Towards Formal Analysis of Vehicle Platoons Using Actor Model

Zeinab Sharifi
School of ECE
University of Tehran
Tehran, Iran
z.sharifi@ut.ac.ir

Ramtin Khosravi
School of ECE
University of Tehran
Tehran, Iran
r.khosravi@ut.ac.ir

Marjan Sirjani
School of IDT
Mälardalen University
Västerås, Sweden
marjan.sirjani@mdh.se

Ehsan Khamespanah
School of ECE
University of Tehran
Tehran, Iran
e.khamespanah@ut.ac.ir

***Abstract*—Vehicle platooning is a promising technology to save the road capacity and also fuel consumption by reducing the distance between the vehicles in the platoon. The closer the cars are to each other, the closer we are to the goals. But, this will increase the need for safety verification. In this paper we use formal methods to verify safety distance in a platoon. To do so, we present a formal actor-based model for a vehicle platoon which incorporates vehicle dynamics and communication protocol. Also, we present a method to do the analysis based on model checking that applies mathematical analysis to reduce the state space. The method uses an upper bound and a lower bound value as network delay, and verifies if a specified vehicle in a platoon has enough distance to the leader during its traveling.**

***Index Terms*—Vehicle Platoon, CAM, Formal Verification, Timed Rebeca, Actor Model**

I. Introduction

Vehicle platoon is a promising Intelligent Transport System (ITS) that has been studied for several years with the objectives such as efficient use of road capacity and reduced fuel consumption [1], [2]. A platoon system is a set of vehicles that drive closely behind one another. Each vehicle autonomously follows the one in front of it and a manually driven leader vehicle is in the first place of the platoon. Currently, vehicle to vehicle (v2v) communications and automated driving are two new technologies applied in these systems.

Cooperation in a vehicle network is achieved via periodic Cooperative Awareness Messages (CAMs) sent from each vehicle under the conditions drafted in ETSI EN 302 637-2 [3]. The conditions are based on the dynamics of the originating vehicle and are checked periodically at a certain sampling rate.

Achieving the specified objectives depends on keeping the distance between the vehicles in the platoon within an appropriate range. The smaller the distance, the more vehicles can travel in a road, and also the less fuel is consumed, since more aerodynamic drag is reduced. However, reducing the distance between vehicles leads to safety issues. Therefore, safety certification is an important concern in these systems [4]. Collision avoidance is one of the issues that has gained the most attention in the research concerning platooning. Various subjects related to this goal were studied in the literature, like ”time to collision” property [5], [6], safety distance [7], string stability [8] (the condition to ensure that disturbances in the control system of a platoon do not lead in vehicles collision [4]). Moreover, there are investigations on the impact of communication protocols on platoons through estimating the performance metrics of communication network, such as collision probabilities of the packets that are exchanged between vehicles to provide cooperative awareness, and data age of each vehicle (which refers to the time elapsed since the last received packet of a follower vehicle by the leading vehicle.)

In this work, we study the impact of CAM delivery delays on the distance between the vehicles. The followers in the platoon react to the decisions of the leader by accelerating or decelerating after receiving CAMs. Hence, different (and varying) message delays may cause the distance between the vehicles to fall outside the desirable range. As mentioned before, both lower and upper bounds of the distance are important as smaller distances raise safety issues and larger distances reduce the efficiency of the platoon. It is important to note that our analysis addresses not only different delays for different pairs of vehicles, but also varying delays, in the sense that the communication delay between two specific vehicles may not be always the same. Effectively, we check if the distance between the vehicles stays within a desirable and safe margin considering a lower and an upper bound on the network delay.

We use model checking, as a formal verification technique to make sure our analysis exhaustively covers all possible scenarios as opposed to simulation, which is more commonly used in this domain, but limits the analysis to some random scenarios. To the best of our knowledge, comparing to the existing works based on formal methods, our method incorporates more details about the vehicle behavior in the formal model used in the verification process. By examining combinations of different values for message delays, we abstract from different road traffic patterns and events and effectively cover a wide range of road traffics for the purpose of verification, in contrast to verifying some random traffic scenarios.

As a first step in our analysis, a formal model is presented for a vehicle platoon. The presented model incorporates the CAM protocol [3], and the dynamics of each vehicle according to the Intelligent Driver Model (IDM) [9]. IDM is a well-known car-following model [1]. The model also provides an abstraction of the inter-vehicular communication network.

Transmission delays in an inter-vehicular communication network may increase as a result of different weather conditions, different sizes of platoons, non-platoon vehicle disturbance, etc. For a platoon of a reasonable size, many CAMs may be exchanged during the analysis period. To investigate how different network delays affect the distance between the leader and the followers, a variation range for the transmission delay of a CAM is assumed. To enable the analysis in practice, we abstract the (infinite) possible values of the network delay into two discrete values which are the upper and lower bounds of the specified range. Whenever a CAM message is sent, the network delay is chosen to be one of the two mentioned bounds nondeterministically. In the following sections, we show that model checking the model using only these two values is enough to cover all the range between the two specified bounds. This will cover the (cumulative) effect of the variations in network delays on the distances between the vehicles.

