Byzantine Consensus in Vehicle Platooning via Inter-Vehicle Communication

Martin Wegner, Wenbo Xu, Rüdiger Kapitza, Lars Wolf
Institute of Operating Systems and Computer Networks
Technische Universität Braunschweig, Germany
Email: (wegner|wxu|kapitza|wolf)@ibr.cs.tu-bs.de

***Abstract*—Cooperative driving and platooning have recently gained focus. Letting vehicles reach consensus to implement joint decisions is an essential service in this context. To address this problem, we propose a novel consensus protocol named BFT-ARM that fits real sensor values and can tolerate $t(< n/3)$ Byzantine nodes out of n. BFT-ARM guarantees that the decision is close to the median of all good nodes. We also present the simulation framework ArteryLTE to evaluate our protocol.** [1]

***Keywords*—Cooperative driving, Byzantine consensus, Inter-Vehicle Communication**

I. Introduction

In recent years, an increasing amount of Advanced Driver Assistance Systems (ADASs) can be seen in automobiles. This also includes the development of cooperative driving functions. Many of these applications can be improved by exchanging data via Inter-Vehicle Communication (IVC) to improve safety, resource usage, energy efficiency and driving experience [1]. One example of cooperative driving is *platooning*, where a group of vehicles can follow each other automatically and keep the optimal distance among each other [2, 3]. Apart from following passively, there are also a number of useful applications which first need to agree on a common value for a cooperative decision. For example, in certain application scenarios vehicles will need to detect the traffic condition or weather condition in their surroundings to adjust their operations to, or to calculate the best route according each vehicle's own navigation device, or to set a preferred speed for the cruise control, etc. The common properties of such applications are: 1) The value is required to be agreed among all vehicles. 2) The value to be agreed upon can be measured individually by each vehicle. 3) Even if some faulty vehicles do not follow the common decision, the safety of others is not violated.

We also consider the existence of faulty or malicious nodes in the group. Faulty behaviours do not only include crash faults but also arbitrary faults like bit flips or providing inaccurate, inconsistent and even malicious values. All these faults are referred to as Byzantine faults [4]. The Byzantine consensus protocol aims at achieving consensus among all correct participants, despite of a limited number of faulty nodes. Examples of such protocols can be found in [5, 6].

Most Byzantine consensus problems assume the value domain is discrete and limited, for instance the binary consensus [5] or multi-value consensus [7]. However, in automobile applications, especially those involving sensor values, the values can be continuous and "smooth", e. g., speed, distance, temperature, etc.

In this work, we designed a new consensus protocol for the continuous value domain in vehicle platooning named BFT-ARM (Byzantine Fault Tolerant and Asynchronous Real-value consensus with Median validity). We also build a simulation framework based on our previous work and will use it to evaluate the consensus protocol.

The paper is organized as follows. Section II discusses related work. Section III defines the system model and gives the problem statement. Section IV presents the design of BFT-ARM and section V introduces the evaluation framework. Section VI concludes the paper.

II. Related Work

In the *Byzantine consensus problem*, each node has an input value and tries to reach consensus on one of these, and there are a limited number of Byzantine nodes [5]. An important aspect of the Byzantine consensus problem is how to define the validity of the agreed value. There are different opinions from different viewpoints. E. g., Neiger distinguishes the *Validity* and *Strong Validity* [8]. The former requires that—if all correct nodes have the same input value—they will decide on that value, but does not guarantee anything when the input values are different. The latter, Strong Validity, requires that the decided value comes from a correct node. Then Neiger proves that achieving Strong Validity requires at least $t \cdot |\mathcal{D}|$ nodes, where t is the maximum tolerable Byzantine nodes and $\mathcal{D}$ is the domain of the input values. This means a tremendous number of nodes are necessary when the input value domain is large.

