve the problem of protocol evaluation scenarios involving interactions with non-formal models. We propose to investigate the other solution consisting in the integration of formal models into a larger DEVS simulation.

III. From formal modeling to DEVS

A. General idea

The proposed approach consists in developing a methodology in order to integrate formal models in DEVS-based multi-modeling. As shown in Figure 1, this methodology operates in two phases:

- The first phase consists in modeling some components using formal tools such as Event-B in order to obtain a set of proofs using the automatic prover, and a set of proof obligations that necessitate an interactive proving process involving an expert (denoted here by RPO for Residual Proof Obligations);
- In the second phase, the formal models, the proven properties and the RPOs are transferred into a DEVS

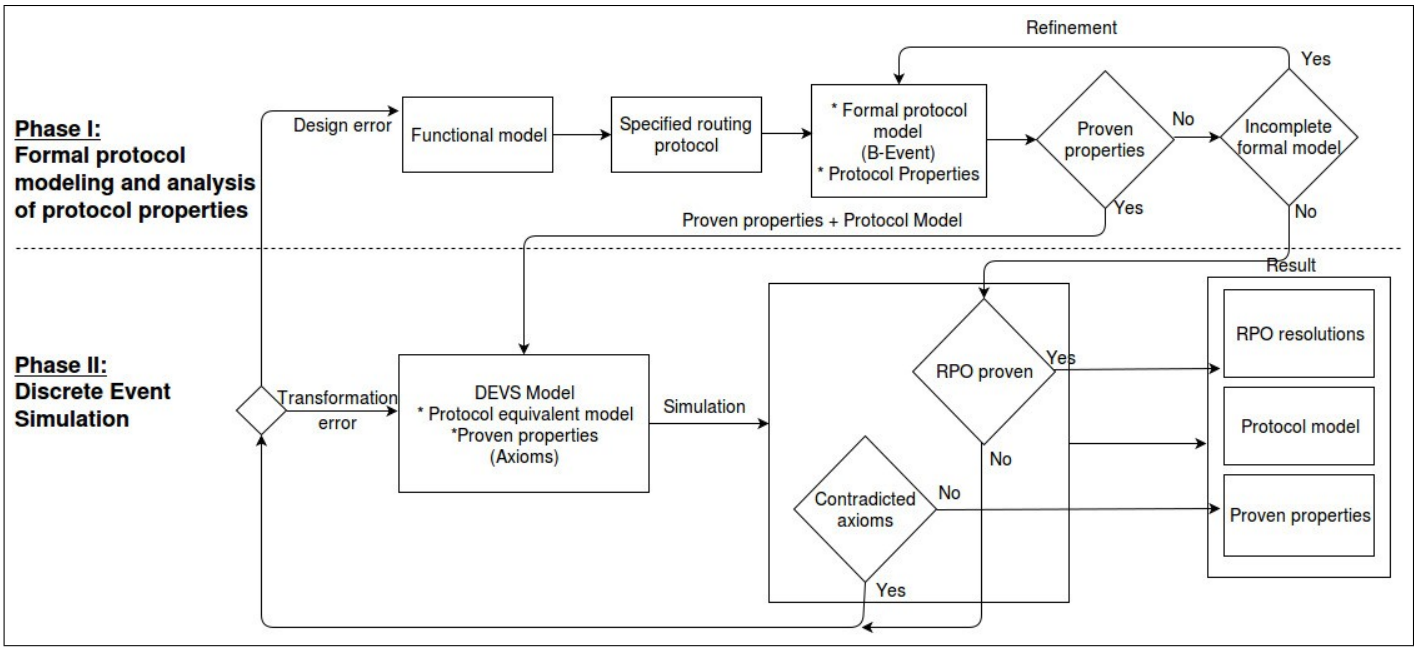

Figure 1. Main steps of a methodology for enhancing the proofs on the properties of an ITS component using Event-B and DEVS simulation. [1]

multi-modeling, which integrates the models of the components that do not fit to formal modeling. Then a simulator generated from the multi-modeling allows evaluating the entire system through a discrete event simulation.

This approach will allow obtaining proven properties through formal tools. It will also address the issue of interacting with the components that are not prone to formal modeling, and that of large-scale scenarios. Moreover, it will allow detecting design errors in formal models when the simulation results are in contradiction with some theorems. Finally, it may help increasing the number of proven properties if the simulation results bring new data that allow solving the RPOs. However, the implementation of this approach raises some issues that need to be addressed:

- Building automatically a DEVS representation of the models, the related proven properties and the RPOs obtained using the formal tool (Event-B). This issue raises itself several implementation problems.
- Designing the DEVS multi-modeling in such a way that the simulation results allow verifying the properties that were proven in the formal tool, and also producing data that could be used in an interactive proving process.

B. Application to VANET evaluation

In this work, we are extending a virtual laboratory based on multi-modeling in order to simulate the communication systems dedicated to transport, especially a routing protocol dedicated to ad hoc communications between the vehicles (Figure 2). The goal is to design a formal model of the components that support formal specification (e.g., the routing protocol). In this way, it is possible to verify and prove some of its properties by resorting to formal methods and tools such Event-B. The formal model of the routing protocol and its proven properties can then be integrated into a larger simulation, by the means of multi-modeling. Practically, the formal models (from Event-B) can be transformed into DEVS models (Discrete Event System Specification), and connected with the models of the other components of the transport system. In this way, it will be easier to integrate real-world data into DEVS simulation and to manage the interactions with other specialized simulators for the different components (e.g., MATLAB for propagation models, OPNET or NS3 for communications, SUMO for mobility models, etc). The goal is to achieve realistic evaluations of the entire vehicular ad hoc network system.

IV. Application to Event-B and DEVS

Following the steps described in Figure 3, we will show how the proposed approach can be developed from the functional model of the protocol up to the corresponding models respectively in a formal tool such as Event-B and a DEVS multi-modeling tool such as VLE.

