CUBA: Chained Unanimous Byzantine Agreement for Decentralized Platoon Management

Emanuel Regnath
emanuel.regnath@tum.de
Technical University of Munich, Germany

Sebastian Steinhorst
sebastian.steinhorst@tum.de
Technical University of Munich, Germany

***Abstract*—Autonomous driving, vehicle platoons and smart traffic management will dramatically improve our transportation systems. In contrast to centralized approaches, which do not scale efficiently with the actual traffic load, a decentralized traffic management based on distributed consensus could provide a robust, fair and well-scaling solution for infrastructures of variable density.**

In this paper, we propose a distributed platoon management scheme, where platoon operations such as *join* or *merge* are decided by consensus over a Vehicular ad hoc network (VANET).

Since conventional consensus protocols are not suitable for Cyber-Physical Systems (CPS) such as platoons, we introduce *CUBA*, a new validated and verifiable consensus protocol especially tailored to platoons, which considers their special communication topology.

We demonstrate that CUBA only introduces a small communication overhead compared to the centralized, Leader-based approach and significantly outperforms related distributed approaches.

***Index Terms*—V2V, VANET, Internet of Vehicles, Consensus**

I. Introduction

Platoons are energy efficient, communication efficient and increase the road throughput by allowing the vehicles to drive with a reduced inter-vehicle distance [1].

Most conventional platoons declare a leader that is responsible for managing the platoon behavior and communication. However, participating vehicles need to trust the leader, which offers many attack vectors on the security, safety, and performance of the platoon itself as well as third parties that might query information about the platoon from the leader.

We therefore aim for a consensus-based platoon management where decisions must be approved by all vehicles as shown in Figure 1. This distributed management would increase the robustness of the platoon by reducing the chance that failures are not detected. Since all vehicles are involved, each vehicle will observe and approve the messages of other vehicles. While this increased communication overhead is often seen as a drawback compared to a centralized management, we argue that the number of vehicles in one platoon is small enough to keep the overhead low.

The consensus can also be used to create a tamper proof specification of the platoon that can be communicated to third parties, such as an Intersection Manager (IM). The IM would verify that the platoon specification is correct by consensus and could then safely assign a slot for the entire platoon at once. This would not only improve the intersection throughput but also reduce the communication overhead and scheduling complexity for the IM.

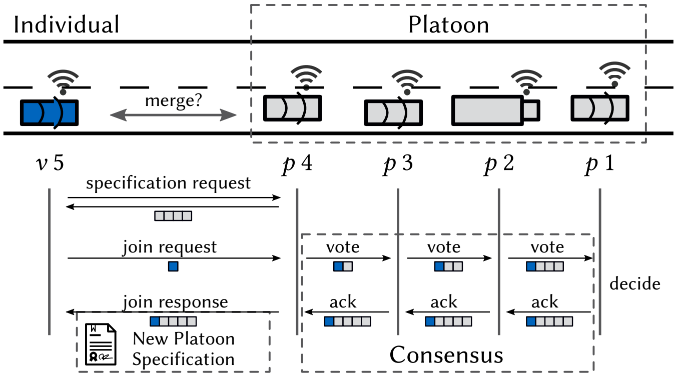

Figure 1: Our idea of a consensus-based join maneuver. An individual vehicle requests to join an existing platoon. The join request is forwarded to each platoon vehicle, which will vote on the request. If all vehicles agree, the new vehicle is accepted and the new specification is sent to all vehicles.

However, the linear spatial structure of a platoon and its dynamic changes via merge and split operations introduce challenging constraints to the topology of the traditional consensus problem. In this paper we analyze the possible topology in a static and dynamic way to create a formalization about the consensus guarantees within a platoon. Furthermore, vehicles in a platoon can not only communicate their own states (conventional consensus) but also observe and measure certain states of neighboring vehicles, such as velocity, and could contribute these information to a consensus protocol.

Even if consensus is reached about a certain operation, any vehicle within the platoon can prevent the actual execution of that operation simply by its physical presence. For example, if the platoon decides to accelerate, any single vehicle not performing the acceleration blocks the entire operation. Therefore, we aim to detect faults rather than be able to tolerate them. Furthermore, misbehaving vehicles should be identified in order to penalize them in a reputation system.

