Application of Consensus Protocols to Vehicular Communications Scenarios for the Negotiation of Cooperative Traffic Maneuvers

Miguel Tavares[1], Emanuel Vieira[1][a], João Almeida[2][b] and Paulo Bartolomeu[1][c]

[1]*Instituto de Telecomunicações - Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal*

[2]*Instituto de Telecomunicações, Universidade de Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal*

Keywords: Connected and Automated Vehicles (CAVs), Vehicle-to-Everything (V2X) Communications, Consensus Protocols, Cooperative Traffic Maneuvers, Intelligent Transportation Systems (ITS), Maneuver Coordination, Byzantine Fault Tolerance (BFT).

Abstract: The introduction of Connected and Automated Vehicles (CAVs) has changed the face of the automotive sector, enhancing further developments in cooperative mobility, public safety and improved transportation system management. This paper presents an example study of the application of consensus algorithms in Vehicle-to-Everything (V2X) environments to enable reliable communication among vehicles for the realization of cooperative traffic maneuvers. Among others, an important mechanism employed in this work is the Verifiable Event Extension (VEE), which adds the reliability feature to the V2X communications and ensures trust. In addition to assessing various network conditions in detail, this work analyzes the performance and resiliency trade-offs of different consensus protocols applied to maneuver coordination scenarios, namely Zyzzyva, Pratical Byzantine Fault Tolerance (PBFT), HotStuff, and Three-Phase Commit (3PC). The obtained results underline the feasibility of applying robust, highly scalable fault-tolerant solutions to open the way towards a safe deployment of next-generation cooperative and autonomous driving systems.

1 INTRODUCTION

The rapid development of the automotive industry in the 21st century is transforming the transportation landscape by embracing Connected and Automated Vehicles (CAVs) at its core. Beyond improving driver convenience, CAVs are redefining public safety standards, optimizing traffic management, fostering cooperative mobility, and unlocking new opportunities for economic growth and transportation system efficiency. One of the pillars of this technological revolution is Vehicle-to-Everything (V2X) communications technology, which provides seamless interaction among vehicles and with the environment to enhance intelligent decision-making and coordinated maneuvers.

CAVs are equipped with a wide range of sophisticated sensors, such as On-Board Units (OBUs), Global Navigation Satellite Systems (GNSS), Light Detection and Ranging (LiDAR), Radio Detection and Ranging (RADAR), cameras and ultrasonic sensors (Kiraz et al., 2024)(Paluszczyszyn et al., 2024), to provide real-time awareness of their surroundings. However, efficiently utilizing these data requires the creation of strong frameworks for decentralized decision-making in dynamic and uncertain scenarios. Cooperative traffic maneuvers, such as overtaking and merging, are particularly challenging; vehicles need not only to process sensor data accurately but also to agree with their counterparts to ensure the safety and reliability of their maneuvers.

As an example, Figure 1 illustrates a scenario in which the green vehicle attempts to overtake the yellow car. The arrow lines represent the future trajectories of the vehicles. Two critical questions emerge, encapsulating the core challenges of this exploration:

- **How Can the Green Vehicle Ensure the Maneuver Is Safe?**
- **How Can the Green Vehicle Trust the Decisions of Others?**

These scenarios highlight the high stakes of cooperative maneuvers, where communication failures or

[a] https://orcid.org/0000-0001-9466-4649
[b] https://orcid.org/0000-0001-6634-6213
[c] https://orcid.org/0000-0002-7471-5135

Tavares, M., Vieira, E., Almeida, J. and Bartolomeu, P.
Application of Consensus Protocols to Vehicular Communications Scenarios for the Negotiation of Cooperative Traffic Maneuvers.
DOI: 10.5220/0013484900003941
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2025)*, pages 235-246
ISBN: 978-989-758-745-0; ISSN: 2184-495X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

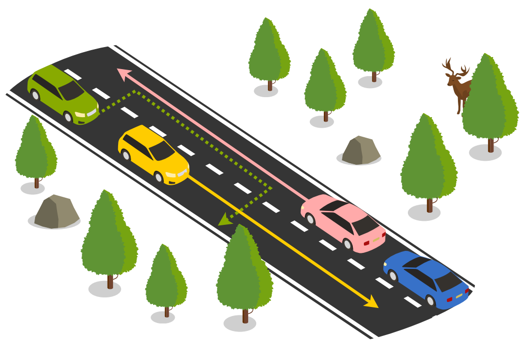

Figure 1: Scenario for vehicular consensus.

errors can cause serious hazards. Establishing trust and coordination among vehicles is essential to avoid these hazards and move towards reliable autonomous transportation systems.

To this end, this paper presents a detailed comparison of four consensus protocols employed in the negotiation of cooperative traffic maneuvers: Zyzzyva, Practical Byzantine Fault Tolerance (PBFT), HotStuff, and Three-Phase Commit (3PC). This maneuver coordination process takes place over short-range vehicular networks, relying either on IEEE 802.11p or C-V2X based communications technologies. Performance analysis of the different consensus protocols is conducted in a variety of network configurations, including node density and packet loss scenarios, thus providing deep insights into their efficiency, fault-tolerance, and practical applicability. The findings highlight the feasibility of employing scalable and fault-tolerant solutions that can empower future cooperative autonomous driving systems.

The remainder of this document is organized as follows. Section 2 presents an overview of related works, followed by Section 3, that provides the system model and the assumptions of the proposed design. Then, Section 4 presents the consensus algorithms used in this work and a detailed explanation of one of them (Zyzzyva). After that, Section 5 describes the implementation of the framework developed for the analysis of the consensus processes. Section 6 presents the evaluation of the different consensus protocols, including the setup used, the tests performed, and the results obtained. Finally, Section 7 provides the final conclusions drawn from the research findings and outlines the future work.

2 RELATED WORK

In this section, related work that contributed to the development of the concepts and methodologies employed in this study is explored. By reviewing existing literature, key advancements in the field are highlighted, positioning this work within the broader context of ongoing research.

Feng *et al.* identify key challenges in achieving consensus in autonomous vehicle networks, particularly under critical conditions (Feng et al., 2023). A notable issue with consensus algorithms like PBFT is their reliance on simple majority voting, which may fall short in scenarios requiring unanimous agreement. To address this, the authors introduce a veto-collection phase with a feasibility-proofing procedure preceding the standard consensus process. As a result, the authors propose two types of consensus:

- **Type 1:** Follows the traditional PBFT protocol, relying on majority voting to reach an agreement.
- **Type 2:** Incorporates the veto-collection phase to address scenarios where unanimity is required.