We use Timed Rebeca (Reactive Objects Language) [11] [12] [13] as our modeling language. Timed Rebeca is an actor-based [14] language capable of modeling timing behaviors of asynchronous systems. An actor model contains a number of actors which communicate via message passing. Asynchronous communication model of Rebeca and its matching with the communication model of vehicle platoons together with the modularity of an actor model, leads to a natural and easy to understand model. This fact makes the model easy to evolve. Also, the Java like syntax and high abstraction of Timed Rebeca makes the model more readable. Moreover, we exploit the exhaustive verification using its supporting model checking tool that efficiently reduces the state space [12] [13].

In summary, the contributions of this work are:

1) Presenting a model for a vehicle platoon system based on actors: The model considers vehicle dynamics according to the IDM model and the cooperative awareness is provided by ETSI CAM protocol. Comparing to the existing formal approaches, to the best of our knowledge, our model of a platoon includes more details of the system. We also use the nondeterministic constructs in the language to cover more scenarios.
2) Presenting a method based on mathematical and formal techniques for checking safe distance in a range of delays: The method gives a range as network delay and checks the distance between the leader and a targeted follower of the platoon. This way, it investigates the impact of varying delays, which may be the result of the large size of the platoon or environmental events.

The remainder of the paper is organized as follows. After a brief review of the preliminaries in Section 2 and the related work in Section 3, we show how to model a vehicle platoon in Timed Rebeca in Section 4, and how to analyze the models in Section 5. The effectiveness of the method is illustrated using a couple of experiments in Section 6. Finally, Section 7 concludes the paper.

[1]Car-following model is used to determine how vehicles follow one another in a road. One of the goals of such a model is to adjust the speed of a vehicle so that its distance to the vehicle in front lies within a desirable range [10].

II. Preliminaries

A vehicle platoon is a group of vehicles traveling together on a road. It contains a leader vehicle, which drives in front of the platoon, and one or more followers. Details of platooning may vary among different solutions according to their goals and technical approaches [15]. Different technologies and devices are used in platooning systems, but it is common to contain local sensors, vehicle-to-vehicle (v2v) technology and automated driving support systems, to have a safe cooperation.

A. CAM Protocol

CAMs [3] are messages that are periodically exchanged in a vehicular network to provide cooperative awareness. Each vehicle sends a CAM to other vehicles, according to some triggering conditions which are defined in [3]. The triggering conditions are periodically checked at a specified sampling rate. A CAM contains kinematic data (such as position, velocity, and acceleration) and other data about the status of the originating vehicle.

According to [3], each vehicle must comply with the following limits of the transmission interval of CAM messages:

- The CAM generation interval shall not be less than $T_{Min} = 100ms$ (CAM generation rate of 10 Hz).
- The CAM generation interval shall not be more than $T_{Max} = 1000ms$ (CAM generation rate of 1 Hz).

CAM generation triggering conditions [3] shall be checked repeatedly every TS ($TS \leq T_{Min}$). The triggering conditions are as follows:

- the absolute difference between the current direction of the vehicle and the direction included in the latest transmitted CAM exceeds $4°$;
- the distance between the current position of the vehicle and the position included in the latest transmitted CAM exceeds 4 m;
- the absolute difference between the current speed of the vehicle and the speed included in the latest transmitted CAM it exceeds 0.5 m/s.

B. Vehicle Platoon Behavior

As stated previously, a vehicle platoon involves a leader followed by one or more vehicles. In the following, the behavior of the leader and the followers are explained.

1) Leader behavior: The leader vehicle is manually driven by a professional human driver. The driver continuously determines the amount of acceleration. The velocity and position change according to the acceleration. The relationship between these three parameters are as follows:

$$a = dv/dt \tag{1}$$

$$v = dx/dt \tag{2}$$

According to [3], the kinematic parameters are checked at a specified sampling rate. Whenever the changes in position and velocity reach the threshold specified in CAM protocol, a CAM message containing the velocity and position of the leader vehicle will be sent to all followers.

2) Follower behavior: As stated previously, each of the followers uses the IDM car following mobility model [9]. When a follower receives a CAM, it calculates the proper acceleration according to the position and the velocity of the leader, which is involved in the CAM. The acceleration is calculated using the following formula:

$$a_{IDM}(s, v, \Delta v) = a \left[1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right] \quad (3)$$

In the above formula a_{IDM} is the acceleration that a follower needs to keep a desirable gap to its front vehicle, a is the maximum acceleration constant and the value of v and s are the velocity and the distance of the follower vehicle respectively, δ is the acceleration power constant, v_0 is the desired velocity, Δv is the difference between velocities of the leader and the follower, and s^* is a function for calculating the desired distance:

$$s^*(v, \Delta v) = s_0 + vT + \frac{v \Delta v}{2\sqrt{ab}} \quad (4)$$

The function s^* depends on the velocity and the velocity difference of the leader and the follower. The constant s_0 is minimum desired gap between vehicles of a platoon (jam distance) and T is the time gap between two vehicles. We set the parameter values according to [9].

