Consensus-Based Fault-Tolerant Platooning for Connected and Autonomous Vehicles

Tzu-Yen Tseng[1], Ding-Jiun Huang[1], Jia-You Lin[1], Po-Jui Chang[1], Chung-Wei Lin[1], Changliu Liu[2]
[1]National Taiwan University, [2]Carnegie Mellon University

***Abstract*—Platooning is a representative application of connected and autonomous vehicles. The information exchanged between connected functions and the precise control of autonomous functions provide great safety and traffic capacity. In this paper, we develop an advanced consensus-based approach for platooning. By applying consensus-based fault detection and adaptive gains to controllers, we can detect faulty position and speed information from vehicles and reinstate the normal behavior of the platooning. Experimental results demonstrate that the developed approach outperforms the state-of-the-art approaches and achieves small steady state errors and small settling times under scenarios with faults.**

***Index Terms*—Autonomous vehicles, connected vehicles, consensus, fault-tolerance, platooning.**

I. Introduction

Platooning is a representative application of connected and autonomous vehicles. Usually supported by the Cooperative Adaptive Cruise Control (CACC) on each vehicle, the information exchanged between vehicles and the precise control provide great safety and traffic capacity. As the technology of connectivity and autonomy advanced in recent years, platooning has been studied from many different perspectives [1], [2], [3], [4], [5]. Especially, Santini *et al.* [1] proposed a state-of-the-art consensus-based approach for platooning. The main goal of the consensus-based approach is to construct a control system that establishes consensus between vehicles and forms the platoon. Although the proposed approach introduces the concept of consensus into platooning, it does not exploit the full strength of consensus to protect platooning against faulty information (the details are explained in Section IV-A). As a result, a platoon may converge to a wrong state if some vehicles send faulty messages, no matter they are general network faults or malicious attacks. To address this problem, Petrillo *et al.* [2] proposed a collective voting procedure to detect faulty position information. Although the approach in [2] can address the issue of faulty position information, it still has some limitations, including its settling time, communication assumption, and faulty speed information.

In this paper, we develop an advanced consensus-based approach consisting of *faulty-vehicle detection* and *controller-gain adjustment*. The faulty-vehicle detection analyzes the messages from vehicles and detects faulty position information from vehicles; the controller-gain adjustment eliminates the impacts of faulty *position* information and reinstates the normal behavior of the platooning. We then develop a two-phase mechanism to detect faulty *speed* information from the leading vehicle and also reinstate the normal behavior of the platooning. By integrating CUBA [6], we can further relax the assumption that each vehicle in the platoon can receive messages from all other vehicles in the platoon. The features of considering faulty speed information and relaxing the communication assumption are not covered in [1], [2]. Experimental results demonstrate that the developed approach outperforms the existing approaches [1], [2] and achieves small steady state errors and small settling times under scenarios with faults.

The rest of this paper is organized as follows: Section II reviews related work. Section III presents the system model and the problem formulation. Section IV describes the developed consensus-based fault-tolerant controller which detects faulty position information and reinstates the platooning. Section V describes the two-phase mechanism which detects faulty speed information and reinstates the platooning. Section VI provides experimental results, and Section VII concludes this paper.

II. Related Work

Collaborative Adaptive Cruise Control (CACC) is an emerging system for platooning connected and autonomous vehicles. However, CACC and platooning algorithms are vulnerable to external malicious attacks, causing safety and security threats to connected and autonomous vehicles. Van Nunen *et al.* [7] focused on sensor faults and proposed an algorithm to calculate safe distances and guarantee the safety for CACC. Han *et al.* [8] came up with an admission protocol for platooning against impersonation attacks. Ucar *et al.* [9] proposed a security protocol for ensuring platooning stability under jamming attacks. Santhosh and Sankara [10] modeled Sybil attacks and provided a defense mechanism against the attacks. Benchi *et al.* [11] provided an approach reducing the influence of false-position attacks, and Taylor *et al.* [12] showed the impact of false-data-injection attacks. Some other work has been done to defend the attacks through intrusion detection. Jagielski *et al.* [13] considered attacks that modify the position, speed, or acceleration information in messages and proposed a learning-based approach for intrusion detection. Alotibi and Abdelhakim [14] proposed another intrusion detection approach fusing the information of the leading vehicle from other vehicles and infrastructure and precluding the deviated data to

This work is partially supported by Ministry of Education (MOE) in Taiwan under Grant Number NTU-111V1901-5, National Science and Technology Council (NSTC) in Taiwan under Grant Number NSTC-112-2636-E-002-010, and Qualcomm.

TABLE I
NOTATION THROUGHOUT THIS PAPER.

Index	i, j	the index of a vehicle
	t	the time or the time step
Set	$\mathcal{I}^*$	the index set of faulty vehicles
Element	κ_i	the i-th vehicle in the platoon (from 0)
	$\mathcal{C}_i$	the controller of κ_i
Given Parameter	N	the number of vehicles in the platoon
	X_i	the initial position of κ_i
	V_i	the initial speed of κ_i
	A_i	the initial acceleration of κ_i
	U_i^+	the maximum acceleration of κ_i
	U_i^-	the maximum deceleration of κ_i
	D_i	the safety distance between κ_{i-1} and κ_i
	Θ	the threshold for settling time
	F_i	the difference from the true position of κ_i
	E	the difference from the true speed of κ_0
	M	the parameter in CUBA [6]
Decision Variable	$g_i(t)$	the controller gain of κ_i toward the speed reference from κ_0 at t
	$g_{i,j}(t)$	the controller gain of κ_i toward the position reference from κ_j at t
Dependent Variable	$x_i(t)$	the (true) position of κ_i at t
	$v_i(t)$	the (true) speed of κ_i at t
	$a_i(t)$	the (true) acceleration of κ_i at t
	$x'_i(t)$	the (received) position of κ_i at t
	$v'_i(t)$	the (received) speed of κ_i at t
	$a'_i(t)$	the (received) acceleration of κ_i at t
	$u_i(t)$	the controlled acceleration from $\mathcal{C}_i$ at t
	s_i	the settling time of κ_i
	$\delta x_i(t)$	the position error of κ_i at t
	$\delta v_i(t)$	the speed error of κ_i at t

detect attacks in CACC. Recently, Taylor *et al.* [15] listed multiple kinds of attacks and pointed out open challenges.

