VeriVANca: An Actor-Based Framework for Formal Verification of Warning Message Dissemination Schemes in VANETs

Farnaz Yousefi[1], Ehsan Khamespanah[2,3], Mohammed Gharib[4], Marjan Sirjani[3,5], and Ali Movaghar[1]

[1] Department of Computer Engineering, Sharif University of Technology - Iran
[2] School of Electrical and Computer Engineering, University of Tehran - Iran
[3] School of Computer Science, Reykjavik University - Iceland
[4] School of Computer Science, Institute for Research in Fundamental Science (IPM) - Iran
[5] School of IDT, Mälardalen University - Sweden

Abstract. One of the applications of Vehicular Ad-hoc NETworks, known as VANETs, is warning message dissemination among vehicles in dangerous situations to prevent more damage. The only communication mechanism for message dissemination is multi-hop broadcast; in which, forwarding a received message has to be regulated using a scheme regarding the selection of forwarding nodes. When analyzing these schemes, simulation-based frameworks fail to provide guaranteed analysis results due to the high level of concurrency in this application. Therefore, there is a need to use model checking approaches for achieving reliable results. In this paper, we have developed a framework called VeriVANca, to provide model checking facilities for the analysis of warning message dissemination schemes in VANETs. To this end, an actor-based modeling language, Rebeca, is used which is equipped with a variety of model checking engines. To illustrate the applicability of VeriVANca, modeling and analysis of two warning message dissemination schemes are presented. Some scenarios for these schemes are presented to show that concurrent behaviors of the system components may cause uncertainty in both behavior and performance which may not be detected by simulation-based techniques. Furthermore, the scalability of VeriVANca is examined by analyzing a middle-sized model.

Keywords: Model Checking, Warning Message Dissemination,Vehicular Ad-Hoc Networks (VANETs), Rebeca, Actor Model

1 Introduction

VANETs have attracted much attention in both academia and industry during the last years. The emergence of autonomous vehicles and the safety concerns regarding the use of these vehicles in the near future have highlighted the possible use of VANETs in safety enhancement of future transportation system. Using

VANETs in such mission critical applications, calls for reliability assurance of algorithms. One of the applications in this domain is the use of vehicle to vehicle communication for Warning Message Dissemination (WMD) in dangerous situations to prevent further damage. In this application, vehicles broadcast warning messages to inform each other of the upcoming hazard. To increase the number of vehicles receiving the warning message, the receiving nodes should forward the message. To hold the trade-off between the traffic in the network and maximum number of vehicles receiving the message, a number of schemes regarding the selection of forwarding nodes has been proposed [14]. More details about WMD in VANETs are presented in Section 2.

A number of simulation-based tools and techniques have been used for the analysis of these WMD schemes. However, concurrent execution of system components reduces the effectiveness of simulation-based approaches for such mission critical applications. This is because simulation-based approaches cannot provide high level of confidence for the correct behavior of the system. In such cases, there is a need to apply formal verification for achieving reliable results. Formal verification is used in applications of VANETs such as cooperative collision avoidance [7], intersection management using mutual exclusion algorithms [2], and collaborative driving [10]. However, to the best of our knowledge, there is no work on formal verification of WMD application in VANETs.

In this paper, we introduce VeriVANca as a framework for the analysis of WMD schemes in VANETs. To this end, we develop VeriVANca in Timed Rebeca [9], a real-time extension of Rebeca [15]. Rebeca is an operational interpretation of the actor model with formal semantics, supported by a variety of analysis tools [8]. In the actor model, all the elements that are running concurrently in a distributed system are modeled as actors. Communication among actors takes place by asynchronous message passing. These structures and features match the needs of VANETs as they consist of autonomous nodes which communicate by message passing. This level of faithfulness helps in having a more natural mapping between the actor model and VANETs, making models easier to develop and understand. In Section 3 Timed Rebeca is briefly introduced using the counting-based scheme example.

To illustrate the applicability of this approach, we have modeled a distance-based scheme [16] and a counting-based scheme [17] using VeriVANca. Results of model checking for the distance-based scheme show that concurrent execution of the system components enables multiple execution traces some of which cause starvation and may not be detected using simulation-based techniques (Section 4.1). We also observed that, in a given scenario, multiple values may be achieved for the performance when considering the interleaving of concurrently executing components. Our further investigations yield that having multiple performance results is not limited to one scenario but is common. More details on these cases are presented in Section 4.2. Furthermore, to examine the scalability of VeriVANca, a middle-sized model of a four-lane street with about 40 vehicles is analyzed. We observed that if scaling up the number of vehicles results in creation of very congested areas, the size of the state space and analysis time is

increased dramatically. However, scaling up the model without creation of new congested areas, results in smooth increase in the size of the state space and analysis time as presented in Section 4.3.

2 Warning Message Dissemination in VANETs

WMD is an application developed for VANETs that tends to increase the safety and riding experience of passengers. In this application, a warning message is disseminated between vehicles in the case of any abnormal situations such as car accidents or undesirable road conditions. Received warning messages are used either to activate an automatic operation such as reducing speed to avoid chained accidents (increasing safety) or are shown as alerts to inform the driver of the upcoming hazard so that the driver can do operations such as changing their route (improving the riding experience).