C. Timed Rebeca

Rebeca is an actor-based modeling language with a Java-like syntax for modeling distributed reactive systems. Actors can be considered as a reference model for object-oriented concurrent computation based on asynchronous communication. A Rebeca model consists of reactive classes and a main part that contains instantiation of reactive objects (rebecs) from reactive classes. Rebecs have encapsulated states and their own execution thread. Each rebec contains a set of state variables, methods and a set of known rebecs with which it can communicate. Communication is through asynchronous message passing. Sending a message to a rebec will cause the invocation of corresponding message server. Each rebec has an initialization method (like a constructor) with the same name as the rebec which is executed while instantiating of the rebec. Parameters of this method are used for initializing the rebec and its known-rebecs. Each rebec has a message queue, for buffering the arriving messages. The scheduling policy of queues is FIFO. State of a rebec consists of the valuation of state variables, and the messages in its message queue. In each step, a rebec is executed by removing a message from top of its queue and executing the corresponding message server. The execution of a message server is an atomic execution of its body and may not be interleaved with other message servers.

Timed Rebeca is an extension to Rebeca, capable of modeling timing features of distributed systems. In a Timed Rebeca model each rebec has its own local clock. The local clocks can be considered as synchronized distributed clocks. The execution of message servers is still atomic and can lead to progress of time in that rebec. To model timing behaviors of a system like computation time, message delivery delay, message expiration, and periodic occurrence of events, the three constructs are provided as follows:

1) $delay(t)$: causes a delay of t time units.
2) $after(t)$: paired with sending a message, causes the message to be sent with a delay of t units of time (it does not cause a delay for the sending rebec).
3) $deadline(t)$: paired with sending a message, the message will be purged from the queue after t time units.

Timed Rebeca is used for modeling and analyzing of distributed systems in different ways, for example in [16], schedulability analysis of wireless sensor networks is performed, different design decisions and routing algorithms in Network on Chips are analyzed in [17].

III. Related Work

One class of the related work use simulation techniques to analyze the performance of communication protocols in vehicle platoons. In [18], the authors use simulation tools to investigate the effect of sampling rate at which CAM conditions are checked. They show some scenarios that improperly increasing the sampling rate leads in more collisions in IEEE 802 11.p MAC (Medium Access Control) layer, hence decreasing the efficiency of CAM protocol. An analysis on the sending rate of CAM protocol is done in [19]. It compares network performance when sending CAMs, in two cases: when using the CAM sending conditions specified in ETSI standards, and when sending CAMs using a constant rate of 10Hz. It considers scenarios with different velocities and platoon sizes to investigate the effect of sending rate on network delay, especially for the last member of the platoon. In [20], the performance of ETSI CAM protocol is evaluated for two mobility patterns. The two patterns are similar in that the velocity of all vehicles changes together, hence, as the ETSI CAM protocol triggering condition is based on kinematic parameters of the originating vehicle, the communication network congests due to the phenomena of message synchronization. In [21], the authors explore the trade-off between the convoy message frequency and the communication performance, to do so they evaluate four communication metrics of convoys using this simulation framework. In contrast to our work, these works use simulation tools which are not as reliable as formal approach, as they only examine some random scenarios in the model.They analyzed the impact of a communication protocol on the performance of the inter-vehicular communication (as they investigate on network metrics), but not on the overall performance of a vehicle platoon.

Another class of the related work uses formal techniques to model and/or analyze vehicle platoons. Kamali et.al. [22] present a modular formal technique for verification of vehicle platoon as a hybrid system. They verify the system against both real-time and high level decision making aspects of a vehicle platoon (such as joining or splitting a platoon). Each of the aspects are verified separately and during the verification the other aspects are abstracted. They extended their work in [23] by adding a spatial controller and thus verifying spatial properties. In both of the works they did not model kinematic parameters of the vehicles (they use a simulator to model the controller) and the work assumes that each vehicle keep its distance to the next and previous vehicles.

[24] Proposes a framework for modeling cooperating vehicular systems. The main idea of the framework is separating low level and high level behaviors of the system and to model them in any flexible modeling language, like the work by [22] [23]. The framework is finally applied to model a vehicle platoon, using a simulator in its lower layer to mention the physical dynamics of each vehicle. The model was not verified against safety properties.

In [25], two models are proposed with two different approaches. The first model is a timed automata model which abstracts from the vehicle dynamics. It contains management operations of a platoon, such as splitting and joining. This model is verified against safe distance and some functional properties about a controller that they proposed. The second model is modeled in Webots simulator. It contains the dynamic of the vehicle and mainly used to assess the implemented controller quality. As they conclude, their Webots model provides better evaluation of the controller compared to their timed automata model, since the vehicle dynamics is abstracted in their formal verification.