A. Functional description of CBL

CBL [14] is a completely distributed algorithm: each communication node initiates its own process. It creates a hierarchy between the nodes in order to build 1-hop clusters so that each node of a cluster can directly communicate to the cluster-head without going through another intermediary node. Some definitions are specified as follows (Figure 4):

- A branch node is a cluster-head node which is elected by other nodes (branch or leaf). It emits HELLO messages like every node, but it is the only allowed to emit topology control messages (TC), to forward application messages, and to participate in the construction of a chain. In order to control the propagation of a

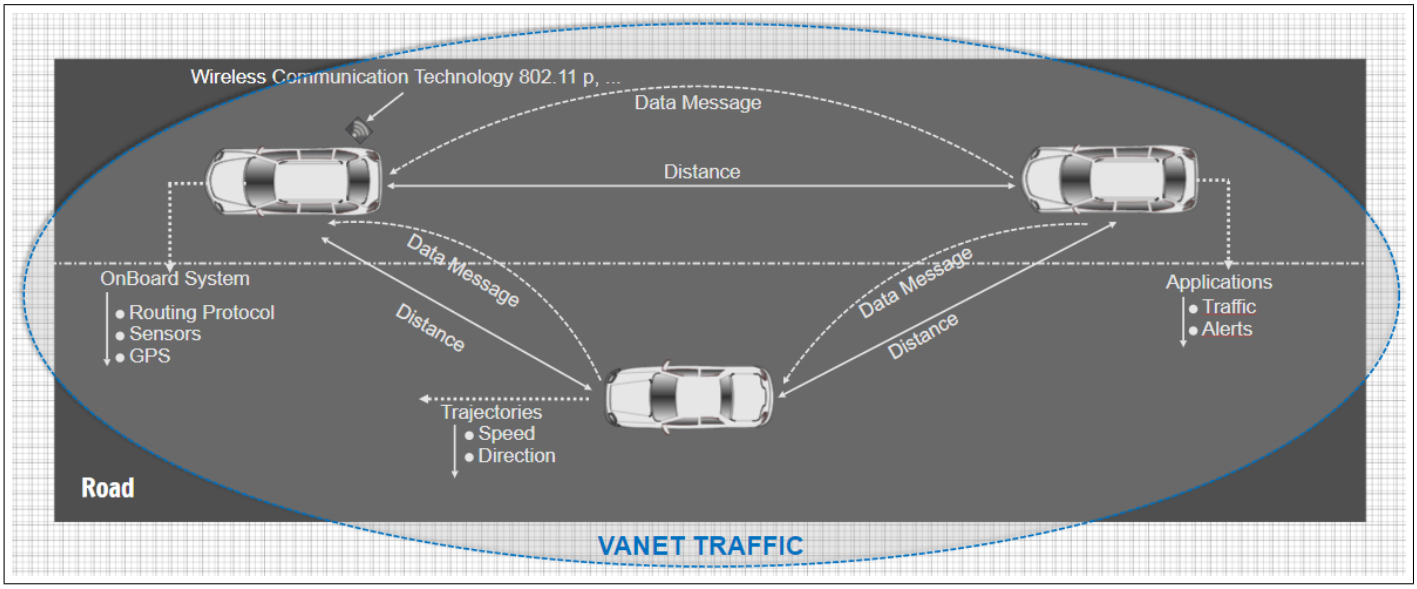

Figure 2. Vehicular Ad Hoc Network sample elements

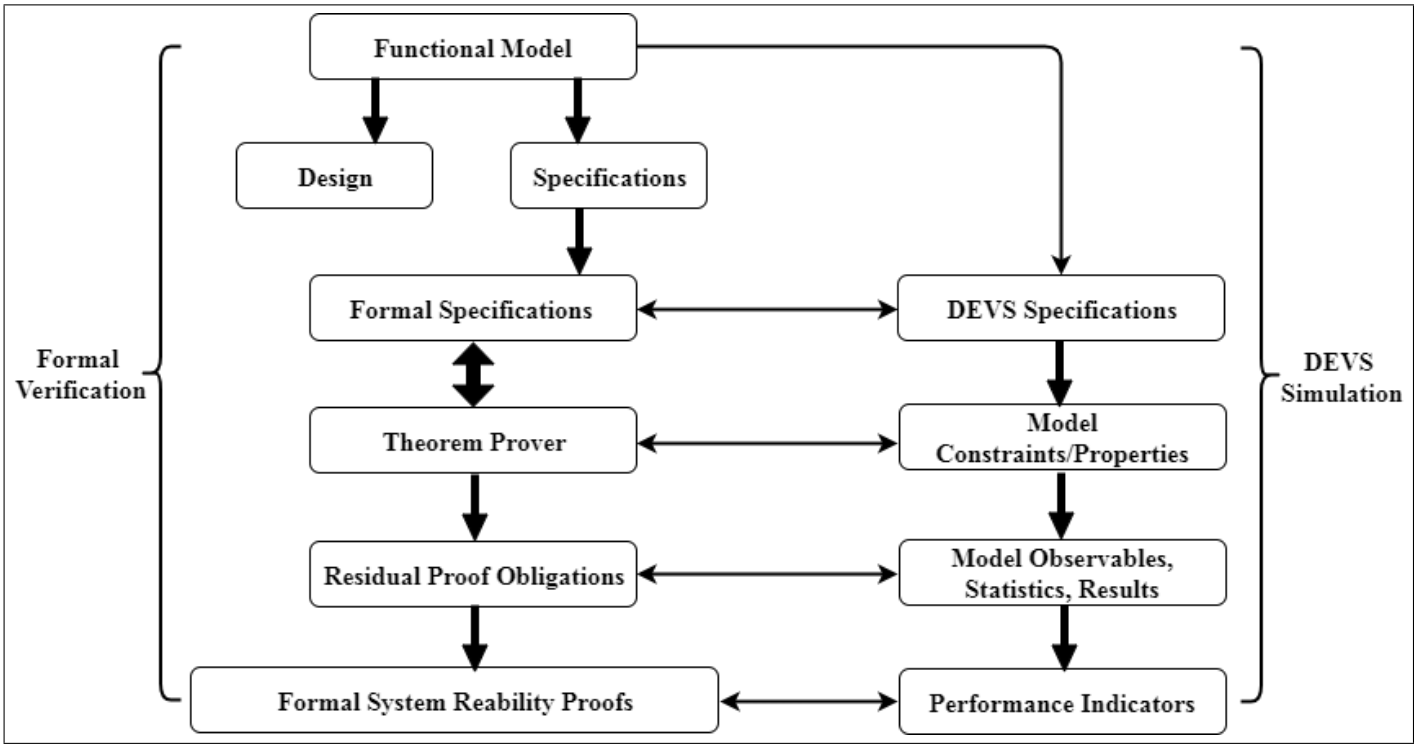

Figure 3. Relations between a functional model of a protocol with its related Event-B and DEVS models