Using WMD in safety-critical applications, requires providing high reliability for the application in developed solutions. Besides, some characteristics of VANETs such as high mobility of the nodes and fast topology changes, makes routing algorithms commonly used in MANETs (Mobile Ad-hoc NETworks) inapplicable to VANETs [20]. Therefore, the only approach for implementation of message dissemination in VANETs is multi-hop broadcast of the message. In this approach, the receiving nodes are responsible for re-broadcasting the message to the others. However, this can result in broadcast storm problem in the network. In order to tackle this problem, a number of schemes have been proposed for WMD as described in the following subsection.

2.1 Message Dissemination Schemes

Message dissemination schemes are algorithms that specify how a forwarding node is selected in a VANET. The selection of a forwarding node is performed based on some criteria such as distance between senders and receivers, number of received messages by a node, probabilities associated with nodes, topology of the network, etc. [14]. In this paper, two schemes —a distance-based and a counting-based scheme— are modeled using the proposed framework.

The distance-based scheme, called TLO (The Last One) [16], makes use of location information of the vehicles to select the forwarding node. In this scheme, upon a message broadcast, the farthest receiver in the range of the sender is selected as the forwarding TLO node. Other vehicles in the range know that they are not the farthest node and do not forward the received message. However, they wait for a while to make sure of successful broadcast of the TLO node. Receiving the warning message from the TLO node, means that the sending of the message has been successful and they do not forward the warning message. Otherwise, the algorithm is run once again to select the next TLO forwarding node.

In the counting-based scheme [17], an integer number is defined as counter threshold. Each receiving node counts the number of received messages in a time interval. At the end of that time interval, the receiver decides on being a

forwarding node based on the comparison of the value of its counter and the value of counter threshold. If the value of the counter is greater than the value of counter threshold, the receiver assumes that enough warning messages are disseminated in its vicinity; therefore, it avoids forwarding the message. Otherwise, the receiver forwards the warning message.

2.2 Analysis Techniques

Different analysis techniques have been developed for the analysis of message dissemination schemes in VANETs. Simulation-based approaches are widely used for the analysis of applications of in this domain. Gama et. al. developed a model and analyzed three different message dissemination schemes using Veins simulator [4]. Sanguesa et. al. have used ns-2 simulator in two independent works regarding the selection of optimal message dissemination scheme. In [12], they aim to select the optimal broadcasting scheme for the model in each scenario and in [13], the selection of the optimal scheme is performed for each vehicle based on vehicular density and the topological characteristics of the environment where the vehicle is located in. In a more comprehensive work [14] authors have developed a framework in ns-3 simulator for comparing different schemes. Note that although this approach is used in many applications, it does not guarantee correctness of results as it does not consider concurrent execution of system components.

Another technique used for the analysis of WMD in VANETs is the analytical approach. In this approach, a system is modeled by mathematical equations and the analysis is performed by finding solutions to the equation system. For example, in [11], Saeed et. al. have derived difference equations that their solutions yield the probability of all vehicles receiving the emergency warning message. This value is computed as a function of the number of neighbors of each vehicle, the rebroadcast probability, and the dissemination distance. In another work, a probabilistic multi-hop broadcast scheme is mathematically formulated and the packet reception probability is reported for different configurations, taking into account the topology of the network and as a result, major network characteristics such as vehicle density and the number of one-hop neighbors [6]. This approach guarantees achieving correct results but it is not modular and developing mathematical formula needs a high degree of user interaction and a high degree of expertise.

As the third technique, model checking is a general verification approach which provides ease of modeling similarly to simulation-based approaches in addition to guaranteeing the correctness of results due to its mathematical foundation. To the best of our knowledge, there is no framework which provides model checking facilities for the analysis of WMD schemes in VANETs.

3 Rebeca Language

Rebeca is a modeling language based on Hewitt and Agha's actors [1]. Actors in Rebeca are independent units of concurrently running programs that communicate

with each other through message passing. The message passing is an asynchronous non-blocking call to the actor's corresponding message server. Message servers are methods of the actor that specify the reaction of the actor to its corresponding received message. In the Java-like syntax of Rebeca, actors are instantiated from reactive class definitions that are similar to the concept of classes in Java. Actors in this sense can be assumed as objects in Java. Each reactive class declares the size of its message buffer [6], a set of state variables, and the messages to which it can respond. Reactive classes have constructors with the same name as their reactive class, that are responsible for initializing the actor's state.

Basically, in Rebeca the concept of known rebecs was introduced for an actor to specify the actors to which it can send messages. However, to implement applications in ad-hoc networks, a more flexible sending mechanism is needed. Two Rebeca extensions b-Rebeca [18] and w-Rebeca [19] have been proposed to provide more complex sending mechanism. In b-Rebeca the concept of known rebecs is eliminated and it is assumed that the only communication mechanism among actors is broadcasting; hence, only a fully connected network can be modeled. Note that the type of broadcasting introduced in b-Rebeca is not the same as the location-based broadcasting in VANETs. In location-based broadcasting, only the actors in the range of each other are connected in the Rebeca model. Regarding this assumption, a counter-based reduction technique is used in b-Rebeca to reduce the state space size of the model making it impossible to send messages to a subset of actors.