Compared to these works, we capture the dynamics of vehicles together with the CAM protocol in our formal model. Having formally modeled the lower level behavior of the system we are able to apply nondeterministic constructs to model a wider range of scenarios for the purpose of verification. Also, we use an abstraction technique to reduce the state space and make exhaustive examination of the scenarios tractable. There already exist some tools that enable bounded model checking on range variables (e.g., [26]). Compared to these tools, our analysis targets a different goal. we applied mathematical reasoning to reduce the state space, regarding a platoon system's assumptions, as we aim to address a lot of independent range variables in our analysis (since a lot of CAM messages are sent from a leader to a follower, possibly with varying sending delays).

IV. The Actor Model of the Platoon

To start the analysis using model checking, the system must be modeled at first. We have chosen Timed Rebeca as the modeling language in this step since actor model is capable of naturally model a vehicle platoon. In a Timed Rebeca model we have independent actors communicating through asynchronous message passing, like in a vehicle platoon where vehicles communicate through asynchronous message passing.

To verify the property that the distance between the vehicles lies within the desired range, we must model vehicles, both the leader and the followers. For each of them it is needed to consider their dynamics and their behavior regarding sending CAM. We also need to capture the model of the communication channels and the topology of the platoon on the road. During a vehicle movement, kinematic parameters change continuously. But, as those parameters are checked at a sampling rate, we can model the changes with a discrete model. To do so, the value of each kinematic parameter can be calculated in each sampling.

The way we model each component, as well as the assumptions made to simplify the analysis, are stated below.

A. Model Assumptions

- We consider linear acceleration in our model (it can simply be extended to support other functions for acceleration).
- Vehicles are in the communication range of each other.
- Each follower adjusts its distance to the leader, whenever it receives a CAM from the leader.

B. Model of the Components

1) Leader: The leader is modeled as an actor, communicating to each follower through message passing. To model the dynamics of the leader, the leader reactive class contains three state variables as its kinematic parameters. The leader contains a message server namely `doSample` that models the behavior of periodically checking of the leader kinematic parameters to determine if it is time to send a CAM. It calculates the values of the kinematic parameters periodically at the specified sampling rate of 10 Hz.

As mentioned in the previous section, the equations needed to calculate the kinematic parameters are based on acceleration, thus a simple way to calculate the value of the parameter is to keep the latest time that the acceleration has changed. Then, the change in the value of each kinematic parameter is calculated from that point to the current time. The three kinematic parameters are calculated as follows,

- **Acceleration(acc)**: is defined by the user, as input of the model. As stated before, we assume acc changes linearly. Then, the user must specify the slope of the acc function (against time), whenever he/she wants to change it. The slope of the line is initially 0 and can be changed via calling a message server `changeAccCoef`.
- **Velocity(v)**: is calculated by integrating the acceleration function, during the time 0 to dt, where dt is the time between the latest time that acc has changed (`rcntTime`) till the current time. The constant value of the integral is the current value of velocity.
- **Position(x)**: is defined by second integral of the acceleration function, during the time 0 to dt, where dt is the time between `rcntTime` till the current time. The constant value of the integral is the current value of position.

After the calculation, the parameters are checked against CAM sending conditions. If they satisfy the conditions, a CAM is sent to all followers via `giveCAM`. To check the conditions, the leader model needs to keep the information about the recently sent CAM. Thus, it also contains three other variables to keep the recent CAM velocity, distance and sending time.

Listing 1 shows the pseudo code for Leader model in Timed Rebeca (the complete code can be found at Rebeca Homepage [27]). As shown in the pseudo code, the leader is declared as a reactive class (line 5). Four followers are defined as the known rebecs (line 8). The information needed to be kept as state variables (line 10) are: the kinematic parameters, information about the recently sent CAM, the slope of acceleration linear function and the latest time that the *acc* has changed.

