Wireless Distributed Consensus in Vehicle to Vehicle Networks for Autonomous Driving

Chenglin Feng, Zhangchen Xu, Xincheng Zhu, Paulo Valente Klaine and Lei Zhang *Senior Member, IEEE*

***Abstract*—Vital societal and industrial autonomous components are increasingly interconnected through communication networks to complete critical tasks cooperatively. However, as the reliability and trust requirements for connected autonomous systems continue to grow, the centralized communication and decision approaches that are in use today are reaching their limits. Focusing on autonomous driving applications, this paper proposes a resilient and trustworthy framework on wireless distributed consensus networks, where the communication links are less reliable or are even in the presence of incorrect local sensor readings/decisions. To accomplish that, a novel three stages consensus mechanism is proposed based on the practical Byzantine fault tolerance (PBFT), where the veto collection and gossip stages are designed to meet the stringent and complex requirements for a vehicle's maneuvers. A plan tree synthesis is also proposed to make consensus on a series of decisions while adopting network members' decision preferences. A detailed protocol including the distributed consensus, plan tree synthesis, dynamic grouping, etc. is proposed. Simulation results show that the proposed consensus mechanism is able to be reached and propagated through the network under poor wireless communication conditions and the presence of faulty vehicles with incorrect sensor readings. The result can be extended to other autonomous systems to significantly enhance safety in critical industrial applications.**

***Index Terms*—V2V network, autonomous driving, connected autonomous systems, wireless distributed consensus, Byzantine fault tolerance**

I. Introduction

The application of IoT devices has seen a significant increase in critical applications, such as in industrial environments and in intelligent transportation systems (ITS) in order to aid these processes to make critical real-time decisions [1]. For example, today inside a car there are around 60 to 100 sensors, including Inertial Measurement Unit (IMU) camera, radar and Lidar. Those sensors collect data and help an autonomous system to make decisions on its own [2]. Such a ego-only autonomous driving system has received wide research interests. Tech-giants have demonstrated their ego-only system in recent years, including Uber, Tesla, Waymo and Baidu to name a few [3] [4] [5] [6] [7]. Though numerous solutions have been proposed, the perception range is limited within the line-of-sight due to the inherit drawback of on-board sensors. Besides, the devices inside a car only aid in the process of local decision making, or in other words, other vehicles may not be aware of a specific vehicle's decision. In addition, sensors are also prone to fail, which can lead to casualties. As such, local decision making and sensor faults can be extremely dangerous, especially in autonomous transportation systems where human lives can be lost, as local decisions taken by different cars or false sensor readings can be conflicting, and lead to accidents. As an example, in a fatal crash of a Tesla car, in which a car's sensor failed to recognize a large truck and trailer crossing the highway, leading the vehicle to drive full speed under the truck [8].

Thus, in order to overcome these issues, vehicle to vehicle (V2V) networks, or a broader concept of vehicle to everything (V2X) networks were introduced, in which communication networks, such as cellular networks, can be used to exchange information between vehicles as well as infrastructures [9] [10] [11]. Such V2X networks also enhance the functionalities of vehicular control in ITS. It extends the management of an ITS beyond traveling speed and vehicle spacing in platoon control. As the sensors and hardware of autonomous driving are becoming increasingly sophisticated, self-driving vehicles are now more capable than adaptive cruise control (ACC) and autonomous emergency braking (AEB), which brings more complex coordination in managing a cluster of vehicles. To maximize road safety and efficiency, the cooperative control of vehicles is studied in this paper to utilize this improved capability and managed complicated inter-vehicle interactions. To realize such cooperative control in V2X networks, vehicles within a network are further managed logically in either centralized or distributed, depending on its requirements. In centralized approaches, vehicles send their collected data to a specified vehicle or a central server, which is then responsible for making decisions. These decisions are then sent back to the vehicles, which act accordingly. Although centralized systems are simpler and bring more control over the decisions, it comes with its disadvantages. With the continuous growth of IoT devices and connected vehicles, centralized approaches are expected to serve more and more autonomous cars in the near future, which can result in a very expensive system as well as cause scalability problems, since all cars would have to send their data to a single point. In addition, single point of failure or malicious attacks can also disrupt the server operation, or even steal sensitive information from the connected devices [12].

C. Feng is with School of Information and Communication Engineering, University of Electronic Science and Technology of China, Sichuan, Chengdu, No.2006, Xiyuan Ave, West Hi-Tech Zone, China (e-mail:202111012027@std.uestc.edu.cn).

Z. Xu is with the Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195 USA (email: zxu9@uw.edu).

X. Zhu is with School of Integrated Circuits, Peking University, Beijing, No.5, Yiheyuan Road, Haidian District, China (e-mail: Xincheng.Zhu@stu.pku.edu.cn).

C. Feng, P.V. Klaine and L. Zhang are with James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, United Kingdom (e-mail:2357707F@student.gla.ac.uk; {Paulo.ValenteKlaine, Lei.Zhang}@glasgow.ac.uk).

Another alternative for management consists in adopting a distributed approach through distributed consensus protocols (also known as consensus algorithms), in which vehicles share information to one another and then agree on decisions jointly, instead of relying on a central authority. We further conclude such distributed approach as a **Perception-Initiative-Consensus-Action (PICA)** scheme. In PICA, a node makes initial decision based on local sensing and computing, then consent through a distributed consensus protocol among the relevant nodes before performing an action. Compared to centralized mechanisms, the networks in the distributed PICA scheme relies on peer-to-peer communication and organized in a distributed manner, which tend to have shorter routing path and lower costs [13]. Moreover, a central server is replaced with a distributed database, which is not controlled by any single party.

The distributed consensus algorithms is the key in PICA scheme, which can guarantee the nodes agree on an identical value, despite the existence of faulty nodes. It can be classified as crash-fault-tolerance (CFT) and byzantine-fault-tolerance (BFT) by whether the faulty nodes act randomly or simply crash [14] [15]. Two well-known applications based on distributed consensus are the newly emerged blockchain technology and the classic state machine replications (SMR) [16] [17] [18]. In the context of blockchain, the non-faulty members of a network agree on the contents and order of a block, where ordered blocks form a chain-link structure. Algorithms like proof-of-work (PoW) and proof-of-stack (PoS) are introduced to validate this process [19] [20] [21]. For the classic SMR, however, the service is provided by replicating the state of the nodes across the whole network [22] [23]. Compared the blockchain with SMR, the former has better scalability while the latter is less computational demanding and more efficiency in terms of consensus finalization. Therefore, we believe the SMR is more suitable for controlling numbers of vehicles in a time-critical situation. In particular, the well-known SMR protocol-Practical Byzantine Fault Tolerance (PBFT) [24]-is very simple to implement compared with the extensively-used PoW-based blockchain consensus.

Despite its advantages, the performance of the consensus might also be a bottleneck in V2V systems. First, the consensus can significantly be affected by the performance of the wireless communication network, especially in terms of latency, reliability and throughput [25] [15]. Second, unlike wired systems, wireless systems bring extra channel uncertainty, scarcity of spectrum provision, thus entailing different security thresholds. Third, the PBFT systems consider node failure and when it happens, all associated communication links are faulty. However, with dynamic wireless communication channels, a node may work fine, but some links connected with the node might be unstable. Fourth, traditional PBFT algorithms do not consider negative votes for a transaction, only abstentions are available, thus few but critical objections may be overlooked. Thus, there is a need to adapt existing consensus mechanisms to wireless environments and more specifically in this paper for the case of V2V networks for autonomous driving.

Since traditional consensus mechanisms have their limitations to be applied in V2V systems for platoon control, in this paper we carefully analyzed the properties that a consensus algorithm in V2V networks should follow, and presents some modifications to deal with the aforementioned concerns of the consensus. The main contributions of this paper are as follows:

1) We provide a novel perspective on the process of consensus/decision making for vehicles in autonomous systems. This proposal-based protocol utilizes PBFT-based consensus mechanism to enable vehicles within the network jointly reach an agreement, even in the presence of incorrect sensor reading/decision. In this proposal-based mechanism, a vehicle (i.e., proposer) sends a proposal of subsequent actions to other vehicles. Other vehicles then evaluate the feasibility of the proposal and jointly decide the execution under the control of the consensus algorithm.
2) We propose veto-collection phase with feasibility proofing procedure before the normal consensus process. The traditional PBFT algorithms do not consider negative votes in the consensus reaching process, only abstentions are available. The veto-collection considers critical minority, improving safety in vehicular applications.
3) We propose the plan tree synthesis to improve the efficiency and success rate of consensus reaching. Since the execution of proposals is jointly decided, a proposal may be rejected by part of the vehicles and fail the consensus reaching. To improve the probability of successful consensus reaching, we propose plane tree synthesis that combines multiple proposals into a plane three, which provide more proposal options.
4) We use a gossip algorithm as a complementary means to increase the success rate of the modified PBFT consensus algorithm in connected vehicular networks. The gossip algorithm enables nodes that fail in intermediate phases in PBFT to be synchronized by nodes that are successfully committed, resulting in a higher consistency throughout the vehicles in the network.