The other extension w-Rebeca, which is developed for model checking of wireless ad-hoc networks, uses an adjacency matrix in the model checking engine, to consider connectivity of actors. In this approach, by random changes in the value of adjacency matrix, all the possible topologies of the network are considered in the model checking. Note that users are allowed to define a set of topological constraints and the topologies that do not fulfill the constraints are not considered in the model checking. w-Rebeca does not support timing in the model which is essential for developing models in the domain of VANET, since there are some real-time properties that need to be considered. Besides, considering all possible topologies —some of which may not be possible in the reality of the model— results in a bigger state space for the model. In addition, considering these infeasible topologies, may cause false-negative results when checking correctness properties.

3.1 Explaining Rebeca by the Example of Counting-Based Scheme

In this subsection, we introduce Timed Rebeca [9] using the example of the counting-based scheme presented in the previous section. A Timed Rebeca model consists of a number of reactive class definitions which provide type and behavior specification for the actors instantiated from them. There are two reactive classes `BroadcastingActor` and `Vehicle` in the implementation of counting-based WMD in VeriVANca as shown in Listing 1.

[6] Message queue in Rebeca and message bag in Timed Rebeca

Each reactive class consists of a set of state variables and a message bag with the size specified in parentheses after the name of the reactive class in the declaration. For example, reactive class `Vehicle` has state variables `isAv`, `direction`, `latency`, `counter`, etc. The size of the message bag for this reactive class is set to five. The local state of each actor consists of the values of its state variables and the contents of its message bag. Being an actor-based language, Timed Rebeca benefits from asynchronous message passing among actors. Upon receiving a message, the message is added to the actor's message bag. Whenever the actor takes a message from the message bag, the routine which is associated with that message is executed. These routines are called message servers and are implemented in the body of reactive classes.

```
1  env int RANGE = 10;
2  env int THRESHOLD_WAITING = 4;
3  env int MESSAGE_SEND_TIME = 1;
4  env int C_THRESHOLD = 3;
5  abstract reactiveclass BroadcastingActor (5) {
6    statevars { int id, x, y; }
7    abstract msgsrv receive(int data);
8    void broadcast(int data) { ... }
9    double distance(BroadcastingActor bActor, BroadcastingActor cActor){...}
10 }
11 reactiveclass Vehicle extends BroadcastingActor(5){
12   statevars{
13     boolean isAV;
14     int direction, latency, destX, destY, counter;
15   }
16   Vehicle (/*List of Parameters*/){
17     /*Variables Initializations*/
18     if (isAV) {
19        self.alertAccident();
20     } else
21       self.move() after(latency);
22   }
23   msgsrv alertAccident(){ ... }
24   msgsrv move() { ... }
25   msgsrv stop () { ... }
26   msgsrv finishWait(int hop) { ... }
27   msgsrv receive(int hopNum) { ... }
28 }
29 main {
30   Vehicle v1():(0,0,10,RIGHT,1,10,10,true), v2():(1,10,0,UP,2,10,10,false),
31     v3():(2,-1,0,RIGHT,1,10,0,false), v4():(3,0,1,DOWN,2,0,-10,false),
32     v5():(4,3,0,LEFT,1,-10,0,false);
33 }
```

Listing. 1. Counting-based scheme in Timed Rebeca

As depicted in Listing 1, the message servers of the reactive class `Vehicle` are `move`, `receive`, `alertAccident`, `stop`, and `finishWait`. In order for an

actor to be able to send a message to another actor, the sender has to have a direct reference to the receiver actor. For example, in Line 19, the message `alertAccident` is sent to `self` which represents a reference to the actor itself. However, in order to model a WMD scheme in VANETs, the warning message should reach actors which are in the range of the sender actor. In other words, actors should receive messages based on some criteria, i.e., their location in this application. We used the inheritance mechanism of Timed Rebeca to implement this customized sending strategy.

3.2 Customized Message Sending in VeriVANca

In object-oriented design, inheritance mechanism enables classes to be derived from another class and form a hierarchy of classes that share a set of attributes and methods. Using this approach, we encapsulated a broadcasting mechanism in a reactive class called `BroadcastingActor` and all other behaviors of vehicles are implemented in `Vehicle` reactive class which is derived from `BroadcastingActor`. In `BroadcastingActor`, the `broadcast` method shown in Listing 2 mimics the sending mechanism of vehicles in VANET.