In the constructor (lines 20-28), the initial value of acceleration slope (`ap1`) is set. Here, the leader driving scenario can be defined as input of the model, through calling `changeAccCoef()` message server and set it to be triggered after a desired time (line 24). The leader calls `doSample()` message server to start sampling itself.

```
1  env byte TSampling = 100; //mili seconds
2  env byte minDistance = 11;
3  env byte maxDistance = 30;
4
5  reactiveclass Leader(20){
6
7    knownrebecs{
8      Follower f1, f2, f3, f4;
9    }
10   statevars{
11     //values sent by the last CAM
12     int vCAM , xCAM, tCAM;
13     //a: acc, v: velocity, dx: distance
14     int a, v, x;
15     //our assumption: a = ap1 t + a0
16     int ap1;
17     //lastest time that ap1 has changed
18     int rcntTime;
19   }
20   Leader(int x0, int v0){
21     ap1 = 0;
22     v = v0/3.6; //km/h => m/s
23     //a = -2t + 0
24     self.changeAccCoef(-2) after(1000);
25     //a = -2
26     self.changeAccCoef(0) after(2000);
27     self.doSample();
28   }
29   msgsrv changeAccCoef(double Acc){
30     //calculate and save
31     calculateKinematicParams();
32     //check the distance to follower1
33     f1.checkDistance(x);
34     //setting the rcntTime
35     rcntTime = now;
36     //changing the slope of acc = ap1 t+ap0
37     ap1 = Acc;
38   }
39   msgsrv checkDistance(int xf) {
40     //saving in temp variables
41     int A = a; int V = v; int X = x;
42     calculateKinematicParams();
43     double dist = realScale(x - xf, x_scale);
44     //compare to minDistance
45     assertion(dist >= minDistance);
46     //retrieving the variables
47     a = A; v = V; x = X;
48   }
49   msgsrv doSample (){
50     int A = a; int V = v; int X = x;
51     calculateKinematicParams();
52     if(checkCAMConds() == true){
53       //saving latest CAM info
54       vCAM = v; xCAM = x; tCAM = now;
55       //nondeterministic network delay
56       int d1 = ?(1,5);
57       //sending CAM to all followers
58       f1.giveCAM(v, x) after(d1);
59       //...
60     }
61     a = A; v = V; x = X;
62     //sampling periodically
63     self.doSample() after(TSampling);
64   }
65   void calculateKinematicParams(){
66     dt = now - rcntTime;
67     //a0: current value of 'a'
68     a = (ap1*dt)+a0;
69     //integral of 'a'
70     v = (ap1*pow(dt,2)/2) + (a0*dt) + v0;
71     //second integerl of 'a'
72     x = (ap1*pow(dt,3))/6) + (a0*pow(dt,2))
         /2 + (v0*dt) + x0;
73   }
74   boolean checkCAMConds(){
75     int T = now - tCAM;
76     if (T >= 1000)
77       return true;
78     else if(T >= 100) {
79       if ((realScale(x-xCAM,x_scale) >= 4)
80         ||(realScale(v-vCAM,v_scale) >= 0.5)
81         ||(realScale(vCAM-v,v_scale) >= 0.5)
82       ) {
83         return true;
84       }
85     }
86     return false;
87   }
88 }
```

Listing 1. Psuedo Code of the `Leader` Reactive Class in the Platooning Rebeca Model

In the `changeAccCoef` message server, the kinematic parameters are calculated and stored. Because this point of the time is a changing point for kinematic parameters, as discussed before, we need to store the time and the kinematic parameters for later calculations. The slope of the acceleration function is then set. In the `doSample` message server the kinematic parameters are first stored in temporary variables (to be finally retrieved at the end of this message server, line 50). Then, their current values are calculated and checked whether they satisfy CAM sending conditions. If the conditions are satisfied, a CAM is sent to the first follower with a nondeterministic delay (lines 56 and 58). After that, the value of `vCAM` and `xCAM` is set and the recent values of the kinematic parameters are retrieved (line 61). Finally, a `doSample` message is scheduled to be processed after the sampling time.

2) Follower: Each of the followers are mapped to an actor in our model. According to the property we are checking, a follower just needs to communicate to the Leader. But, according to [4] there exist two ways for a follower to choose an acceleration. The first way is to adjust itself to the leader, and the second is to adjust to the front vehicle. We chose the first way for our analysis. Although our model is capable of considering CAMs between followers we have not included this feature for current analysis, and our properties refer to the distance of each follower to the leader.

A follower periodically checks its kinematic parameters to send CAM to its rear vehicle if the CAM sending conditions are satisfied. Thus, as Listing 2 shows, the body of the follower model is very similar to the leader model, but each follower adjusts its acceleration when it receives a CAM message from the leader according to the velocity and distance to the leader (line 31). The other two kinematic parameters are calculated like those of the leader.

```
 1 reactiveclass Follower(10){
 2   knownrebecs{
 3     Leader lead; Follower f;
 4   }
 5   statevars{
 6     int a, v, x;
 7     int rcntTime;
 8     int xCAM, vCAM, tCAM;
 9   }
10   Follower(int x0, int v0) {...}
11   boolean checkCAMConds() {...}
12   msgsrv doSample () {...}
13   msgsrv checkDistance(int xl) {...}
14   msgsrv giveCAM(int vl, int xl) {
15     // calculate & save
16     calculateKinematicParams();
17     lead.checkDistance(x);
18     // absolute value
19     double dv = vl-v;
20     double s = xLead - xSelf;
21     // idm parameters
22     double v0 = 120/3.6;
23     byte sigma = 4;
24     double maxAcc = 1.4;
25     byte b = 2;
26     byte s0 = 2;
27     double T = 1.5;
28     double ss = 0;
29     //4.3 = 2 * sqrt(ab)
30     ss = s0 + max(0, (vSelf * T) + (vSelf * dv) /
          3.4);
31     a = maxAcc * (1 - pow((vSelf / v0), sigma)
            - pow((ss / s), 2));
32     rcntTime = now;
33   }
34   //Completed and checked
35   void calculateKinematicParams(){
36     double dt = now - rcntTime;
37     //acc: a constant calculated in giveCAM
38     //v: the integration of acc in dt
39     v = (a0 * dt) + v0;
40     x = (a0 * pow(now,2))/2 + (v0 * currTime)
41         - ((a0 * pow(rcntT, 2))/2
42         + (v0 * rcntT)) + x0;
43   }
44 }
```