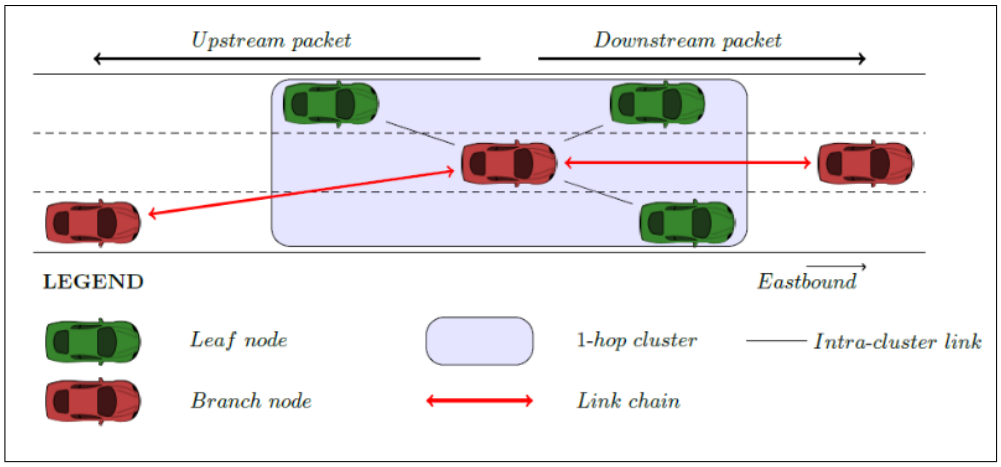

Figure 4. Building a virtual infrastructure with a distributed algorithm: Chain-Branch-Leaf[14]

message, based on the application request specified in the header fields, a branch node can forward it to:

- its leaf nodes;
- upstream branch node;
- downstream branch node;
- all branch nodes (including branch nodes of another traffic direction).

These destination options are coded into the link code of the original format of the packets defined in OLSR protocol. However, CBL can be implemented inside any other ad hoc routing protocol.

- A leaf node is an ordinary node which tries to connect itself to the closest branch node. If no branch node is detected, the leaf node elects the neighbor moving with the lowest speed and in the same traffic direction, as a branch. A leaf node sends both HELLO and application messages of which it is the originator.
- A chain is a virtual backbone made up of a sequence of branch nodes. Ideally, one chain should be created per traffic direction. On longitudinal road context such as highways, the chains behave as a virtual backbone similar to the one that should be obtained with an infrastructure. It offers to its branch nodes a path to forward application messages over long distance.
- Branch Choice is a field added in the HELLO message and containing the address of the elected branch to which the HELLO originator node is connected.
- The Connection Time (CT) is the time during which two nodes Ni and Nj could communicate if they kept the same speed.

CBL builds a chain formed by particularly stable vehicles called “branch” to which attach vehicles located in their coverage area, called “leaf”. Through OLSR routing messages, the vehicles exchange information that allows each one deciding in a completely distributed way if it is a branch or a leaf. Each vehicle which is a leaf designates the branch to which it is attached. As for the MPR in native OLSR, when a broadcast message is sent, it is retransmitted only by the vehicles called “branch”. In addition, CBL realizes an additional optimization which makes it possible to indicate that a broadcast message must be flooded only upstream or downstream in the chain. The “vehicle” component including its routing protocol (OLSR implementing the clustering scheme CBL) is a component that can be modeled using formal tools [6].

B. Event-B model of a CBL based VANET

In this paper, we present the formal modeling of Chain Branch Leaf protocol using Event-B . Reasoning on complex systems and software development are ensured by the formal method B [15] [16]. Event-B is an evolution utilizing only the notion of events, the latter makes describing the actions of abstractly modeling the behavior and specifications of our protocol in the B language possible. The development in Event-B is a list of formal models. This model contains all the complete mathematical development of a Discrete Transition System. The semantic of Event-B focuses on the simulation, transition systems, and the simulation between all described parts in the system. Each Event-B model is organized in two basic constructs that are machines and contexts. Contexts define the static part and machines define the dynamic part of the model. In an Event B framework, we can develop and structure asynchronous systems using abstract systems. We use refinements to augment the functionality to be modeled or to introduce details about the dynamic properties of a model. In refinement steps we refine one model M1 to another model M2, model M2 to model M3 and so on, until getting the desired functionality. To analyze our Event-B model, Rodin Core platform [17] was used. This platform is composed of two main components: the first is Rodin repository and the second is Rodin builder. They are integrated into Eclipse derived from the Java Development Tools. The following models are largely inspired from[11]. INITIALISATION event is the only auto-created event by Rodin tool when we define a new machine because every variable in the machine must be initialized in a way that is consistent with the model as illustrated in Figure 5.