On the other hand, consensus algorithms have been well-studied. Lamport [16] proposed the Paxos algorithm, which can be implemented for a fault-tolerant distributed system. Cavin *et al.* [17] reformed the traditional consensus algorithms to handle the dynamic in mobile ad hoc networks. Ongaro and Ousterhout [18] developed the Raft algorithm, which simplified and optimized the design of Paxos algorithm. Benchi *et al.* [19] presented a new consensus algorithm that can be applied to opportunistic networks.

Some research concentrated on the automotive applications of consensus algorithms. Zhang *et al.* [20] proposed a consensus tracking algorithm for a multi-vehicle system with a dynamic topology. Petit and Mammeri [21] presented a consensus application in vehicular ad hoc networks (VANET) with dynamic inputs according to the uncertainty of surrounding environments. Wegner *et al.* [22] proposed a consensus protocol for platooning, but the protocol purely considers high-level value exchanges (without considering controllers).

III. System Model and Problem Formulation

The notation throughout this paper is listed in Table I, and the system model and problem formulation are provided in this section.

Vehicle. There are N vehicles indexed from 0. All vehicles are connected and autonomous, and they are on the same lane to form a platoon. Each vehicle κ_i periodically broadcasts its position, speed, acceleration, and other relevant information.

- The (true) position, speed, and acceleration of κ_i at time t are $x_i(t)$, $v_i(t)$, and $a_i(t)$, respectively.
- The position, speed, and acceleration of κ_i at t, as messages received by other vehicles, are $x'_i(t)$, $v'_i(t)$, and $a'_i(t)$, respectively, *i.e.*, if there is no fault, $x'_i(t) = x_i(t)$, $v'_i(t) = v_i(t)$, and $a'_i(t) = a_i(t)$.
- The initial position, speed, and acceleration of κ_i are X_i, V_i, and A_i, respectively, *i.e.*, $x_i(0) = X_i$, $v_i(0) = V_i$, and $a_i(0) = A_i$.

The system model in [1] assumes that each vehicle in the platoon can receive messages from the leading vehicle κ_0. Here we also assume that each vehicle in the platoon can receive messages from all other vehicles in the platoon. We can also relax the assumption with the modified CUBA method (which is described in Section IV-C).

Controller. The controller $\mathcal{C}_i$ of κ_i decides the acceleration $u_i(t)$ of κ_i at t. For each activation period at t, $\mathcal{C}_i$ collects the messages from other vehicles and computes $u_i(t)$, where the computation involves the controller gains toward the messages from other vehicles. Note that κ_i has its maximum acceleration U_i^+ and its maximum deceleration U_i^-. If the computed acceleration is larger (smaller) than U_i^+ (U_i^-), $u_i(t)$ will be set to U_i^+ (U_i^-).

Target States. The target state of κ_i includes the *position target state* and the *speed target state*. The position target state of κ_i ($i \geq 1$) is the safety distance between κ_{i-1} and κ_i, denoted as D_i, *i.e.*,

$$x_i(t) - x_{i-1}(t) = D_i, \tag{1}$$

and the speed target state of κ_i is the speed of κ_0, *i.e.*,

$$v_i(t) = v_0(t). \tag{2}$$

Settling Time. The settling time s_i of κ_i is the time for κ_i to reach and stay within a threshold Θ of its final states (state values), *i.e.*,

$$s_i = \min \left\{ t \mid \forall t' > t, \left| 1 - \frac{x_i(t') - x_{i-1}(t')}{\lim\limits_{t'' \to \infty} (x_i(t'') - x_{i-1}(t''))} \right| < \Theta \right.$$
$$\left. \wedge \left| 1 - \frac{v_i(t')}{\lim\limits_{t'' \to \infty} v_i(t'')} \right| < \Theta \right\}. \tag{3}$$

The system settling time is the maximum settling time of all vehicles. Some examples of Θ are 5%, 3%, and 1%. If Θ is smaller, the settling time will become larger; if Θ is larger, the settling time will become smaller.

State Errors and Steady State Errors. The state errors of κ_i at t include the *position-state error* and the *speed-state error*. The position-state error of κ_i at t is

$$\delta x_i(t) = \left| x_i(t) - x_{i-1}(t) - D_i \right|, \tag{4}$$

and the speed-state error of κ_i at t is

$$\delta v_i(t) = \left| v_i(t) - v_0(t) \right|. \tag{5}$$

The steady state errors of κ_i are the corresponding state errors of κ_i when $t \geq s_i$.

Fault Model. In this paper, we consider *faulty positions* and a *faulty speed.* A faulty position of κ_i indicates

$$x'_i(t) = x_i(t) + F_i, \tag{6}$$

where F_i is the difference from the true value. A faulty speed of κ_o indicates

$$v'_0(t) = v_0(t) + E, \tag{7}$$

where E is the difference from the true value. We assume that the index set of faulty vehicles $\mathcal{I}^*$ is *unknown* (not given) in advance. The faults, whether they are malicious or not, will lead to steady state errors and impair system properties, *e.g.*, safety (smaller gaps between vehicles) and efficiency (larger gaps between vehicles).