Overall, platoons are highly relevant CPS but the conventional leader-based management poses severe risks on the security and safety of all platoon vehicles because it inherits a single point of failure by design. We therefore propose a distributed, consensus-based management of the platoon, analyze the challenges that arise for this specific application, and provide a first solution to demonstrate the feasibility.

A. Scope and Contributions

This paper addresses consensus mechanisms using vehicle-to-vehicle (V2V) communication over a vehicular ad-hoc network (VANET) for platoon management decisions. We do nei-

With the support of the Technische Universität München – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763.

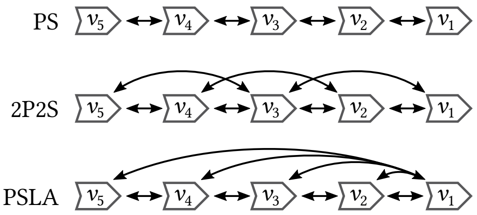

Figure 2: Considered communication topologies of neighboring vehicles. An arrow indicates that a vehicle can directly send messages using a wireless V2V VANET.

ther address the motion control of vehicles within the platoon nor the performance of the VANET itself. We consider three different V2V communication topologies and assume a synchronized message delivery with timeouts.

We propose a distributed platoon management scheme that distributes the validation of platoon properties across all platoon members and thus removes the leader as a single-point of failure. In particular, we

- analyze under which conditions the consensus problem can be solved for Cyber-Physical Systems (CPS), such as platoons (Section III).
- propose *CUBA*, a consensus scheme specifically tailored to platoons that communicates over VANET. (Section IV).
- discuss related approaches (Section V) and compare our CUBA protocol illustrating its feasibility (Section VI).

II. Scenario, Model, and Assumptions

For describing and evaluating the platooning scenario, we consider a set of vehicles $\mathcal{V} = \{v_1, v_2, ...\}$ on a highway and some of them form a platoon $\mathcal{P} = \{p_1, p_2, ...\} \subseteq \mathcal{V}$. Vehicles v_x can join a platoon $\mathcal{P}$, two platoons $\mathcal{P}_1$ and $\mathcal{P}_2$ can merge to $\mathcal{P}_3 = P_1 \cup P_2$ and a vehicle v_x can leave a platoon $\mathcal{P}' = \mathcal{P} \setminus \{v_x\}$. As pointed out by [2], we assume a maximum platoon size of $|\mathcal{P}| = N = 20$. Each vehicle has specific properties, such as color, length, or maximum velocity.

1) Interaction Topologies Due to the linear structure of a platoon, the communication and sensing capabilities of each vehicle can be described in general by the range in forward and backward direction within the platoon.

Following the notation of [3], we consider three communication topologies (see Figure 2) that are suitable for consensus based platoon management:

- Predecessor – Successor (PS)
- Two-Predecessor – Two-Successor (2P2S)
- Predecessor – Successor – Leader to All (PSLA)

Despite communication, we also need to consider the sensing topology of each vehicle. Sensing is important to verify the sent claims of vehicles, such as its physical presence and its identity. Since there are numerous vehicle properties that could be sensed in different ways, it is very difficult to model all sensing capabilities in a unified topology graph. We leave the general specification of heterogeneous sensing topologies as an open problem for future work. For simplicity, we only consider sensing the license plate of a vehicle and that each vehicle is able to read the license plate of its predecessor and successor in order to verify the authenticity of a vehicle. This sensing topology would correspond to the Predecessor-Successor (PS) communication topology.

We furthermore assume that it is possible to verify a vehicle specification according to a license plate using a trusted third-party certification service or distributed certification (e.g. using Blockchain) [4].

III. Consensus within Platoons

In this section we will analyze the differences between the conventional consensus problem of state replication in databases and the consensus problem to achieve self organization in platoons as a Cyber-Physical System (CPS). We will especially address the problems of heterogeneous voting and zero fault tolerance.

A. Conventional Consensus

In a consensus protocol, each agent needs to decide which state transitions are applied and in which order, which is equivalent to providing a reliable totally ordered broadcast. The consensus problem is considered solved if three properties hold:

- Agreement: All correct vehicles decide the same value.
- Integrity: All correct vehicles decide only once.
- Termination: All correct vehicles decide before timeout.