Listing 2. A Psuedo Code of the Follower Reactive Class in the Platooning Rebeca Model

3) Communication Channels and Topology: Channels can be modeled by message passing. The delay of v2v communication may vary depending on the network load, environmental conditions, cut-in situations, etc. We capture this fact by assuming a minimum and a maximum for the delay of the network and choosing the delay in the corresponding range nondeterministically (Listing 1, line 58). These two bounds can be modeled as the delay of calling a message server `giveCAM` using `after` construct. Listing 3 shows the instantiation of actors.

```
main{
  Leader L(F1):(100, 90);
  Follower F1(L, F1):(78, 90);
  Follower F2(L, F2):(66, 90);
  Follower F3(L, F3):(54, 90);
  Follower F4(L, F4):(42, 90);
}
```

Listing 3. Instantiating the reactive classes in the Timed Rebeca Model for Platooning

The leader and the four followers are instantiated in the main function. Their initial positions and velocities are passed as parameters. Each of the followers is passed to the leader, and also, the leader and the rear vehicle of a follower are passed to each follower as `knownrebecs`. This way, the topology of the platoon is defined.

V. Verification of the Model

In this section, we explain how to verify the model against the property that the distance *d* between the vehicles lies within a range, such that $d_{min} <= d <= d_{max}$ while avoiding state explosion.

The main challenge is that there are infinitely many values in the range of possible network delays. Even if we assume the interval contains discrete values, the possible combination of the values for different CAMs will cause state space explosion. To overcome this challenge, we let the leader nondeterministically choose between the upper bound and the lower bound of the possible network delay range before sending a CAM. As several CAMs are exchanged between the leader and the followers during their travelling, the model checker must cover all possible combination of nondeterministic choices for sending the CAMs. Here, a question arises: "How examining every combination of the upper bound and the lower bound for network delay can guarantee finding the minimum and the maximum distance points for the whole range?"

To answer the question, observe that in a vehicle platoon a follower is assumed to have a velocity close to that of the leader. Thus, the amount of Δv in formula 4 is negligible. Thus, by removing the last term of formula 4, the function defined in formula 4 and consequently the one defined in formula 3 will be monotonic relative to the velocity of the follower. Between two receiving CAMs, the velocity and position functions of a follower are monotonic in terms of time (since they are the first and the second integral of the IDM acceleration function which is kept constant during two CAMs). Therefore, according to the features of monotonic functions, the minimum and the maximum values of IDM acceleration, velocity and position of a follower in a time interval occur in the lower bound or upper bound of the time interval.

To perform verification against the specified property, the distance between the leader and a follower must be checked at every point in time in which their distance can be at the minimum or maximum. Now, we should answer this question: ”At which point in time the minimum and the maximum distance may occur?”

Let us call the position function of the leader and the follower x_L and x_F respectively. Their distance is $D = x_F - x_L$ which is always positive (according to the platoon system definition). The minimum and the maximum value of D during a time interval can be located at the first and the last points of the interval, and also in the extremum points of the function in the interval. The extremum points are the root of the derivative of D in the specified interval. As both x_F and x_L are position functions, their derivative is their velocity function.

As previously stated, the curve of the acceleration input of the model is composed of some consecutive linear parts. The integral of each part, the velocity function, is a polynomial function at most of degree 2. Therefore, the root of D can be easily calculated in each of the linear acceleration parts (also, for more complicated acceleration functions, Newton-Raphson method can be efficiently used to find the roots, as the specified interval is relatively small).

In our model, to divide the acceleration function of the leader and the follower into linear parts we must take into account all points in time at which the acceleration changes. As mentioned in the previous section, the acceleration of the leader is changed through calling the `changeAccCoef` message server. Then, we must check the distance between the leader and the follower whenever this message server is executed. Moreover, we must include all the points at which the follower receives a CAM, since it changes its acceleration at these points to adjust to the leader.

A. Verifying the Timed Rebeca Model

In the following, we will explain how we perform the verification in our Rebeca model:

To verify $dmin <= d$, we declare a message server called `checkDistance` (Listing 1, line 30) in both leader and follower models. In both models, `checkDistance` receives the position of the sending vehicle as parameter and compares it to that of itself. If the distance is greater than a threshold, an assertion fails and the execution path will be returned as a contradiction example. The leader sends the `checkDistance` message to the follower in its `changeAccCoef` message server (and passes its own position to the follower to be compared to that of the follower). Also, a follower does the same in its `giveCAM` message server. This way, we just check the distances in the first point and the last point of the intervals. To keep the model simple, we do not check the distance in those points at which the root of the velocity function is zero. As we assume the frequency of CAMs is relatively high, this does not make a significant error in our results. Adding this check to the model is not hard and will be included in the future steps.

