Statistical Verification Framework for Platooning Systems of Systems with Uncertainty

Sangwon Hyun, Jiyoung Song, Seungchyul Shin, and Doo-Hwan Bae
School of Computing
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, South Korea
{swhyun, jysong, scshin, bae}@se.kaist.ac.kr

***Abstract*—Platooning system is a well-known technology for alleviating traffic congestion and increasing fuel efficiency by grouping vehicles. It has the major characteristics of Systems of Systems (SoS), such as uncertainty. Several internal and external factors of uncertainty exist in the platooning system, such as car accidents, network disconnections, and simultaneous requests from other platoons. These factors make it difficult to guarantee that the system operates correctly in unpredictable scenarios and environments. The existing techniques used to verify the platooning system have two limitations: 1) the lack of consideration of uncertainty in scenarios and environments; 2) the application of exhaustive verification techniques which are vulnerable to the state-explosion problem. Thus, we suggest a statistical verification framework for a platooning SoS to address the above two limitations. The proposed framework automatically generates platooning configurations and scenarios considering the internal and external uncertain factors and bypass the state-explosion problem using a statistical verification technique. In this study, experimental results showed that the proposed approach generates 50% more valid scenarios than purely random strategy. In addition, we found two types of undiscovered failures and their causes in the VENTOS platooning system. These results indicate that our approaches enable the deep analysis of the platooning management system.**

***Keywords*—Platooning System of Systems, System Uncertainty, Statistical Verification**

I. Introduction

As network technology advances and systems have become larger and more complex, the interest in System of Systems (SoS) such as smart transportation, smart home, and smart plant systems has increased. The System of Systems (SoS) is a system in which the heterogeneous Constituent Systems (CS) cooperate to achieve a common goal through synergism [1]. When analyzing the behavior of an SoS, it makes the analysis difficult that the SoS dynamically reconfigures its structure or strategies in response to uncertainties within or outside the system. The unexpected changes within the SoS such as adding, removing, and modifying CSs, cause the internal uncertainty of the system. In addition, changes in the environment cause the external uncertainty of the SoS, such as network disconnections, and unexpected behavior of environmental objects [2], [3].

The platooning system, which is an representative example of an SoS, has recently gained attention because it is considered a fuel-efficient strategy for alleviating the traffic congestion of driving vehicles. The platooning system groups

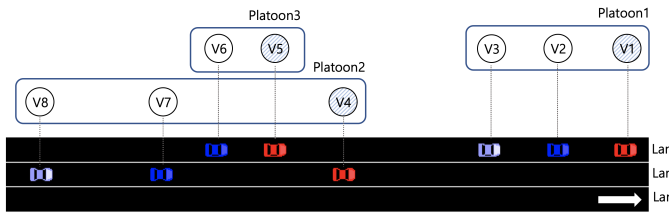

Fig. 1. An example platooning scenario execution in VENTOS.

vehicles and drives them with narrow intervals as depicted in Fig. 1. In the Fig. 1, there are two platoons in lane 1 and one platoon in lane2. The platoon 1, 2, and 3 consist of three, two, and three vehicles, respectively. Vehicles with the ids 1, 4, and 5 are the leaders of the platoons and they decide the driving strategies in each group. The leaders and followers in the platoons communicate by the Vehicular Adhoc Network (VANET) [4] to request operations such as *Merge* and *Split*. There exist several uncertain factors within or around the platooning system, because of its complexity and constantly changing environment. The *internal uncertainty* of the platooning system occurs in a sequence of platooning system operation such as simultaneous *Merge*, *Split* requests from other platoons or constituent vehicles. The *external uncertainty* occurs in the environment surrounding the platooning system, such as vehicle accidents, network disconnections, and unexpected lane change of Human-Driven Vehicles (HDV). To deal with these uncertainties of the platooning system, it is necessary to verify the system with considering the uncertainty factors in the verification process.

The verification of a platooning SoS currently relies on simulating few specific scenarios that are generated manually. Much time and effort are required to analyze the vehicle operation data to generate realistic scenarios [3]. Moreover, the generated scenarios do not address sufficient uncertain factors that can occur in the platooning system. Meinke et al. [5], for example, verified a platooning system with generating scenarios, but only use a single *Speedchange* event. Kamaili et al. [6] simulated a platooning protocol on a spacing and timing constraint without considering external environments like HDVs. In addition, most of the studies verified a platooning system by applying exhaustive approaches which are vulnerable to the state-explosion problem. This problem negatively affects to address uncertain factors of the platooning system by limiting the scalability of the verified system. For instance, Elgharbawy et al. [7] exhaustively check all traces of the automated truck driving system, but it only simulates a single vehicle in the verification. Therefore, to verify the platooning SoS, we need to consider the internal and external uncertainties of the system and bypass the state-explosion problem.

We suggest a StarPlateS, which is a statistical verification framework for a platooning SoS that effectively deals with the uncertainty issues in the system. We extend a VENTOS simulator [4] by implementing a scenario generation module, simulation module, and verification module. The scenario generation module addresses internal uncertain factors of the system by generating random configurations and scenarios. We suggest a condition-based scenario generation approach to minimize the number of invalid and meaningless scenarios. Then the simulation module executes the system with stochastic environmental objects. We add heterogeneous HDVs in the simulation to cover the external uncertain factors of the system. Finally, the verification module applies a statistical model checking technique to verify the system and to alleviate the state-explosion problem.