Problem Formulation. Given the system model, decide

- The controller gain of κ_i toward the speed reference from κ_0 at t, $g_i(t)$, and
- The controller gain of κ_i toward the position reference from κ_j at t, $g_{i,j}(t)$,

to minimize

- The average of steady state errors of all vehicles (primary objective), and
- The system settling time (secondary objective).

IV. Consensus-Based Fault-Tolerant Controller

In this section, we first introduce the state-of-the-art approaches [1], [2] in detail (integrated with the notation of this paper) and explain the motivations in Section IV-A. We then describe the faulty-vehicle detection, the modified CUBA method (also for the faulty-vehicle detection), and the controller-gain adjustment in Sections IV-B, IV-C, and IV-D, respectively.

A. Motivations

Santini *et al.* [1] proposed a consensus-based approach for platooning. They observed that the previous research lacks the consideration of changing communication delays and thus proposed a controller which takes communication delays into account. The leading vehicle κ_0 is a reference for platooning and is assumed globally reachable, meaning that $\kappa_1, \kappa_2, \ldots, \kappa_{N-1}$ can receive the information sent from κ_0. The objective of the controller $\mathcal{C}_i$ is to fix a desired safety distance D_i between κ_{i-1} and κ_i and set the speed $v_i(t)$ to approach $v_0(t)$ by computing

$$\begin{aligned} u_i(t) &= -g_i(t)(v_i(t) - v_0(t)) \\ &\quad - \sum_{j=0}^{N-1} g_{i,j}(t) \cdot f(x_i, x_j, v_0, \tau_{i,j}, t), \end{aligned} \tag{8}$$

where

$$\begin{aligned} f(x_i, x_j, v_0, \tau_{i,j}, t) &= x_i(t) - x_j(t - \tau_{i,j}(t)) \\ &\quad -\tau_{i,j}(t)v_0(t) - h_{i,j}v_0(t) - d_{i,j}, \end{aligned} \tag{9}$$

where $\tau_{i,j}(t)$ is the communication delay between κ_i and κ_j at time t, $h_{i,j}$ is the constant time headway (*i.e.*, a constant time where each vehicle κ_i travels to maintain the desired following distance to its front vehicle κ_{i-1}), and $d_{i,j}$ is the summation of the inter-vehicular spacing at standstill and the vehicle length [1]. After formulating the problem, $u_i(t)$ is rewritten with the terms of the state errors $\delta x_i(t)$ and $\delta v_i(t)$, recasting the problem to be a closed-loop platooning dynamics. The controller is theoretically proven to be asymptotically stable. The controller gains $g_i(t)$ and $g_{i,j}(t)$ also guarantee string stability if the communication delay has an upper bound.

The approach in [1] does not consider faulty messages (no matter whether they are general network faults or malicious attacks), *i.e.*, $F_i = 0$ for each vehicle κ_i. The controller $\mathcal{C}_i$ of the approach in [1] only considers the speed of κ_0 and the positions of κ_0 and κ_{i-1}, which means that the positions of other vehicles are not considered, *i.e.*, the controller gain $g_{i,j}(t) = 0$ if $j \neq 0$ and $j \neq i-1$. This setting is reasonable when there is no faulty message because the speed of κ_0 and the positions of κ_0 and κ_{i-1} are the most relevant information for $\mathcal{C}_i$. However, when there are faulty messages, $\mathcal{C}_i$ which relies on only a few specific vehicles may converge to a wrong state. Also, because there is no faulty message, the controller gain $g_{i,j}(t)$ is set to be constant in the approach in [1].

Following the approach in [1], Petrillo *et al.* [2] proposed a collective voting procedure to detect vehicles with faulty messages at each time step. Each vehicle κ_i compares the distance to every considered vehicle κ_j with an average gap value and takes κ_j as abnormal if the difference is larger than a threshold. Although the strategy in [2] can address the issue of faulty messages, the system settling time can be reduced.

Starting from the approaches [1], [2], we develop a more advanced approach with faulty-vehicle detection and controller-gain adjustment. The major features are as follows:

- We consider faulty positions (as messages), *i.e.*, $F_i \neq 0$. Then, we exploit the strength of consensus, consider the positions of other vehicles, and further adaptively adjust the controller gain $g_{i,j}(t)$, *i.e.*, $g_{i,j}(t)$ is not necessarily zero if $j \neq 0$ and $j \neq i-1$, to eliminate the impacts on the platoon from faulty positions. The above features are not in [1].
- We further reduce the system settling time, compared with the approach in [2].
- The system model in [1], [2] assumes that each vehicle in the platoon can receive the messages from the leading vehicle κ_0. Although we also first assume that each vehicle in the platoon can receive the messages from all other vehicles in the platoon, we can also relax the assumption with the modified CUBA method. The feature is not covered in [1], [2].
- We also consider the faulty speed of the leading vehicle κ_0, which will be described in Section V. The feature is not covered in [1], [2].

B. Faulty-Vehicle Detection

For κ_i to detect a faulty vehicle κ_j, the intuitive solution is to check the inconsistency of the true dynamics $x_j(t), v_j(t), a_j(t)$ and the received dynamics $x'_j(t), v'_j(t), a'_j(t)$, as illustrated in Figure 1. However, given

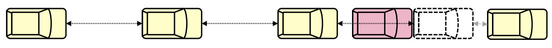

Fig. 1. Faulty-vehicle detection.