Additionally, Feng *et al.* tackle the challenge of maneuver coordination, often requiring multiple small actions. They propose a plan tree to consolidate a sequence of actions into a single proposal, reducing the frequency of consensus processes. A gossip algorithm further enhances the PBFT consensus, ensuring robust information dissemination across the network. Simulations demonstrate the efficacy of their approach, particularly in mitigating communication failures and handling faulty vehicles.

With the purpose of safeguarding cooperative maneuver information, Vieira *et al.* propose a system comprising two key protocols designed to enhance vehicle coordination and data reliability in connected environments: the Maneuver Coordination Protocol and the Maneuver Data Consensus Protocol (Vieira et al., 2023). The former focuses on negotiation and agreement among vehicles, leveraging European Telecommunications Standards Institute (ETSI) ITS standards to resolve conflicts using basic Maneuver Coordination Messages (bMCMs). While this protocol focuses on timeliness and cryptographic security, its lack of fault-tolerance poses challenges under adverse conditions. The Maneuver Data Consensus Protocol, in contrast, emphasizes fault-tolerance and traceability, employing PBFT for data agreement and blockchain for storage. Hardware-in-the-loop (HiL) simulations highlight its strengths and limitations, with packet loss significantly impacting performance. Despite this, the protocol's adaptability and scalability make it a promising solution for connected environments.

Expanding on the use of alternative consensus algorithms, the AIR-RAFT system integrates the Raft algorithm with LoRa-based wireless communication to address unreliable V2V scenarios (Li et al., 2022). The proposed solution focuses on decentralized data consistency over long distances, leveraging LoRa's capabilities. While effective in scenarios like platooning, the system faces limitations in transmission speed and latency, suggesting a need for further research into advanced communication technologies like Cellular-V2X (C-V2X).

In a more generic proposal (Vieira et al., 2024a), that is able to support different consensus protocols, *Vieira et al.* introduce the Verifiable Event Extension (VEE), a lightweight and modular applicational extension designed specifically for Intelligent Transportation System (ITS) messages. It enhances vehicular networks by seamlessly adding functionality for data security, consensus, and trading while maintaining compatibility with existing V2X communication standard protocols. This design ensures that the implementation does not disrupt or require significant modifications to the current Cooperative ITS (C-ITS) framework.

VEE is structured into three core modules that provide specific functionalities. The Ledger Module utilizes localchains, a geographically based distributed ledger technology, to ensure data immutability and traceability, making vehicular data tamper-proof and highly reliable. The **Consensus Module** handles consensus processes, providing an agreement mechanism for various applications. The PBFT algorithm is given as an example, to enable fault-tolerant consensus among network participants, particularly for non safety-critical scenarios. Lastly, the Token Module facilitates cryptocurrency-based transactions, enabling the secure and efficient exchange of digital assets or services, such as toll payments or cooperative maneuver rewards, within the vehicular network.

The VEE framework is designed with performance efficiency in mind, minimizing overhead on the communication channel by leveraging existing ITS messages as a transport medium. This approach avoids the need for additional message headers, reducing the network load. The feasibility and lightweight nature of the system have been validated through HiL setups, demonstrating its capability to function effectively under real-world vehicular constraints. VEE significantly enhances the security and accountability of cooperative maneuvers by recording and verifying vehicular event data. It also supports distributed trading mechanisms, such as instant road toll payments and reward systems for cooperative maneuvers, providing a robust platform for value-added vehicular services. The modular design allows for customization, enabling its deployment in various vehicular scenarios while addressing diverse use cases.

Finally, V2X messages may lack reliability, for instance due to sensor errors or malicious nodes, even when essential security mechanisms (e.g., standard authentication and authorization protocols) are in place (Vieira et al., 2024b). To address this issue, the authors propose the inclusion of consensus information on top of the standard V2X messages, in order to be validated using the PBFT algorithm. This way, it is possible to achieve agreement among vehicles, even in the presence of faults or malicious actors. The methodology was tested using a HiL setup that simulates real-world conditions, by introducing a configurable packet loss rate in the communications between four OBUs running the ETSI ITS-G5 protocol stack. The obtained results showed that dedicated messages were faster for consensus, while extended messages were better suited to dense traffic scenarios due to their lower wireless medium impact. Channel Busy Ratio (CBR) measurements confirmed the feasibility of both methods. The extended V2X messages approach is suitable for non-safety-critical scenarios but may require optimization for time-sensitive tasks.

The reviewed works collectively highlight significant progress in developing consensus mechanisms tailored for vehicular networks. From enhancing PBFT with veto-gathering phases and gossip algorithms to exploring alternative approaches such as Raft and integrating blockchain technologies, these efforts address critical challenges such as fault tolerance, scalability, and reliability under diverse conditions. However, certain gaps remain unaddressed.

Existing methods typically rely on majority-based consensus, which, while efficient, can fall short in scenarios that require unanimous agreement for safety-critical applications. Although the one-vote veto mechanism addresses this to some extent, its reliance on unanimity can introduce additional delays and inefficiencies, particularly under moderate/high packet loss conditions or when the network encompasses untrusted nodes. Furthermore, alternative algorithms such as 3PC and other BFT protocols have not been extensively explored in the context of vehicular networks, leaving questions about their comparative performance under varying packet loss rates and operational constraints.

To address these limitations, this work focuses on the systematic evaluation and comparison of different consensus algorithms, including PBFT, 3PC, and other BFT protocols, under diverse network conditions and packet loss rates.

3 SYSTEM MODEL

This section presents the system model for the proposed cooperative maneuvers framework and outlines the assumptions considered in the study. Figure 2 provides an overview of the system model. The system consists of multiple OBUs, each representing a vehicle, employing the Verifiable Event Extension (VEE) proposed in (Vieira et al., 2024a). This way, the ITS-G5 protocol stack running in each of the OBUs is enhanced with the Verifiable Event Protocol (VEP) that is able to support the execution of different consensus algorithms. This extension allows OBUs to process and interpret additional data embedded within ITS messages, facilitating the implementation of consensus processes. It is worth noting that vehicles in which the stack does not use the extension simply disregard the extended data encapsulated in the ITS messages. Therefore, these vehicles can still exchange standard V2X messages and decode VEP-enhanced packets, but are not able to participate in the consensus mechanisms.