Agreement and Integrity are safety properties, Termination is a liveness property. A vehicle is *correct* if it follows the consensus protocol.

For useful applications, the decision should also be correct with respect to some policy or desired behavior. Derived from the notion of [5], we require two additional properties:

- Validity: The decision of a correct *Acceptor* was validated by a correct *Validator*. The terms Acceptor and Validator will be defined in Subsection III-C.
- Provability: The validity of a decision can be verified by any vehicle.

B. Assumptions

Solving consensus in self-organizing systems is difficult, especially if vehicles could stop communicating or could even send malicious messages. In order to address these challenges, many systems (including ours) make the following assumptions:

- Partial synchronous: Message delays are in general unknown and unbounded but every message will eventually be delivered before a fixed deadline.
- Known participants: Each participant knows every other participant that is allowed to vote.
- Signatures: Each participant can verify the sender of a message using signatures. As shown by [6] signatures prevent clients from forging messages and can be used to identify malicious nodes.

C. Heterogeneous Agreement

In contrast to conventional consensus systems where each node can validate and vote on all possible requests, not all vehicles within a platoon might be able to validate the same set of requested transitions. For example, a vehicle approaching the tail of a platoon can probably only be sensed by the last vehicle of the platoon.

We therefore propose a more detailed role assignment. In our consensus system there are seven distinct roles:

- Requester: vehicle that requests an operation
- Receiver: processes requests from a *Requester*
- Responder: responds to *Requester* when consensus is reached
- Proposer: proposes a new system state
- Validator: can validate a proposed state
- Acceptor: can vote for a proposed state
- Learner: will receive accepted state

D. Zero Fault Tolerance

Conventional consensus algorithms, which are mostly applied to databases, can tolerate a certain amount of f failures as long as a sufficient majority $(2f+1)$ operates correctly. Since, the correct state of such a system is determined by majority, each agent can determine the current state by receiving a majority of correct votes. The remaining votes of unresponsive, outdated, or malicious agents can simply be ignored. As a result, the final decisions of faulty agents have no influence on the decisions of correct agents, allowing correct agents to maintain a consistent system state.

However, for CPSs these assumptions do not apply and solving consensus is fundamentally different from conventional consensus. Platoons, for example, are safety critical CPSs and the safety of a single passenger weights more than the decision of any majority. As a result, we cannot tolerate a single negative vote or failure.

In our consensus scheme, we therefore require *all* participants to agree on the same state transition. Note that the goal of our scheme is not to tolerate failures but to reliably detect if an unanimous decision was reached by consensus, or in case consensus could not be reached, which vehicles are responsible for the failing decision.

These responsible vehicles could have failed to transmit their vote, voted against the decision, or tried to send a malicious message.

E. Promise vs. Reality

Considering a platoon in which vehicles want to perform a merge operation, then reaching consensus is only a virtual agreement of the vehicles to perform certain actions. Besides changing some virtual data structures, these actions could also involve a physical process that can be blocked, slowed down, or changed by external constraints.

Many physical processes within the platoon require each vehicle to participate and work correctly. Even in the case consensus could be reached, it might not be possible to execute the overall operation safely. In the moment of execution, a vehicle that promised to perform a certain action might be forced to postpone or stop the action, or even perform a completely different action to ensure the safety of its passengers. Despite external influences that are beyond the consensus decision, any vehicle could also be subject to a critical failure or even maliciously block operations within the platoon by its mere physical presence.

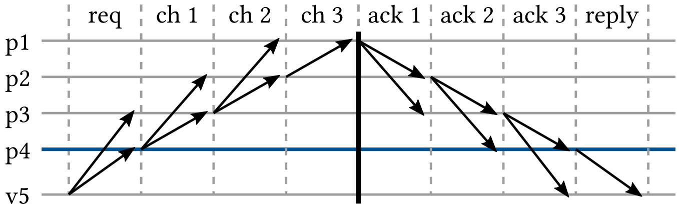

Figure 3: Join request, normal operation, 2P2S: Vehicle v_5 sends the request to the platoon vehicles p_4 and p_3. p_4 will start a consensus round and when p_1 receives a valid chain of accepting votes, it decides for the new platoon and sends an ACK back to v_5.