In experiments, we first checked the feasibility of the scenario generation algorithm by comparing it with a purely random strategy. The result showed that the suggested approach generated more than 50% of valid scenarios instead of the purely random generation strategy. Next, we verified the platooning management system in 20 cases of configurations and scenarios. In the verification of average speed maintenance, we found that a deficiency of *Lanechange* operation negatively affected the maintenance of the speed in the system. In addition, we found a "busy-leader" scenario pattern that always causes the rejection of the *Merge* operation.

This paper is organized as follows. Section 2 explains the background. Section 3 elucidates the main features of the framework, and Section 4 describes implementation details. Section 5 presents the experimental results of this framework. Section 6 describes the related works of this research. Section 7 discusses the implications of our findings, recommends directions for future work, and concludes the paper.

II. Background

A. Platooning as an SoS

The platooning system has been studied to increase the capacity of roads [4], [8]. The system is used to group vehicles into platoons with a common purpose. The head of the platoon becomes the leader, and the other members of the platoon become the followers. The platooning system is based on the Cooperative- Adaptive Cruise Control (C-ACC) technique, and it enables that vehicles in a platoon actively communicate with other vehicles to make synergism in their cooperation. For example, it enhances the fuel economy of the vehicles by reducing air resistance. Narrow intervals between the vehicles reduce the total amount of air resistance acting on the entire platoon. These intervals also have positive side effects, such as reducing traffic congestion and the number of traffic collisions.

A platoon has representative five characteristics of SoS: *Autonomy*, *Belonging*, *Connectivity*, *Diversity*, *Emergence* [1]. First, the vehicles in a platoon have the *Autonomy* to decide whether to join a new platoon, or leave current platoon for its own purpose. Second, vehicles *Belong* to a platoon to achieve specific goals, such as gaining economic advantages. Next, vehicles in a platoon are *Connected*. Not only does every vehicle inside a platoon communicate but also platoons communicate with each other. They maintain inter-platoon and intra-platoon spaces by continuously communicating. Fourth, the vehicles in a platoon are *Diverse* because they include heterogeneous types, such as trucks and passenger vehicles with two types of roles: a leader and followers in a platoon. The last characteristic is *Emergence*. The cumulative actions and interactions of vehicles and platoons result in achieving common goals such as the reduction of traffic congestion.

In this work, we use a notable and open traffic simulator, VENTOS, which provides a platooning management system with the five characteristics of an SoS. The VENTOS consists of the OMNET++ network simulator [9] and the SUMO traffic simulator [10]. The VENTOS provides a platooning management system with hierarchical operation protocols which contains *Merge*, *Split*, and *Leave* operations [4]. The *Merge* operation is conducted when the leader in the front platoon approves a request from the leader in the back platoon. In the *Split* operation, after a follower in a platoon requests a *Split* from a platoon leader, if the leader approves it, the platoon is divided into two platoons. In the *Leave* operation, a vehicle leaves a platoon after approval by the platoon leader.

B. Statistical model checking

To determine whether the system satisfies the properties, the Statistical Model Checking (SMC) technique is used to simulate and monitor the system. It is based on a hypothesis testing to provide statistical evidence for judging the property [11]. Because SMC is less intensive in time and memory than exhaustive approach, it is used to verify large and complex systems [12]. When the verification property ϕ, the system and environment model M and E are used to perform SMC, the probability $Pr(M, E \Longrightarrow \phi)$ is calculated statistically.

The satisfaction of simulation execution trace σ in verification property ϕ is checked by the verification module in the proposed framework, which can handle Probabilistic Computational Tree Logic (PCTL) properties. An example of a PCTL property is as follows: "$P\ =?\ [trueU\ <= 100(num\_vehicle\_platoon\_A)\ >\ 5]$". This example could be used in the platooning system scenario, where it checks whether the number of vehicles in the platoon A is more than five within 100 seconds in each trace. Then the verification module returns the result of the trace ($\sigma \models \phi$).

In this framework, the verification module calculates the number of traces that are statistically sufficient for a reliable verification result using the Sequential Probability Ratio Test (SPRT) algorithm. To bound the error probability of the probabilistic result, false positive probability α and false negative probability β are used as two precision parameters [13].