Verifying $d <= dmax$ is just like the minimum distance. The only difference is the assertion in `checkDistance` message server.

VI. Experiments

We use Afra model checker [28] to perform a set of experiments on the system model to check the distance between the leader and a follower, during their movement. The analysis is done in a scenario for disruption maneuvers of a platoon which is introduced in [9]. In the scenario, all vehicles are running at the speed of 90 km/h for a while, then the leader slows down to 60 km/h because of an obstacle it faces, which may be a vehicle moving with a lower speed. After a short time, the leader turns back to 90 km/h and moves for a while.

In the first experiment, there is a 12 meters initial gap between the leader and the follower. The communication delay was considered to vary in the range of [1, 5] milliseconds. The delay was extracted from the experimental results reported in [21] as the communication delay for a platoon of size 5 to 40 vehicles.

The scenario was analyzed for different values of acceleration -1, -2, -4 and -8. In all of these cases the following property is satisfied: " Is the distance between the leader and the follower always at least 12 meters?"

In this experiment, the distance between the leader and the first follower was analyzed. To do so, we consider variable network delay that was assigned to the communication delay of different platoon sizes. This way we investigate the impact of large platoons on the distance between them. Thus, our method enable us to consider the impact of different platoon sizes without strictly modeling different sizes of platoons.

The second experiment is performed with the goal of showing that using a deterministic maximum or minimum network delay may not necessarily lead to the minimum or the maximum distance between two vehicles. To find the minimum and maximum distances, we verified the model against the two following properties:

1) Is the distance between the vehicles always at least d_{min} meters?
2) Is the distance between the vehicles always at most d_{max} meters?

To do so, we tried the model checking with different threshold as d_{min} and d_{max}. (Approximately, we need to verify the model log n times to find the maximum or minimum distance, where n is d_{min} or d_{max}.)

The experiment used the same scenario as the first one, but used an initial gap of 15 meters between the leader and the follower. The delay of sending CAMs was considered to vary in the range [1, 80] ms. Table. 1 illustrates the verification result of three scenarios with different delays.

The result shows that the minimum and the maximum distances between the vehicles may not necessarily occur in the cases that the network delay for sending all of the CAMs is maximum or minimum. Thus, we need an exhaustive method to examine all of the possible delays in a variation range of network delay to find the minimum and the maximum distances.

VII. Conclusion and Future Works

We presented a formal model based on actor model for a vehicle platoon that uses the well known IDM car-following model and CAM protocol for v2v communications. The model is analyzed to verify that the distance between the leader and a

TABLE I
THE MINIMUM AND MAXIMUM DISTANCES FOR DIFFERENT NETWORK DELAYS

	1 ms	80 ms	[1,80] ms
Minimum Distance	9m	12m	9m
Maximum Distance	66m	64m	75m

targeted follower always lies within an appropriate range. This enables us to efficiently use the road capacity while keeping the safe distance between the vehicles. To make the analysis practical, we assumed a possible range for the transmission delays and showed that (nondeterministically) choosing one of these two values for each transmission will cause both the minimum and the maximum distances between the vehicles to appear in the examined scenarios. We used model checking as the means of analysis to check every combination of the network delays of the messages. This allowed us to abstract from the events that affect the network delay, such as environmental events and the packet loss resulting from network congestion in large platoons. Our method can be applied to check whether the CAM protocol (or any other beaconing protocol) provides enough awareness for a platoon to reach its goals. The analysis results allow us to evaluate the communication protocol, or choosing appropriate values for its parameters.

We have used an approximation in computing the minimum and maximum distances. The accuracy of the results can be improved by including other comparison points (other extremum points), which is left as a future work. As another future work, we will investigate on how the results of the analysis of smaller platoons can be related to the larger ones. Since the network delay increases as the size of the platoon gets bigger, characterizing the relation between these two parameters may enable us to analyze bigger platoons by first analyzing a smaller platoon that uses the same communication protocol and then extend the results for larger platoons, probably through some approximations.

ACKNOWLEDGMENT

The work of the third and forth authors is supported in part by KKS SACSys Synergy project (Safe and Secure Adaptive Collaborative Systems), and KKS DPAC Project (Dependable Platforms for Autonomous Systems and Control) at Malardalen University, and MACMa Project (Modeling and Analyzing Event-based Autonomous Systems) at Software Center, Sweden.

REFERENCES

[1] L. Alvarez and R. Horowitz, "Safe platooning in automated highway systems part i: Safety regions design," *Vehicle System Dynamics*, vol. 32, no. 1, pp. 23–55, 1999.

[2] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, "A survey on platoon-based vehicular cyber-physical systems," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 263–284, 2016.