A recent work of Stolz and Wattenhofer proposes a weaker requirement compared to Strong Validity, called *median validity* [9]. It only requires that the agreed value is close to the median of all good nodes which, especially with

[1]This work is part of the DFG Research Unit *Controlling Concurrent Change*, funding number FOR 1800.

a continuous value domain, is useful. However, their protocol assumes a synchronous communication where message transmission time has a known upper bound. This assumption is not applicable in Inter-Vehicle Communication. So, we designed our new BFT-ARM protocol to achieve median validity.

There is also some other work from the viewpoint of control theory to manage platooning with a consensus strategy [10]. While this is useful for another family of applications like instant speed and distance control, it is an orthogonal direction to our work.

III. System Model and Problem Statement

System Model: In the following, a platoon consists of n vehicles, or—more abstract—nodes: $\{p_1, p_2, \dots, p_n\}$. Every node has an input value $x_i \in \mathbb{R}$ [2], e. g., from its sensor or configuration. A node is called *correct* if 1) its input value is correct and 2) it exactly follows the protocol. Among all the nodes, up to t ($< n/3$) nodes can be *faulty*, meaning that they can behave arbitrarily such as taking an incorrect value from a malfunctioning sensor or actively work against the protocol.

Consensus problem: BFT-ARM achieves consensus on a value $v \in \mathbb{R}$ satisfying the following conditions:

- *Agreement*: No two correct nodes decide differently.
- *Termination*: Every correct node eventually decides.
- *Validity*: The decided value of correct nodes v is *valid* (see definition below).

Inspired by the work of Stolz et al. [9], the validity is defined in the following way: Assuming there are actually f ($\leqslant t$) faulty nodes during runtime, not known by the consensus algorithm. Let SG be the sorted array of input values of all good nodes (the index starts from 0). Then $SG[\lceil\frac{n-f}{2}\rceil - 1]$ represents the median value of SG.

Definition 1. *Validity: a decision v is* valid*, if*

$$SG[\lceil\frac{n-f}{2}\rceil - 1 - t] \leqslant v \leqslant SG[\lceil\frac{n-f}{2}\rceil - 1 + t] \quad (1)$$

In other words, a valid value is the one within the range of the middle $(2t+1)$ correct nodes.

Network: Nodes *asynchronously* communicate via messages over the network. That means that messages can be duplicated or corrupted or, in principle, can experience an unbounded delay, i. e. they are lost. However, eventually synchronicity is required to overcome the FLP impossibility [11]. While previous approaches achieve termination only with known upper bounds for delays (cf. Section II), BFT-ARM is able to cope with *unknown* bounded delays of messages.

[2] In practice, the value space is still a finite set limited by the platform. We do not discuss Turing uncomputable numbers or real computation here.

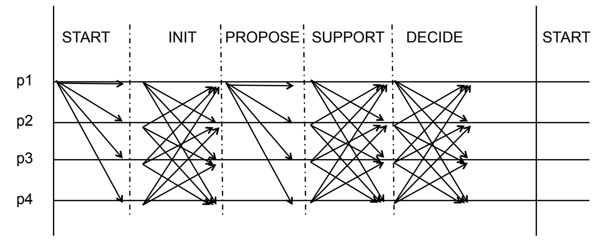

Figure 1. BFT-ARM in normal case.

Digital Signature and Trusted Subsystem: The messages are signed with digital signatures. A message m signed by a node i is notated as $\langle m \rangle_{\sigma_i}$. We also assume that every node possesses a trusted subsystem for message authentication and verification with a monotonic counter. The value of the counter can be used to authenticate some special messages, and increases by 1 after that. The receiver of one of these messages can also verify that this message has the valid counter value without any gaps to the previous one. Examples of such a subsystem applied in Byzantine fault tolerant systems can be found in [12, 13]. Furthermore, we assume that faulty nodes cannot break the digital signature mechanism nor the trusted subsystem.

IV. BFT-ARM Design

A. Normal case operation

The normal case protocol is illustrated in Figure 1.