Therefore, we try to reliably detect any failures and then separate sub-platoons from the failed vehicle.

Since failed or malicious vehicles pose a severe risk to other vehicles, another important goal is to distinguish these vehicles from vehicles that were forced to deviate from the consensus but otherwise operate correctly and honestly.

The matter gets more complicated as we also need to consider that malicious vehicles may send false messages but not all vehicles might be able to sense the true nature of the deviation.

In our platoon scenario we therefore distinguish two types of consensus rounds:

1) consensus about a planned action, which requires all vehicles to agree.
2) consensus whether and which vehicle failed, which requires $f+1$ vehicles to agree.

We assume that only f failures can occur at maximum and that all other vehicles participate honestly.

IV. Our CUBA protocol

We now introduce our Chained Unanimous Byzantine Agreement (CUBA) protocol, which is suitable for platoons and other distributed CPS that want to reliably detect failures because they cannot afford to tolerate (ignore) any single failure. CUBA works by passing messages hop-by-hop instead of using broadcasts. Each hop confirms the messages sent by previous hops. This concept is similar to BChain [7], a general consensus protocol, which, however, is *not* feasible for platoons (For discussion, see the related work Section V).

In contrast to BChain, our protocol is designed to terminate successfully only if all involved acceptors agree on the same value. In case no consensus can be reached, our protocol will detect which acceptor was responsible for the failing consensus, such that the correct nodes can take action. The maximum number of failures f that can reliably be detected depends on the underlying network topology.

A. Role Assignment

We apply the following roles:

- Any vehicle v_x (including platoon vehicles) is a Requester.
- Any platoon vehicle p_x is Acceptor, and Learner.
- The platoon vehicles at the top p_1 and tail p_N are Receivers, Responders, and Proposers.
- The direct neighbors of a vehicle v_x are Validators for that vehicle as they can physically sense v_x and read its license plate.

B. Normal Operation

In normal operation, all vehicles operate correctly and need to agree on the same proposal. The proposal could be any planned platoon operation. We use three message types: $\langle \textsc{Ch} \rangle$, $\langle \textsc{Ack} \rangle$, and $\langle \textsc{Nak} \rangle$.

Each message is structured as $\left\langle T, s, (h), l_o, (l_n), (m), \sigma \right\rangle$ where fields in parentheses are optional. T is the type of the message (chain, ACK, NAK), s is an integer indicating the sequence number of the current consensus round, h is the cryptographic hash of the previous message, l_o is the own license plate number of the sender, l_n is the license plate number of the next (succeeding) vehicle if present, m is the proposed message or state to vote upon, and σ the signature of the sending vehicle.

The protocol execution is illustrated in Figure 3.

1) A Proposer proposes a new state or planned operation and forwards the proposal in form of a $\langle \textsc{Ch} \rangle$ towards the other end of the platoon.
2) Each intermediate vehicle validates the proposal, and votes for it by appending its own $\langle \textsc{Ch} \rangle$ message to the proposed $\langle \textsc{Ch} \rangle$. If an intermediate vehicle receives several chained $\langle \textsc{Ch} \rangle$ messages, it will first validate each vote by verifying four predicates:
   a) the sequence number in each message matches the current sequence number,
   b) h of the current message matches the hash of the previous $\langle \textsc{Ch} \rangle$,
   c) l_o of the message matches l_n of the previous message,
   d) the signature σ is valid using the public key corresponding to l_o.
3) Once the last Acceptor (other Proposer) received the proposal and all votes, it decides. If all votes agree on the proposal, it decides for the proposal and sends an $\langle \textsc{Ack} \rangle$ including all signatures back to the other Acceptors (now in the role of Learners). If there is one vote against the proposal, it decides against the proposal and sends an $\langle \textsc{Nak} \rangle$ together with *all* signatures.
4) Each intermediate vehicle (now Learner) receives and validates the signatures of the $\langle \textsc{Ack} \rangle$ or $\langle \textsc{Nak} \rangle$ and decides accordingly until the last Learner (=Proposer) is reached.