Note that apart from reaching consensus for autonomous driving, our protocol can also be applied in other distributed systems with similar requirements. However, since autonomous driving and V2V networks are emerging technologies, we mainly focus on the consensus for autonomous driving in this paper.

The remainder of this paper is organized as follows. In Section II, we present a literature review of consensus mechanisms applied in V2V scenarios. A brief introduction to PBFT is presented together with our proposed consensus scheme for autonomous driving through V2V networks in Section III. The Section IV elaborates the protocol of the consensus scheme. The further refinement of the consensus scheme is given in Section V. The performance of our scheme is analyzed in Section VI through simulation experiments, while conclusions are reported in Section VII.

II. Related Work

Distributed consensus mechanisms have attracted wide interest in autonomous systems and other related fields including ITS, Cooperative Adaptive Cruise Control (CACC),

Connected Autonomous Vehicles (CAV), Vehicular Ad-hoc Networks (VANET), autonomous vehicles, etc. [26] Preliminary consensus mechanisms, which focus on managing the inter-vehicle spacing in platoons, have been studied extensively in recent years [27]. Running under different traffic dynamics models [28], these platoon-based consensus require the parameters to be exchanged among adjacent vehicles. It describes the interactions of vehicles on roads and ensures a stable platoon formation, while also optimizing the traffic flow. With the known speed, acceleration and inter-vehicle spacing of vehicles, adjustments are taken accordingly [29]–[32]. To further improve the road throughput, multi-platoon consensus and other more general formation are proposed [33], [34]. Though the inter-vehicle spacing converges and stabilizes via consensus with traffic dynamics models, their inability in conducting complicated vehicle maneuvers limits their application.

On the other hand, other researchers have applied blockchain consensus mechanisms for constructing a more general information sharing in V2V networks [35]. A great number of works with proof-based consensus have been proposed recently. For example, in [36], Kang et al. deploy distributed smart contracts based on proof-of-work (PoW) and proof-of-storage consensus to help data auditing and verification. In their work, PoW nodes collect and verify local metadata to find available hash value with certain level of difficulty to get vehicle coins (a specific crypto-currency) as rewards to upgrade computation resources, while proof-of-storage nodes contribute storage resources to get vehicle coins for further upgrades. In [37] Yang et al. present a joint consensus consisting of both PoW and PoS, later they proposed a trust management system for data credibility assessment using Bayesian inference model. Yet the aforementioned solutions are computationally demanding since nodes are competitively mining.

Despite traditional PoW-based consensus mechanisms being thoroughly investigated, a few other studies implement alternative consensus schemes. In [38], for example, the information sharing is secured by the proof of data contribution frequency and proof of energy contribution amount. In their distributed consensus mechanism, data coins and energy coins serve as cryptocurrency for vehicular applications. Besides, the Byzantine fault tolerance is identified as another feasible solution. In [39], a privacy-preserving announcement network enabled vehicles to forward and receive information by using reputation points and the Byzantine fault tolerance consensus algorithm. In [40], Hu et al. introduced an authentication based on an improved Byzantine consensus algorithm for the Internet of vehicles. In [41], Liu et al. combined the Byzantine-fault-tolerant consensus algorithm for connected vehicles (BFCV) with the concept of Proof-of-Eligibility challenge to ensure information security with compromised vehicles and without privileged members. Wegner et al. [42] in their paper proposed an innovative consensus called BFT-ARM, which can fit real sensor values and guarantee that the decisions only require median validity instead of strong validity [43].

However, despite the validity of all approaches, there are still gaps. As previously mentioned, PoW/PoS consensus mechanisms have high computational complexity, can be quite slow, and can also generate competition among vehicles for their rewards, while other consensuses such as PBFT do not scale up very well. Moreover, traditional consensus mechanisms were primarily designed for stable environments, not for unreliable or fast-paced applications such as the ones involving vehicles or a wireless channel. Based on that, it is clear that novel consensus mechanisms that take into account the instability of the wireless channel as well as the stringent constraints of V2V networks need to designed.

III. The Distributed Consensus for Autonomous Driving

In this section we briefly introduce PBFT, a distributed consensus mechanism. Based on PBFT, we explore the use of distributed consensus for joint decision making in autonomous systems and introduce a proposal-based mechanism. Then, we discuss the use of a vehicle cluster as an organisation for the consensus, and discuss its size, threshold and self-organization scheme. After that, we analyze the shortcomings of PBFT applied to V2V consensus networks, i.e., the inability to perform a one-vote veto and the incapability to guarantee that all nodes are aware of the result of the consensus, and propose a novel solution. For a more efficient consensus, we introduce the plan tree, which packages multiple related operations together and make decisions through a single consensus. Our detailed consensus protocol for autonomous driving based on these discussions is then proposed in Section IV. Note that in this context, each vehicle in V2V networks is equivalent to one node in the consensus algorithms, so the term *node/replica* and *vehicle* have the same meaning.

A. PBFT-based Distributed Consensus for Autonomous Systems

The practical Byzantine fault tolerance (PBFT) is a three-phase state machine replication (SMR) protocol widely used in distributed systems [24]. It based on a partially synchronous network assumption that the message delivery is guaranteed but with an uncertain amount of delays. As a Byzantine fault tolerance algorithm, it is capable of reaching a consensus in the presence of up to 1/3 of Byzantine nodes (i.e., malicious or faulty nodes) in a network. Here we briefly describe the conventional process of PBFT.

In a PBFT consensus network there are many replicas among which one is designated as the primary node, and the others are considered as backups. Let us assume that there are N replicas in the consensus network and f is the maximum number of faulty replicas. To ensure safety and liveness, $N \geq 3f + 1$. In a normal-case consensus process, the consensus is triggered by a client sending a request to the primary node. On receiving the request, the primary node enters the *pre-prepare* phase by broadcasting pre-prepare messages. After receiving the pre-prepare message from the primary node, the replicas broadcast the prepare message and enter the $prepare$ phase, if the pre-prepare message is valid. Then in the prepare phase, if a replica receives more than $N - f$ valid prepare messages from the consensus network, it

reaches the prepared state and broadcasts the commit message. Similarly, a node reaches a committed state once more than $N - f$ valid commit messages are received. The node is then able to execute the request and reply the execution result to the client. Fig. 2 (normal PBFT consensus process) illustrates the details of this process.

However, despite its success, the traditional PBFT algorithm needs improvements, especially when applying to joint decision making in autonomous systems. To enable the consensus reaching process involving the preferences of nodes instead of solely relying on executing requests from a client, we introduce a **proposal-based** mechanism, which is different from the traditional request-based mechanism of PBFT. In this proposal-based mechanism, the request from the client can be regarded as a proposal of an action that the client expects everyone to implement. The client can be one of the replicas, and the consensus process is a procedure that all replicas vote on the proposal. Also, there is no need for the replicas to return the execution result back to the client node. For example, in autonomous driving, a vehicle (i.e., a client in the consensus process) in a cluster makes a proposal to increase the group's speed. After receiving the proposal, the consensus network that is composed of the nearby vehicles will evaluate if this proposal is reasonable. After reaching the committed state, the proposal is considered to be agreed by the network members and the vehicles in the consensus network can execute the contents of the proposal (i.e., increase the vehicles' speed).