[3] "Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. ETSI EN 302 637-2 v1.3.2," 2014.

[4] J. Axelsson, "Safety in vehicle platooning: A systematic literature review," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1033–1045, 2017.

[5] V. E. Balas and M. M. Balas, "Constant time to collision platoons," *International Journal of Computer Communications & Control*, vol. III, pp. 33–39, 2008.

[6] M. Aramrattana, "A simulation-based safety analysis of CACC-enabled highway platooning," *PhD thesis, Halmstad Univercity Press*, 2018.

[7] A. Alam, A. Gattami, K. H. Johansson, and C. J. Tomlin, "Guaranteeing safety for heavy duty vehicle platooning: Safe set computations and experimental evaluations," *Control Engineering Practice*, vol. 24, pp. 33–41, 2014.

[8] X. Liu, A. J. Goldsmith, S. Mahal, and J. K. Hedrick, "Effects of communication delay on string stability in vehicle platoons," *ITSC*, pp. 625–630, 2001.

[9] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, pp. 4585 – 4605, 2010.

[10] R. W. ROTHERY, "Car following models," *Trac Flow Theory, Transportation Research Board*, 1992.

[11] M. Sirjani, A. Movaghar, A. Shali, and F. S. de Boer, "Modeling and verification of reactive systems using rebeca," *Fundam. Inform.*, vol. 63, no. 4, pp. 385–410, 2004.

[12] E. Khamespanah, Z. Sabahi-Kaviani, R. Khosravi, M. Sirjani, and M. Izadi, "Timed-rebeca schedulability and deadlock-freedom analysis using floating-time transition system," in *AGERE*, 2012, pp. 23–34.

[13] M. Sirjani and M. M. Jaghoori, "Ten years of analyzing actors: Rebeca experience," in *Formal Modeling: Actors, Open Systems, Biological Systems*, ser. Lecture Notes in Computer Science, vol. 7000, 2011, pp. 20–56.

[14] C. Hewitt, "Description and theoretical analysis (using schemata) of planner: A language forproving theorems and manipulating models in a robot," 1972.

[15] C. Bergenheim, S. Shladover, and E. Coelingh, "Overview of platooning systems," in *19th ITS World Congress*, 2012.

[16] E. Khamespanah, M. Sirjani, K. Mechitov, and G. Agha, "Modeling and analyzing real-time wireless sensor and actuator networks using actors and model checking," *Int. J. Softw. Tools Technol. Transf.*, vol. 20, no. 5, pp. 547–561, 2018.

[17] Z. Sharifi, M. Mosaffa, S. Mohammadi, and M. Sirjani, "Functional and performance analysis of network-on-chips using actor-based modeling and formal verification," *ECEASST*, vol. 66, 2013.

[18] N. Lyamin, A. V. Vinel, and M. Jonsson, "Does ETSI beaconing frequency control provide cooperative awareness?" in *ICC Workshop Proceedings*, 2015, pp. 2393–2398.

[19] A. V. Vinel, L. Lan, and N. Lyamin, "Vehicle-to-vehicle communication in c-acc/platooning scenarios," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 192–197, 2015.

[20] N. Lyamin, A. V. Vinel, M. Jonsson, and B. Bellalta, "Cooperative awareness in vanets: On ETSI EN 302 637-2 performance," *IEEE Trans. Vehicular Technology*, vol. 67, no. 1, pp. 17–28, 2018.

[21] I. Llatser, A. Festag, and G. P. Fettweis, "Vehicular communication performance in convoys of automated vehicles," in *ICC*, 2016, pp. 1–6.

[22] M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres, "Formal verification of autonomous vehicle platooning," *Sci. Comput. Program.*, vol. 148, pp. 88–106, 2017.

[23] M. Kamali, S. Linker, and M. Fisher, "Modular verification of vehicle platooning with respect to decisions, space and time," in *FTSCS*, vol. 1008, 2018, pp. 18–36.

[24] J. Campbell, C. E. Tuncali, P. Liu, T. P. Pavlic, Ü. Özgüner, and G. E. Fainekos, "Modeling concurrency and reconfiguration in vehicular systems: A π-calculus approach," in *CASE*, 2016, pp. 523–530.

[25] O. Karoui, M. Khalgui, A. Koubâa, E. Guerfala, Z. Li, and E. Tovar, "Dual mode for vehicular platoon safety: Simulation and formal verification," *Inf. Sci.*, vol. 402, pp. 216–232, 2017.

[26] S. Gao, S. Kong, and E. M. Clarke, "dreal: An SMT solver for nonlinear theories over the reals," in *CADE-24*, vol. 7898, 2013, pp. 208–214.

[27] "Rebeca model for vehicle platoons," 2020. [Online]. Available: http://rebeca-lang.org/assets/projects/Seada/case-studies/PlatoonMultiVehicles.rebeca

[28] "Rebeca formal modeling language." [Online]. Available: http://www.rebeca-lang.org