that a vehicle can only know the dynamics of non-adjacent vehicles through received messages (implying that $x_j(t), v_j(t), a_j(t)$ are unknown to κ_i), the position-state error of κ_j, *observed by other vehicles*, can be written as

$$x'_j(t) - x'_{j-1}(t) - D_j. \tag{10}$$

If κ_j is not faulty, the position-state error of κ_j should converge to zero, *i.e.*,

$$x'_j(t) - x'_{j-1}(t) - D_j \longrightarrow 0, \tag{11}$$

when all vehicles become steady. However, if κ_j is faulty, the position-state error of κ_j should converge to F_j, *i.e.*,

$$x'_j(t) - x'_{j-1}(t) - D_j \longrightarrow F_j, \tag{12}$$

when all vehicles become steady. We would like to point out that whether κ_{j-1} is faulty or not does not change Equations (11) and (12) because κ_j behaves according to $x'_{j-1}(t)$, not $x_{j-1}(t)$. Based on Equations (11) and (12), the faulty-vehicle detection checks the position-state errors when all vehicles become steady. If the gap between κ_j and κ_{j-1} is Θ (the threshold in Equation (3)) larger than or smaller than D_j, the faulty-vehicle detection decides that κ_j is a faulty vehicle. Note that there is a case that κ_j and κ_{j-1} are both faulty with the same difference from the true positions, and κ_{j-1} is considered as a faulty one. In this case, κ_j and κ_{j-1} will no longer be considered with Equation (10), κ_j and κ_0 will be considered instead due to the controller-gain adjustment (which is described in Section IV-D).

The faulty-vehicle detection will lead to the controller-gain adjustment (which is described in Section IV-D). In practice, when to trigger the detection is still an issue. If it is triggered when all vehicles become steady, then we can achieve high fault-tolerance, but the system settling time may become much larger; if it is triggered too early, then the dynamics of most vehicles may have not converged, and thus many vehicles will be detected as faulty vehicles. To address this issue, we also propose to trigger the detection and the following adjustment when $\frac{N}{2}$ vehicles become steady, which can improve (shorten) the system settling time. Here, $\frac{N}{2}$ can be replaced by other ratios, but $\frac{N}{2}$ balances between settling time and over-sensitivity and fits if there is an assumption that $\frac{N}{2}$ vehicles are non-faulty. This *fast fault-tolerant* method does not have any additional communication overhead as each vehicle can decide the stability of each other vehicle by the received information.

Note that, if the assumption that D_j is the same for each vehicle is not held, we can assume that each vehicle knows the safety distances for all vehicles in front of it, which can be achieved by additional information exchanges between vehicles and/or roadside units. Similarly, if the gap between κ_j and κ_{j-1} is Θ larger than or smaller than the safety distance, κ_j is regarded as a faulty vehicle.

C. Modified CUBA Method for Faulty-Vehicle Detection

Here, we relax the assumption that each vehicle in the platoon can receive the messages from all other vehicles in the platoon. There are a consensus round and a suspect round in CUBA [6]. We modify and use the *suspect round* of CUBA for the problem. When the consensus round fails, a proposer starts a suspect round for detecting faulty vehicles. In the suspect round, it only requires at least one message to be received by each vehicle.

We modify the suspect round of CUBA by splitting the protocol into three stages:

- In the first stage, each vehicle κ_i sends its own position and speed information to nearby $2M+1$ vehicles (*i.e.*, from κ_{i-2M-1} to κ_{i+2M+1}), where the parameter M is set based on the network status and the distance between the vehicles.
- In the second stage, we need to consider the cases that the number of hops between two vehicles is larger than $2M+1$. For each κ_i that receives a message from κ_j, it sends the message to $2M+1$ vehicles in the direction away from κ_j. For example, when $(M,i,j) = (1,3,4)$, κ_4 sends the message from κ_3 to κ_5, κ_6, and κ_7. Note that if the total number of vehicles is fewer than $2M+3$, the protocol degenerates to the assumption that each vehicle can receive the messages from all front vehicles.
- In the third stage, based on the messages received in the second stage, each vehicle detects faulty vehicles with Equations (11) and (12).

When the modified CUBA method detects faulty vehicles, it executes the controller-gain adjustment in Section IV-D.

With the modifications, we can guarantee that the protocol works if the number of faulty vehicles is fewer than or equal to M (*i.e.*, $|\mathcal{I}^*| \leq M$). We can prove it by induction with the following steps:

- Considering a vehicle κ_j and the following vehicles κ_{j+k}, where $k > 0$. We know that $\kappa_{j+1}, \kappa_{j+2}, \ldots, \kappa_{j+2M+1}$ can successfully receive the message from κ_j. We will show that, for each $k > 2M+1$, κ_{j+k} can also receive the information of the message.
- When $k = 2M+2$, κ_{j+k} can receive messages from κ_{j+1}, κ_{j+2}, …, κ_{j+k-1}. Since $|\mathcal{I}^*| \leq M$, κ_{j+k} receives at least $2M+1-M = M+1$ messages with correct information. Because $M+1 > M$, which is the number of vehicles that may be faulty, κ_{j+k} can correctly learn the information from κ_j. Note that we do not assume that κ_{j+2M+2} is non-faulty. We only prove that κ_{j+2M+2} can learn the information from κ_j.
- When $k > 2M+2$, all non-faulty vehicles between κ_j and κ_{j+k-1} send the messages including the information from κ_j. Following the same reason when $k = 2M+2$, κ_{j+k} can also correctly learn the information from κ_j because the number of non-faulty vehicles among $\kappa_{j+k-2M-1}$, κ_{j+k-2M}, …, κ_{j+k-1} is at least $M+1$.

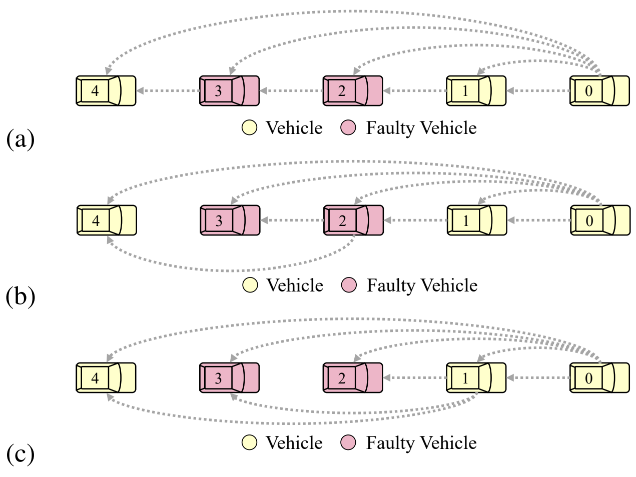

Fig. 2. Controller-gain adjustment.