B. Ordinary Consensus and Full Consensus

Typically, consensus defines that a majority (or a certain percent) of nodes agree on the requested information. For example, in PBFT, up to 1/3 of the nodes can be malicious nodes or faulty nodes (if more than that, then the consensus cannot be performed completely or collaboratively). In such an **ordinary consensus** protocol, some nodes (e.g., due to communication failure) cannot notice whether the consensus has been achieved or not. Whereas, in static systems where the client does not participate in, but only triggers the consensus reaching process, it does not matter that some nodes do not know the consensus result. Let us take vehicle navigation in autonomous driving as an example. In this case, when a vehicle requests map data from other vehicles, the cluster reaches a consensus on the request and returns a map to the client. However, the act of sending the map to the client does not affect the behavior of a vehicle in real time, so whether a committed state is reached or not has no effect on the vehicles in the consensus network. However, there are scenarios that each node can be affected by the consensus result. Consider the previous example of lane change in Section III-D, however this time, let us assume that there is no danger in changing lanes to overtake. In order to ensure that every vehicle performs the lane change operation, we have to make sure all nodes are aware of the results of the decision, even though there is communication uncertainty. We call this a **full consensus**, in such case, we propose that there is a synchronization step following the consensus, where the consented nodes send the consensus result to other nodes that are not aware of the result. This can be done either through broadcasting communication or point to point communications. Thus a successful full consensus is only achieved if and only if all nodes are synchronized. We introduce a gossip algorithm in Section V-B as a possible synchronization method.

C. Proposal Combination

Another issue in wireless distributed consensus network for autonomous driving is that of complicated maneuvers. The more complicated the maneuver, the less likely it is to coordinate the whole process by a single proposal. As such, an intuitive approach is to decompose one maneuver into multiple actions, then propose and execute them in a specific sequence. Yet, this leads to an increase in delay as a consensus process is required for each action. To avoid that hazard, the proposals for a series of actions may be combined together and processed within one consensus as shown in Fig 1.

Since the combination of actions are executed in a specific sequence, any action that fails to pass the consensus process would result in the fail of the whole maneuver. As such, the failure probability increases as it gets more complicated, since there are more actions to consider in the consensus. Furthermore, there may be different solutions for one maneuver with similar effects that can be implemented. Thus, it is necessary to propose multiple plans for one maneuver simultaneously and collect preferences from network members, which constructs a plan set similar to a decision tree, referred to as the **plan tree** in this paper. With this plan tree, the consensus network may select the best plan among different proposals at one time. The detailed protocol about this plan tree synthesis is introduced in Section IV-G.

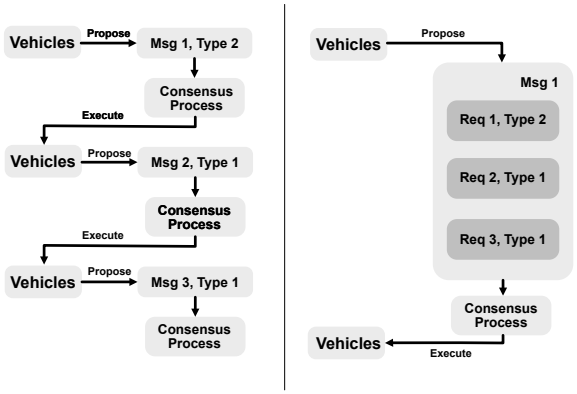

Fig. 1. Combine a series of actions into one proposal.

D. One-vote Veto

A challenge to apply traditional BFT consensus protocol in autonomous systems such as autonomous driving is that it does not take the "one-vote veto" into consideration. For example, a member at the back of the cluster has proposed a lane change maneuver to overtake another vehicle. Yet, another member at the front of the cluster detects that there is a vehicle in the target line, and this lane change action may lead to a collision. Therefore, the member at the front may need a one-vote veto

to terminate this consensus, thus stopping the lane merging action. However in traditional BFT protocols, a vehicle that opposes a proposal cannot vote against it, but is rather absent from voting when collecting responses in intermediate phases. Therefore, consensus may still be reached and an accident may occur. Due to the lack of the "one-vote veto" in tradition PBFT, few but critical objections may be overlooked.

Recognizing the above challenge, a V2V consensus network should retain the ability for its members to have a right of this kind of rejection. Based on that, we introduce a feasibility proofing procedure before the normal consensus process to address this issue. In this procedure, a member in the network puts forward a proposal to the leader like normal. On receiving the proposal, the leader does not go directly to the pre-prepare phase of PBFT, but requires the rest of the cluster to prove the feasibility of this proposal in a new phase called **veto-collection phase**. Here we use a cryptography method: threshold signature [44] to perform this process. In a (k, N)-threshold signature algorithm, there is a single public key held by all replicas, while N replicas hold a different private key fragment. If there are at least k replicas that signed a partial signature to the proposal, the k partial signatures can then be combined into a complete signature, which can be verified by the public key held by all members. In the veto-collection phase, the replicas sign a partial signature to the proposal if they have no objections, and then unicast the signature back to the primary node. The feasibility proofing can be proved by the feedback from replicas if there are N partial signatures, i.e., all vehicles agree on the proposal. Then, the primary vehicle combines the partial signatures to a complete one and send as evidence in the following consensus procedures.

The potential hazards can be avoided by feasibility proofing, but this comes at a cost. On the one hand, it consumes more time than a normal PBFT, and on the other hand, the malicious nodes may keep voting against the consensus to prevent it from being reached. As a result, one-vote veto is recommended only for operations that require the awareness and consent of all nodes, e.g., dangerous but necessary operations. Meanwhile, ordinary PBFT still has its advantages (e.g. less time consuming and higher consensus success rates) and can be used in relatively common situations. For simplicity, we define the following two types of proposals:

- Type 1: A proposal that requires a number of proofs from other members.
- Type 2: A proposal that requires no veto from other members.

In the above two types, type 1 proposals only need to satisfy the PBFT threshold (more than 2/3 nodes agree), while type 2 needs to use the veto-collection phase to ensure that there are no negative votes.

E. Vehicle Clustering

A wireless distributed consensus network for autonomous driving requires high reliability and low latency. This becomes increasingly challenging as the number of vehicles increase, as vehicles are geographically distributed. Moreover, because of high communication complexity, PBFT is proven to not scale up very well [45]. Thus, instead of managing all vehicles in a V2V network, one common approach is to manage vehicles in different clusters. Clustering is an effective technique to manage and organize vehicle networks while keeping the system decentralized. It limits the signaling and routing overhead, and the links within a cluster are more stable if well chosen [46] [47]. In terms of its structure, each cluster can be composed of a leader and multiple members, where the leader plays a management role. By comparison, PBFT also has a primary node (can be regarded as the leader), which invokes and manages the consensus reaching process and several replicas (group members). In other words, the consensus mechanism is able to be mapped to a clustered vehicle network. For the above reasons, vehicle clustering techniques can be used in a distributed consensus network.

Regarding the cluster size, we propose that it is based on the consensus reliability and communication reliability requirements. For example, larger network sizes are generated if the consensus reliability requirement is high, while a smaller sizes are considered if the consensus reliability requirement is low. However, we should also consider the latency and the wireless communication range between vehicles. For instance, if the task requires a lower latency or high communication successful rate, smaller networks can be built, and vice versa. Meanwhile, the size of a cluster can also influence the threshold of a voting-based BFT consensus. When the vehicles are far away, the success rate of communication between vehicles is relatively low. For a vehicle, failing to receive responses from others can be regarded as Byzantine behaviours. Moreover, considering time-varying network conditions, the vehicle may calculate a threshold for a consensus process based on the non-faulty response probability between itself and other vehicles in a period of time. Details of the cluster size and dynamic threshold are discussed in Section V-A.

In terms of self-organization scheme of consensus networks, in a static distributed system, all nodes are typically static, thus the consensus network can also be static. As a result, the grouping and degrouping can be done in the initial instalment. In the case of a dynamic environment that the nodes are moving (either physically or logically), e.g., in autonomous driving, initial grouping to form a consensus network is required, this can be done either by a broadcasting or peer to peer network communication. In the case of a certain period without consensus tasks or a consensus cannot be achieved in a determined period of time (in case vehicles are too far from each other, thus communication links cannot be established), the consensus network can be degrouped. In the case of an emergency, such as a vehicle breakdown, e.g., is unable to keep up with the cluster's speed, the vehicle may leave the cluster. Also, when merging, there may be new vehicles wishing to join the already existing cluster. The protocol related to dynamic change of cluster members is introduced in Section IV-F.