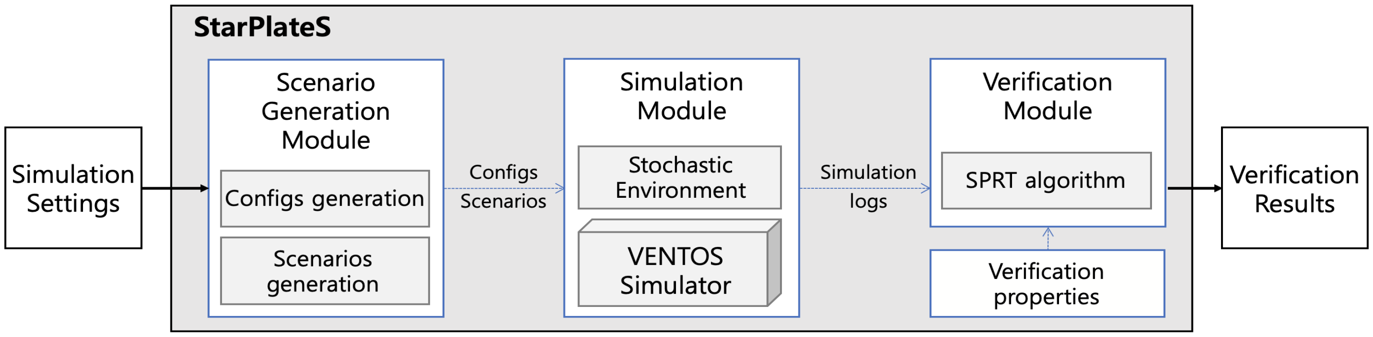

Fig. 2. Overall architecture of the verification framework.

III. Statistical Verification Framework of Platooning SoS: StarPlateS

In this section, we explain the main features of this framework. Fig. 2 shows the architecture of the framework which is composed of the scenario generation module, simulation module, and verification module. The proposed framework, StarPlateS operates as follows. First, the scenario module generates random configurations and scenarios of the platooning SoS using condition-based approach. Then, the simulation module executes the system on the generated scenarios with stochastic environment. Finally, the verification module applies a statistical model checking technique, especially the SPRT, and returns the verification results for each configuration and scenario. The following parts detail each component of the framework in sequence.

A. Simulation settings

There are several options for executing a simulation: simulation time, repetition numbers, verification options, duration between events, and GUI. We define these options as simulation settings. There are two modes of simulation settings according to the use of the StarPlateS framework. First, users can use our framework in ‘verification mode’. In this mode, users set high repetition numbers, such as 1,000, and the verification option that is true for statistical verification. Another way of finding configurations and scenarios with specific purpose is called the ‘single simulation mode’. In this mode, users assign the repetition number as 1 and set the verification option at false; thus, the framework generates as many scenarios as possible in the given amount of time. For example, we use the single simulation mode to compare two scenario generation approaches and the verification mode to conduct statistical model checking. The simulation settings are utilized in the scenario generation module and simulation module.

B. Scenario generation module

In the scenario generation module, the main goal is to generate diverse scenarios and configurations to address the internal uncertainty of the platooning system. Prior to explaining the module, we define *a configuration* as *a set of platoon generation features* and *a scenario* as *a sequence of operations on the created vehicles*. This module first generates random

Configuration
```
<vehicle_platoon id="veh1" type="TypeCACC1" size="6" route="route1"
departPos="100" departLane="0" platoonMaxSpeed="0" pltMgmtProt="true"
 optSize="4" maxSize="10"/>
<vehicle_platoon id="veh2" type="TypeCACC1" size="4" route="route1"
departPos="100" departLane="2" platoonMaxSpeed="0" pltMgmtProt="true"
 optSize="3" maxSize="8"/>
```
Scenario
```
<speed id="veh1" begin="5" value="20" />
<speed id="veh2" begin="5" value="30" />
<pltSplit pltId="veh1" splitIndex="3" begin="25" />
<optSize begin="45" value="5" />
<pltLeave pltId="veh2" leaveIndex="1" begin="65" />
<pltMerge pltId="veh1.3" begin="85" />
…
```

Fig. 3. An example of platoon configuration and scenario.

platoon configurations, and then generates scenarios based on each configuration.

When generating configurations of the platoons, We assign random values to every parameter of the platooning configuration. For example, in the configuration part of Fig. 3, the first generated platoon consists of “6” homogeneous cars with id “veh1” created in the “0” lane of “route1” at position “100”. Its optimal size is “4” and its maximum size is “10”. The second configuration shows the “4” sizes of the platoon with id “veh2” created in the “1” lane of “route1”. Its optimal size is “4”, and its maximum size is “8”. The pltMgnProt option is used to check whether the vehicles use platooning management protocols or not. In this work, we assume that the platoons always use management protocols; thus, the value of pltMgnProt is always “true”. We also add heterogeneous types of vehicles to generate configurations. For example, Fig. 4 shows a heterogeneous platoon that consists of a truck leader (V5) and three following passenger vehicles (V6, V7, and V8).

Next, to generate various scenarios, this module uses a condition-based approach to generate the scenarios for each configuration. A key point of the proposed approach is the pre-simulation of the scenarios to prevent the generation of meaningless and invalid scenarios by using the conditions and actions of each event. For example, if the module selects events in a purely random way, there could be meaningless scenarios, such as executing a *Split* operation on a platoon with size 1, and invalid scenarios, such as executing a *Leave* operation to a vehicle which has already left the platoon.

To alleviate this problem, we first define the available events and their conditions and actions in the platooning system.

TABLE I. Utilized events with its types, conditions, and actions

Event Types	Events	Condition	Action
Vehicle management	1) Speed change	None	None
Platoon management	1) Platoon merge	Two platoons exist in the same lane	Merge platoons
	2) Platoon split	A platoon with more than size 1 exists	Split the platoon into two platoons
	3) Platoon leave	A platoon with more than size 1 exists	Decrease the size of the platoon
Policy management	1) Optimal size change	None	Split all platoon with size more than the optimal size

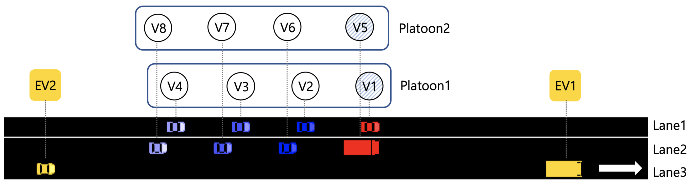

Fig. 4. Generated platoons and Humman-Driven Vehicles (HDVs) in StarPlateS.