In order to identify the type of each variable in the machine, we must add the invariants. Figure 6 shows every invariants of the machine. The sent variable (*inv1*) represents the set of sending data packets by any source node. The got variable (*inv2*) contains the set of successfully received data packets by any destination node. The lost variable (*inv3*) is the set of lost data packets due to network failure. New variables Branch,

```
INITIALISATION:
THEN
    act1:   sent ≔ ∅
    act2:   got ≔ ∅
    act3:   lost ≔ ∅
    act4:   Branch ≔ ∅
    act5:   Leaf ≔ ∅
    act6:   OneHop ≔ ∅
    act7:   link_chain ≔ ∅
    act8:   intra_cluster_link ≔ ∅
    act9:   z :∈ ∅
END
```

Figure 5. Initialisation.

```
inv1:  sent ⊆ Msg
inv2:  got ⊆ Msg
inv3:  lost ⊆ Msg
inv4:  Branch ⊆ ND
inv5:  Leaf ⊆ ND
inv6:  OneHop ⊆ ND
inv7:  z ∈ ND
inv8:  z ∉ OneHop
inv9:  got ∪ lost ⊆ sent
inv10: got ∩ lost = ∅
inv11: Branch ∩ Leaf = ∅
inv12: Branch ∪ Leaf = ND
inv13: (Leaf ∩ OneHop) ∩
       (Branch ∩ OneHop) = ∅
inv14: link_chain ∈ ND ↔ ND
inv15: intra_cluster_link
       ∈ ND ↔ ND
```

Figure 6. Invariants.

Leaf, OneHope (*inv6-inv8*) represent respectively the set of the cluster-head node elected by the other nodes, the set leaf nodes which tries to connect them selfs to the closest branch node and the set of the elected branch by other node in 1-hope neighborhood. The invariant (*inv9*) presents that the got and lost data packets are subset of the sending data packets. The disjointedness between the sets got and lost (*inv10*) means that a data packet cannot be simultaneously both received and lost. (*inv11*) makes clear that a node cannot be in the same time a branch and a leaf. (*inv12*) defines that the branch and leaf node are subsets of the set of all the network nodes. *OneHope* node cannot be in the same time a branch and a leaf as specified in (*inv13*). *LinkChain* set and *IntraClusterLink* define respectively the link between only branch nodes and the link between all networks nodes in a precise cluster as declared in (*inv14*) and (*inv15*).

To initiate the communication between the nodes, we use The event *Sent Hello* which represent the sending of a Hello message from the source node (*s*) to the destination node (*t*). The Guards of an event specify the conditions under which it can be executed. In our case, they declare that a Hello message can be sent between (*s*) and (*t*) provided that the node (*s*) is different to node (*t*) as shown in Figure 7.

```
sending_hello:
ANY
      s
      t
      hello
WHERE
      grd1: hello ∈ Msg
      grd2: hello ∉ sent
      grd3: s ∈ ND ∧ t ∈ ND
            ∧ s ≠ t
      grd4: source(hello) = s
      grd5: target(hello) = t
THEN
      act1: sent ≔ sent ∪ {hello}
END
```

Figure 7. Sent Hello.

```
sending_TC:
ANY
      c
      t
      TC
WHERE
      grd1: TC ∈ Msg
      grd2: TC ∉ sent
      grd3: c ∈ Branch
            ∧ t ∈ ND ∧ c ≠ t
      grd4: source(TC) = c
      grd5: target(TC) = t
THEN
      act1: sent ≔ sent ∪ {TC}
END
```

Figure 8. Sent TC.

Figure 8 presents that a sent TC event make it possible to branch nodes to send a A Topology Control message to any other node. Guards here specify that only branch node (*c*) can send a TC message to any other node (*t*) from the network nodes.

```
receiving_hello:
ANY
      s
      t
      hello
WHERE
      grd1: hello ∈ sent \
            (got ∪ lost)
      grd2: source(hello) = s
            ∧ target(hello) = t
THEN
      act1: got ≔ got ∪ {hello}
END
```

Figure 9. Got Hello.

Receiving events are illustrated in Figures 9 and 10 suc-

cessively:

A receiving Hello event represents the success of receiving of the Hello message by the destination node (*t*). (*grd1*) precises that the sending hello message is a part of sent only and not received by got or lost. (*grd2*) presents that hello message has a correct references of the source node (*s*) and the destination node (*t*).

```
receiving_TC:
ANY
      c
      t
      TC
WHERE
      grd1: TC ∈ sent \
            (got ∪ lost)
      grd2: source(TC) = c
            ∧ target(TC) = t
THEN
      act1: got ≔ got ∪ {TC}
END
```

Figure 10. Got TC.

A receiving TC event notifies the success of receiving of the TC message by the destination node (*t*) from the source node (*c*). Its Guards maintain the same ideas of a receiving Hello event.

```
losing_hello:
ANY
      s
      t
      hello
WHERE
      grd1: hello ∈ sent
            \ (got ∪ lost)
      grd2: source(hello) = s
            ∧ target(hello) = t
      grd3: s ↦ t ∉
            closure(link_chain)
      grd4: s ↦ t ∉
      closure(intra_cluster_link)
THEN
      act1: lost ≔ lost ∪ {hello}
END
```

Figure 11. Lost Hello.

Losing Hello event means the lost of a hello message due to any problem. Such problems can be a network failure or a powered off of any node or a moving of one node to a new location, and disconnection of a node from the network. As shown on Figure 11, the guards state that the Hello message is sent but not received neither by got or lost, and they precise that there is not any valid route between the source node (*s*) and the destination node (*t*). They state also in case of broken paths (*linkChain* and *intraClusterLink*) that Hello message cannot reach the destination node (t).

```
losing_TC:
ANY
      c
      t
      TC
WHERE
      grd1: TC ∈ sent \
            (got ∪ lost)
      grd2: source(TC) = c
            ∧ target(TC) = t
      grd3: c ↦ t ∉
            closure(link_chain)
      grd4: c ↦ t ∉
      closure(intra_cluster_link)
THEN
      act1: lost ≔ lost ∪ {TC}
END
```

Figure 12. Lost TC.

```
add_link_chain:
ANY
      x
      y
WHERE
      grd1: x ↦ y ∉ link_chain
      grd2: x ≠ y
      grd3: x ∈ Branch ∧ x ∉ Leaf
            ∧ y ∈ Branch ∧ y ∉ Leaf
THEN
      act1: link_chain ≔
            link_chain ∪ {x ↦ y}
END
```