3.1 Assumptions

The proximity of vehicles in the tested scenarios indicates that packet loss is unlikely to exceed 20%, aligning with realistic conditions for vehicular networks experiencing minimal external interference. Typically, in V2X communications technologies, packet loss remains below 20% for distances of several hundred meters between transmitting and receiving vehicles. Therefore, this value represents a practical communications range for vehicles engaged in a traffic maneuver (Molina-Masegosa et al., 2020).

To evaluate the robustness and fault-tolerance properties of the consensus algorithms, scalability testing was conducted using configurations of 4, 7, and 10 nodes, in addition to the client vehicle proposing the maneuver. These configurations were chosen to represent cooperative traffic maneuvers involving varying numbers of vehicles, providing information on the performance of the algorithms under different levels of complexity and communication challenges.

In the system, replicas (i.e., vehicles participating in the consensus protocol) may begin the process at different states due to asynchronous starts or operational conditions. These replicas are designed to progress through states independently as they meet the necessary criteria, offering flexibility and resilience in achieving system-wide consensus despite operational variability.

The network model assumes that failures are restricted to message losses, without accounting for the presence of malicious nodes or intentional disruptions. This simplifies the fault model, allowing the focus to remain on the system's ability to effectively handle communication losses, which is critical for reaching consensus in vehicular networks.

Finally, while the vehicles are moving, the group membership for a specific consensus process (cooperative maneuver) remains fixed. Only the nodes present at the beginning participate in the process, guaranteeing stability.

4 CONSENSUS ALGORITHMS

In this section, an overview of the four consensus algorithms employed in this work is presented, as well as a more detailed analysis of a specific one (Zyzzyva), which is given as an example. PBFT was selected due to its widespread adoption in related work, while Zyzzyva was chosen for its speculation mechanism, which holds promise for enhancing performance. 3PC was considered since it can be easily applied to scenarios in which all vehicles need to agree on the proposed maneuver. And finally, HotStuff was employed as a BFT alternative with small communications complexity. Since Zyzzyva is the algorithm chosen to give examples in the next sections, it is the only algorithm explained here in more detail. The four algorithms are summarized in Table 1, with the following aspects being used to compare them:

- **Consensus Type:** Distinguishes consensus algorithms based on their fault-tolerance characteristics. It divides them into BFT and non-BFT algorithms.
- **N**: Indicates the minimum number of nodes needed to achieve consensus. f refers to the number of faulty nodes in the system.
- **Communications Model:** Refers to the method or pattern of communications between nodes in the network. It could include decentralized communications, master-slave, or other models that define how nodes exchange information to achieve consensus.
- **Message Complexity:** Measures the complexity of the algorithm in terms of number of messages exchanged between nodes during the execution of the consensus process.

4.1 Zyzzyva Protocol

Zyzzyva (Kotla et al., 2010) is a BFT protocol designed to improve the replication of state machines

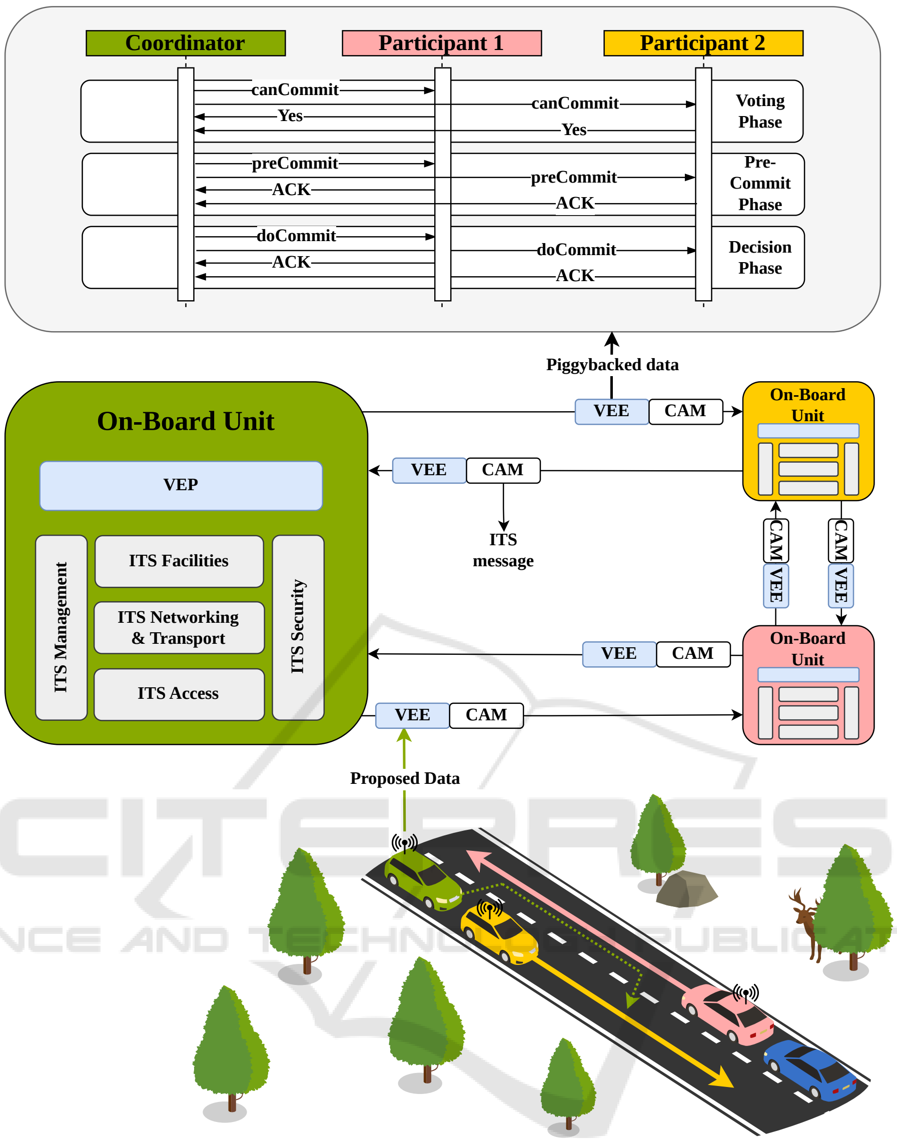

Figure 2: System model overview.

by using speculative execution. This approach reduces overhead and enhances performance by allowing replicas (corresponding to CAVs participating in cooperative maneuvers in the context of this work) to optimistically follow the primary's proposed order and respond to client requests immediately. Temporary inconsistencies are resolved by the clients, who ensure a single, total ordering of requests.