Table I shows five event types, such as vehicle management, platoon management, and policy management events as well as their conditions and actions. The *Speedchange* operation changes the speed of the leader of the platoon, and the *Optimalsizechange* operation changes the optimal sizes of all platoons. This operation causes *Split* events in the platoons that are larger than the assigned optimal size.

Considering the condition and action of each event, this algorithm generates a status set in each step of the scenario generation. For example, according to the configuration of the two platoons shown in Fig. 3, the initial status set is as follows: {“veh1:6”, “veh2:4”}, which describes the platoon ids and their sizes in the initial state. Next, the proposed approach randomly selects an available event by comparing the current status set with the condition of each event. In the example, all but *Merge* operations are available, because two platoons in the same lane don’t exist. Then, the algorithm randomly selects the *Split* event among the available events as shown in the Fig. 3[1]. Then the attributes of the *Split* operation are randomly assigned. In the example, the *Split* operation is executed in the platoon “veh1” with an index of “3” at “25 seconds”. Using these attributes and actions, the algorithm updates the status set. Because the *Split* operation divides “veh1” into two platoons from the “veh1.3” vehicle, the example status is changed to {“veh1:3”, “veh1.3:3”, “veh2:4”}. In this way, we can select an available event after specific sequences of events to successfully generate valid scenarios. Fig. 3 shows a scenario that consists of *Split*, and *Optsize*, *Leave*, *Merge* with the duration of 20 seconds.

C. Simulation module

After the scenario generation module returns the sets of configurations and scenarios, the simulation module executes the platooning management system on the configurations and scenarios using the VENTOS. This module performs two main functionalities to generate execution traces for each configuration and scenario, and to address the environmental uncertainty of the platooning SoS. In addition, as mentioned in subsection A, there are two types of simulation modes: “verification” and “single simulation”. These modes repeat the same configurations and scenarios many times with verification and only one time without verification, respectively.

To address the external environmental uncertainty problems in the platooning system, this module adds stochastic environmental objects, such as HDVs, which are not communicatable with C-ACC vehicles; thus, they cannot anticipate the movement of the HDVs in the simulation, which randomly change lanes and speeds, and even stop randomly. These features of HDVs could make collision of vehicles in the simulation. Fig. 4 shows generated HDVs with ids of “EV1” and “EV2”. “EV1” is a truck vehicle and “EV2” is a passenger vehicle. In addition, in this module, users can change the generation period of the HDVs. Therefore, users can execute the simulation with various environments, such as rush hour or an empty road by changing the vehicle generation period. With stochastic environmental objects, this module finally returns the executions traces of each configuration and scenario to the verification module.

D. Verification module and results

Lastly, the verification module applies the statistical model checking algorithm to check the achievement rates of specific goals. The verification module needs verification properties of the platooning SoS which are related to the goals of the system. Previous research [6] provided the formal definition of the verification properties in a platooning management system. However, in this definition, the properties are focused on verifying the performance of basic operations of the system without considering the macro-level goals, and assume that there is only one platoon in the simulation. Because our framework assumes that there are more than two platoons in the simulation, and it attempts to verify systems with high-level goals, the existing properties do not match our verification goals. Therefore, we defined new verification properties that are appropriate for verifying the high-level goals in a multi-platoon situation.

The verification properties of the platooning system can be written in property specification languages, such as Linear Time Logic (LTL), and Computational Tree Logic (CTL), Probabilistic CTL (PCTL) [14]. We chose the PCTL verification property specification language. The defined properties are as follows:

[1]In the VENTOS, there is an implementation issue that it returns an error when platoons move at 0 second in a simulation. Thus, in the VENTOS manual, they suggest to assign the platoon with speed 0 at first and change its speed at specific times. Therefore, the first two *Speedchange* events are automatically generated in Fig. 3

1) $P =? \ [F <= t \ num\_passed\_veh\_P > n]$
2) $P =? \ [(op\_reject\_rate > x) \ U \ sim\_Terminate]$

We defined these two properties to check the CS-level goal, which checks arrival of participant vehicles, and SoS-level goal, which checks the success rate of the operation in the platoon. The meaning of the first verification property is "the probability that more than *n* cars passed through the specific point *P* within the first *t* seconds". This verification property checks whether the average speed of the platoon is maintained by the end of the simulation. For example, in the experiment, we set the average velocity of all generated platoons at 20 m/s and assigned the *P* value to 1,800m point to check the average speed maintenance. The second property checks "the probability that the rate of $operation\_reject$ is over x before the simulation terminates". In the experiments, we assigned the value x to 0 for checking the existence of $reject$ signal of the operations. Thus, by using this property, we can see how smoothly the platoon operations were done at the occurrence rate of the $reject$ signal.