As mentioned before, broadcasting data results in receiving a message containing that data by the vehicles in the range of the sender actor. In the body of this method, all actors —that are derived from `BroadcastingActor`— are examined in terms of their distance to the sender (Line 5). If the distance between an actor and the sender is less than the specified threshold, called `RANGE` (Line 6), the data is sent to the actor by an asynchronous message server call of `receive` (Line 7). As `BroadcastingActor` has no idea about the behavior of vehicles, upon receiving the `receive` message, the template method design pattern [5] is used in the implementation of `receive`. So, the `receive` message server is defined as an abstract message server in `BroadcastingActor` and its body is implemented in `Vehicle`. The behavior of the WMD scheme is implemented in `Vehicle`.

```
1  void broadcast(int data) {
2    ArrayList<ReactiveClass> allActors = getAllActors();
3    for(int i = 0; i < allActors.size(); i++) {
4      BroadcastingActor ba = (BroadcastingActor)allActors.get(i);
5      double distance = distance (ba , self);
6      if(distance < RANGE) {
7        ba.receive(data) after (MESSAGE_SEND_TIME);
8      }
9    }
10 }
11 double distance(BroadcastingActor bActor , BroadcastingActor cActor){
12   return sqrt(pow(cActor.x - bActor.x, 2) + pow(cActor.y - bActor.y, 2));
13 }
```

Listing. 2. Body of broadcast Method in Broadcasting Actor

3.3 Counting-Based Scheme in VeriVANca

For the case of counting-based scheme, three message servers `alertAccident`, `finishWait`, and `receive` provide the behavior of the scheme. When `Vehicle` actors are instantiated, their constructor methods are executed resulting in sending one of the following messages to themselves:

- `alertAccindent`: sent by the accident vehicle to start the WMD algorithm (Line 8)
- `move`: sent by the other actors to begin moving with their pre-defined `latency`; an actor performs this through sending `move` message periodically to itself (Line 10).

```
1  reactiveclass Vehicle extends BroadcastingActor(5){
2
3    statevars{ ... }
4    Vehicle (...){
5      ...
6      counter = 0;
7      if (isAV) {
8         self.alertAccident();
9      } else
10       self.move() after(latency);
11   }
12   msgsrv alertAccident(){
13     broadcast(0);
14   }
15   msgsrv finishWait(int hop){
16     if (counter < C_THRESHOLD)
17       broadcast(hop++);
18   }
19   msgsrv receive(int hopNum) {
20     if (counter == 0) {
21       self.finishWait(hopNum) after (THRESHOLD_WAITING);
22       counter = 1;
23     } else {
24       counter++;
25     }
26   }
27 }
```

Listing. 3. Body of message servers in Vehicle Actor

The algorithm of counting-based scheme, as implemented in Listing 3, begins by serving `alertAccident` message in the accident vehicle. Upon the execution of `receive`, if the `counter`, which is initially set to zero for all actors (Line 6), is zero — meaning that it is the first time the actor is receiving the warning message— a watchdog timer is started. This is implemented by sending the `finishWait` message to the actor itself with the arrival time of `THRESHOLD_WAITING`. In

addition, the value of `counter` is set to one to indicate that this is the first call of `receive` (Lines 20-22). The next calls of `receive` result in increasing the value of `counter`, which represents the number of received warning messages. When message server `finishWait` is executed by an actor, showing that the watchdog timer is expired, the value of `counter` is compared with the threshold considered for the counter (`C_THRESHOLD`). By not exceeding the threshold, i.e., the area around the actor is not covered by enough number of warning messages, the actor broadcasts the warning message (Lines 16 and 17).

3.4 Reusability of VeriVANca

To illustrate the reusability of VeriVANca, we show how the model of the counting-based scheme can be altered to present another scheme (the TLO scheme) by making minor modifications to the code. At the first step, we implemented the algorithm in a method called `runTLO`. As shown in Listing 4, the bodies of the message servers `finishWait` and `receive` are rewritten to mimic the behavior of the scheme in the event of expiration of the watchdog timer and receiving a warning message respectively.

```
1  msgsrv finishWait(int hopNum) {
2    if (isWaiting)
3      runTLO(hopNum);
4  }
5  msgsrv receive(int hopNum) {
6    if(!isWaiting)
7      runTLO(hopNum);
8    else
9      isWaiting = false;
10 }
11 void runTLO(int hopNum) {
12   if (!received) {
13     if (isTLO()) {
14       broadcast(hopNum++);
15       received = true;
16     } else {
17       isWaiting = true;
18       self.finishWait(hopNum)
             after(THRESHOLD_WAITING);
19     }
20   }
21 }
```

Listing. 4. Needed modifications for TLO scheme

In the TLO scheme, explained in Section 2.1, upon receiving the warning message for the first time, the `runTLO` method is called. In the body of this method, if the value of state variable `received` is false —meaning that the actor has not received the duplicate warning message from a selected TLO node as a sign of its successful broadcast—, the `isTLO` method is called. This method is implemented in the `BroadcastingActor` and checks if the actor is the furthest node in the range of the sender and returns the result as a boolean value. If the return value is true, the actor is the last one in the range and is selected as the TLO node to forward the warning message; so, it broadcasts the message by increasing the value of `hopNum` by one (Line 15). Then the value of `received` is set to true to show that broadcasting has been successful. In case the actor is not the last one in the range (Line 17), the actor should wait for a while to make sure that the selected TLO node has successfully broadcasted the warning message. To this end, the actor sets the value of `isWaiting` to true to show that

the actor is in the waiting mode, and then sets the watchdog timer by sending message `finishWait` to itself by execution time of `THRESHOLD_WAITING` (Line 19). The message server `receive`, like in the previous scheme, mimics receiving the warning message. In the body of this message server, if the value of `isWaiting` is false, meaning that the actor is not in the waiting mode, `isTLO` is executed to select the TLO forwarding node. Otherwise, `isWaiting` is set to false since this message is interpreted as a successful broadcast of the TLO node. The `finishWait` message server is executed upon expiration of the watchdog timer and it checks the value of `isWaiting`. In the case of false value for `finishWait`, the actor has not received the warning message from the selected TLO node, so, `runTLO` is called to select the next TLO forwarding node.