A key feature of Zyzzyva is speculative execution, where replicas process requests without engaging in time-consuming agreement protocols. They send immediate responses to clients, significantly reducing cryptographic overhead and increasing throughput compared to protocols like PBFT and Query/Update (Q/U) (Abd-El-Malek et al., 2005). This speculative mechanism enables concurrent processing and optimizes system performance.

Zyzzyva employs a client-centric approach to ensure consistency. Clients verify stability using replies that include history information. If the replies are consistent, the client considers the request complete. If inconsistencies are detected, clients prompt the system to converge to a stable state, driving overall consistency and reliability.

The agreement phase is responsible for ordering client requests for execution by replicas. The process begins when a client sends a request to the primary replica. The primary replica then forwards the request to the other replicas in the system. Once the replicas receive the request, they analyze it and send their re-

Table 1: Comparison of the four selected consensus algorithms.

Algorithm	Consensus Type	N	Communications Model	Message Complexity[1]
HotStuff (Yin et al., 2019)	BFT	$3f+1$	Client-Primary-Replicas	$O(N)$
PBFT (Castro and Liskov, 1999)	BFT	$3f+1$	Client-Primary-Replicas	$O(N^2)$
Zyzzyva (Kotla et al., 2010)	BFT	$3f+1$	Client-Primary-Replicas	$O(N)$
3PC (Al-Houmaily and Samaras, 2009)	Non-BFT	Unanimous	Coordinator-Participants	$O(N)$

sponses directly to the client.

Upon receiving the responses, the client evaluates them to determine whether the request can be considered complete. This decision is based on the consistency of the responses and the history information contained within them. There are two possible scenarios for determining completion:

1. In a gracious execution scenario, if the client receives $3f+1$ consistent responses, it considers the request complete and acts on it accordingly. This case represents the ideal outcome where sufficient consistent responses are obtained without issues.
2. In cases involving faulty replicas, if the client receives between $2f+1$ and $3f$, it aggregates these into a commit certificate. The client then sends the commit certificate to the replicas. Once $2f+1$ replicas acknowledge the commit certificate, the client considers the request complete and acts on the reply. This approach ensures robustness in the presence of potential faults while maintaining the integrity of the process.

Figure 3 shows the protocol workflow for these two cases. In green, the situation in which there is an immediate conclusion of the consensus process and, in orange, the case where a faulty replica does not respond, thus leading the client to initiate the commit phase.

In addition to the agreement procedure, there are other protocol phases (view change and checkpoint) that are not employed in this work, given that each maneuver is executed independently and therefore the state of the system does not need to be stored from one maneuver request to the other. The view change phase ensures the liveliness of the system by electing a new primary if the current one is faulty or slow. This phase is initiated when enough replicas suspect the behavior of the primary and therefore transition to a new view. The checkpoint phase limits the state that replicas must store and reduces the cost of view changes. Periodically, replicas create checkpoints to manage storage requirements and provide a consistent state for new view transitions.

Zyzzyva offers several advantages, including reduced latency, increased throughput, and strong fault-tolerance capabilities. By minimizing cryptographic overheads and enabling concurrent request processing, it achieves higher performance while maintaining safety and liveliness in asynchronous distributed systems. However, the speculative execution mechanism may result in wasted computational resources if inconsistencies or incorrect ordering lead to discarded speculative work.

5 IMPLEMENTATION

This section delves into the implementation details of such algorithms, illustrating how replicas manage consensus tasks while handling timeouts, retransmissions, and state synchronization. The message structure used for the implementation of consensus processes is also explained.

Every replica can run multiple consensus processes simultaneously, always saving the IDs of processes that have already been removed, each with its own lifecycle. The tasks involved include checking timeouts, handling messages according to the algorithm, and removing timed-out processes. If the chosen algorithm is BFT (HotStuff, PBFT or Zyzzyva), each process includes three types of nodes: client nodes, primary nodes (if applicable), and backup nodes. In contrast, if the algorithm is 3PC, the nodes are categorized into two types: coordinator and par-

[1]Zyzzyva obtains an O(N) message complexity only in case of gracious execution; otherwise, the message complexity is O(N^2) (Zhang et al., 2024).