With the two verification properties, we applied a statistical verification technique, the SPRT [13] algorithm which gradually checks the achievement rate of a specific property. Most of the existing research tried to verify the platooning system with simulating a single platoon because of the state-explosion problem. In contrast, we overcome this limitation by applying the statistical approach. Thus, we used 2 to 4 platoons which consist of 3 to 6 vehicles respectively with more than 20 HDVs to verify the platooning system in the VENTOS.

IV. Implementation

We implemented the framework on Ubuntu 16.04, as the VENTOS was optimized on the OS version. As we described above, our framework consists of three modules: a scenario generation module, a simulation module, and a verification module. We implemented these modules in a wrap-up system of VENTOS, and we open the details of the implementation in our Github[2].

We decided to use the VENTOS simulator and its platooning system for three reasons. First, VENTOS is based on the SUMO traffic simulator, and it has many advantages for traffic simulation. For example, it is easier to generate a map for traffic simulation using SUMO than by using other simulators, such as TORCS [15]. After a specific map file is downloaded from the web, the map file can be easily imported to SUMO by its own module. Second, VENTOS has been used in several previous studies on traffic simulation [16]–[18]. Thus, up-to-date techniques can be used in a simulation, such as an online vehicle routing algorithm and accident control. Third, in VENTOS, there are various kinds of vehicle models and hierarchical platooning management protocols [4]. VENTOS enables users to generate heterogeneous types of vehicles in a simulation. It also contains hierarchical platooning management protocols that consist of 5 operations and 17 micro-commands. These operations are used to simulate basic events in the scenario generation module, and the micro-commands are used to examine logs in the verification module.

[2]https://github.com/abalon1210/StarPlateS

The proposed framework set a scenario generation module to modify input files of the VENTOS simulator, which contains the configurations of the platoons and scenarios. The module accesses the files and changed them to newly generated configurations and scenarios. In the implementation of this module, one issue arises in the lane change event: it is not enabled to change the lane of a platoon in the VENTOS simulator. The developers of the VENTOS simulator blocked the functionality of *Lanechange* in all vehicles in platoons. Therefore, we could not generate *Lanechange* events in this scenario generation module. Instead, we resolved this issue by generating random configurations of platoons, especially with randomly assigned values of lanes and routes. This module generates various platoon configurations with diverse scenarios. Thus, it effectively covers the internal uncertainties in the platoon management system.

Before executing the VENTOS, the simulation module adds stochastic environment objects to the input files. We use a vehicle flow generator to continuously generate the stochastic objects. This function enables the generation of various types of vehicles within a specific period. For example, in the experiments, we generated a passenger vehicle and a truck vehicle every five seconds. These generated vehicles could not communicate with the platoons because we turned off the communication setting in the vehicles. As shown in the Fig. 4, platoon 1 consists of four homogeneous vehicles with leader(V1), and platoon 2 consists of truck leader(V2) and three passenger followers. There are two HDVs: a truck(EV1) and a passenger vehicle(EV2).

After generating simulation traces, the framework runs the verification module to check the achievement rate on the verification properties. We used existing verification property patterns and checkers from a SIMVA-SoS [19]. SIMVA-SoS provides the abstract functions of seven verification property patterns and its checkers which use the SPRT statistical verification algorithm. By using these abstract functions, we generated verification properties and checkers for the platooning management system.

V. Experiment

A. Experimental setup

We performed experiments to determine the effectiveness of the condition-based scenario generation algorithm which covers the internal uncertainty of the platooning SoS and to analyze the verification results of the platooning management system in the defined verification properties in order to address that covering the uncertainties of the system enables the deep analysis of the platooning management system. The experiments were designed to answer the following research questions:

- **RQ1.** Does the random scenario generation algorithm effectively alleviate meaningless and invalid scenarios?
- **RQ2.** Does the framework enable further analysis of the platooning SoS by addressing uncertainties?

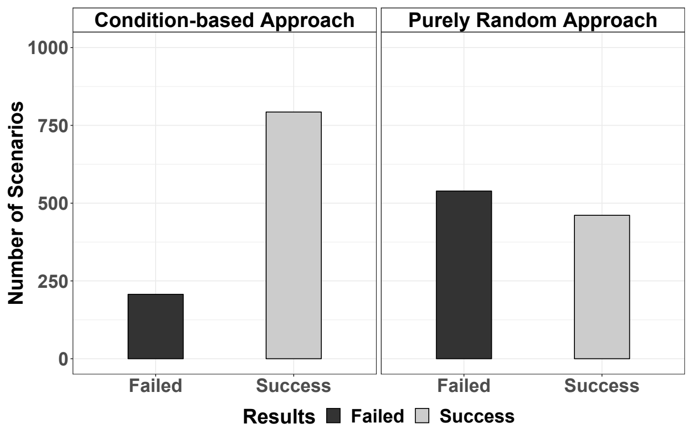

Fig. 5. Comparison results of the generation of 1,000 scenarios.