IV. V2V Network Consensus Protocol

In this section, we propose a practical wireless distributed consensus protocol for autonomous driving. This is a proposal-based protocol (as discussed in Section III-A), where each

vehicle declares its plan and executes the corresponding vehicle operation if it is approved by the distributed consensus network[1]. It is also worth noting that the proposed protocol can not only be used in vehicular networks, but can be applied in other general scenarios that is composed of multiple parties, from multi-agent robots and cooperative UAVs control to data update of servers clusters and block-chain applications. As the application of vehicular networks are emerging and the coordination in V2V networks are an exemplar of multi-agent coordination, we mainly focus on vehicular application in this paper.

The protocol is based on PBFT, and we consider unintentionally incorrect decisions made by nodes based on sensor errors or failures as Byzantine errors. As the vehicular applications requires high reliability for safety issue, we classify the proposals with different validation requirements into three modes, which will be shown in Section IV-A. The maneuvers for vehicles can be regarded as combinations of multiple proposals in different modes and have been executed by the network in a commonly agreed sequence, as discussed in Section III-C. We also consider the actual case for the one-vote veto of the proposal and propose a veto-collection phase before the pre-prepare phase.

We assume the total number of vehicles in a cluster is N. f is the number of replicas that may be faulty and can be calculated as $f = \lfloor (N-1)/3 \rfloor$. Typically, the threshold is the minimum T that satisfies the inequality $2T - N - f \geq 1$. However, in the case of large networks or high sensor failure rates, the threshold for nodes to proceed to the next phase can be calculated using the method in Section V-A. We show a complete V2V network consensus process in Fig. 2 below.

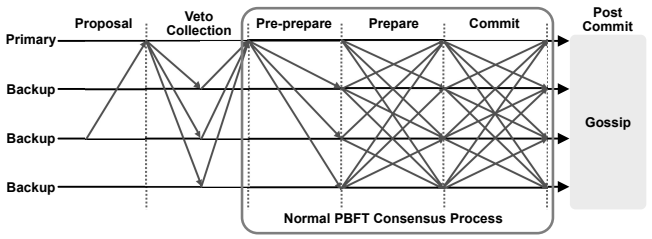

Fig. 2. Full PBFT consensus protocol with veto collection and gossip.

In the following content, we will focus the details of the protocol, including the operations of each phase, three sub-protocols (view change, dynamic grouping, and plan tree synthesis) and the proof of safety and liveness.

A. Client Proposal

When a situation arises that requires a consensus, any vehicle can be the initiator of the consensus, which refers to the proposal-based consensus in Section III-A. Based on the type and structure of the proposal, we divide the proposal into the following three modes[2]:

- Mode 1: A proposal that requires a number of proofs from other members, i.e. normal PBFT requirements.
- Mode 2: A proposal that requires feasibility proofing in the veto-collection phase, i.e., no veto is collected.
- Mode 3: A plan tree, which is discussed in detail in Section IV-G.

We design the format of a proposal as $[mode, o, r, t, (t_{exec})]_{proposal}$, which contains the proposal $mode$, a vehicle's operations requiring consensus o, the reason to propose this proposal in standardized expression r, current timestamp t, and appointed execution time of the operation t_{exec} (optional). The parameter t_{exec} is the absolute time set by the proposer according to the average time needed for consensus (with a margin), in order to make each car perform operations at the same time and to better unify the behavior of each vehicle. After generation, the proposal messages are multicast to the primary node, which then differentiates the modes and performs different actions.

B. Veto-collection Phase

The veto-collection phase aims to provide the ability of "one vote veto" and will only be activated when required. When the leader (i.e., the primary of PBFT) receives a proposal $M = [mode, o, r, t, (t_{exec})]_{proposal}$, it authenticates the proposal as well as the client's identity. If the proposal is of mode 1, it omits the veto-collection phase and enters the normal PBFT consensus process (Section IV-C). Instead, if the message is of mode 2, the nodes enter the veto-collection phase, in which the primary node calculates the message's digest d generated by collision-resistant hash functions, and multicast message $[mode, o, r, t, (t_{exec}), d]_{veto-collection}$. Mode 3 is for plan tree synthesis and details will be discussed in Section IV-G.

On receiving a veto-collection phase message, the vehicle invokes sensors or in-vehicle data to verify that the actual situation and in-message reason r are consistent to make decisions. If the in-message operation o is not considered problematic and is accepted by the vehicle, it signs a partial signature ps of this veto message (mentioned in Section III-D, threshold signature algorithm) and unicast message's digest and partial signature together as $[d, ps]_{veto-reply}$ to the primary node. Else, if there is a conflict between the operation o and its own situation or the car is unable to execute the command, the vehicle rejects this message by unicasting $[d, VETO]_{veto-reply}$ to the primary node.

Only when the primary node has received N valid $veto-reply$ messages (including the primary node itself) and there is no veto, can the primary node implement the following normal consensus process, otherwise the consensus is terminated by the primary. After validation, partial signatures from all N vehicles are combined into a complete signature noted as sig in the following sections.

C. Normal Consensus Process

After successfully proving the feasibility of the veto-collection phase (i.e., a valid complete signature sig is combined by the primary), the protocol proceeds to the normal

[1]Note that the network can be composed of vehicles, or also road side units, such as cellular base stations, or a mixture of them.

[2]Note that the mode defined here is not the same as the types defined in Section III-D.

consensus process of the PBFT protocol. The normal consensus process includes the pre-prepare, prepare and commit stages. Here, we integrate the vehicular decision-making and implementation into the normal PBFT process.

1) Pre-prepare: Assume that the primary node, after receiving the message, has checked the authenticity of the message, classified, and completed the relevant steps of the veto-collection phase. In the pre-prepare phase, the primary node assigns a serial number n to the proposal, packages it into a pre-prepare message and broadcasts it to the cluster. The pre-prepare message is $M_{pp} = [v, n, d, M, (sig)]_{pre-prepare}$, where v is the current view, d is the digest of the proposal M and sig is the complete signature, if the request message belongs to mode 2.

When receiving pre-prepare messages, the node validates the message in the following steps: 1) First, it checks if signatures and digits are correct; 2) Then, it checks if the view is the same as this node; 3) In view v, it checks if it has not accepted a pre-prepare message with sequence number n containing a different digest. Once the validation passes, the pre-prepare message is accepted, and the node enters the prepare phase, broadcasting the prepare message $[v, n, d, i]_{prepare}$ where i is the vehicle's serial number.

2) Prepare: In the prepare phase, nodes check whether there is a message M, a pre-prepare message M_{pp} of M, and T prepare messages from other nodes in its logs.

If all of the above points are satisfied, in the case of the mode 1 of the proposal, then the vehicles detect the surrounding area using their information (typically, by their own sensors) to see if it matches the description in the message and check the validity of the operation instructions. If all checks passes, the node reaches the prepared state, enters the commit phase and multicasts the commit message $[v, n, d, i]_{commit}$.

Unlike mode 1, in the case of mode 2 proposals, the vehicle has already checked the surroundings and acknowledged the reasonableness of the proposal in the veto-collection phase, so another inspection is not necessary. Instead, it verifies the signature sig provided by the primary node to confirm the consent of all other nodes for this proposal. If verification passes, it passes to the commit phase and multicasts the commit message $[v, n, d, i]_{commit}$.

3) Commit: If a node is prepared and more than T commit messages from other nodes have been received (possibly including its own) for a proposal, the consensus of the validity of the operations in the proposal is confirmed by the consensus network and the node reaches the committed state, so operation can be done by the vehicles immediately or in the appointed time if t_{exec} is defined.

D. Post-commit

The post-commit phase can be considered as a synchronization step for better reaching the full consensus (as defined in Section III-B). It utilizes the gossip algorithm proposed in Section V-B to spread post-commit messages in the V2V network and synchronize consensus results. When a vehicle reaches the committed state for a proposal message, it broadcasts a post-commit message $[M_{pp}, C, i]_{post-commit}$ to other nodes where C is the collection of at least T commit messages this vehicle has received. Then, the vehicles that receive the post-commit message also reach a committed state and continue to broadcast this message to other vehicles. This action continues until all nodes reach the committed state.

The post-commit message has two main functions. First, it enables replicas which did not receive the pre-prepare message to regain the pre-prepare information. Second, it let other nodes that fail to collect enough messages in previous phases enter the committed state directly by proving adequate commit messages.