To ensure that all vehicles will receive the $\langle \textsc{Ack} \rangle$ of a successful consensus round we make the critical assumptions that each vehicle can send messages to the next $f + 1$ vehicles in one direction. Otherwise we could not guarantee that decisions for a proposal are propagated to all vehicles. This assumption limits the possible topologies to 2P2S.

C. Example: Platoon Formation

Platoon formation can happen between two single vehicles, between a single vehicle and an existing platoon or between two existing platoons. The Proposers of a platoon will also react to join requests from vehicles outside of the platoon. In the following, we consider a single vehicle v_5 that wants to join an existing platoon $\mathcal{P} = \left\{p_1, p_2, p_3, p_4\right\}$ as illustrated in Figure 1. The single vehicle v_5 approaches the tail vehicle p_4 of the platoon until it can read its license plate and then runs the following protocol:

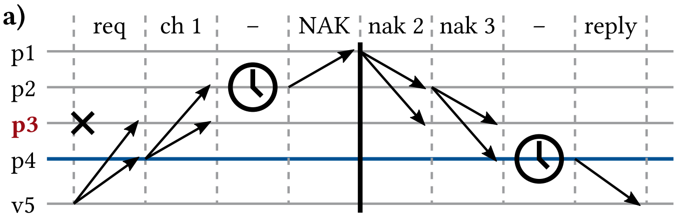

Figure 4: Join request, p_3 failure/timeout: When p_4 starts the consensus round, p_2 receives the message and starts the timeout timer. Since p_3 is unresponsive, p_2 will wait until the timeout and then forward a $\langle \textsc{Nak} \rangle$.

1) The Requester v_5 requests and receives the current platoon specification $\langle \textsc{Spec} \rangle$ from p_4.
2) The Requester v_5 verifies $\langle \textsc{Spec} \rangle$ and decides whether it wants to join.
3) The Requester v_5 sends a join request to the tail p_4. The request contains information about the vehicle v_5.
4) The Proposer p_4 starts a new consensus round by forwarding a $\langle \textsc{Ch} \rangle$ message including the join request.
5) When the consensus round was successful, the Proposer p_4 replies with the new platoon specification $\langle \textsc{Spec} \rangle$ including all $\langle \textsc{Ch} \rangle$.
6) The Requester v_5 verifies $\langle \textsc{Spec} \rangle$ and if valid, accepts it and becomes p_5. The next consensus decision now requires votes from 5 platoon vehicles.

a) Initial Formation The initial platoon formation between two individual vehicles v_1 and v_2 works similar to the described join maneuver. The difference is that the Requester v_2 will receive a $\langle \textsc{Spec} \rangle$ that includes only v_1. If v_2 sends the join request, v_1 can immediately decide as it represents a platoon with only one vehicle. Once v_1 appends a $\langle \textsc{Ch} \rangle$ with its signature to the request and sends the chain back to v_1, a new platoon $\mathcal{P} = \left\{p_1, p_2\right\}$ is formed.

D. Failed Consensus

Each vehicle sets a timer with the time period that corresponds to the expected consensus execution time for the remaining set of vehicles. This time is calculated as

$$t_{\mathrm{TO}} = (N - i) \cdot \tau \tag{1}$$

where $(N - i)$ represents the number of remaining vehicles that need to vote and τ is a fixed and pre-defined timeout value for every vehicle. In case, the successor vehicle can provide a consensus proof (valid $\langle \textsc{Ack} \rangle$) before the timer runs out, the vehicle decides for the consensus value and forwards $\langle \textsc{Ack} \rangle$. In all other cases, the vehicle decides that the consensus failed and forwards a $\langle \textsc{Nak} \rangle$ in which case it *may* also include the ID of another vehicle it suspects to deviate from the protocol or to have timed out.

An example for a timeout of p_4 is shown in Figure 4. p_2 will wait until p_3 times out and then forwards a $\langle \textsc{Nak} \rangle$. When the $\langle \textsc{Nak} \rangle$ returns from p_1, p_2 will forward it to p_3, giving it a second chance to respond. Forwarding the messages to all vehicles despite an early $\langle \textsc{Nak} \rangle$ is important for determining the failed vehicle. p_4 will wait until its timeout timer runs out and decides that the consensus failed.