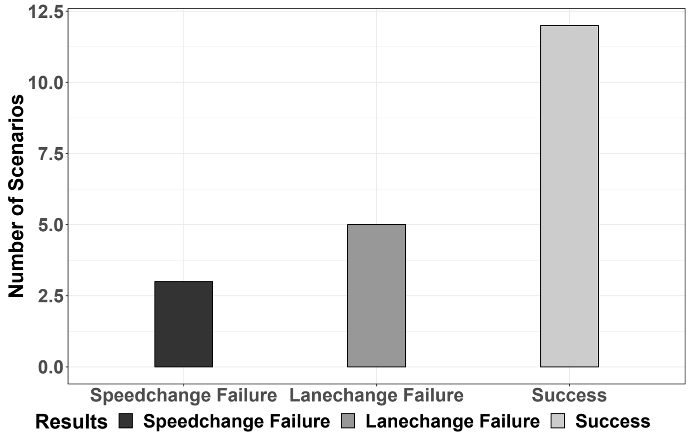

Fig. 6. Verification results of average speed maintenance.

First, to answer RQ1, we compared the condition-based algorithm with the purely random algorithm. We generated 1,000 scenarios, which contain five events with 20 seconds interval and checked the availability of the scenarios by checking whether the scenarios were executed without any run-time errors on the VENTOS. Next, to answer RQ2, we described and analyzed the results of each verification property. We set the first verification property to check "whether 80% of the vehicles in the platoon arrived at the point of 1,800 m in 100 seconds". We set the average speed of the vehicles at 20 m/s and checked whether the average speed was maintained in the generated scenarios and environment. We indirectly checked the average speed by extracting the positions of the vehicles at the end of the simulation. In this property, we set the standard number of vehicles at 80% of the generated platoon because the numbers of generated vehicles were different in each configuration. In the second property, we checked "whether rejection commands are occurred in the simulation". We checked the rejection commands of the *Merge*, *Split*, and *Leave* operations in the execution logs, and we analyze the specific configurations and scenarios that make the rejection commands.

To answer the research questions, we used the following settings in the experiments:

- Number of generated platoons: 2∼4
- Size of platoons: 2∼6
- Stochastic HDV generation: 1 vehicle per 5 seconds
- Simulation time: 100 seconds

The experiments were executed on an infinite size of a one-way road with three lanes. The generated platoons and HDVs were randomly assigned to one of the three lanes. Platoons are randomly generated with 2 to 4 sizes consisting of 2 to 6 vehicles. We set the HDV generating duration five seconds. Therefore, there were 20 HDVs at the end of the simulations. We also changed the non-accident option to enabing accidents.

B. Experimental results

In the experiments performed to answer RQ1, we compared the condition-based approach with a purely random approach to generate scenarios. We generated 1,000 scenarios and executed them in the VENTOS simulator to check their availability. As shown in Fig. 5, the condition-based approach generated more than 50% of the valid scenarios compared with the purely random approach. By examining the failed cases using the purely random method, we found that most of the failed scenarios were caused by the *Operation parameter error*, which requests an operation with an invalid vehicle id to a platoon, such as sending a *Leave* operation to the vehicle that has already left. This error occurs mainly in the *Leave* and *Split* operations because we found that the errors of the *Merge* operation are handled as an exception in the platooning system. We further checked the successful scenarios in two generation approaches. We found that no scenario contained meaningless operations using the condition-based approach, but 126 scenarios, almost 30% of the successful scenarios using the purely random approach had meaningless operations, such as *Merge* operation to a participant of a platoon. Therefore, in the condition-based approach, more than 50% of available scenarios were generated and the condition-based approach did not have any meaningless operation in the available scenarios, while the purely random approach generated 30% of available scenarios involving meaningless operations.

Fig. 6 shows the verification results of the platooning management system for the first verification property, which checked the average speed of the generated platoons. We generated 20 configurations and scenarios, and repeated each simulation 1,000 times. We generated about 100 GB of execution logs, which are used to verify and analyze the platooning system. We identified the success and failure of the verification results by whether they achieved 100% or not. As shown in Fig. 6, 12 scenarios, about 60% of the generated scenarios achieved that more than 80% of the vehicles in the generated platoons passed the 1,800 m point, which means that they maintained the average speed: about 20 m/s. We analyzed the failed cases and found that most of them were caused by accidents on the same lane with one of the generated platoons. If a vehicle had a collision, it took some time to clear the road. However, as we mentioned in section IV, in this platooning system, there is no *Lanechange* operation to manipulate the platoons. Therefore, the platoons in the same lane in which

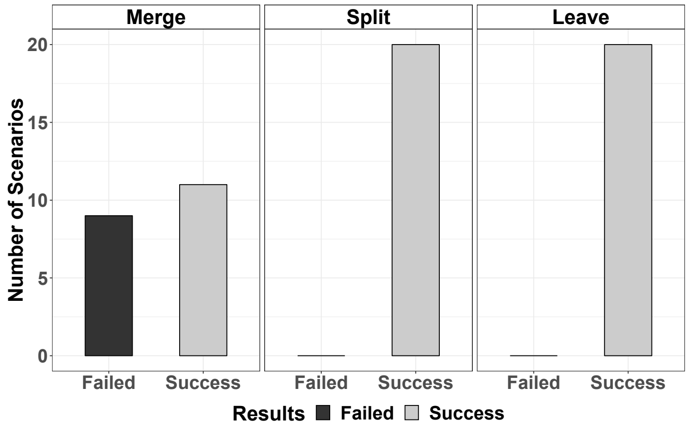

Fig. 7. Verification results of rejection commands in each operation.

an accident occurred had no choice but to wait for the debris to be cleared. It is a limitation of the platooning management system provided by VENTOS, and it reduces the realism of the simulation. The other failed cases were caused by the speed change operation in the generated scenario, which reduced the speed of the platoon to much lower than 20 m/s.