E. View Change

Consistent with the original PBFT protocol, when a node detects a timeout on a proposal, it sends a view change proposal in the following format: $[v+1, n, C, P, i]_{view-change}$. The parameter n is the sequence number of the last stable checkpoint, C is the set of valid T messages in that checkpoint and P is a set of P_m, where P_m is the set of messages with the serial number m ($m > n$) that have reached the prepared state. When a vehicle with serial number $v+1$ receives T valid view change messages, it broadcasts $[v + 1, V, O]_{new-view}$ where V is the set of view-change messages, and O is the set of pre-prepare messages. In the new view, the vehicles are in the consensus process based on the prepared state of each vehicle in the previous view, which ensures that the message serial number does not change before and after view change. Through view change, the safety of the consensus network can be guaranteed.

F. Dynamic Grouping Protocol

In a V2V network, the vehicle cluster may change dynamically, as vehicles may join or leave the network at any time. For a consensus, the current number of vehicles N of the network stored in each node's memory should be the same in order to ensure the safety and liveness of the consensus. To ensure that all nodes have the same N, allowing vehicles to join or leave also requires consensus, so nodes can update N based on the consensus result. For the above reason, a dynamic grouping is designed on the basis of our V2V network consensus protocol. Considering that there are no strict requirements or restrictions, for a faster consensus process, the accession and exit proposal belongs to mode 1 proposal which only requires number of proofs.

1) Vehicle Accession: The format of an accession proposal from a vehicle outside the network can be expressed as $M_{accession} = [mode1, info, t, t_{exec}]_{accession-request}$, where $info$ contains information of the vehicle and t is the current timestamp. The accession request message is then unicast to the nearest vehicle which is already in the network. This vehicle plays the role of the client and forwards the request to the primary node. Since the accession proposal belongs to mode 1, a normal PBFT consensus process is executed, and in this process the vehicles determine if the current network is too large or not. If enough positive responses are collected and the vehicles in the network reach a committed state, it

renews its $N' = N + 1$ at $t = t_{exec}$. The vehicle that served as client also replies to the requesting vehicle the consensus result and the value of N'. After $t = t_{exec}$, the vehicle automatically becomes a member of the network and participates in the consensus. However, if there are any other consensus in progress, where the request timestamp $t < t_{exec}$, the consensus is still calculated with the previous value of N.

2) Vehicle Exit: When a vehicle needs to leave the V2V consensus network, it sends an exit request $M_{exit} = [mode1, t, t_{exec}]_{exit-request}$ where t is the current timestamp. After a normal consensus process, if it reaches the committed state, the vehicles in the network renew its $N' = N - 1$ at $t = t_{exec}$. The vehicle that requests to leave stops participating in the consensus after $t > t_{exec}$. Similarly to the previous case, any other consensus in progress where request timestamp $t < t_{exec}$ is still calculated with the previous N. When a vehicle leaves silently and does not make an exit request, any other vehicle in the cluster can make a request to remove the vehicle that has left. As vehicles in the cluster collect broadcast messages from other vehicles, once they found that one of the vehicles has been out of contact for a long time, it may propose the removal of that vehicle.

G. Plan Tree Synthesis

As mentioned in Section III-C, proposal combination can decrease the delay caused by multiple proposals and is vital for complex maneuvers. However, when combining the veto-collection phase with the proposal combination, even if one sub-proposal in the combination is voted against, the whole maneuver cannot be executed. Therefore, the acceptance rate of the combined proposal might be low for complex maneuvers. Besides, considering that the vehicles are closely operated with each other in dense traffic, executing inappropriate decisions may pose a threat to adjacent vehicles. For instance, an obstacles is detected in front and the platoon can either breaks or switches lane. If one vehicle in the platoon is heavily loaded, a breaking may lead to rear-end collision. To overcome these two problems, we propose a plan tree synthesis protocol which synthesises multiple plans composed of basic proposals of two modes (mode 1 and 2 defined in Section IV-A). For the former problem, as the number of alternatives increases with each consensus, the probability of infeasibility of all options decreases, thus the success probability of a single consensus increases. For the second case, the plan tree synthesis can combine multiple proposals (line switching and breaking in the example), and infeasible proposals (breaking in this example) can be voted against by nodes who detected the anomaly. We define a plan tree as a *mode 3* proposal.

One complete process is depicted in Fig. 3. The vehicle that is about to execute one maneuver proposes several candidate plans, and it then synthesizes them into a plan tree. When there is more than one option after one action, branches are created. Every path which starts from the root node to the leaf node represents a plan. Then the plan is packed into one message and handled as in Fig. 2. Instead of voting for one proposal, a member votes for each action that is possible on its view in the plan tree, with the preferences of members being expressed

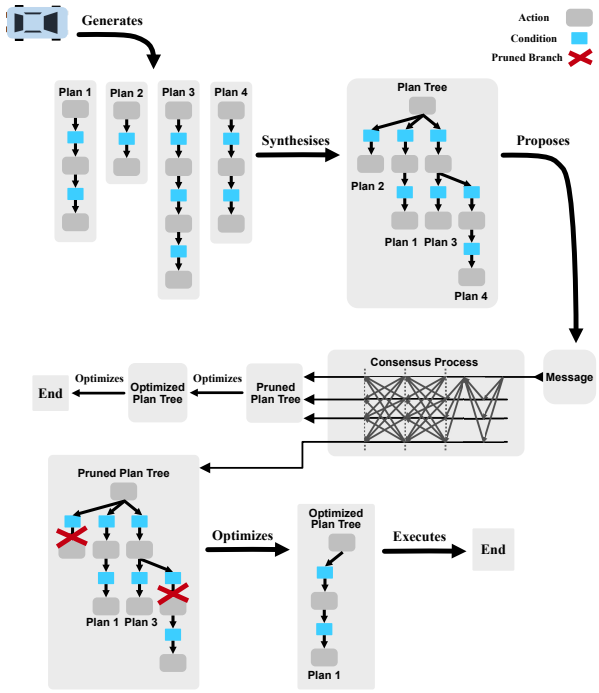

Fig. 3. Complete plan tree consensus process.

as vetoes in the veto-collection phase. When one member has identified one action in a plan is impossible to be executed, it gives out a veto. Thus, the sub-tree below that condition node is therefore pruned.

In terms of protocols, the client generates a proposal $[mode3, tree-structure, tree-content, t, (t_{exec})]_{proposal}$, where $mode3$ indicates that this proposal belongs to a plan tree. Apart from t for the current timestamp and t_{exec} for the appointed execution time for coordination, there are two parameters that are different from a normal proposal. Parameter $tree-structure$ contains the structure of the plan tree, and $tree-content$ contains the operations and reasons of different plans in a plan tree. Then, the message is sent to the primary node and enters the veto-collection phase. After receiving all veto-collection messages, the vehicles evaluate the reasonableness of each action of this plan tree and veto the actions that are considered to be unreasonable. The result is then unicast back to primary node with format $[d, marked-tree, ps]_{veto-reply}$, where $marked-tree$ stores the veto for each action in the plan tree. The primary node then collects N valid $veto-reply$ messages from all nodes, packages them as $mark-collection$ and enters the normal consensus process with pre-prepare message $M_{pp} = [v, n, d, M, mark-collection, sig]_{pre-prepare}$.

The chances are that more than one plan survives after the consensus process (which is desired since it does not require further retries). After receiving the preferences of all nodes packed by the primary node, to ensure that all vehicles execute one identical plan at a time, an optimization mechanism should be predefined on each vehicle. For example, the plan with the shortest execution duration can be chosen. However, if there still exists more than one plan, the final decision could

be based on predefined priorities. With plan tree synthesis and optimization, the preferences and opinions from other members are considered in the consensus process to some extent. However, the detailed optimization depends on specific application scenarios thus is not covered in this paper.

H. Proof of Safety and Liveness

Safety of the protocol guarantees that all non-faulty vehicles agree on the sequence numbers of proposals that commit locally. In a system where at most f out of $3f+1$ nodes is faulty, the threshold/quorum size is set to $2f+1$. Thus for two consecutive rounds of communication, there exists an intersection of at least $f+1$ nodes. Since there are at most f faulty nodes, there must be at least one non-faulty in this intersection. This is how the safety can be guaranteed. If a dynamic threshold mentioned in the following V-A is implemented, the algorithm also guarantees that there is at least one non-faulty node in the intersection.