By induction, we prove that all vehicles receive the information from all other vehicles and detect faulty vehicles based on the information.

D. Controller-Gain Adjustment

Since we cannot find the faulty vehicles in the beginning, all controllers are first set to be the same as that in [1], meaning that the controller $\mathcal{C}_i$ of κ_i only considers the positions sent from κ_0 and κ_{i-1}, *i.e.*, $g_{i,0}(0) > 0$, $g_{i,i-1}(0) > 0$, and $g_{i,j}(0) = 0$ for $j \neq 0$ and $j \neq i-1$. With this setting, the system settling time can be small, but the steady state errors will be affected by faulty positions. The goal of the controller-gain adjustment here is to minimize the average steady state errors by adaptively adjusting the controller gains.

When to trigger the detection has been described in Section IV-B. Once it is triggered, it starts from the last vehicle κ_{N-1} to the leading vehicle κ_0. If κ_j is detected as a faulty vehicle, then the controller of κ_i will adjust the controller gain by the following rule:

$$\text{if } g_{i,j}(t) \neq 0, \text{ then } g_{i,j-1}(t) \leftarrow g_{i,j}(t) \text{ and } g_{i,j}(t) \leftarrow 0. \tag{13}$$

Following the design in Section IV-B, if $F_j \geq D_j \cdot \Theta$ for each $j \in \mathcal{I}^*$, then the controller-gain adjustment can reinstate the normal behavior of the platooning (Θ is the threshold in Equation (3)).

An example with five vehicles is shown in Figure 2. In the example, κ_2 and κ_3 are faulty vehicles. In the beginning, all controllers are set to be the same, meaning that κ_i only considers the positions of κ_0 and κ_{i-1}, and the logical communication topology is shown in Figure 2(a). Each vehicle triggers the detection and the following adjustment are triggered at the same time. As the rule above, they start from the last vehicle κ_4 to the leading vehicle κ_0. Once κ_4 detects κ_3 as a faulty vehicle, $g_{4,2}(t)$ is set to $g_{4,3}(t)$, and then $g_{4,3}(t)$ is set to zero. At this point, the logical communication topology is shown in Figure 2(b). Next, once κ_4 and κ_3 detects κ_2 as a faulty vehicle, $g_{4,1}(t)$ and $g_{3,1}(t)$ are set to $g_{4,2}(t)$ and $g_{3,2}(t)$, respectively, and $g_{4,2}(t)$ and $g_{3,2}(t)$ are set to zero. At this point, the logical communication topology is shown in Figure 2(c). Since there is no other faulty vehicle, the logical communication topology will stay as Figure 2(c).

V. Extension to Faulty Speed

We have developed the consensus-based fault-tolerant approach to deal with faulty positions in Section IV. In this section, we extend the approach and develop a two-phase mechanism to deal with a faulty speed of the leading vehicle κ_0. It should be emphasized that the first phase in the two-phase mechanism is exactly the approach in Section IV, meaning that we do not need to know the types (positions or speed) of faults in advance. We will also show that the developed approach can deal with the combination of faults in Section VI.

The model of the faulty speed of κ_0 is defined as Equation (7). Each vehicle κ_i in the platoon only considers the positions of other vehicles and the speed of κ_0, so the faulty speeds of vehicles, except of κ_0, do not cause steady state errors or break the platoon. With the controller defined in Section IV, we observe that the faulty speed of κ_0 results in position-state errors, while all vehicles still converge to the true speed of κ_0. This is because the controller tends to stabilize the corresponding vehicle. If the vehicles in the platoon have different speeds from the true speed of κ_0, the platoon will not become stable as the distances between vehicles constantly change. As a result, the controller follows Equation (8) and generates position-state errors.

Based on the observation, we develop a two-phase mechanism to deal with a faulty speed of κ_0. In the first phase, we can apply the fault-tolerant method, the fast fault-tolerant method, and the modified CUBA method in Section IV. The controller-gain adjustment can still be triggered, but adjusting $g_{i,j}$ does not eliminate the position-state errors as the errors result from the faulty speed of κ_0. Therefore, if the position-state errors are not eliminated after the controller-gain adjustment, we know that the errors result from the faulty speed of κ_0. In the second phase, since the position-state errors are not eliminated, all vehicles except κ_0 give up the speed of κ_0 and regard κ_1 as the dummy leading vehicle, *i.e.*, $g_i(t)$ becomes the controller gain of κ_i toward the speed reference from κ_1 at t. Then, $g_{i,0}$ is set to 0, and the other controller gains are reset to the initial values because the controller-gain adjustment in the first phase is misled by the faulty speed of κ_0.

VI. Experimental Results

The whole experiments are simulated with Plexe [23] running on a virtual machine with VirtualBox, 4 processors, and 4GB memory. The virtual machine runs on a laptop with Ubuntu 22.04, Intel HM470 Chipset, 12 i7-8750H processors, 16GB memory, and VT-x enabled. As the approach in [1] is open-sourced and also experimented with Plexe, we have successfully reproduced the results by plugging in the same parameters except for some missing initial states. There are five scenarios, all with 8 vehicles ($N = 8$), in our experiments:

TABLE II
EXPERIMENTAL RESULTS OF SCENARIO 1A.

Method	Avg. Steady Position-State Error (*m*)	System Settling Time (*s*)
[1]	1.61	43.3
All-Front	0.46	75.4
[2]	< 0.01	97.2
Fault-Tolerant	< 0.01	62.9
Fast Fault-Tolerant	< 0.01	55.3
Modified CUBA	< 0.01	58.0

TABLE III
EXPERIMENTAL RESULTS OF SCENARIO 1B.