It can be divided into 6 steps:

1) The leader p_i periodically activates a consensus request with a broadcast $\langle \textsc{Start}, seq, p_i \rangle_{\sigma_i}$. seq is a sequence number generated by the trusted counter.
2) Upon a received Start message, each node p_j firstly verifies the sequence number. If it is a valid sequence number, it broadcasts (including to itself) with its input value in $\langle \textsc{Init}, seq, p_j, x_j \rangle_{\sigma_j}$.
3) When the leader received $(n-t)$ values (including itself), it sorts the received values and picks the median value v_{med}. Then it proposes v_{med} together with the $(n-t)$ original signed Init messages attached as a certificate $\vec{cm}$. Namely: $\langle \textsc{Propose}, seq, p_i, v_{med}, \vec{cm} \rangle_{\sigma_i}$.
4) Upon a node p_j received the Propose message, it verifies that v_{med} is really the median of all the values in $\vec{cm}$. If so, it broadcasts $\langle \textsc{Support}, seq, p_j, v_{med} \rangle_{\sigma_j}$.
5) Upon a node p_j received $\lceil (n+t+1)/2 \rceil$ Support for the same v_{med}, it broadcasts $\langle \textsc{Decide}, seq, p_j, v_{med} \rangle_{\sigma_j}$.
6) Upon a node p_j received $\lceil (n+t+1)/2 \rceil$ Decide for the same v_{med}, it decides v_{med}.

From step 3 on, BFT-ARM is similar to the PBFT protocol [6]. So if the leader proposes the correct v_{med} matching the certificate, all correct nodes will decide v_{med}.

Now we prove that v_{med} is valid according to Definition 1.

Theorem 1. *Let SA be the sorted array of the input values of any $(n-t)$ nodes. The median of SA is denoted as $v = SA[\lceil\frac{n-t}{2}\rceil - 1]$. Then v is valid.*

Proof. According to the definition of median, there are at least $(\lceil\frac{n-t}{2}\rceil - 1)$ nodes whose value is no greater than v. Among them there are at least $(\lceil\frac{n-t}{2}\rceil - 1 - f)$ good nodes. And because $f \leqslant t < n/3$, we have $\lceil\frac{n-t}{2}\rceil - 1 - f \geqslant \lceil\frac{n-f}{2}\rceil - 1 - t \geqslant 0$. So $v \geqslant SG[\lceil\frac{n-t}{2}\rceil - 1 - f] \geqslant SG[\lceil\frac{n-f}{2}\rceil - 1 - t]$. Similarly, we can prove that $v \leqslant SG[\lceil\frac{n-f}{2}\rceil - 1 + t]$. Because of the Definition 1, v is valid. □

Thus, the validity of the proposal can be confirmed by comparing with the certificate of $(n-t)$ values in step 4.

We use the trusted counter to generate a sequence number for every START message from the leader. The sequence number is monotonically increasing by one at a time, so there is exactly one sequence number assigned to every consensus period. In this way, faulty nodes cannot provide an outdated value (replay attack). If a node detects that the sequence number does not belong to this period, it will discard the message. A synchronized clock is not required here, but the interval of the period is known to everyone. From the first time a node receives the sequence number from the leader, it can determine the correspondence between the sequence number and period.

B. Suspect leader protocol

When the leader is faulty or disconnected from the group, leading to a fail of consensus within a predefined timeout, the other nodes will initiate a suspect leader protocol, basically similar to the PBFT view change protocol (without considering the history). When a node p_j suspects the leader p_{cur}, it broadcasts a $\langle \text{SUSPECT}, p_j, p_{cur}, p_{new}\rangle_{\sigma_j}$, where p_{new} is the next leader according to a deterministic rule, e. g., based on the position of the platoon's vehicles such as choosing the one behind the current leader, possibly wrapping to the front vehicle.

When p_{new} receives $(\lceil(n+t+1)/2\rceil - 1)$ messages suspecting current leader, it takes over the leader role and broadcasts $\langle \text{NEWLEADER}, p_{new}, seq_{new}\rangle_{\sigma_{new}}$ with its own sequence number, and operates as in the normal case.