Liveness means that the consensus continues without getting stuck. If we consider the communication failures of the vehicles as Byzantine errors, the network can be considered to be a partially synchronous network, which is similar with traditional PBFT algorithm. In this case, the view change replaces fault primary nodes to keep the consensus going, so the liveness can also be guaranteed.

V. Consensus Refinement

The PBFT is introduced in our consensus scheme. However, it is impractical to predetermine a fixed cluster and threshold size under wireless connection quality varies. It is also noticeable that a extra synchronization is required to reach full consensus as discussed in Section III-B. Therefore, we try to improve the performance and reliability of our consensus scheme in this section by introducing the group and threshold size selection as well as epidemiology gossip [48].

A. Cluster and Threshold Size Selection

In the traditional PBFT, the upper-limit of faulty nodes is a known fixed number. Thus, the threshold/quorum size is set as $N-f$ to ensure non-faulty nodes intersection. In practice, however, sensor and communication failure is more practically to be modelled as a probability rather than determining number assumed in the original PBFT protocol. Note that here communication failures could be caused by either the communication link is too weak (only affects the communication link with a particular node) or the node is faulty (thus all communication with the node will fail).

Similarly, to reach a consensus in an autonomous driving network, both the cluster and the threshold size should meet certain conditions (i.e., larger than certain numbers) for achieving required minimum possibility of existing an intersection (i.e., minimum consensus reliability). In this subsection, we will first derive the analytical relationship between the size of a cluster and the probability that a non-faulty vehicle exist in the intersection. Then we derive the relationship between the size of the threshold for a vehicle to proceed into the next phase of PBFT protocol and the probability that at least one feedback is received from a non-faulty vehicle in the intersection. Assume that the success rate of point-to-point communication between two nodes is P_c, the probability that one vehicle replies a faulty response (node failure rate) is P_r and the size of vehicles in a cluster is chosen as N_C. The probability of such existence is denoted as $P_{Cint}(N_C)$.

If a vehicle produces an incorrect response due to sensor failure, this vehicle is said to be faulty, therefore, the probability that then number of feedbacks from non-faulty vehicles in each round of communications N_b equals to n_b is given as

$$P_{(N_b=n_b)} = \binom{N_C}{n_b}(P_cP_r)^{n_b}(1-P_cP_r)^{(N_C-n_b)}. \quad (1)$$

The existence of vehicles with correct response in the intersection indicates $2n_b - N \geq 1$. The probability that the intersection $P_{Cint}(N)$ exists is given as

$$P_{Cint}(N_C) = \sum_{n_b=\frac{N_C+1}{2}}^{N_C} P(N_b = n_b). \quad (2)$$

Equation (2) gives an indication when selecting size of a vehicle cluster, and can be used as a criterion for accepting other vehicles to join the consensus network. For one who expects the probability that the existence of an intersection to be higher than a level Δ_C, N_C should be chosen so that $P_{Cint}(N_C) \geq \Delta_C$.

In the consensus reaching process, one node steps into the next phase once the number of valid feedbacks it receives exceeds a threshold. Correspondingly, the threshold for each node to proceed into the next phase should also be considered, since the exact number of faulty responses is unknown to a vehicle. Here we derive the relationship between the size of the threshold T and the probability of the existence of a non-faulty intersection $P_{Tint}(T)$. Assume that one vehicle has already received N_r feedbacks, and that the number of faulty feedbacks, F, is unknown to that vehicle. To ensure at least one non-faulty feedback intersection, the parameters T, N_r and F should satisfy the inequality $2T-N_r-F\geq 1$. Therefore we obtain

$$F \leq 2T - N_r - 1, \quad (3)$$

where the right side of the inequality is the upper limit of the number of faulty nodes (to ensure the non-faulty intersection existence). For simplicity, we denote the limit as $L(T) = 2T - N_r - 1$. Assuming that $P_{(N_F=n_F)}$ indicates the probability that the number of faulty feedbacks N_F equals to n_F, $P_{(N_F=n_F)}$ can be expressed as follows

$$P_{(N_F=n_F)} = \binom{N_r}{i}\left(P_r^{N_r-i}(1-P_r)^i\right). \quad (4)$$

Since the existence of a non-faulty intersection requires the number of faulty vehicles to be less than $L(T)$, $P_{Tint}(T)$ is then calculated by summing up the node failure rate $P_{N_F=n_F}$ from $n_F = 0$ to $n_F = L(T)$, as

$$P_{Tint}(T) = \sum_{n_F=0}^{L(T)} P_{(N_F=n_F)}. \quad (5)$$

For a vehicle that expects the probability of the existence of correct feedbacks in an intersection to be higher than a level Δ_T, the threshold T should be chosen so that $P_{Tint}(T) \geq \Delta_T$. Apart from a unified P_r, this probability can also be distinct for each vehicle. It is noticeable that the value of (5) is not always close to the threshold expectation $T = 2f_{exp} + 1$, where f_{exp} is summation of each vehicle's P_r. As an example, we randomly generate P_r for each vehicle and combine them into a vector $\boldsymbol{P_R}$ =[0.0152, 0.0133, 0.0849, 0.0954, 0.0251, 0.0015, 0.0632, 0.0619, 0.0447, 0.0726, 0.0905, 0.0868, 0.0141, 0.0450, 0.0578, 0.0137, 0.0464, 0.0703, 0.0735, 0.0006], the threshold T calculated by expectation is $T = 3$. This threshold expectation is impractical since it is less than the no intersection limit, i.e. half of the number of nodes. On the other hand, the intersection of $\Delta_T = 0.999$ is guaranteed at $T = 13$ as shown in Fig. 4.

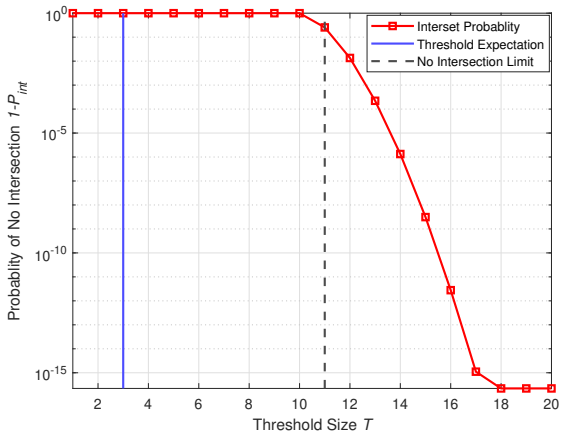

Fig. 4. Dynamic threshold under specific P_r with $N = 20$, $\Delta_T = 0.999$.

B. Consensus Reaching and Execution

Full consensus is necessary as vehicles may require to take identical course of actions under V2V scenarios, yet PBFT does not provide this property. Therefore, we utilize a Gossip algorithm epidemiology [48] to make up for the deficiencies in PBFT. In our design, the node who has reached consensus (committed) is considered to be infected, and thus is contagious. Then, contagious nodes communicate with others to spread the committed information. After that, the nodes that receive the committed information from a contagious node turn contagious as well, while the previous contagious node becomes immune and stops broadcasting committed information. Nodes repeat the process described above until all the nodes in the network are infected with this information. Generally, there are two schemes to implement this infection process. A node can either passively wait for an information update from others, or it actively contacts others for an update. The mathematical analysis of the two schemes are given in the following. In addition, since the performance differences are negligible with relatively high communications quality, the passive scheme is chosen in our protocol to reduce inter-node communications.

1) Passive:: in a passive gossip, nodes without the information passively wait for the communication from those who have the committed message. The probability of a node to be uninfected in the i^{th} round of infection is denoted as P_i. The number of total nodes is N and the success rate of point-to-point communication is P_c. Here we consider the average number of infected nodes to be $N(1 - P_i)$ for simplicity. In a round of infection, an uninfected node passively waits for the messages from infected nodes. If this node is still uninfected in the $(i + 1)^{th}$ round, all the infected nodes have failed to establish the connection, which yields

$$P_{i+1} = (1 - P_c)^{N(1-P_i)}. \tag{6}$$

It is clear that there exists an asymptote by plotting the iteration results (passive in Fig. 5). Since $\lim_{i\to\infty} P_{i+1} - P_i = 0$, we force $P_{i+1} = P_i$ to derive the expression of the asymptote. By denoting $(1 - P_c)^N$ as a, substituting P_{i+1} with (6), and multiplying both sides with a^{P_i}, gives

$$a = P_i a^{P_i}. \tag{7}$$

One obvious solution for (7) is $P_i = 1$ for (7). To find the real solution, we choose the principal branch W_0 of the Lambert W function [49] then substitute a with $(1 - P_c)^N$ to derive expression of P_i, given as

$$P_i = \frac{W_0(N(1 - P_c)^N ln(1 - P_c))}{Nln(1 - P_c)}. \tag{8}$$

From (8), we conclude that the asymptote of the uninfected probability, P_i, is solely affected by the success rate of communication P_c for a fixed number of nodes N.