Method	Avg. Steady Position-State Error (*m*)	System Settling Time (*s*)
[1]	No Steady State	∞
All-Front	No Steady State	∞
[2]	0.03	97.5
Fault-Tolerant	No Steady State	∞
Fast Fault-Tolerant	< 0.01	50.5
Modified CUBA	< 0.01	51.6

- **Scenario 1A: Single Faulty Position**, where the position of one vehicle is faulty with a constant.
- **Scenario 1B: Single Faulty Position**, where the position of one vehicle is faulty with a sinusoidal function.
- **Scenario 2: Multiple Faulty Positions**, where the positions of multiple vehicles are faulty.
- **Scenario 3: Different Safety Distances**, where the position of one vehicle is faulty, and there are different safety distances between vehicles.
- **Scenario 4: Faulty Speed of Leading Vehicle**, where the speed of κ_0 is faulty.
- **Scenario 5: Combination of Faults**, where the positions of multiple vehicles are faulty, and the speed of κ_0 is faulty.

There are six experimented methods (approaches):

- **The Approach in [1]**, where κ_i only considers the positions of κ_0 and κ_{i-1}, and the controller gain $g_{i,j}(t)$ is kept constant.
- **The "All-Front" Approach**, where κ_i considers the positions of $\kappa_0, \kappa_1, \ldots, \kappa_{i-1}$, and the controller gain $g_{i,j}(t)$ is kept constant.
- **The Approach in [2]**, where all vehicles collaboratively detect the faulty vehicles at each time step, and κ_i considers the positions of a specific set of vehicles.
- **Our Fault-Tolerant Method** which starts the faulty-vehicle detection when all vehicles are steady and executes the controller-gain adjustment.
- **Our Fast Fault-Tolerant Method** which starts the faulty-vehicle detection when $\frac{N}{2} = 4$ vehicles are steady and executes the controller-gain adjustment.
- **Our Modified CUBA Method** which also starts the faulty-vehicle detection when 4 vehicles are steady and executes the controller-gain adjustment. M is set to 1.

For Scenarios 4–5, our two-phase mechanism is integrated with our methods for a faulty speed of the leading vehicle.

A. Scenario 1A: Single Faulty Position

The given parameters are listed as follows (the units are in meter (*m*), second (*s*), or meter per second (*m/s*)):

- $X_i = 46 \cdot (8 - i)$ for $i = 0, 1, \ldots, 7$.
- $V_i = (27, 25, 23, 22, 21, 20, 19, 18)$ for $i = 0, 1, \ldots, 7$.
- $A_i = 0$, $U_i^+ = 2.5$, and $U_i^- = 9$ for all i [1].
- $D_i = 37.2$ for $i = 1, 2, \ldots, 7$. [1].
- $g_{1,0}(0) = 460$, $g_{i,0}(0) = 80$ for $i = 2, 3, \ldots, 7$, $g_{i,i-1}(0) = 860$ for $i = 1, 2, \ldots, 7$, and $g_{i,j}(0) = 0$ for the remaining cases.
- $F_3 = -10$, each other $F_i = 0$, and $E = 0$.
- $\Theta = 5\%$.

The experimental results with single constant faulty position are listed in Table II. All vehicles controlled by all methods have almost zero steady speed-state errors, so we only report and discuss steady position-state errors. As the approach in [1] only considers the positions of κ_0 and κ_{i-1}, κ_4 is affected by the faulty positions sent from κ_3, resulting in a large average steady position-state error (1.61*m*). On the other hand, as the all-front approach considers the positions of $\kappa_0, \kappa_1, \ldots, \kappa_{i-1}$, κ_4 is still affected by the faulty positions sent from κ_3 but mitigated by the correct positions sent from $\kappa_0, \kappa_1, \kappa_2$, resulting in a smaller average steady position-state error (0.46*m*), compared with the approach in [1]. The approach in [2] successfully detects κ_3 as a faulty vehicle and has an almost zero average steady position-state error, but its system settling time is the longest (97.7*s*). The fault-tolerant method applies the faulty-vehicle detection and the controller-gain adjustment and has an almost zero average steady position-state error. The system settling time is shortened (from 62.9*s* to 55.3*s*) by the fast fault-tolerant method. The modified CUBA method also has an almost zero average steady position-state error with comparable system settling time (58.0*s*).

B. Scenario 1B: Single Faulty Positions

The given parameters are the same as Scenario 1A, except the following parameter (the unit is in meter (*m*)):

- $F_3 = 10\sin(t)$ and each other $F_i = 0$.

The experimental results with single sinusoidal faulty position are listed in Table III. The approach in [1] and the all-front approach do not reach a steady state due to the oscillating faulty positions sent from κ_3. The fault-tolerant method does not reach a steady state, either, because it starts the faulty-vehicle detection only when all vehicles are steady. The approach in [2] successfully detects κ_3 as a faulty vehicle, but its average steady position-state error (0.03*m*) and system settling time (97.5*s*) are larger than those of the fast fault-tolerant method (<0.01*m* and 50.5*s*) and the modified CUBA method (<0.01*m* and 51.6*s*).

C. Scenario 2: Multiple Faulty Positions

The given parameters are the same as Scenario 1A, except the following parameters (the units are in meter (*m*)):

- $F_1 = 15$, $F_3 = -10$, $F_4 = 5$, and each other $F_i = 0$.

The experimental results with multiple faulty positions are listed in Table IV. Similarly, all vehicles controlled by all

TABLE IV
EXPERIMENTAL RESULTS OF SCENARIO 2.