Figure 13. Add Link Chain.

```
add_intra_cluster_link:
ANY
      x
      y
WHERE
      grd1: x ↦ y
            ∉ intra_cluster_link
      grd2: x ≠ y
THEN
      act1: intra_cluster_link
      ≔ intra_cluster_link ∪ {x ↦ y}
END
```

Figure 14. Add Intra Cluster Link.

```
remove_link_chain:
ANY
      x      ›
      y      ›
WHERE
      grd1: x ↦ y ∈ link_chain
      grd2: x ≠ y
THEN
      act1: link_chain ≔
            link_chain ∖ {x ↦ y}
END
```

Figure 15. Remove Link Chain.

```
remove_intra_cluster_link:
ANY
      x
      y
WHERE
      grd1: x ↦ y
      ∈ intra_cluster_link
      grd2: x ≠ y
THEN
      act1: intra_cluster_link ≔
      intra_cluster_link ∖ {x ↦ y}
END
```

Figure 16. Remove Intra Cluster Link.

Losing TC event means the lost of TC message sent by a branch node (c) to destination node (t) as presented in Figure 12. The guards of this event maintain the same ideas as those of the Losing Hello event.

Figures 13 to 16 show the links between the nodes of our protocol:

An Event Add Link Chain creates a link between two nodes (x) and (y). (*grd1-grd2*) state that there is no *linkChain* between the two different nodes (x) and (y). (*grd3*) presents that both (x) and (y) must be only branch nodes.

An Event Add Intra Cluster Link creates a link between two nodes (x) and (y) which can be branch node or leaf node in the same cluster. A cluster is composed by branch node and leaf nodes.

An Event Remove Link Chain deletes a link between two nodes (x) and (y). (*grd1-grd2*) state that there is a *linkChain* between the two different nodes (x) and (y).

An Event Remove Intra Cluster Link deletes a link between two nodes (x) and (y). (*grd1-grd2*) state that there is a *intraClusterLink* between the two different nodes (x) and (y).

C. DEVS models of an OLSR-CBL based VANET

Another step in the development of the proposed approach is to implement a DEVS-based multi-modeling of all the components used to simulate an ITS (e.g., a VANET). The DEVS modeling that we propose, early described in [18], is based on two variants of the DEVS formalism: P-DEVS[19] (Parallel-DEVS) and DS-DEVS[20] (DEVS Dynamic Structure). The first variant manages simultaneous external events and internal transitions by introducing the conflict function. The notion of transient state is also implemented by resorting to zero lifetime events. DS-DEVS and its improvements like DS-DE [21] introduce the possibility to modify the graph model during the simulation. For example, it is possible to create and destroy atomic or coupled models, or to create and destroy connections between the models. In our case, all these possibilities, which are not available in the classical DEVS formalism, are fundamental. Indeed, we chose, for the moment, to represent a vehicle as an atomic model whose connections represent the communication channels of the vehicles in the ad hoc network. The second important aspect is the management of the vehicles movements in a 3D continuous space (the road traffic lanes). Several space management options exist: discretization of the space, which raises the problem of the discretization step, distributing the space definition within each model (vehicle) or centralizing the definition and the management of the space into a specialized model. We chose the third option. As shown in Figure 17, the model "space" collaborates with the model "controller" which has a special type: it is an executive from the point of view of DS-DEVS. An executive is an atomic model, unique within a coupled model, that can modify the structure of the coupled model. All these operations are performed by the abstract algorithm of the associated coordinator so that we can guarantee the causality. The couple "space"-"controller" is responsible for: location management of the vehicles, the detection and dynamic creation of potential connections between the vehicles and the appearance and disappearance of the vehicles in the studied section of traffic lane according to their respective trajectory. The "space" model is notified by the "controller" when a vehicle enters or leaves the section. The "space" model calculates the connections based on the changes in the speed and direction sent by each vehicle.

For each new connection, the "controller" is notified and it updates the connection graph. Depending on its connections, the "vehicle" model transmits and receives messages that allow it to execute the routing protocol and the clustering method. As discussed in Section II-A, proactive routing protocols offer a bigger range of possibilities because they maintain a local topology of the network. So, we chose a variant of the OLSR protocol that implements the clustering algorithm CBL[14].

V. Validation of the Models

A. Validation of Event-B models of CBL

Our initial model presented how the data packets were transferred in only one step from their source to their destination node in our abstract model. On the contrary, actual protocols usually transfer data packets from source node (s) to destination node (t) by hop to hop concept. Thus, in a setting where not all nodes are directly connected, our goal is to model the storing and forwarding architecture. For that purpose, our model represents the variable *gstore* by invariant (*inv1*) that is the relation between *ND* and *Msg*. (inv2 and inv3) present that *linkChain* is the link between two branch nodes and *intraClusterLink* is the link between any two node in the cluster. Each data packet is stored using (*got U lost*) in the network, or any node can similarly store it in local variable *gstore* using invariant (*inv4*). Distributed data packets that are represented by invariant (*inv5*) as (*ran(gstore) ł (got ł lost)*) are known in the network as sending data packets (*sent*). Each data packet that belongs to (*sent*) in the network is given by