Under the assumption that it is not possible to forge messages, consensus is only reached if and only if *all* vehicles

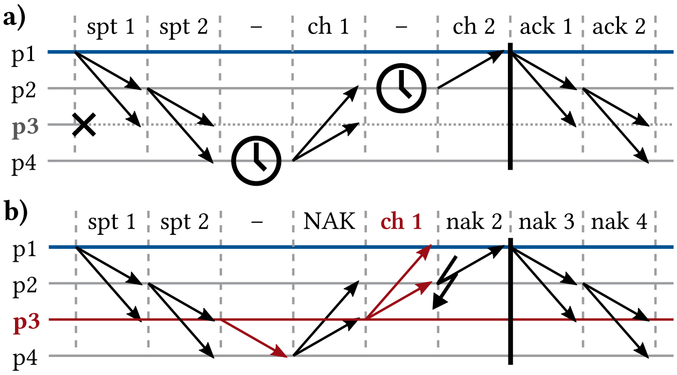

Figure 5: Suspect round to determine which vehicle failed proposed by p_1. a) p_3 was suspected by p_4 and really timed out. b) p_3 is malicious and suspected p_4, but p_4 is still running.

respond correctly and in time $< \tau$. Any non-correct response will result in a non-valid ⟨Ack⟩ or a timeout. Therefore each vehicle is able to reliably detect a failed consensus.

E. Detection of Failing Vehicle

However, detection of the vehicle that is responsible for the failure is more complicated. For example, a malicious vehicle could falsely accuse a neighboring vehicle to have timed out.

After a failed consensus round, the last acceptor checks if the ⟨Nak⟩ includes a suspected vehicle and – when true – starts a suspect round. This special consensus round requires only $f+1$ chain votes from neighbors in communication range of the suspected vehicle to be successful because with a maximum of f faults at least one vote will be correct. Furthermore, before chaining votes, ⟨Spt⟩ messages are forwarded to all vehicles. This serves two purposes: 1. each vehicle knows which vehicle is suspected, 2. each neighbor of the suspected vehicle has the chance to observe a timeout or conflicting messages of the suspected vehicle before deciding against it.

An example of a suspect round for a timeout of p_3 is illustrated in Figure 5a which would be triggered by p_1 after the timeout round in Figure 4. In the normal round, p_2 would suspect p_3 for the timeout and after the consensus failed, p_1 would start a suspect round suspecting p_3 by forwarding a ⟨Spt⟩ message. p_4 would witness another timeout by p_3 and voting with a ⟨Ch⟩ against it. Once the ⟨Ch⟩ reaches p_2 and p_3 does not respond in time, p_2 will also vote against p_3, reaching $f+1$ votes. Thus, p_1 will decide that p_3 timed out and forwards the decision with the signatures of the $f+1$ votes in an ⟨Ack⟩ message. Once the failed vehicle is identified, the platoon is split to ensure the safety of the remaining vehicles.

For the network topology, we need $f+1$ communication hops in both directions to reliable identify f failing vehicles, which means a 2P2S topology for $f = 1$.

V. Related Work

A. BFT-ARM Platooning

The authors of [8] designed BFT-ARM, a consensus protocol for continuous sensor values in an asynchronous inter-vehicle network. The protocol uses median validity, where the decided value is only required to be close to the median of all correctly proposed values and claims to tolerate up to $f < \frac{N}{3}$ Byzantine nodes. This is achieved by calculating the median only over the sorted $(2f+1)$ middle values in the full range of all $(3f+1)$ proposed values, cutting off $f/2$ values at each end of the range.

Protocol	Messages per Round
PBFT	$2N^2 - 2N$
BFT-ARM	$3N^2 - N - 2$
BChain	$2Nf + N - 4f^2 - 1$
CUBA	$2Nf + 2N - f^2 - 3f - 2$

Table I: Number of messages per consensus round depending on the number of agents N and the maximum number of possible failures f.