4 Experimental Results

To demonstrate the applicability of VeriVANca, both of the schemes presented in the former section are analyzed in different configurations. As mentioned before, concurrent behaviors of the system components may cause uncertainty which is clearly observable in the presented scenarios, but may not be detected using simulation-based techniques. For the case of the TLO scheme, we show that nondeterminism causes starvation and for the case of the counting-based scheme, it causes different results in the performance of the algorithm. Furthermore, we illustrate that the approach is scalable regarding the number of cars with traffic patterns that do not contain congested areas. Note that the following experiments have been executed on a Macbook Air with Intel Core i5 1.3 GHz CPU and 8GB of RAM, running macOS Mojave 10.14.2 as the operating system. Development of these experiments are performed in Afra, modeling and verification IDE of Rebeca family languages [3].

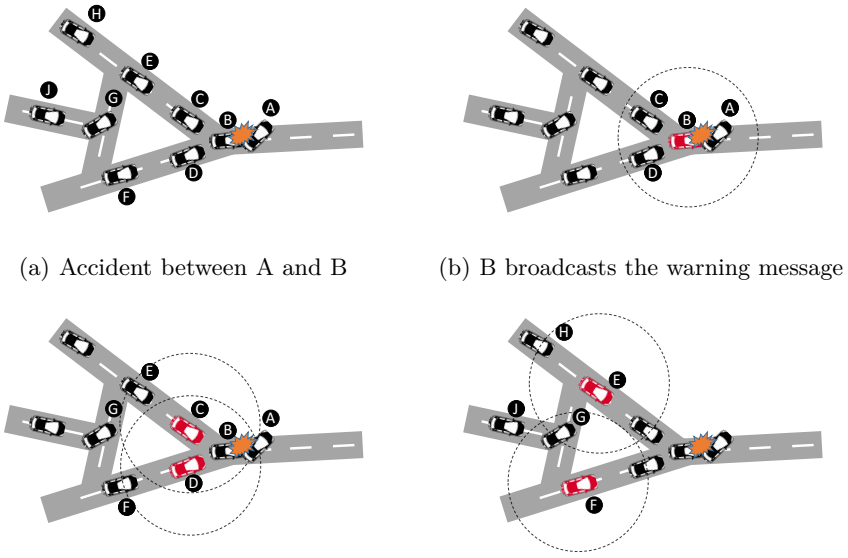
4.1 Starvation Scenario in TLO Scheme

In this section, we present an observed scenario that using the TLO scheme causes starvation and affects the reliability of the scheme in some executions. The steps of the scenario is depicted in Figure 1. In 1(a), position of the vehicles is shown in the time of the accident between vehicles A and B. In the next step, vehicle B starts broadcasting the warning message and vehicles C and D receive the message as they are in the range of B (Figure 1(b)). Upon receiving the warning message, these vehicles execute the TLO algorithm and since they both have the same distance from B, they forward the received warning message and the vehicles E and F receive the warning message from these two vehicles. When vehicles E and F execute the TLO algorithm, racing between the following two scenarios happen.

1. **E broadcasts before F:** vehicles G and H receive the warning message from E. Upon execution of TLO algorithm by G and H, Vehicle H is selected as the TLO forwarding node and forwards the message. Meanwhile, vehicle

G is waiting for receiving the warning message from H to make sure that the broadcasting has been successful. If in the waiting time of G, vehicle H forwards the warning message, the message will be interpreted as acknowledgement of the successful broadcast of H and although G is TLO node in this step, it will not forward the message. In this case, the vehicle J does not receive the warning message.

2. **F broadcasts before E:** vehicle G receive the warning message from F and after the execution of TLO algorithm, it forwards the message as the selected TLO node and vehicle J will receive the warning message in this scenario.