Method	Avg. Steady Position-State Error (m)	System Settling Time (s)
[1]	4.37	46.2
All-Front	2.72	79.3
[2]	< 0.01	82.7
Fault-Tolerant	< 0.01	71.8
Fast Fault-Tolerant	< 0.01	64.7
Modified CUBA	< 0.01	67.8

TABLE V
EXPERIMENTAL RESULTS OF SCENARIO 3.

Method	Avg. Steady Position-State Error (m)	System Settling Time (s)
[1]	1.61	32.8
All-Front	0.46	57.0
[2]	< 0.01	78.5
Fault-Tolerant	< 0.01	58.7
Fast Fault-Tolerant	< 0.01	45.6
Modified CUBA	< 0.01	47.0

methods have almost zero steady speed-state errors, so we only report and discuss steady position-state errors. The results are similar to those in Scenario 1A. The approach in [1] has a large average steady position-state error (4.37*m*), and the all-front approach has a smaller average steady position-state error (2.72*m*), compared with the approach in [1]. The approach in [2] has an almost zero average steady position-state error, but its system settling time is the longest (82.7*s*). Again, the fault-tolerant method has an almost zero average steady position-state error, and the system settling time is shortened (from 71.8*s* to 64.7*s*) by the fast fault-tolerant method. The modified CUBA method also has an almost zero average steady position-state error with comparable system settling time (67.8*s*).

D. Scenario 3: Different Safety Distances

The given parameters are the same as Scenario 1A, except the following parameters (the units are in meter (*m*)):

- $D_1 = 52.2$, $D_7 = 72.2$, and each other $D_i = 37.2$.

The experimental results with different safety distances are listed in Table V. With the assumption that each vehicle knows the safety distances for all vehicles in front of it, the experimented methods have the same advantages and disadvantages as described in Scenarios 1 and 2. The fault-tolerant method has an almost zero average steady position-state error, and the system settling time is shortened (from 58.7*s* to 45.6*s*) by the fast fault-tolerant method. The modified CUBA method also has an almost zero average steady position-state error with comparable system settling time (47.0*s*).

E. Scenario 4: Faulty Speed of Leading Vehicle

The given parameters are the same as Scenario 1A, except the following parameters (the units are in meter (*m*) or meter per second (*m/s*)):

- Each $F_i = 0$ and $E = -2$.

The experimental results of faulty speed of κ_0 are listed in Table VI. Note that we do not include the steady state errors of

TABLE VI
EXPERIMENTAL RESULTS OF SCENARIO 4.

Method	Avg. Steady Position-State Error (m)	System Settling Time (s)
[1]	1.23	42.4
All-Front	1.98	71.4
[2]	3.30	69.9
Fault-Tolerant	< 0.01	69.0
Fast Fault-Tolerant	< 0.01	69.0
Modified CUBA	< 0.01	67.6

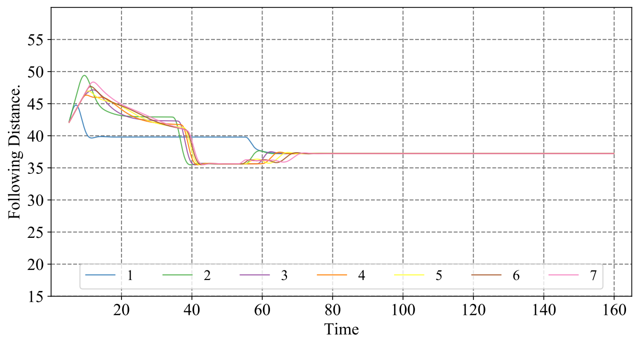

Fig. 3. The gaps (*m*) to the front vehicles of $\nu_1, \nu_2, \ldots, \nu_{N-1}$ of the fast fault-tolerant method along the time (*s*) in Scenario 4 (faulty speed of ν_0).

the second vehicle in the calculation because κ_0 is like leaving the platoon, and the second vehicle becomes the dummy leading vehicle after applying the two-phase mechanism. Also, as mentioned in Section V, the faulty speed of κ_0 results in steady position-state errors, not steady speed-state errors. The approach in [1] has a large average steady position-state error (1.23*m*). The approach in [2] also has a large average steady position-state error (3.30*m*) because it does not consider the faulty speed of the leading vehicle. The two-phase mechanism (no matter it works with the fault-tolerant method, the fast fault-tolerant method, or the modified CUBA method) has an almost zero average steady position-state error. In the first phase of the two-phase mechanism, due to the position-state errors resulting from the faulty speed of κ_0, the controller-gain adjustment is triggered, and we can observe a significant turning point at $t = 34.3s$ in Figure 3. At $t = 53.0s$, all vehicles become stable, but the position-state errors are not eliminated, which implies that the errors result from the faulty speed of κ_0. Therefore, in the second phase of the two-phase mechanism, the vehicles behave as described in Section V and achieve an almost zero average steady position-state error.

F. Scenario 5: Combination of Faults

The given parameters are the same as Scenario 1A, except the following parameters (the units are in meter (*m*) or meter per second (*m/s*)):

- $F_2 = 15$, $F_3 = 15\sin(t)$, $F_4 = 5$, and each other $F_i = 0$.
- $E = -2$.

The experimental results of faulty speed of κ_0 are listed in Table VII. The approach in [1], the all-front approach, the approach in [2], and the two-phase mechanism with the fault-tolerant method do not reach a steady state due to the

TABLE VII
EXPERIMENTAL RESULTS OF SCENARIO 5.

Method	Avg. Steady Position-State Error (m)	System Settling Time (s)
[1], All-Front, [2]	No Steady State	∞
Fault-Tolerant	No Steady State	∞
Fast Fault-Tolerant	< 0.01	98.0
Modified CUBA	< 0.01	98.2

sinusoidal fault. The two-phase mechanism with the fast fault-tolerant method and the modified CUBA method detects the faulty speed of κ_0 and regard the second vehicle as the dummy leading vehicle. Then, the faulty-vehicle detection detects the faulty positions, and the controller-gain adjustment reinstates the platooning. As a result, we have an almost zero average steady position-state error, showing the capability of dealing with the combinations of faults.