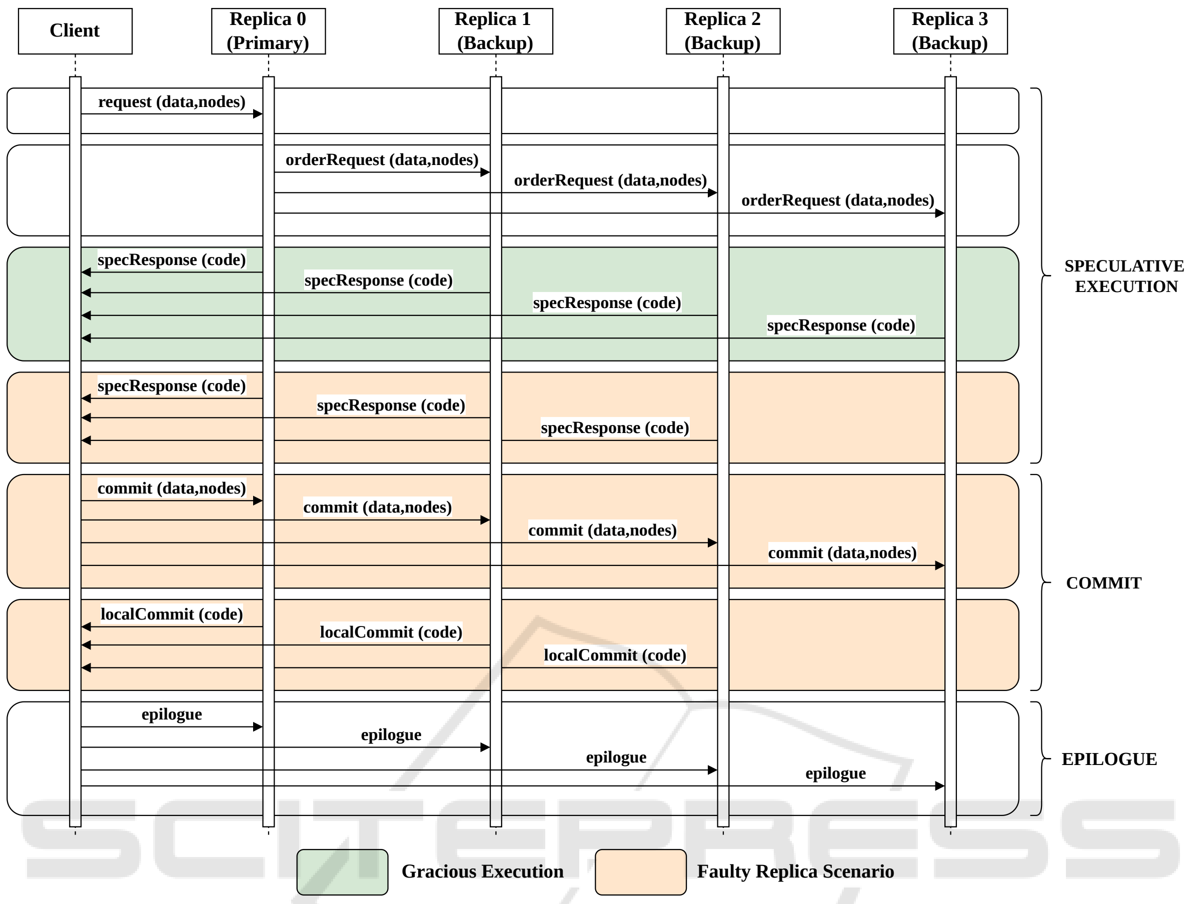

Figure 3: Agreement phase of Zyzzyva consensus protocol.

ticipants.

A process typically begins when a client sends a request to the system. In terms of vehicular networks, this means that a process will start when a vehicle sends a maneuver request to the group of vehicles in close proximity, after detecting a trajectory conflict. The replicas previously selected by the client (the relevant ones for the maneuver) receive the request and, depending on the algorithm, exchange messages to reach consensus. The other replicas assume roles as either primary or backup nodes, being those roles determined by their IDs. The replica with the lowest ID (excluding the client) is designated as the primary, provided that the algorithm requires a primary replica.

To illustrate such scenarios, consider a consensus process that uses the Zyzzyva algorithm. In this process, the primary replica has reached the *localCommit* state (see Figure 3), but the process remains unfinished due to missing messages on the client side. In response, the client retransmits its request. Upon receiving this repeated request, the primary replica will retransmit its *localCommit* message, ensuring the client is updated with the current state. On the other hand, consider a scenario where a replica, due to message loss, finds itself in a delayed state. For instance, if a backup replica receives a *commit* message while still in an earlier state — specifically, the `requestHold` state, where it awaits the *orderRequest* message — it will respond with both the *specResponse* and the *localCommit* messages. By handling such scenarios, the system ensures that all replicas eventually synchronize and maintain consistency.

Now, consider a client initiating a consensus process using Zyzzyva. Upon receiving the request, replicas determine which node will act as the primary. Once identified, the primary creates the process. In Zyzzyva, only the primary initiates the process upon receiving the client request. The primary then sends an *orderRequest* and a *specResponse*. Backup replicas receive the *orderRequest*, create the process locally, and send their own *specResponses*. These responses are collected by the client, which analyzes the number of unique responses received. If the client receives $3f+1$ (where f is the maximum number of Byzantine nodes tolerated) *specResponse* messages, the process is successfully completed. However, if it receives between $2f+1$ and $3f$ responses, it then sends a *commit*. Replicas that receive a *commit* message reply with a *localCommit*. Once the client receives $2f+1$ *localCommit* messages, the process is considered complete, and its state transitions to `finalized`.

Periodic mechanisms are also in place to iterate through all ongoing processes and perform two key verifications:

- **Timeouts for Stage-Specific Retransmissions:** this involves checking if the retransmission timer for the current stage has exceeded the retry timeout. If it has, the replica retransmits messages, up to a maximum of five retransmissions for that process. When the limit of 5 is reached, the process is marked as timed out.
- **Process Timeout Status:** if a process is considered to have timed out, it is removed. Depending on the circumstances, the process is marked as completed (if the success criteria were met) or as failed.

A process is considered timed out if any of the following conditions are met:

- **Retry Timeout:** the time since the last activity exceeds a certain threshold value (11 seconds by default).
- **Retry Count:** the number of retries has exceeded the maximum allowed (5).
- **Lifetime Limit:** the total duration of the process from its initiation to the current time exceeds the maximum allowed (30 seconds by default).

5.1 VEE Message Structure

In this work, ASN.1 description language is used to define the message structure. Figure 4 illustrates the structure of VEE, highlighting its main modules: consensus, ledger and token. In the original VEE proposal (Vieira et al., 2024a), all these modules could be optionally used, but for the purpose of this work the consensus module is always employed, serving as a mandatory component for the execution of the consensus process. This module contains the chosen consensus algorithm, such as Zyzzyva, PBFT, HotStuff, or 3PC, along with a nonce representing the process ID and a timestamp indicating when the message extension was formed. The figure specifically uses Zyzzyva as an example to demonstrate the consensus representation.

Each algorithm is represented by different elements corresponding to the various stages of its workflow. As shown in Figure 4, Zyzzyva can use one of the following six elements in each VEE transmission:

1. **request** – sent from the client to the primary node, containing the proposed operation (maneuver) and the identification of the participating nodes.
2. **orderRequest** – used by the primary replica to forward the client request to all remaining replicas.
3. **specResponse** – sent by the primary or secondary replicas to the client, containing only a code. In this work, it is ensured that a positive reply (“OK”) is always conveyed in the code field.
4. **commit** – transmitted by the client, when it receives between $2f+1$ and $3f$ *specResponses*. The proposed operation (maneuver) and the participating nodes are retransmitted to ensure synchronization.
5. **localCommit** – operates similarly to *specResponse*, allowing primary and secondary replicas to send a reply to the client.
6. **epilogue** – sent by the client to signal when the consensus process has been successfully completed.