(a) Accident between A and B

(b) B broadcasts the warning message

(c) C and D are both selected as TLO nodes to forward the warning message

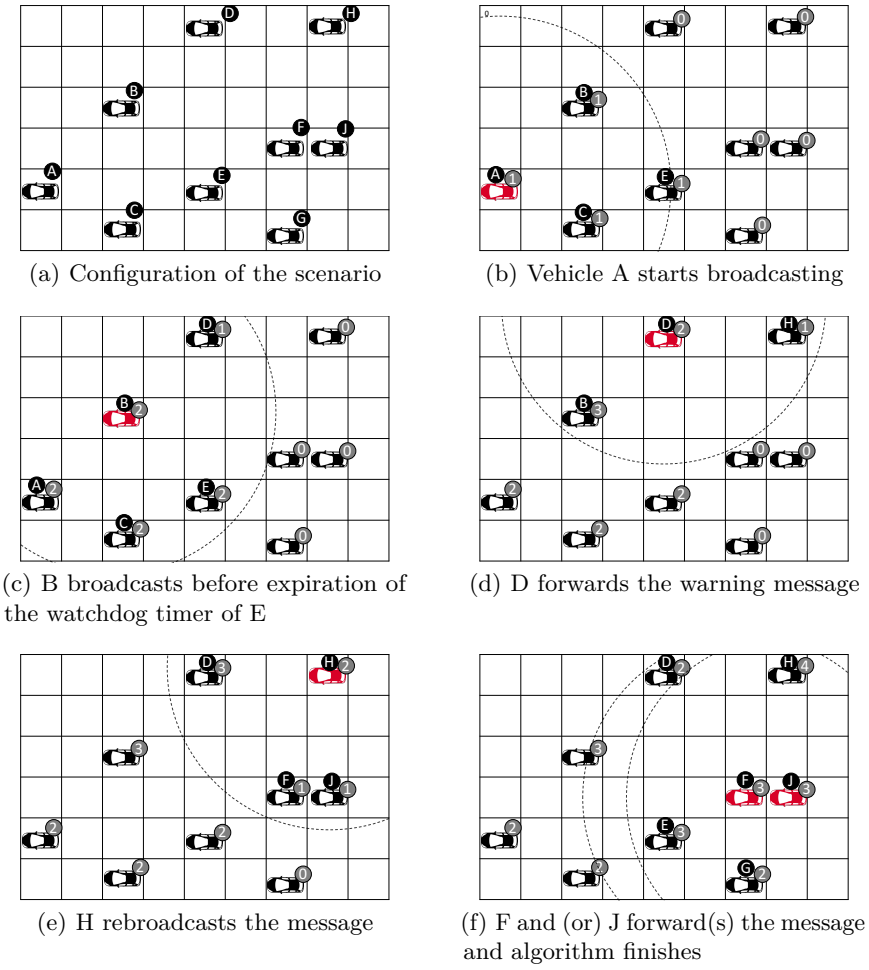
(d) Order of broadcasting between E and F results in two cases

Fig. 1. A scenario of TLO scheme which results in two execution alternatives that one of them causes starvation for vehicle J

This example shows that concurrent execution of the algorithm in nodes causes nondeterministic behavior which may violate correctness properties of the application. To avoid such cases, all the possible nondeterministic behaviors have to be considered in any analysis framework. However, simulation-based techniques, commonly used for the analysis of these systems, fail to report a result by considering all the possible execution traces. This highlights the necessity of applying formal methods in the development of applications of VANETs with critical mission.

4.2 Nondeterminism in Performance of the Counting-Based Scheme

The configuration depicted in Figure 2(a) is used for the analysis of the counting-based scheme (explained in Section 3.1). In this scenario, the value of `C_THRESHOLD` is set to 2 and the `RANGE` is set to 4. The scenario begins with the vehicle A broadcasting the warning message (Figure 2(b)). This broadcast results in increasing the counters of the vehicles A, B, C, and E by one. In the next round two following cases may happen.



(a) Configuration of the scenario

(b) Vehicle A starts broadcasting

(c) B broadcasts before expiration of the watchdog timer of E

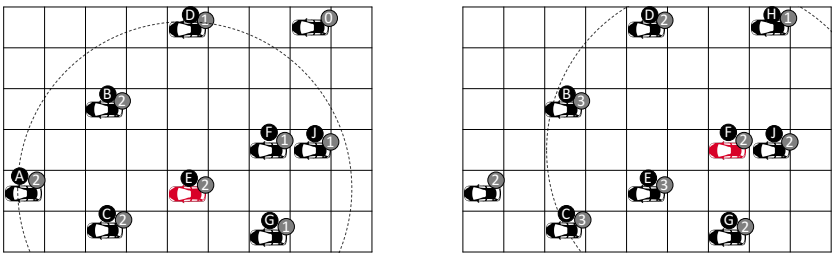
(d) D forwards the warning message

(e) H rebroadcasts the message

(f) F and (or) J forward(s) the message and algorithm finishes

Fig. 2. A case of the scenario for the counting-based scheme

1. **The watchdog timer of vehicle E expires after receiving the message from B:** In this case, as the counter has reached the threshold, E does not



(a) E is selected as forwarder (instead of B as depicted in Figure 2(c))

(b) F broadcasts the message and algorithm finishes

Fig. 3. Another case of the scenario for the counting-based scheme

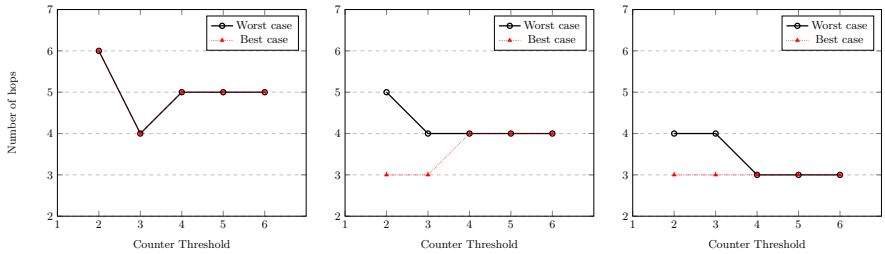
forward the warning message as shown in Figure 2(c). Following this case, the algorithm continues with vehicles D, H, and F being selected as forwarding nodes and rebroadcasting the message (Figures 2(d) to 2(f). As a result, it takes 5 hops for all the vehicles to get informed of the warning message. Note that the same scenario happens when C forwards the message before the expiration of the watchdog timer of E.