V. EVALUATION

To evaluate BFT-ARM in platooning environments, we intend to use an extended version of the *ArteryLTE*[3] simulation framework, which is detailed in [14].

A. Simulation Framework

ArteryLTE is based on the renowned open-source *Vehicles in Network Simulation (Veins)* framework [15]. The *Veins* projectcombines the dedicated network simulator OMNeT++ with the microscopic traffic simulator *Simulation of Urban Mobility* (SUMO). In addition, Veins provides an implementation of the US Wireless Access in Vehicular Environments (WAVE) Dedicated Short Range Communication (DSRC) stack based on IEEE 802.11p.

[3]https://github.com/ibr-cm/artery-lte

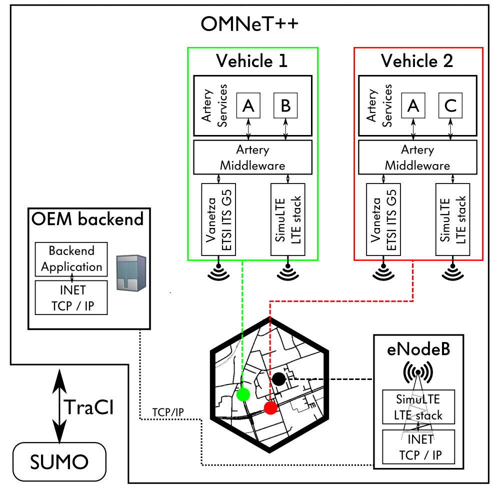

Figure 2. Architecture of the *ArteryLTE* simulation framework.

ArteryLTE integrates several extensions to Veins: First, a modular middleware for Veins called *Artery*[4] [16] is used to implement heterogeneous vehicle capabilities. Multiple applications (so-called Artery services) can be implemented and dynamically configured for vehicles per market penetration rates. Furthermore via *Vanetza*[5], the European equivalent to the WAVE stack, the European Telecommunications Standards Institute (ETSI) Intelligent Transport System (ITS) G5 protocol stack, is brought in and used to disseminate Cooperative Awareness (CA) messages [17].

Secondly, *ArteryLTE* integrates Long Term Evolution (LTE) support for vehicles as introduced to Veins by the *VeinsLTE* [18] project, thus enabling heterogeneous communication technologies on the network nodes. VeinsLTE's *decision maker* is replaced by an option in *Artery*'s middleware that allows Artery services to choose between either the ETSI ITS G5 or the LTE stack for communication.

Thirdly, *ArteryLTE* includes support for backend-based applications. A backend is represented by a static network node in the network which is connected to the eNodeBs of the LTE network.

The architecture of the *ArteryLTE* framework is depicted in Figure 2. In the presented cell of the eNodeB two vehicles are shown, both equipped with an LTE and an ETSI ITS G5 stack. Different Artery services—A, B or C—are deployed on the vehicles. Data transmitted via LTE by the vehicles is forwarded between the eNodeB and the backend.

Furthermore, local perception sensors for advanced ADASs are the latest addition to *ArteryLTE* [14].

[4]https://github.com/riebl/artery
[5]https://github.com/riebl/vanetza

B. Extension of the Framework

As part of this work, we are bringing yet another extension into the *ArteryLTE* framework: The *Plexe* extension [19] to Veins enables the simulation of vehicle platoons with corresponding control algorithms, such as for cruise control, and the implementation of cooperative driving applications. We are in the process of porting the changes made by *Plexe* to Veins to *ArteryLTE*'s codebase so that it is able to interact with Plexe's SUMO version via SUMO's TraCI protocol. We will use the platooning examples and the included control algorithms of Plexe as the basic scenario for our application. Vehicles of the platoon will—in a first step—be equipped with IEEE 802.11p for local communication to run the presented consensus protocol.