6 EVALUATION

In this section, an overview of the experimental setup used for the evaluation process is provided, as well as an analysis and discussion of the obtained results.

6.1 Experimental Setup

In order to evaluate the performance of the different consensus algorithms, an experimental setup was developed using Docker containers. All containers are hosted on the same laptop with the following specifications: 11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz (8 cores), 16GB of RAM, and a 512GB NVMe SSD, running Ubuntu 22.04 LTS.

Each container in the environment is equipped with the ETSI ITS-G5 protocol stack and VEP, enabling the use of ITS message extensions. The setup configuration, illustrated in Figure 5, comprises a minimum of five to a maximum of eleven of these containers:

- **Container 1:** acts as the client, initiating and managing the consensus processes.
- **Container 2:** serves as the primary replica, responsible for coordinating the consensus process when required.

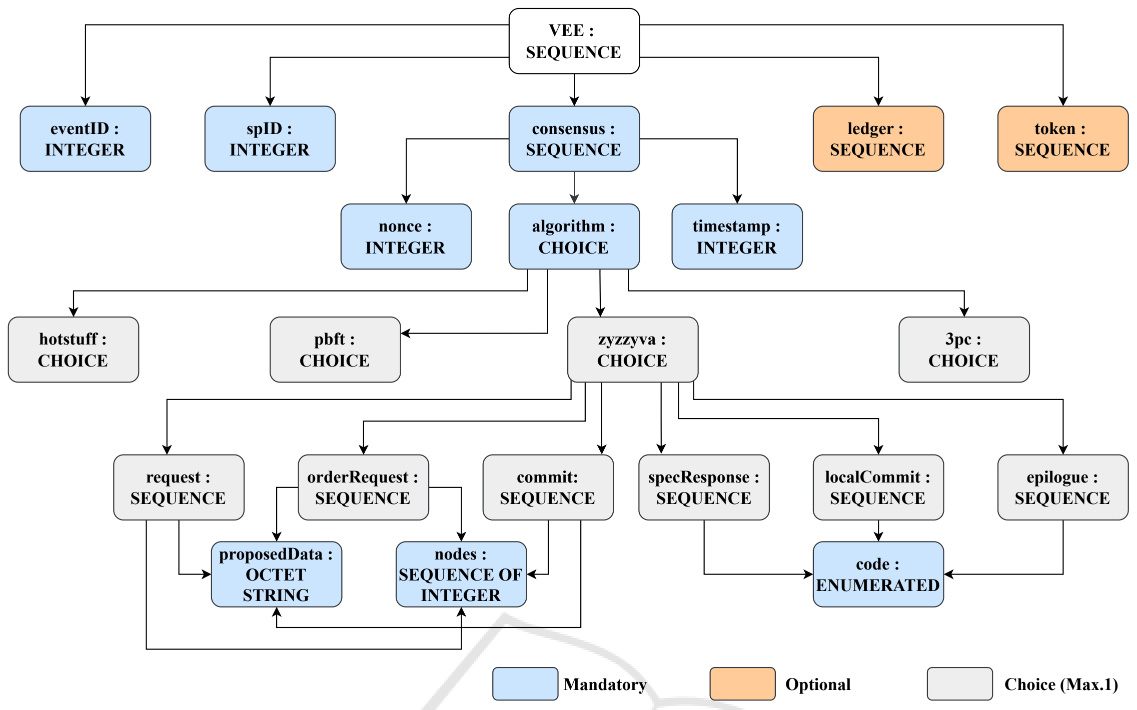

Figure 4: ASN.1 definition of the implemented VEE message format (Zyzzyva used as an example for the detailed structure).

- **Containers 3-11:** operate as backup replicas participating in the consensus process.

The containers are all connected to the same local network (*dsrc_network*) via individual network interfaces. This virtual V2X network, running on the test laptop, is responsible for managing all traffic generated by the various ETSI ITS-G5 protocol stacks. To simulate real-world conditions, **traffic control rules** are applied to each container's interface, specifically affecting incoming traffic. These rules are used to introduce packet loss, mimicking network disruptions and challenging wireless communications channel conditions. This way, the setup attempts to simulate real-world scenarios, providing a controlled environment for systematic testing.

To effectively evaluate the consensus algorithms, the following tests were conducted:

- **Performance Testing:** measured the delays in reaching consensus under various network conditions. Simulated different levels of packet loss rates (e.g., 0%, 1%, 5%, and 20%) to assess the fault-tolerance and efficiency of the algorithms.
- **Scalability Testing:** Increased the number of nodes in the network to observe the algorithm's performance in maneuver scenarios with varying traffic complexity. Examined how the system handles additional replicas and higher communication loads.

In these experimental tests, it was ensured that only one consensus process was active at any given time. This was accomplished by consistently using the same client node to initiate each process and by introducing a sufficiently large interval between the start of consecutive processes. This framework provides a flexible and controlled platform for testing, allowing the simulation of complex, real-world scenarios with distinct packet loss rates, node failures, and varying network loads. It enables the assessment of fault-tolerance and efficiency metrics regarding the consensus algorithms performance under different network stress conditions, as well as the validation of node synchronization and state correctness during the consensus process.

6.2 Results

The total durations of the consensus processes measured during the experiments are presented in Figure 6 as empirical cumulative distribution functions (ECDFs). Each ECDF is derived from 100 consensus process delay measurements for each of the algorithms employed in this work. The results document the time required for the network to reach agreement based on a request with dummy payload, as well as the percentage of successful consensus processes under various conditions of node count and average packet loss rate. Extended messages were the sole