2. **The watchdog timer of vehicle E expires before receiving warning message from B and C:** In this case, since the counter of E is less than the threshold, E must forward the warning message (Figure 3(a)). In the next step, vehicle F broadcasts the message and all non-informed vehicles receive the warning message and algorithm finishes in 3 hops.

Achieving two different numbers for performance of this algorithm shows that beside correctness properties, providing guaranteed values for performance results requires applying formal verification techniques as well. We analyzed this scenario with different values for range and counter threshold, the result of three of them are shown in Figure 4. The results show that this phenomenon is not rare and can be observed in many cases.

4.3 Scalability Analysis

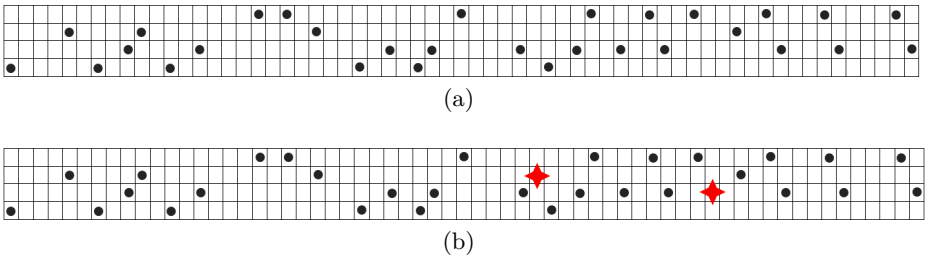
For the purpose of scalability analysis, we have modeled a four-lane street which contains about 30 vehicles. These vehicles are distributed in a way that there is no congested area in the street as shown in Figure 5(a). The execution time of this model is 11 seconds and the number of reached states and transitions are 19,588 and 110,627 respectively. To determine the scalability, we added new cars in two ways. First, we increased the length of the street and added new vehicles to the tail of the street of Figure 5(a). To avoid creating congested areas, we kept the same distribution while adding new vehicles. This way of scaling resulted in 15 seconds, 23,734 states, and 133,255 transitions for 35 vehicles and 18 seconds, 25,872 states, and 143,727 transitions for 40 vehicles (i.e. about 1.3 times more than the first case). As an estimation of the supported maximum size of the



(a) Range is 3 (b) Range is 4 (c) Range is 5

Fig. 4. Analysis results of the counting-based scheme with different values for the range and counter threshold (Note that Y axis shows the number of hops required for termination of the algorithm)

model regarding the state space size limit of Afra, the number of vehicles can be increased up to 100 if having distribution which does not create congested areas. In the second way, new vehicles were added in a way to increase congestion in some areas (Figure 5(b)). Scaling in this way increases the execution time of the model to 120 seconds and the number of reached states and transitions to 157,086 and 1,265,839, respectively (i.e. about 10 times more than the previous case). This is because of the fact that in a congested area, the number of delivered warning messages to each vehicle grows rapidly and all the possible orders of execution for messages with the same execution time are considered in the model checking. This results in a sharp growth in the size of the state space and model checking time consumption.



(a)

(b)

Fig. 5. Configuration of the scenario used for scalability analysis

5 Conclusion and Future Work

Lack of a framework for formal modeling and efficient verification of warning message dissemination schemes in VANETs is the main obstacle in using these

schemes in real-world applications. In this paper, we presented VeriVANca, an actor-based framework, developed using Timed Rebeca for modeling warning message dissemination schemes in VANETs. Model of schemes developed in VeriVANca can be analyzed using Afra, the model checking tool of Timed Rebeca. We showed how warning message dissemination schemes can be modeled using VeriVANca by implementing two of these schemes. Scenarios in these schemes were explored to illustrate the effectiveness of the approach in checking correctness properties and performance evaluation of the schemes. We further explained how easily the model of a scheme can be transformed to present another scheme by making minor modifications. Providing this level of guarantee in correctness and performance of warning message dissemination schemes, enables engineers to benefit from these schemes in the development of smart cars.

Considering different members of Rebeca family modeling language, VeriVANca can be used for addressing other characteristics of schemes such as their probabilistic behavior. Since Afra supports different members of Rebeca family, models with these characteristics can be analyzed using Afra.

VeriVANca can be used for the analysis of scenarios with limited congested areas. However, to be able to use the framework for large-scale models containing congested areas, we are going to develop a partial order reduction technique. This reduction relies on the fact that reaction of a vehicle to received warning messages is independent of their sender; therefore, different orders of execution (interleaving) for messages received at the same time can be ignored without affecting the result of model checking.