VI. Conclusion and Future Work

As soon as the basic setup of BFT-ARM is implemented, in a first step we evaluate the characteristics of the consensus protocol among vehicles via IVC. We then intend to use the whole potential of our communication environment to improve the consensus process. For example, to take advantage of the available heterogeneous networks, we envisage the ability to fall back to cellular communication in cases where local communication of a group is disrupted. Furthermore, in the case of ADASs that are tightly coupled to an Original Equipment Manufacturer (OEM) backend, running an agreement might be assisted by this backend as, e. g., the backend may initiate a consensus, or a leader change based on data available to the backend such as network metrics or local traffic data [14].

References

[1] Theodore Willke, Patcharinee Tientrakool, and Nicholas Maxemchuk. "A survey of inter-vehicle communication protocols and their applications". In: *IEEE Communications Surveys & Tutorials* 11.2 (2009), pp. 3–20.

[2] Pedro Fernandes and Urbano Nunes. "Platooning with IVC-enabled autonomous vehicles: Strategies to mitigate communication delays, improve safety and traffic flow". In: *IEEE Transactions on Intelligent Transportation Systems* 13.1 (Mar. 2012), pp. 91–106.

[3] Michele Segata et al. "Supporting platooning maneuvers through IVC: An initial protocol analysis for the JOIN maneuver". In: *11th Annual Conference on Wireless On-Demand Network Systems and Services, IEEE/IFIP WONS 2014 - Proceedings*. IEEE, Dec. 2014, pp. 53–60.

[4] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (July 1982), pp. 382–401.

[5] Gabriel Bracha. "Asynchronous Byzantine agreement protocols". In: *Information and Computation* 75.2 (1987), pp. 130–143.

[6] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance". In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. New Orleans, Louisiana, USA: USENIX Association, 1999, pp. 173–186.

[7] Kim Potter Kihlstrom, Louise E Moser, and P Michael Melliar-Smith. "Byzantine fault detectors for solving consensus". In: *The Computer Journal* 46.1 (2003), pp. 16–35.

[8] Gil Neiger. "Distributed consensus revisited". In: *Information Processing Letters* 49.4 (1994), pp. 195–201.

[9] David Stolz and Roger Wattenhofer. "Byzantine Agreement with Median Validity". In: *19th International Conference on Priniciples of Distributed Systems (OPODIS), Rennes, France*. 2015.

[10] S Santini et al. "A consensus-based approach for platooning with inter-vehicular communications". In: *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE. 2015, pp. 1158–1166.

[11] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. "Impossibility of distributed consensus with one faulty process". In: *Journal of the ACM (JACM)* 32.2 (1985), pp. 374–382.

[12] Giuliana Santos Veronese et al. "Efficient byzantine fault-tolerance". In: *Computers, IEEE Transactions on* 62.1 (2013), pp. 16–30.

[13] Rüdiger Kapitza et al. "CheapBFT: Resource-efficient Byzantine Fault Tolerance". In: *Proceedings of the EuroSys 2012 Conference*. Ed. by European Chapter of ACM SIGOPS. Switzerland, 2012, pp. 295–308.

[14] Julian Timpner et al. "Towards a Multi-Protocol Microscopic IVC Simulation Environment for ADASs". In: *4th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication*. Berlin, 2016.

[15] C. Sommer, R. German, and F. Dressler. "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis". In: *IEEE Trans. Mobile Comput.* 10.1 (Jan. 2011), pp. 3–15.

[16] R. Riebl et al. "Artery - Extendig Veins for VANET applications". In: *Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. 2015.

[17] *ETSI EN 302 637-2 V1.3.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. ETSI, Sept. 2014.

[18] F. Hagenauer, F. Dressler, and C. Sommer. "Poster: A simulator for heterogeneous vehicular networks". In: *Proc. Vehicular Networking Conference (VNC)*. IEEE, Dec. 2014, pp. 185–186.

[19] Michele Segata et al. "Plexe: A platooning extension for Veins". In: *IEEE Vehicular Networking Conference, VNC*. IEEE, Dec. 2014, pp. 53–60.