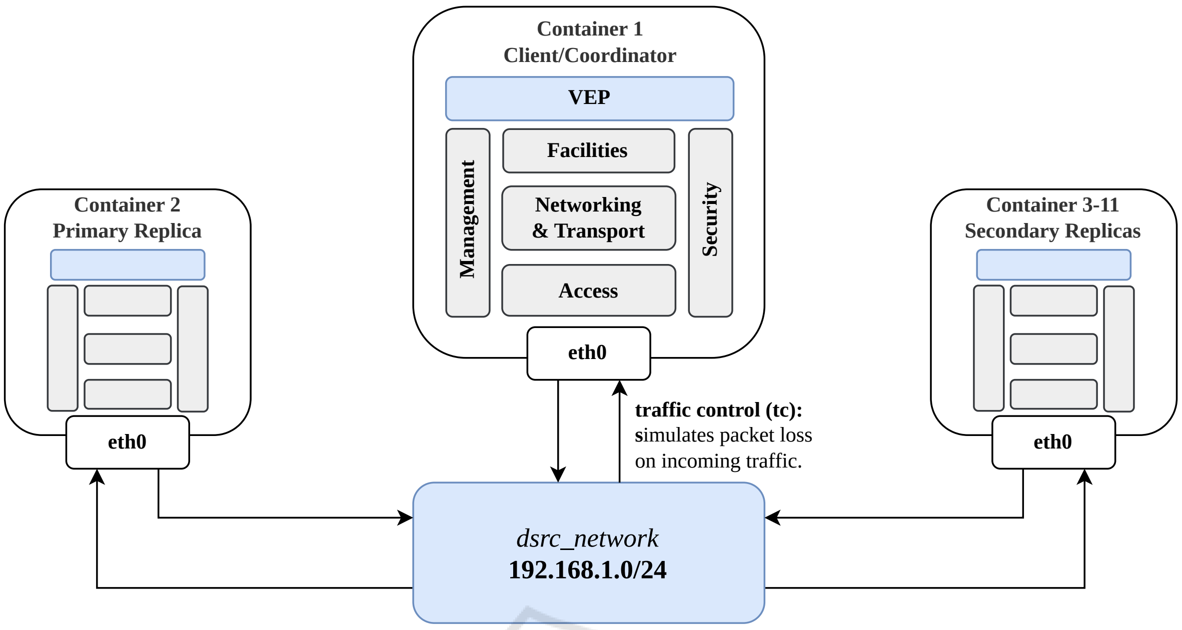

Figure 5: Experimental setup used to evaluate consensus algorithms performance.

type of messages tested. Although this approach requires more time to achieve consensus compared to dedicated messages, it results in a lower channel busy ratio (CBR) (Vieira et al., 2024b), thereby minimizing interference with other C-ITS messages transmitted over the wireless medium.

The findings reveal that variations in packet loss rates and the number of nodes participating in the maneuver significantly influence consensus delays and the percentage of finalized processes. Higher packet loss results in longer consensus times and a reduced rate of successful process completions across all algorithms. This outcome is largely driven by delays caused by retransmissions and the handling of missing messages. The impact is especially pronounced under extreme packet loss conditions, such as 20%, where delays increase substantially, and success rates decline sharply.

The impact of increasing the number of nodes is also noticeable. A larger number of nodes in the consensus process introduce greater communication overhead, extending consensus times across all algorithms. However, increasing the node count can enhance fault-tolerance, leading to higher success rates for some algorithms. For example, with PBFT under 20% packet loss, the success rate improves from approximately 54% (4 replicas) to 68% (7) and 86% (10). On the other hand, the success rate for 3PC drops significantly as node count increases, highlighting the limitations of non-BFT algorithms in lossy environments.

Among the evaluated algorithms, Zyzzyva consistently outperforms the others in terms of robustness and efficiency. It maintains shorter consensus times and higher completion rates even in unreliable network conditions, showcasing strong resilience. PBFT performs adequately under moderate conditions but suffers from longer delays and reduced success rates as packet loss rises. In contrast, HotStuff and 3PC exhibit high sensitivity to packet loss, with significant increases in total delays and steep drops in completion rates, rendering them less reliable in such scenarios.

7 CONCLUSION

In this article, a comparative analysis is presented regarding the application of four different consensus algorithms to the negotiation of cooperative traffic maneuvers. Three of the selected protocols exhibit Byzantine Fault-Tolerance (BFT) properties, while one is non-BFT (3PC). This evaluation was conducted using VEP to extend standard C-ITS messages, ensuring that the Channel Busy Ratio (CBR) was not overloaded. The consensus framework was validated in a local environment with several containers running the ETSI ITS-G5 protocol stack and VEP. The obtained results show that both packet loss and the number of nodes involved in the maneuver have a significant impact on the time taken to reach consensus and the rate of successfully completed processes. While this research focuses on decision-making for traf-