2) Active:: in an active gossip, the node without information actively asks for it. If an uninfected node at the i^{th} round of infection is still uninfected in the $(i + 1)^{th}$ round, this implies that all nodes it communicated with in the i^{th} round must be uninfected as well. The average number of nodes it can communicate with in the i^{th} round is NP_c, which yields:

$$P_{i+1} = P_i^{NP_c}. \tag{9}$$

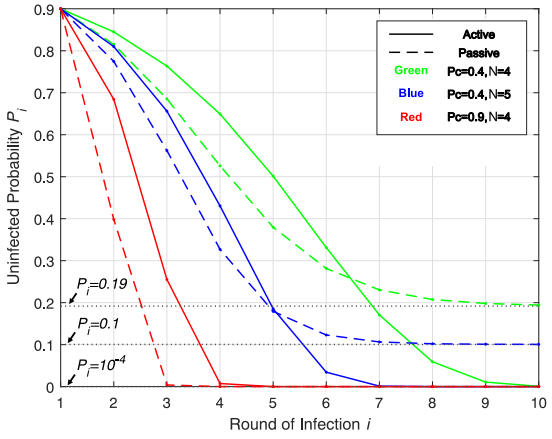

Fig. 5. Uninfected probability under active and passive gossip with N=4, 5 and P_c=0.4, 0.9.

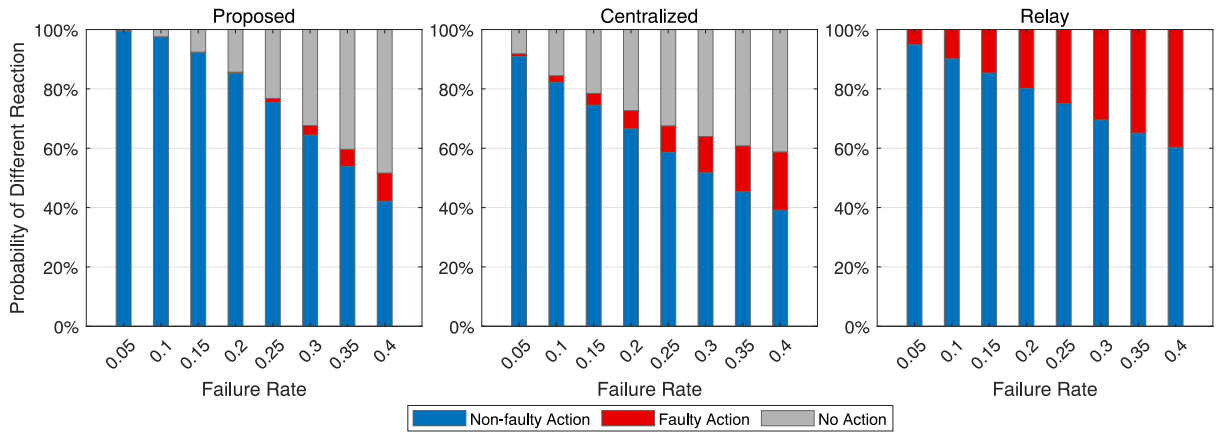

Fig. 6. Execution failure rate per vehicle against number of Faulty vehicles with $N = 7$.

When the initial value of P_i is less than 1, it approaches to zero as the infection proceeds. For comparison, we plot both uninfected probabilities under active and passive in Fig 5, for $N = 4, 5$, $P_c = 0.4, 0.9$ and $P_{i=0} = 0.9$.

Though the performance difference do exist between the passive and active schemes, this difference rapidly drops to negligible numbers as N increases (even slightly). The change in the success rate of communication P_c, also shows a significant impact. Considering the communication complexity, we chose the passive mode as the adopted scheme in this paper. We notice that even if some nodes crash or fail (Byzantine nodes), the status of these nodes are consistent with that of others over a period of time. That is to say, the Gossip algorithm itself has the natural advantage of distributed fault tolerance.

VI. Performance Analysis

To verify the feasibility of applying a PBFT-based consensus in autonomous driving and study the impact of gossip algorithm on full consensus, a simulation is conducted. It operates under a general wireless condition where each node randomly dominates the channel and broadcasts information. The simulation reproduces the complete process including client proposal generation, consensus reaching, consensus messages decoding on vehicle and proposal execution. The simulation considers a platoon of 7 vehicles achieves consensus on the platoon leader's proposal, aiming to reflect the performance enhancements brought by gossip algorithm.

The probability for each vehicle to successfully achieve consensus and execute the proposal P_{ex} is calculated based on the statistics. Results in Fig. 8 plot the failure rate $1 - P_{ex}$ with gossip and without gossip against the communication link failure rate $1 - P_c$ in a log scale. In terms of the impact of wireless communication, significant improvements can be observed for P_c greater than 0.8, which is very easy to be achieved in many communication networks. With the help of gossip, the failure probability drops to 3×10^{-5}.

The impact of byzantine vehicles is further investigated by intentionally setting part of these vehicles to be faulty, where the faulty vehicles are assumed to support the faulty option based on their incorrect observation. The V2V network consensus protocol is compared with relay and centralized protocol. For our protocol, the leader's proposal is a combination of non-faulty and faulty options as described in section IV-G, forming a single layer binary tree. For the relay scenario, the information is acquired from the immediate adjacent vehicles. The proposal decision is made by the head of the platoon and relayed through the platoon. Notice that the tolerance of faulty detection is involved in relay information flow. To evaluate the effect of fault tolerance mechanism, we designed a centralized protocol, where the central vehicle generates a similar proposal with two options. The central vehicle then selects and broadcasts one option based on its observation. Other vehicles execute the action accordingly if it is consistent with their observation, otherwise, it is rejected.

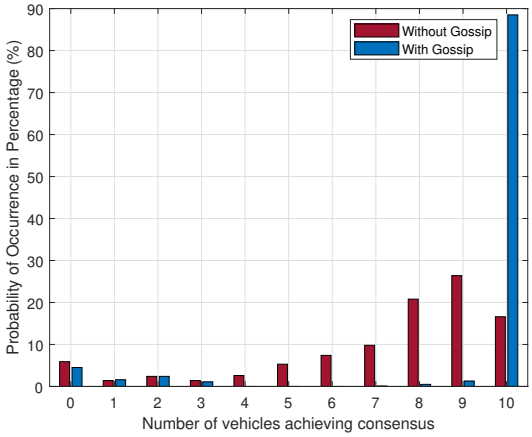

Fig. 7. Percentage of certain number of vehicles achieved consensus in each round of simulation with $N = 10$, $P_c = 0.9$.

As illustrated in Fig. 6, the failure rate indicates the probability of incorrect observation on one of 7 vehicles. The failure rate increase from 0.05 to 0.4 with 0.05 increment. The probability of executing non-faulty action is compromised by incorrect observations of the vehicle for all three protocols. For low failure Rate the non-faulty action execution probability of

proposed protocol is highest action is our proposed protocol (75.41%, 75.01% and 58.62% for proposed, relay and centralized protocol respectively at failure rate equals to 0.25). It decreases more rapidly for proposed and centralized protocol compared with relay protocol for higher failure rate. The faulty action may be executed as the percentage of faulty vehicles increases. However, with the fault tolerance mechanism, this potential risk is partially mitigated by allowing vehicles to take no action by rejecting execution in both centralized and our protocol. For instance, the vehicles could be programmed with with default protective behaviour. So that no action does not necessarily lead to accident. Our protocol further reduce the probability of faulty actions by half compared with the centralized protocol (9.50% and 19.65% for proposed and centralized protocol respectively at failure rate equals to 0.4). In contrast, the probability of faulty action for relay protocol is significantly higher.