G. Summary

From the experimental results of different scenarios, we can observe that the approach in [1] and the all-front approach are vulnerable to the faulty position and speed information. The approach in [2] and our fault-tolerant method are respectively vulnerable to the faulty speed information and the sinusoidal faulty position information. Our fast fault-tolerant method and our modified CUBA method are robust against the faults. They are also efficient to reach almost zero average steady position-state error, especially when there is only faulty position information. Last but not least, we would also like to point out that reaching a steady state is crucial for energy efficiency. With a sinusoidal fault, we compute the power consumption and observe that failing to reach a steady state significantly increases power consumption.

VII. CONCLUSION

We developed an advanced consensus-based approach detecting faulty position and speed information from vehicles and reinstating the normal behavior of the platooning. We also relaxed a communication assumption with the modified CUBA method. Experimental results demonstrated that the developed approach outperforms the existing approaches [1], [2] and achieves small steady state errors and small settling times under scenarios with faults. Potential future work includes considerations for different types of faults, testing methodologies with an exponential number of parameters and scenarios, protection mechanisms against malicious attacks including collusion between vehicles, and consensus problems in other applications (such as lane merging, lane changing, intersection management) of connected and autonomous vehicles.

REFERENCES

[1] S. Santini, A. Salvi, A. Valente, A. Pescapè, M. Segata, and R. Cigno. A consensus-based approach for platooning with inter-vehicular communications. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1158–1166, Apr 2015.

[2] A. Petrillo, A. Pescapé, and S. Santini. A collaborative control strategy for platoons of autonomous vehicles in the presence of message falsification attacks. In *IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 110–115, 2017.

[3] T.-S. Dao, J. Huissoon, and C. Clark. A strategy for optimisation of cooperative platoon formation. *International Journal of Vehicle Information and Communication Systems*, 3(1):28–43, Aug 2013.

[4] P. Hao, Z. Wang, G. Wu, K. Boriboonsomsin, and M. Barth. Intra-platoon vehicle sequence optimization for eco-cooperative adaptive cruise control. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Oct 2017.

[5] J. Heinovski and F. Dressler. Platoon formation: Optimized car to platoon assignment strategies and protocols. In *IEEE Vehicular Networking Conference (VNC)*, pages 1–8, Dec 2018.

[6] E. Regnath and S. Steinhorst. CUBA: Chained unanimous byzantine agreement for decentralized platoon management. In *ACM/IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 426–431, 2019.

[7] E. van Nunen, J. Ploeg, A. Medina, and H. Nijmeijer. Fault tolerancy in cooperative adaptive cruise control. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1184–1189, Oct 2013.

[8] J. Han, M. Harishankar, X. Wang, A. Chung, and P. Tague. Convoy: Physical context verification for vehicle platoon admission. In *International Workshop on Mobile Computing Systems and Applications*, pages 73–78, 2017.

[9] S. Ucar, S. Ergen, and O. Ozkasap. IEEE 802.11p and visible light hybrid communication based secure autonomous platoon. *IEEE Transactions on Vehicular Technology*, 67(9):8667–8681, 2018.

[10] J. Santhosh and S. Sankaran. Defending against sybil attacks in vehicular platoons. In *IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6, 2019.

[11] F. Boeira, M. Asplund, and M. Barcellos. Mitigating position falsification attacks in vehicular platooning. In *IEEE Vehicular Networking Conference (VNC)*, pages 1–4. IEEE, 2018.

[12] S. Taylor, F. Ahmad, H. Nguyen, S. Shaikh, and D. Evans. Safety, stability and environmental impact of FDI attacks on vehicular platoons. In *IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2022.

[13] M. Jagielski, N. Jones, C.-W. Lin, C. Nita-Rotaruand, and S. Shiraishi. Threat detection for collaborative adaptive cruise control in connected cars. In *ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, pages 184–189, 2018.

[14] F. Alotibi and M. Abdelhakim. Anomaly detection in cooperative adaptive cruise control using physics laws and data fusion. In *IEEE Vehicular Technology Conference (VTC-Fall)*, pages 1–7, Sep 2019.

[15] S. Taylor, F. Ahmad, H. Nguyen, S. Shaikh, D. Evans, and D. Price. Vehicular platoon communication: Cybersecurity threats and open challenges. In *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 19–26, 2021.

[16] L. Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121)*, pages 51–58, Dec 2001.

[17] D. Cavin, Y. Sasson, and A. Schiper. Consensus with unknown participants or fundamental self-organization. *Lecture Notes in Computer Science*, 3158, Jul 2004.

[18] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference (ATC)*, pages 305–319, Jun 2014.

[19] A. Benchi, P. Launay, and F. Guidec. Solving consensus in opportunistic networks. In *International Conference on Distributed Computing and Networking*, pages 1:1–1:10, 2015.

[20] J. Zhang, C. Sun, L. Wang, and J. Xia. Consensus tracking for multi-vehicle system with a well-informed leader: Adaptive control method. In *Chinese Control and Decision Conference (CCDC)*, pages 1050–1054, May 2011.

[21] J. Petit and Z. Mammeri. Dynamic consensus for secured vehicular ad hoc networks. In *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8, Oct 2011.

[22] M. Wegner, W. Xu, R. Kapitza, and L. Wolf. Byzantine consensus in vehicle platooning via inter-vehicle communication. In *Fachgespräch Inter-Vehicle Communication*, pages 20–23, 2016.

[23] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Cigno. Plexe: A platooning extension for Veins. In *IEEE Vehicular Networking Conference (VNC)*, pages 53–60, Dec 2014.