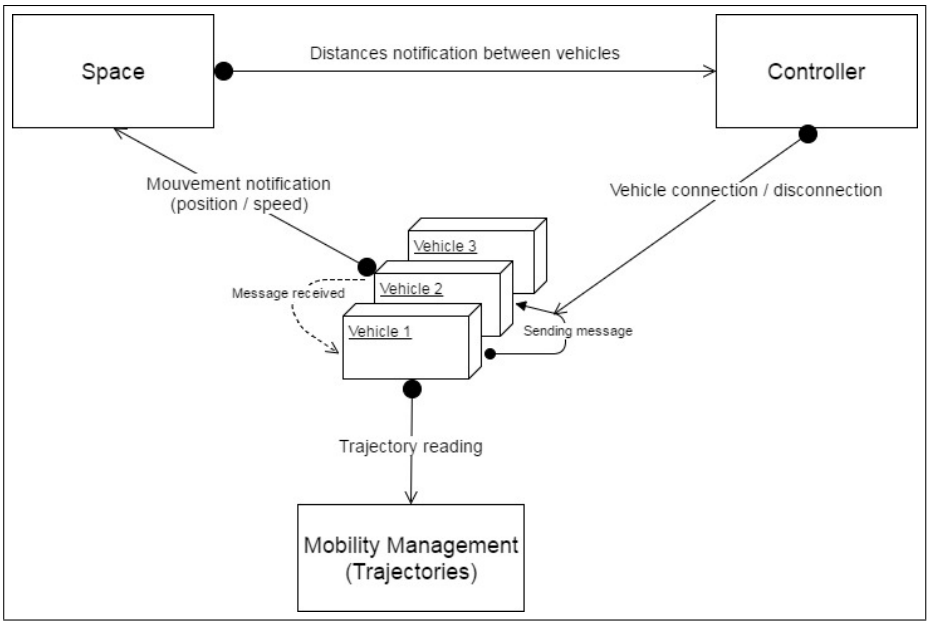

Figure 17. DEVS model of an OLSR-CBL based VANET [18] in VLE

invariant (*inv6*). Invariant (*inv7*) shows that a new data packet is a member of the network distributed data packets when it is not a member of sending data packets (*sent*). The last invariant (*inv8*) represents that it is not possible for different two nodes to map same data packet in relation (*gstore*), this means that a node is not able to store conflicting information regarding a unique data packet (Figure 18).

```
inv1: gstore ∈ ND ↔ Msg
inv2: link_chain ∈ Branch ↔ Branch
inv3: intra_cluster_link ∈ ND ↔ ND
inv4: ∀i·i ∈ ND ∧ i ∈ dom(gstore)
      ⇒ (got ∪ lost) ∩ gstore[{i}] = ∅
inv5: ran(gstore) ∪ (got ∪ lost) = sent
inv6: ∀i·i ∈ ND ⇒ gstore[{i}] ⊆ sent
inv7: ∀m·m ∈ Msg ∧ m ∉ sent
      ⇒ (m ∉ got ∧ m ∉ lost
      ∧ (∀i·i ∈ ND ⇒ i ↦ m ∉ gstore))
inv8: ∀m, i, j ·i ↦ m ∈ gstore
      ∧ j ↦ m ∈ gstore ⇒ i = j
```

Figure 18. Refinement: Invariants.

Figure 19 presents our refinement step that introduces a new forward event which is *forward TC*. This event is used to transfer the data packets from one node to its connected neighbor through the route. The first four guards show whether a new sending TC message is received or not using (*got ł lost*), and whether intermediate nodes *x* and *y* are directly connected or not. The destination node is represented by the target (*t*) and the intermediate node by (*x*) in (*grd4*) and (*grd5*). It is shown that the data packets (*TC*) is stored at the node (*x*) not y in the last two guards.

In this refinement, we introduce a new forward event, guards and actions in events that are *sending hello*, *sending TC*, *receiving hello* , *receiving TC*, *losing Hello* and *losing TC* (Figure 20).

As shown in Table I, these proof statistics of the formal

```
forward_TC:
ANY
      t
      x
      TC
      y
WHERE
      grd1: TC ∈ sent \
            (got ∪ lost)
      grd2: x ↦ y ∈ link_chain
      grd3: x ↦ y ∈
      intra_cluster_link
      grd4: target(TC) = t
      grd5: x ≠ target(TC)
      grd6: x ↦ TC ∈ gstore
      grd7: y ↦ TC ∉ gstore
THEN
      act1: gstore ≔ (gstore \
      {x ↦ TC}) ∪ {y ↦ TC}
END
```

Figure 19. Refinement: Forward TC Message.

TABLE I. PROOF STATISTICS.

Model	*Total number of POs*	*Automatic Proof*	*Interactive Proof*
Abstract Model	32	18 (56%)	14 (44%)
First Refinement	37	7 (19%)	30 (81%)
Total	69	25 (36%)	44 (64%)

development indicate the size of the model, the total number of the proof obligations, the number of automatic proofs and those proved interactively. In our abstract model, there are 32 proof obligations. 18 (56%) of these proof obligations are proved automatically, and 14 (44%) proof obligations are

```
    EVENT sending_hello
⊕ grd6: s ↦ hello ∉ gstore
⊕ act2: gstore ≔ gstore ∪ {s ↦ hello}

    EVENT sending_TC
⊕ grd6: c ↦ TC ∉ gstore
⊕ act2: gstore ≔ gstore ∪ {c ↦ TC}

    EVENT receiving_hello
⊕ grd3: t ↦ hello ∈ gstore
⊕ act2: gstore ≔ gstore ∖ {t ↦ hello}

    EVENT receiving_TC
⊕ grd4: t ↦ TC ∈ gstore
⊕ act2: gstore ≔ gstore ∖ {t ↦ TC}

    EVENT losing_hello
⊕ grd5: x ↦ hello ∈ gstore
⊕ act2: gstore ≔ gstore∖{x ↦ hello}

    EVENT losing_TC
⊕ grd6: x ↦ TC ∈ gstore
⊕ act2: gstore ≔ gstore∖{x ↦ TC}
```

Figure 20. Refinement: Events.

proven interactively. The use of *forward TC* and store in the first refinement generate 37 proof obligations in which 7 (19%) proved automatically and 30 (81%) proof obligations are interactive proofs. In our model there is 69 proof obligations in which 25 (36%) are automatically proved and 44 (64%) are proved interactively by Rodin tool.