Regarding the verification of the success rate of operations in the platooning management system, Fig. 7 shows that in the 20 cases, rejection commands were detected only in the *Merge* operation. By examining the execution logs of the failed cases, we found that detected rejections in the *Merge* operation occurred when 1) the leader of the front platoon was in the process of another operation, such as *Split*, *Leave*, or *Merge*, 2) the leader is changed to follower or disappeared after the operation. In this case, the behind leader continuously requests Merge operation to the vehicle which is not a leader any more. This "busy-leader" problem indicated the presence of performance-degrading factors in the sequential execution of operations. In addition, we found that the requesting vehicle kept sending messages to the vehicle. This repetition could result in a dead-lock problem if another event were operated in the requesting vehicle.

Next, we determined the reasons that the rejection commands in the other operations were not detected. The main reason was that the condition-based approach was not a guided approach; thus, it was difficult to generate specific rejection scenarios among 20 cases. Another potential reason is that the conditions for rejecting other operations were more complicated than the *Merge* operation.

C. Threats to validity

In the experiment for RQ1, we suggested two factors, availability and meaningless to compare the algorithms. However, we only used the meaningless factor as an additional evaluating factor because it is infeasible to distinguish between scenarios involving meaningless operations and scenarios only consisting of meaningless operations in manual way. Therefore, we mainly used the availability of the scenarios and checked whether the success scenarios involve the meaningless operations or not. In addition, there were still failed scenarios using the condition-based algorithm in Fig. 5. They were mostly caused by the optimal size change operation. In the algorithm, it checks the optimal size of each platoon in the beginning and execution of *Optimalsizechange* operation. However, there is an additional condition when platoons are merged and exceed the optimal size of the front platoon. We will improve the algorithm in a future work. Next, we set the empirical values of the verification properties, such as 80% of the vehicles and specific 1,800m point. To the best of our knowledge, we did not find an existing example that matched with our purposes. Thus, we set the acceptable values which evaluate moderate satisfaction of the platooning operation.

VI. Related Works

Several previous studies have provided tools for verifying the platooning system for specific goals, such as safety and resilience [6], [8]. Vieira et al [8] provided an integrated simulator consisting of the robot operating system (ROS) based simulator [20] and OMNET++ to test a communication network model of the platooning system. In the present work, the network frequency of a communication channel was used as test input to verify the maintenance of a generated platoon. Kamali et al. [6] focused on the spatial and timing constraints of the platooning system using an agent-based model with an integrated simulator of TORCS and MATLAB/Simulink [21]. However, they did not use the simulator to verify their model; instead, they used the UPPAAL model checker with a formal model. Both studies used a single platoon and did not consider the environment in the simulation and verification. Vieira et al. considered infrastructural settings in the system by changing the network frequency, but this work may not have sufficiently covered the uncertainty factors in the platooning system. In addition, their verification techniques were not compatible with verifying the platooning system in scalable situations.

In other previous studies [5], [7], [22], [23], formal models of the platooning system were designed and verified using existing tools, such as UPPAAL and MATLAB VnV Toolbox [24]. Elgharbawy et al. [7] verified the safety of an automated truck driving system by including unexpected environmental situations. They added stochastic properties to environmental sensing modules and exhaustively verified the system in several scenarios. However, their definition of a scenario was different from ours. They used a scenario to represent the parameter settings in a vehicle. We infer that they used a single vehicle to verify their system. Achrifi et al. [22] also focused on environmental uncertainty issues. They mainly described the advantages of the MATLAB VnV framework for verifying advanced driving assistant (ADAS) models. In this study, they also used a single ADAS model and did not include environmental factors in testing it. Mallozzi et al. [5] proposed a formal model for selecting platoon leaders in UPPAAL. They used diverse scenarios with different numbers of vehicles and platoon operations to verify the system in UPPAAL. Although they partially covered the internal uncertainty of the platooning system, they only used a single platoon and homogeneous vehicle types in the verification and didn't consider environmental factors. Meinke [23] verified a platooning system in scenarios in which the leaders speed was changed. Their research focused on the scalability of the platooning system and simulated different numbers of vehicles in a platoon. This research only used a *Speedchange* event in the scenarios, thus they locally cover the internal uncertainties of the system.

To the best of our knowledge, no previous study has simultaneously used various numbers of platoons and vehicles with stochastic environmental objects in the verification. Previous studies only partially covered the uncertainties in the platooning system. Moreover, most of the previous studies used exhaustive verification techniques that could not bypass the state-explosion problem. In contrast, our framework covers the internal and external uncertainties using various factors and applies the SPRT algorithm, which alleviates the state-explosion problem.

VII. Conclusion

We proposed a statistical verification framework, StarPlateS for a platooning SoS. This framework addressed uncertainty issues in the platooning system by implementing three extended modules on the VENTOS simulator: a scenario generation module, a simulation module, and a verification module. The scenario module generated random configurations and scenarios using the condition-based algorithm and addressed the internal uncertainty of the system. In the simulation module, we added stochastic HDVs to cover external uncertainty around the platoons. The verification module applied the SMC algorithm to bypass the state-explosion problem. By this approach, we found two undiscovered failure cases in the VENTOS platooning algorithm with 20 scenarios.