The protocol is focused on *tolerating* some faulty measurements in order to agree on a common value and is not designed for safety critical decisions that require the agreement of all involved vehicles. Furthermore, the protocol relies on a trusted subsystem which provides unforgeable counter values. Overall, the application focus of BFT-ARM is different from our protocol and thus not suitable for platoon management.

B. BChain

BChain [7], is a general consensus protocol that does not work for platoons in its original form but shares some concepts with our CUBA. Nodes are ordered within a chain, which is divided into two parts: The first $2f+1$ nodes within the chain are acceptors and the last f nodes are learners. A request is forwarded hop-by-hop from the head (1st node) towards the acceptor tail (node $2f+1$) using ⟨Ch⟩ messages. Once the acceptor tail receives and accepts a ⟨Ch⟩, it will send 3 messages: 1) a reply to the client, 2) an ⟨Ack⟩ message that traverses backwards to the head, and 3) its ⟨Ch⟩ to the learners.

In order to handle failures, each node starts a timer after sending its ⟨Ch⟩ and in case a timer expires before receiving an ⟨Ack⟩, the node will issue a ⟨Spt⟩ to the head. The head is then responsible for re-ordering the chain, such that malicious or crashed nodes are moved towards the end.

While such a reordering is possible in a consensus overlay network where the underlying network topology allows several routes between nodes, it is not feasible in platoons where the routes and connections depend on the spacial location of nodes. It would require changing the position of vehicles by overtaking maneuvers. Furthermore, requests must be always sent to the head, which increases the risk of blocking or slowing down the protocol execution if the head is faulty. Compared to our protocol, BChain can not guarantee to terminate in the first round but would require $3q$ rounds in the worst case, where q is the number of faulty nodes [7].

C. Consensus from Control Theory

In control theory, algorithms such as the Average Consensus are used to solve a distributed control problem. While this family of algorithms is well-studied and suitable for controlling, e.g. the distance between platoon vehicles [9], it does not consider faulty or malicious agents but rather assumes that every agent is running and responding within a certain time period.

VI. Analysis

In this section we discuss the advantages and drawbacks of CUBA and provide an analytical evaluation of its performance.

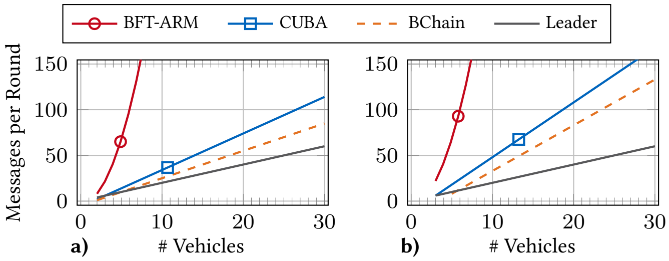

Figure 6: Required number of messages per round for different platoon sizes. a) assuming maximum possible failures $f = 1$ b) assuming maximum possible failures $f = 2$.

A. Consensus Safety

CUBA offers a safe consensus based platoon management. In contrast to majority-based consensus protocols, CUBA requires all vehicles to agree on the same decision and this decision is verified by all vehicles. Therefore, failures can be reliably detected, which prevents any unintended interactions between the vehicles. The sequential execution of all messages in an ordered chain also helps to avoid collision and retransmission of messages because only one vehicle is sending. Furthermore, the fixed timeouts guarantee a deterministic consensus execution within a bounded period of time, where BChain would perform expensive re-chaining in the case of failure.

In contrast to BFT-ARM, our protocol does not require *view changes*, where the Proposer is changed when it is suspected, as any identified failure will lead to a splitting of the platoon. Therefore, we achieve a simpler protocol that only utilizes two types of consensus rounds. Due to the use of signatures, it is not possible to produce wrong decisions as long as one vehicle is correct. Reliable detection of failures is possible as long as the vehicles can reach $f + 1$ neighboring vehicles in each direction.

B. Verifiable Platoon Specification

Using signatures during consensus also enables us to generate a platoon specification that was signed by all involved vehicles. Vehicles outside of the platoon can then query the specification and verify the signatures to ensure that the specification is correct and corresponds to the agreement.