B. Validation of DEVS models of CBL

All the models were developed using VLE, a DEVS-based multi-modeling tool that allows creating coupled models. Except the "vehicle" model which will be modified according to the projection of its related model realized with a formal tool, the other models should remain unchanged. Therefore, it is necessary to validate their design and verify that they behave correctly according to the corresponding vehicular network when compared to other specialized and well-established simulation tools. We performed this validation by simulation using the following configuration:

- Network size: the network includes a total of 358 vehicles on the A27 highway in France. The trajectory data have been generated based on real-world traffic data of the A27.
- Mobility: each vehicle is associated with one of the available trajectories. The trajectory determines the entry date of the vehicle on the road, its movement realized by successive segments with heterogeneous constant speeds (segments of different lengths and different speeds, respectively) and its exit date from the road section.
- The OLSR protocol operates according to referenced settings[14]. For the moment, only the sending and receiving of HELLO messages required by the CBL scheme are modeled. The communication range is fixed to 500 m.

Figure 21 shows the number of vehicles entering to the simulation, the number of vehicles leaving the simulation and

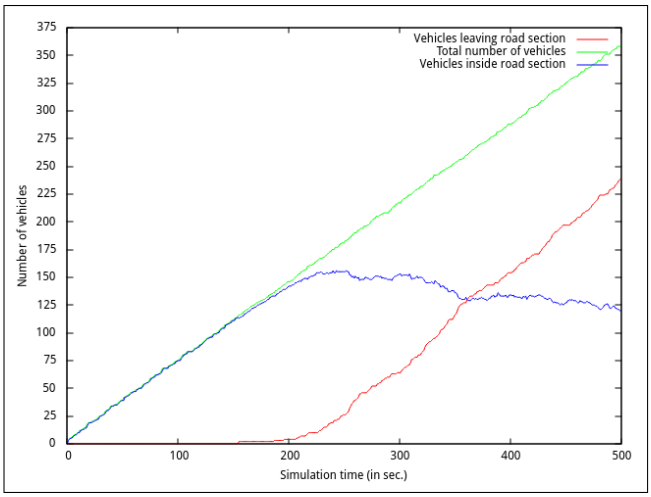

Figure 21. Evolution of the number of vehicles in the simulation.

the number of vehicles simultaneously present in the road section. The total number of branch nodes is 15% up to 35% of the total number of nodes, which reaches its maximum value of 130 simultaneous vehicles present in the road section (Figure 22).

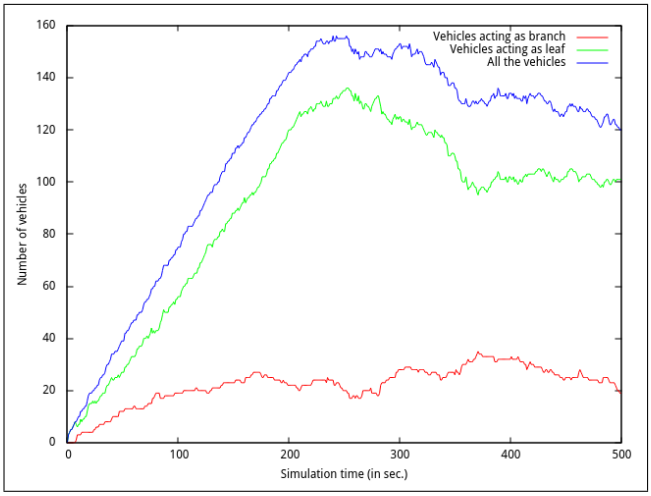

Figure 22. Evolution of the number of branch and leaf nodes in the simulation.

Therefore, about 70% of the vehicles are leaf nodes and do not retransmit broadcast traffic, which confirms CBL performance results[14]. The number of branch nodes per chain indicates that there are 1 to 2 connected subsets in each traffic direction, which confirms that the vehicular network is entirely connected, at least in the same traffic direction.

Figure 23 shows that each vehicle has an average of 80 neighbors in the entire VANET, thus 40 in the same traffic direction. Those which are branch nodes are selected by 20% to 40% of their neighbors which attach as leaf nodes. Each of these leaf nodes remains attached to a branch node 90% of the time (Figure 24), and a node remains without any branch less than 10% of the time.

This demonstrates that isolated nodes remains only few

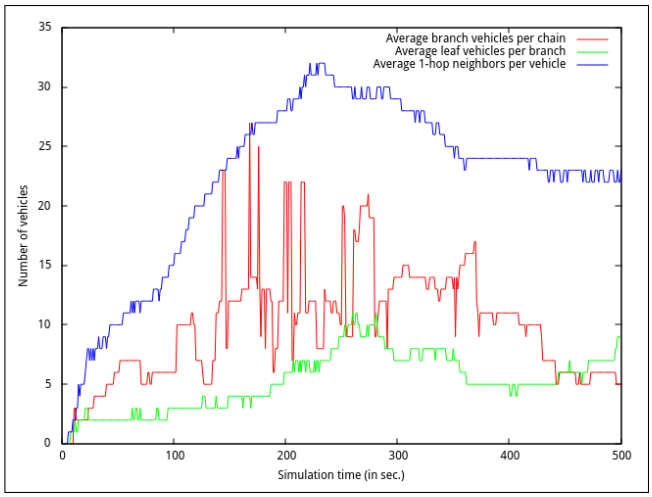

Figure 23. The number of one-hop neighbors and leaf nodes per branch in the simulation.

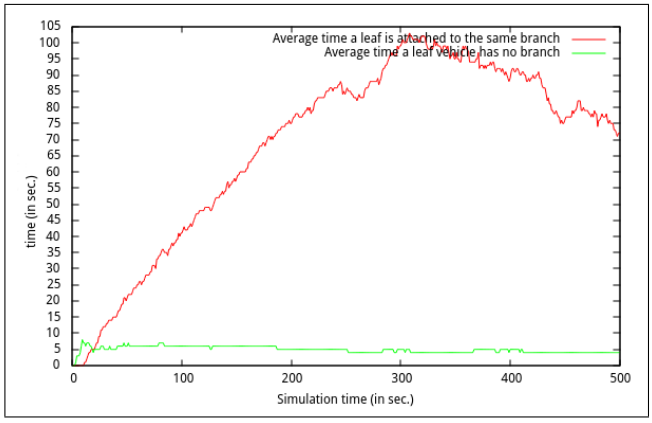

Figure 24. The duration a leaf is attached or not to a branch in the simulation.

time out of the vehicular network. All these results show that these DEVS-based models of a VANET implemented using VLE allow obtaining the same performance for OLSR with the CBL scheme than those obtained with well-established tools such as MATLAB and OPNET[14]. In addition to the testings on each component, these additional results contribute to validate the designed models.