We also notice that the gossip algorithm largely improves the consistency across the vehicles to the full consensus status. In Fig. 7, we present the statistics of a cluster of 10 vehicles that achieves the consensus with $P_c = 0.9$. The probability of occurrence indicates the likelihood that a such number of vehicles achieve consensus in one round of the consensus reaching process. For instance, the bars of 0 vehicles indicate the probability that all vehicles failed to reach consensus, while the bars of 10 vehicles indicate the probability that the whole cluster of 10 vehicles reach consensus. The results show that the gossip promotes the achievement of full consensus (all 10 vehicles achieving consensus in this case). As shown in Fig. 7, when gossip is considered, it is highly likely (88.5%) that the all vehicles reach consensus, while without gossip, only 16.6% of vehicles are capable of achieving full consensus.

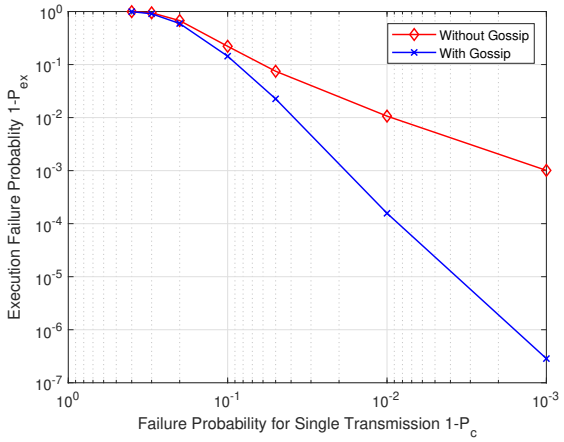

Fig. 8. Execution failure rate per vehicle against success rate of communication with $N = 7$.

VII. Conclusion

In this paper, we provide a proposal-based decision-making solution dedicated to wireless V2V networks for vehicular platoon control. A novel PBFT-based consensus protocol for the joint decision making of vehicles is proposed. The simulation results reveals that the proposed protocol is able to reach a decision in the presence of faulty vehicles and suppress the execution of faulty actions. Besides, the communication reliability is significantly improved through the gossip algorithm which facilitates full consensus reaching. Compare the consensus performance without gossip and with gossip, the result shows the failure rate is reduced from around 10^{-3} to less than 10^{-6}, with communication reliability of 10^{-3}. The single-layer binary tree scenario experiment also suggests that The protocol is capable to select optimal solutions from multiple possible candidates through plan tree synthesis, as such binary is the building block for any complex plan tree.

References

[1] D. Yu, W. Li, H. Xu, and L. Zhang, "Low reliable and low latency communications for mission critical distributed industrial internet of things," *IEEE Communications Letter*, vol. 25, no. 1, pp. 313–317, 2021.
[2] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1671–1678.
[3] S. Gibbs, "Google sibling waymo launches fully autonomous ride-hailing service," *The Guardian*, 2017.
[4] H. Somerville, P. Lienert, and A. Sage, "Uber's use of fewer safety sensors prompts questions after arizona crash," *Reuters*, 2018.
[5] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[6] Baidu, "Apolloauto," https://github.com/ApolloAuto/apollo, 2021.
[7] Tesla, "Full self-driving capability," https://www.tesla.com/autopilot, 2021.
[8] Y. Danny and T. Dan, "Tesla driver dies in fatal crash while using autopilot mode," *The Guardian*, 2016.
[9] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, and L. Zhao, "Vehicle-to-everything (v2x) services supported by lte-based systems and 5g," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 70–76, 2017.
[10] M. Torrent-Moreno, J. Mittag, P. Santi, and H. Hartenstein, "Vehicle-to-vehicle communication: Fair transmit power control for safety-critical information," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 7, pp. 3684–3703, 2009.
[11] X. Cheng, C. Chen, W. Zhang, and Y. Yang, "5g-enabled cooperative intelligent vehicular (5genciv) framework: When benz meets marconi," *IEEE Intelligent Systems*, vol. 32, no. 3, pp. 53–59, 2017.
[12] J. Joy and M. Gerla, "Internet of vehicles and autonomous connected car - privacy and security issues," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–9.
[13] L. Yang and H. Li, "Vehicle-to-vehicle communication based on a peer-to-peer network with graph theory and consensus algorithm," *IET Intelligent Transport Systems*, vol. 13, no. 2, pp. 280–285, 2019.
[14] L. Lamport, "The part-time parliament," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 277–317.
[15] D. Dolev, "The byzantine generals strike again," *Journal of Algorithms*, vol. 3, no. 1, pp. 14–30, 1982.
[16] C. Shen and F. Pena-Mora, "Blockchain for cities—a systematic literature review," *IEEE Access*, vol. 6, pp. 76 787–76 819, 2018.
[17] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 355–362.
[18] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.
[19] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.
[20] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *Self-published Paper*, vol. 19, no. 1, 2012.

[21] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[22] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys (CSUR)*, vol. 22, no. 4, pp. 299–319, 1990.

[23] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 179–196.

[24] M. Castro and B. Liskov, "Practical byzantine fault tolerance," *OSDI 1999: Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 99, pp. 173–186, 1999.

[25] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2018, pp. 1545–1550.

[26] Z. Wang, G. Wu, and M. J. Barth, "A review on cooperative adaptive cruise control (cacc) systems: Architectures, controls, and applications," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2884–2891.

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

[28] D. Helbing, "Traffic and related self-driven many-particle systems," *Reviews of Modern Physics*, vol. 73, no. 4, p. 1067, 2001.

[29] W. Ren, "Consensus based formation control strategies for multi-vehicle systems," in *2006 American Control Conference*, 2006, p. 6.

[30] T.-S. Dao, C. M. Clark, and J. P. Huissoon, "Distributed platoon assignment and lane selection for traffic flow optimization," in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 739–744.

[31] P. Fernandes and U. Nunes, "Platooning with ivc-enabled autonomous vehicles: Strategies to mitigate communication delays, improve safety and traffic flow," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 91–106, 2012.

[32] D. Jia and D. Ngoduy, "Enhanced cooperative car-following traffic model with the combination of v2v and v2i communication," *Transportation Research Part B: Methodological*, vol. 90, pp. 172–191, 2016.

[33] Y. Li, C. Tang, K. Li, X. He, S. Peeta, and Y. Wang, "Consensus-based cooperative control for multi-platoon under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2220–2229, 2018.

[34] M. Porfiri, D. G. Roberson, and D. J. Stilwell, "Tracking and formation control of multiple autonomous agents: A two-level consensus approach," *Automatica*, vol. 43, no. 8, pp. 1318–1328, 2007.

[35] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2906–2920, 2019.

[36] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, 2018.

[37] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2018.

[38] H. Liu, Y. Zhang, and T. Yang, "Blockchain-enabled security in electric vehicles cloud and edge computing," *IEEE Network*, vol. 32, no. 3, pp. 78–83, 2018.

[39] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[40] W. Hu, Y. Hu, W. Yao, and H. Li, "A blockchain-based byzantine consensus algorithm for information authentication of the internet of vehicles," *IEEE Access*, vol. 7, pp. 139 703–139 711, 2019.

[41] H. Liu, C.-W. Lin, E. Kang, S. Shiraishi, and D. M. Blough, "A byzantine-tolerant distributed consensus algorithm for connected vehicles using proof-of-eligibility," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019, pp. 225–234.

[42] M. Wegner, W. Xu, R. Kapitza, and L. Wolf, "Byzantine consensus in vehicle platooningvia inter-vehicle communication," in *Fachgespräch Inter-Vehicle Communication 2016*, 2016, pp. 20–23.

[43] D. Stolz and R. Wattenhofer, "Byzantine agreement with median validity," in *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, vol. 46, 2016, p. 22.

[44] C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography," *Journal of Cryptology*, vol. 18, no. 3, pp. 219–246, 2005.

[45] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer pbft consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2021.

[46] A. Hajami, K. Oudidi, and M. ElKoutbi, "An enhanced algorithm for manet clustering based on multi hops and network density," in *2010 10th Annual International Conference on New Technologies of Distributed Systems (NOTERE)*, 2010, pp. 181–188.

[47] A. Benslimane, T. Taleb, and R. Sivaraj, "Dynamic clustering-based adaptive mobile gateway management in integrated vanet—3g heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 559–570, 2011.

[48] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, 1987, pp. 1–12.

[49] R. M. Corless, G. H. Gonnet, D. E. Hare, D. J. Jeffrey, and D. E. Knuth, "On the lambertw function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329–359, 1996.