Our future research includes the development of a guided-scenario generation algorithm, further definition of verification properties, and the application of localization techniques to determine the causes of failures in the proposed framework.

Acknowledgement

This research was supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2015-0-00250, (SW Star Lab) Software R&D for Model-based Analysis and Verification of Higher-order Large Complex System), and Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (2017M3C4A7066212).

References

[1] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of Systems Engineering: Basic Concepts, Model-based Techniques, and Research Directions," ACM Computing Surveys (CSUR), vol. 48, no. 2, p. 18, 2015.

[2] P. Liu, B. Xia, Y. Tan, and D. Zhao, "Modeling and Robust Optimization for System of Systems Problems under Uncertainty," in 2018 IEEE 4th International Conference on Control Science and Systems Engineering (ICCSSE). IEEE, 2018, pp. 385–390.

[3] Y. Nomaguchi, K. Kawakami, K. Fujita, Y. Kishita, K. Hara, and M. Uwasu, "Robust Design of System of Systems using Uncertainty Assessment based on Lattice Point Approach: Case Study of Distributed Generation System Design in a Japanese Dormitory Town." IJAT, vol. 10, no. 5, pp. 678–689, 2016.

[4] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon Management with Cooperative Adaptive Cruise Control Enabled by VANET," Vehicular communications, vol. 2, no. 2, pp. 110–123, 2015.

[5] P. Mallozzi, M. Sciancalepore, and P. Pelliccione, "Formal Verification of the On-the-fly Vehicle Platooning Protocol," in International Workshop on Software Engineering for Resilient Systems. Springer, 2016, pp. 62–75.

[6] M. Kamali, S. Linker, and M. Fisher, "Modular Verification of Vehicle Platooning with respect to Decisions, Space and Time," in International Workshop on Formal Techniques for Safety-Critical Systems. Springer, 2018, pp. 18–36.

[7] M. Elgharbawy, "A Big Testing Framework for Automated Truck Driving," Urban transportation and construction, vol. 4, no. 1, pp. e27–e27, 2019.

[8] B. Vieira, R. Severino, A. Koubâa, and E. Tovar, "Towards a Realistic Simulation Framework for Vehicular Platooning Applications," arXiv preprint arXiv:1904.02994, 2019.

[9] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops. ICST (Institute for Computer Sciences, Social-Informatics and , 2008, p. 60.

[10] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO–Simulation of Urban Mobility: An Overview," in Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind, 2011.

[11] A. Legay, B. Delahaye, and S. Bensalem, "Statistical Model Checking: An Overview," in International conference on runtime verification. Springer, 2010, pp. 122–135.

[12] H. L. Younes, M. Kwiatkowska, G. Norman, and D. Parker, "Numerical vs. Statistical Probabilistic Model Checking," International Journal on Software Tools for Technology Transfer, vol. 8, no. 3, pp. 216–228, 2006.

[13] A. Wald, "Sequential Tests of Statistical Hypotheses," The annals of mathematical statistics, vol. 16, no. 2, pp. 117–186, 1945.

[14] P. Schnoebelen, "The Complexity of Temporal Logic Model Checking." Advances in modal logic, vol. 4, no. 393-436, p. 35, 2002.

[15] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the Open Racing Car Simulator," Software available at http://torcs. sourceforge. net, vol. 4, no. 6, 2000.

[16] S. Ucar, S. Coleri Ergen, O. Ozkasap, D. Tsonev, and H. Burchardt, "Secvlc: Secure Visible Light Communication for Military Vehicular Networks," in Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access. ACM, 2016, pp. 123–129.

[17] B. Zheng, C.-W. Lin, H. Yu, H. Liang, and Q. Zhu, "CONVINCE: A Cross-layer Modeling, Exploration and Validation Framework for Next-generation Connected Vehicles," in 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2016, pp. 1–8.

[18] S. Ucar, B. Turan, S. C. Ergen, O. Ozkasap, and M. Ergen, "Dimming Support for Visible Light Communication in Intelligent Transportation and Traffic System," in NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2016, pp. 1193–1196.

[19] SE Lab in KAIST, "SIMVA-SoS: Simulation-based Verification and Analysis for SoS," [Online; accessed 2-July-2019]. [Online]. Available: https://github.com/SESoS/SIMVA-SoS

[20] Aaron Blasdel, "Robot Operating System (ROS)," [Online; accessed 2-July-2019]. [Online]. Available: https://www.ros.org/

[21] Math Works, "Simulink," [Online; accessed 2-July-2019]. [Online]. Available: https://www.mathworks.com/products/simulink.html

[22] S. Achrifi, "Coverage Verification Framework for ADAS Models," March 2017.

[23] K. Meinke, "Learning-based Testing of Cyber-Physical Systems-of-Systems: A Platooning Study," in European Workshop on Performance Engineering. Springer, 2017, pp. 135–151.

[24] Math Works, "VnV Toolbox," [Online; accessed 2-July-2019]. [Online]. Available: https://www.mathworks.com/products/transitioned/simverification.html