C. Communication Overhead

While conventional consensus protocols often measure the throughput of processed requests for state machine replication, we are interested in the number of messages that need to be sent to run one consensus round because the bandwidth is limited for VANETs. We therefore derived the equations in Table I for the number of messages from the protocol description in the corresponding papers. Note that the number of messages does not change with the number of actual failures q but only with the maximum number f. The reliability of a consensus-based approach introduces some overhead to the leader-based approach, for which we assume $2N$ messages (send message to each vehicle and receive ACK). Figure 6 illustrates that all decentralized approaches require more messages than the leader-based communication but in contrast to BFT-ARM, CUBA also scales linear to the number of platoon vehicles. Furthermore, the messages are more equally distributed among the vehicles in CUBA compared to the leader-based approach, where the leader needs to process all messages. Note that BChain is not compared but only shown for completeness because this protocol cannot be applied to platoons.

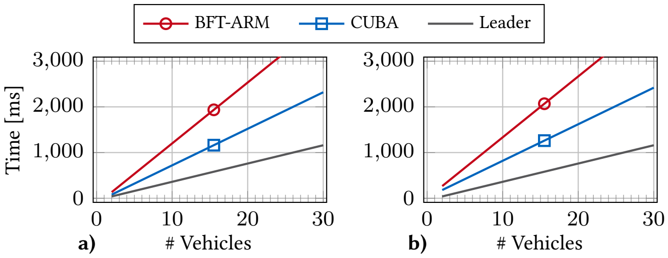

Figure 7: Consensus time per round. a) in normal operation without failures ($f = 1, q = 0$). b) with one failure ($f = q = 1$).

Another important metric is the overall time required to reach consensus, which is illustrated in Figure 7. Here, we assume $40\,\mathrm{ms}$ latency between two vehicles for transmission and processing and $100\,\mathrm{ms}$ as the timeout timer value. For large platoons this results in consensus times around $2\,\mathrm{s}$ but CUBA always terminates faster than BFT-ARM due to the reduced rounds required to reach consensus.

VII. Conclusion

We have illustrated the benefits of distributed, consensus based platoon management over conventional centralized and leader-based platoons and presented CUBA, a new consensus protocol for platoon management, which addresses the challenges of consensus in Cyber-Physical Systems. CUBA focuses on failure detection and failing vehicle identification and guarantees to terminate in a fixed time window. For typical platoon sizes up to 20 vehicles, the communication overhead of CUBA is low compared to leader-based systems and significantly less than related consensus approaches for platoons.

References

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

[2] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, “Platoon management with cooperative adaptive cruise control enabled by VANET,” *Vehicular Communications*, vol. 2, no. 2, pp. 110–123, 4 2015.

[3] Y. Zheng, S. E. Li, J. Wang, D. Cao, and K. Li, “Stability and scalability of homogeneous vehicular platoon: Study on the influence of information flow topologies,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 14–26, 2016.

[4] S. Rowan, M. Clear, M. Gerla, M. Huggard, and C. M. Goldrick, “Securing vehicle to vehicle communications using blockchain through visible light and acoustic side-channels,” *arXiv*, vol. abs/1704.02553, 2017.

[5] J. Sousa and A. Bessani, “From Byzantine Consensus to BFT State Machine Replication: A Latency-Optimal Transformation,” in *2012 Ninth European Dependable Computing Conference*, May 2012, pp. 37–48.

[6] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, “Making byzantine fault tolerant systems tolerate byzantine faults,” in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2009, pp. 153–168.

[7] S. Duan, H. Meling, S. Peisert, and H. Zhang, “BChain: Byzantine Replication with High Throughput and Embedded Reconfiguration,” in *Principles of Distributed Systems*, 2014, pp. 91–106.

[8] M. Wegner, W. Xu, R. Kapitza, and L. Wolf, “Byzantine Consensus in Vehicle Platooning via Inter-Vehicle Communication,” Proceedings of the 4th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication (FG-IVC 2016), Humboldt University, Berlin, Germany, Tech. Rep., 3 2016.

[9] L. Y. Wang, A. Syed, G. Yin, A. Pandya, and H. Zhang, “Coordinated vehicle platoon control: Weighted and constrained consensus and communication network topologies,” in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec 2012, pp. 4057–4062.