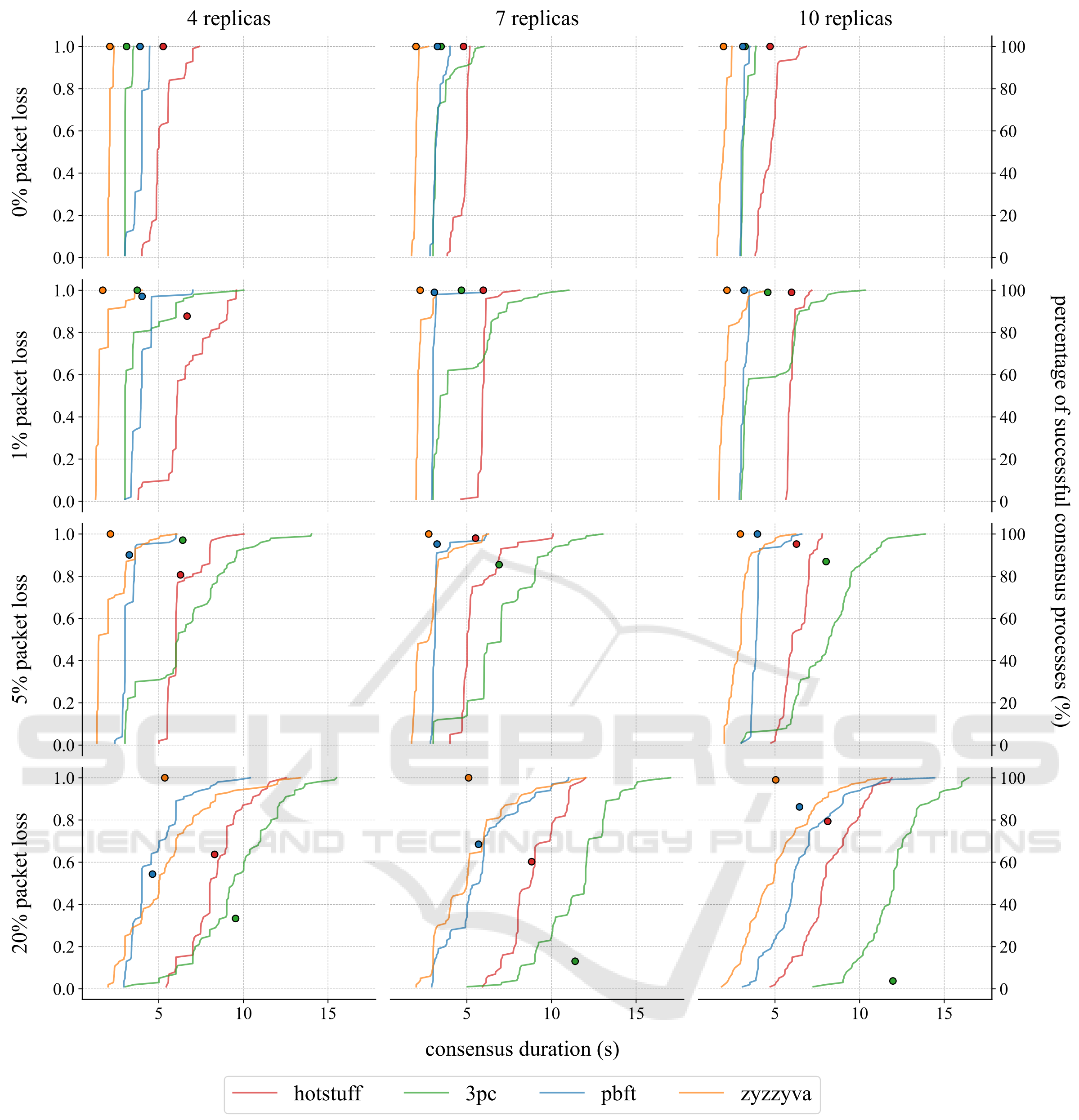

Figure 6: ECDF of consensus duration (s) and percentage of successful consensus processes (%) for the four implemented algorithms. The results were obtained for several packet loss rates and number of replicas. The ratio of successful processes (% in the y-axis) is plotted over the average consensus duration values (seconds in the x-axis) for each algorithm.

fic maneuvers, the proposed methods are adaptable to other scenarios requiring decentralized decision-making, such as fleet management or coordinated responses in emergency situations.

Future work could explore additional consensus algorithms, including hybrid approaches that adapt to the specific requirements of each situation. For instance, in scenarios where unanimous agreement is critical, the system could dynamically select the most appropriate algorithm compared to cases where a majority vote is sufficient. Another possible line of work involves developing mechanisms to deal with malicious nodes. For example, by keeping records of the processes in each vehicle, it could be possible to identify and penalize the nodes responsible for unsuccessful maneuvers, once a consensus has been reached. In addition, further investigation is warranted to optimize the synchronization process for the remaining vehicles. While related work discusses a gossip protocol, exploring alternative methods to reduce message overhead and improve efficiency could yield significant benefits.

ACKNOWLEDGEMENTS

This work is supported by the European Union / Next Generation EU, through Programa de Recuperação e Resiliência (PRR) [Project Nr. 29: Route 25 (02/C05-i01.01/2022.PC645463824-00000063)].

REFERENCES

Abd-El-Malek, M., Ganger, G. R., Goodson, G. R., Reiter, M. K., and Wylie, J. J. (2005). Fault-scalable Byzantine fault-tolerant services. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, SOSP '05, page 59–74, New York, NY, USA. Association for Computing Machinery.

Al-Houmaily, Y. J. and Samaras, G. (2009). *Three-Phase Commit*, pages 3091–3097. Springer US, Boston, MA.

Castro, M. and Liskov, B. (1999). Practical Byzantine Fault Tolerance. In *3rd Symposium on Operating Systems Design and Implementation (OSDI 99)*, New Orleans, LA. USENIX Association.

Feng, C., Xu, Z., Zhu, X., Klaine, P. V., and Zhang, L. (2023). Wireless Distributed Consensus in Vehicle to Vehicle Networks for Autonomous Driving. *IEEE Transactions on Vehicular Technology*, 72(6):8061–8073.

Kiraz, M., Sivrikaya, F., and Albayrak, S. (2024). A Survey on Sensor Selection and Placement for Connected and Automated Mobility. *IEEE Open Journal of Intelligent Transportation Systems*, 5:692–710.

Kotla, R., Alvisi, L., Dahlin, M., Clement, A., and Wong, E. (2010). Zyzzyva: Speculative Byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4).

Li, Z., Zhang, L., Zhang, X., and Imran, M. (2022). Design and Implementation of a Raft based Wireless Consensus System for Autonomous Driving. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 3736–3741.

Molina-Masegosa, R., Gozalvez, J., and Sepulcre, M. (2020). Comparison of IEEE 802.11p and LTE-V2X: An Evaluation With Periodic and Aperiodic Messages of Constant and Variable Size. *IEEE Access*, 8:121526–121548.

Paluszczyszyn, D., Stamenkovic, V. R., and Lane, B. (2024). Toward Development of Ecosystem for Connected Autonomous Vehicles: Challenges of Modeling and Testing Sensors. *IEEE Sensors Letters*, 8(3):1–2.

Vieira, E., Almeida, J., Ferreira, J., and Bartolomeu, P. C. (2023). Safeguarding Cooperative Maneuver Information with Practical Byzantine Fault Tolerance. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 108–115.

Vieira, E., Almeida, J., Ferreira, J., and Bartolomeu, P. C. (2024a). Enabling Seamless Data Security, Consensus, and Trading in Vehicular Networks. *IEEE Transactions on Intelligent Vehicles*, pages 1–22.

Vieira, E., Almeida, J., Ferreira, J., and Bartolomeu, P. C. (2024b). Extending V2X Messages for the Implementation of Consensus-Building Mechanisms in C-ITS. In *CARS@EDCC2024 Workshop - Critical Automotive applications: Robustness & Safety*, Leuven, Belgium.

Yin, M., Malkhi, D., Reiter, M. K., Gueta, G. G., and Abraham, I. (2019). HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 347–356, New York, NY, USA. Association for Computing Machinery.

Zhang, G., Pan, F., Mao, Y., Tijanic, S., Dang'ana, M., Motepalli, S., Zhang, S., and Jacobsen, H.-A. (2024). Reaching Consensus in the Byzantine Empire: A Comprehensive Review of BFT Consensus Algorithms. *ACM Comput. Surv.*, 56(5).