VI. Conclusion and Future Work

This paper presented a new approach in development. This approach consists in projecting a set of formal models and proven properties on these models through formal tools, in a DEVS-based multi-modeling. Our goal is to put in interaction all these formal models with DEVS models of other components not prone to formal modeling in order to perform the evaluation of the global transport system in a single simulation process. This approach will allow verifying, by simulation, that proven properties on formal models that might not be sufficiently refined, are not contradicted in certain scenarios. It would also allow producing the results of the simulation as data that could be used in an interactive formal proving loop instead of a human expert. This article presents the preliminary formal model of an ad hoc network using the OLSR routing protocol and the CBL clustering scheme, and the DEVS-based related models that we have already realized. These models will be used for the development and the proof of concept of the approach we are developing. They models were implemented using an Event-B tool, namely Rodin, and VLE. The formal models were validate through refinements and interactive proof, and DEVS models were validated through a scenario of an ad hoc vehicular network built based on actual real-world traffic data on the A27 highway in France. Future work will concern the implementation of an automatic conversion of the Event-B formal models to DEVS ones.

Acknowledgment

The authors acknowledge the support of the CPER EL-SAT2020 project which is co-financed by the European Union with the European Regional Development Fund, the French state and the Hauts de France Region Council.

References

[1] E. Chebbi, P. Sondi, and E. Ramat, “Enhancing Formal Proofs of Network Protocols for Transport Systems using Discrete Event Simulation,” in The 9th Int. Conference on Advances in System Simulation, SIMUL 2017. IARIA, 2017.

[2] B. T. Sharef, R. A. Alsaqour, and M. Ismail, “Vehicular communication ad hoc routing protocols: A survey,” Journal of network and computer applications, vol. 40, 2014, pp. 363–396.

[3] G. K. Walia, “A survey on reactive routing protocols of the mobile ad hoc networks,” International Journal of Computer Applications, vol. 64, no. 22, 2013.

[4] R. Shenbagapriya and N. Kumar, “A survey on proactive routing protocols in manets,” in Science Engineering and Management Research (ICSEMR), 2014 International Conference on. IEEE, 2014, pp. 1–7.

[5] A. Maghsoudlou, M. St-Hilaire, and T. Kunz, “A survey on geographic routing protocols for mobile ad hoc networks,” Carleton University, Systems and Computer Engineering, Technical Report SCE-11-03, 2011.

[6] P. Sondi, D. Gantsou, and S. Lecomte, “Design guidelines for quality of service support in optimized link state routing-based mobile ad hoc networks,” Ad Hoc Networks, vol. 11, no. 1, 2013, pp. 298–323.

[7] Y.-C. Hu, D. B. Johnson, and A. Perrig, “SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks,” Ad hoc networks, vol. 1, no. 1, 2003, pp. 175–192.

[8] L. Gyesik, “How to formally model features of network security protocols,” International Journal of Security and Its Applications, vol. 8, no. 1, 2014, pp. 423–432.

[9] J. Diaz, D. Arroyo, and F. B. Rodriguez, “A formal methodology for integral security design and verification of network protocols,” Journal of Systems and Software, vol. 89, 2014, pp. 87–98.

[10] A. Singh and V. Singh, “Formal Modeling of Distance Vector Routing Protocol using Event-B,” Electronic and Electric Engineering ISSN 2231-1297, vol. 3, 2013, pp. 91–98.

[11] D. Méry and N. Singh, “Analysis of DSR protocol in e Event-B,” Stabilization, Safety, and Security of Distributed Systems, 2011, pp. 401–415.

[12] M. Kamali, L. Laibinis, L. Petre, and K. Sere, “Formal development of wireless sensor–actor networks,” Science of Computer Programming, vol. 80, 2014, pp. 25–49.

[13] A. Yacoub, C. Frydman et al., “Using dev-promela for modelling and verification of software,” in Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation. ACM, 2016, pp. 245–253.

[14] L. Rivoirard, M. Wahl, P. Sondi, M. Berbineau, and D. Gruyer, “Chain-Branch-Leaf: A clustering scheme for vehicular networks using only V2V communications,” Ad Hoc Networks, vol. 68, 2018, pp. 70–84.

[15] D. Cansell and D. Méry, The Event-B Modeling Method: Concepts and Case Studies. Springer, 2007.

[16] J.-R. Abrial, Modeling in Event-B: system and software engineering. Cambridge University Press, 2010.

[17] J. Coleman, C. Jones, I. Oliver, A. Romanovsky, and E. Troubitsyna, "Rodin (rigorous open development environment for complex systems)," in Fifth European Dependable Computing Conference: EDCC-5 supplementary volume, 2005, pp. 23–26.

[18] E. Chebbi, P. Sondi, E. Ramat, L. Rivoirard, and M. Wahl, "Simulation of a clustering scheme for vehicular ad hoc networks using a devs-based virtual laboratory environment," Procedia Computer Science, vol. 130, 2018, pp. 344 – 351, the 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018).

[19] A. C. H. Chow and B. P. Zeigler, "Parallel DEVS: A parallel, hierarchical, modular modeling formalism," in Simulation Conference Proceedings, 1994. Winter. IEEE, 1994, pp. 716–722.

[20] F. Barros, "Modeling formalisms for dynamic structure systems," ACM Transactions on Modeling and Computer Simulation (TOMACS), vol. 7, no. 4, 1997, pp. 501–515.

[21] F. J. Barros, "Abstract simulators for the dsde formalism," in Proceedings of the 30th conference on Winter simulation. IEEE Computer Society Press, 1998, pp. 407–412.