Acknowledgments

The work on this paper has been supported in part by the project "Self-Adaptive Actors: SEADA" (163205-051) of the Icelandic Research Fund and DPAC Project (Dependable Platforms for Autonomous Systems and Control) at Mälardalen University, Sweden.

References

1. Agha, G., Hewitt, C.: Concurrent programming using actors: Exploiting large-scale parallelism. In: Foundations of Software Technology and Theoretical Computer Science, Fifth Conference, New Delhi, India, December 16-18, 1985, Proceedings. pp. 19–41 (1985)
2. Aoxueluo, Wu, W., Cao, J., Raynal, M.: A generalized mutual exclusion problem and its algorithm. In: ICPP. pp. 300–309. IEEE Computer Society (2013)
3. de Boer, F.S., Serbanescu, V., Hähnle, R., Henrio, L., Rochas, J., Din, C.C., Johnsen, E.B., Sirjani, M., Khamespanah, E., Fernandez-Reyes, K., Yang, A.M.: A survey of active object languages. ACM Comput. Surv. 50(5), 76:1–76:39 (2017)
4. Gama, Ó., Nicolau, M.J., Costa, A., Santos, A., Macedo, J., Dias, B.: Evaluation of message dissemination methods in vanets using a cooperative traffic efficiency application. In: IWCMC. pp. 478–483. IEEE (2017)

5. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
6. Gholibeigi, M., Heijenk, G.: Analysis of multi-hop broadcast in vehicular ad hoc networks: A reliability perspective. In: Wireless Days. pp. 1–8. IEEE (2016)
7. Hafner, M.R., Cunningham, D., Caminiti, L., Vecchio, D.D.: Cooperative collision avoidance at intersections: Algorithms and experiments. IEEE Trans. Intelligent Transportation Systems 14(3), 1162–1175 (2013)
8. Khamespanah, E., Khosravi, R., Sirjani, M.: An efficient TCTL model checking algorithm and a reduction technique for verification of timed actor models. Sci. Comput. Program. 153, 1–29 (2018)
9. Khamespanah, E., Sirjani, M., Viswanathan, M., Khosravi, R.: Floating time transition system: More efficient analysis of timed actors. In: Formal Aspects of Component Software - 12th International Conference, FACS 2015, Niterói, Brazil, October 14-16, 2015, Revised Selected Papers. pp. 237–255 (2015)
10. Lin, S., Maxemchuk, N.F.: The fail-safe operation of collaborative driving systems. J. Intellig. Transport. Systems 20(1), 88–101 (2016)
11. Saeed, T., Mylonas, Y., Pitsillides, A., Papadopoulou, V., Lestas, M.: Modeling probabilistic flooding in vanets for optimal rebroadcast probabilities. IEEE Trans. Intelligent Transportation Systems 20(2), 556–570 (2019)
12. Sanguesa, J.A., Fogue, M., Garrido, P., Martinez, F.J., Cano, J., Calafate, C.M.T., Manzoni, P.: On the selection of optimal broadcast schemes in vanets. In: MSWiM. pp. 411–418. ACM (2013)
13. Sanguesa, J.A., Fogue, M., Garrido, P., Martinez, F.J., Cano, J., Calafate, C.M.T., Manzoni, P.: RTAD: A real-time adaptive dissemination system for vanets. Computer Communications 60, 53–70 (2015)
14. Sanguesa, J.A., Fogue, M., Garrido, P., Martinez, F.J., Cano, J., Calafate, C.T.: A survey and comparative study of broadcast warning message dissemination schemes for vanets. Mobile Information Systems 2016, 8714142:1–8714142:18 (2016)
15. Sirjani, M., Movaghar, A., Shali, A., De Boer, F.S.: Modeling and verification of reactive systems using rebeca. Fundamenta Informaticae 63(4), 385–410 (2004)
16. Suriyapaibonwattana, K., Pomavalai, C.: An Effective Safety Alert Broadcast Algorithm for VANET. In: 2008 International Symposium on Communications and Information Technologies. pp. 247–250. IEEE (oct 2008)
17. Tseng, Y., Ni, S., Chen, Y., Sheu, J.: The broadcast storm problem in a mobile ad hoc network. Wireless Networks 8(2-3), 153–167 (2002)
18. Yousefi, B., Ghassemi, F., Khosravi, R.: Modeling and efficient verification of broadcasting actors. In: Fundamentals of Software Engineering - 6th International Conference, FSEN 2015 Tehran, Iran, April 22-24, 2015, Revised Selected Papers. pp. 69–83 (2015)
19. Yousefi, B., Ghassemi, F., Khosravi, R.: Modeling and efficient verification of wireless ad hoc networks. Formal Aspects of Computing 29(6), 1051–1086 (2017)
20. Zeadally, S., Hunt, R., Chen, Y., Irwin, A., Hassan, A.: Vehicular ad hoc networks (VANETS): status, results, and challenges. Telecommunication Systems 50(4), 217–241 (2012)