Cooperation with Disagreement Correction in the Presence of Communication Failures

Oscar Morales-Ponce$^{1,2}$ and Elad M. Schiller$^{1,2}$ and Paolo Falcone$^{3,*}$†‡

Abstract

Vehicle-to-vehicle communication is a fundamental requirement in cooperative vehicular systems to achieve high performance while keeping high safety standards. Vehicles periodically exchange critical information with nearby vehicles to determine their maneuvers according to the information quality and established strategies. However, wireless communication is prone to failures. Thus, participants can be unaware that other participants have not received the information on time resulting in conflicting trajectories that may not be safe. We present a deterministic solution that allows all participants to use a default strategy when other participants have not received on time the complete information. We base our solution on a timed distributed protocol that adapts its output according to the effect of message omission failures so that the disagreement period occurs for no longer than a constant time (of the order of milliseconds) that only depends on the message delay. We formally show the correctness and perform experiments to corroborate its efficiency. We explain how the proposed solution can be used on vehicular platooning to attain high performance and still guarantee high safety standards despite communication failures. We believe that this work can facilitate the implementation of cooperative driving systems that have to deal with inherent (communication) uncertainties.

1 Introduction

The vision of automated driving systems holds a promise to change the transportation reality. Current deployments focus on autonomous solutions which pose a variety of sensors and actuators for safe driving on the road, e.g., Volvo drive me project in Gothenburg and Google car in California. These autonomous solutions are based on the vehicles’ ability to observe obstacles in their line-of-sight. Vehicle-to-vehicle communication has the potential to improve the system confidence on the sensory information and support advance vehicular coordination, e.g., when changing lanes and crossing intersections, as well as improving the road capacity by reducing the inter-vehicle distances. However, failures on the communication can result in hazardous situations due to coordination based on inconsistent information shared by the participating vehicles.

Consider the architecture depicted in Figure 1 (left) that implements a cooperative driving system in a participating vehicle. The Communication Protocol implements the mechanisms for exchanging information with other vehicles. The Control Algorithm plans the vehicle motion according to the on-board sensors

---

* This work was partially supported by the European Union’s Seventh Programme for research, technological development and demonstration, through project KARYON, under grant agreement No. 288195.
† Computer Science and Engineering. Chalmers University of Technology, Göteborg, Sweden {mooscar, elad}@chalmers.se
‡ Signals and Systems. Chalmers University of Technology, Göteborg, Sweden falcone@chalmers.se
and exchanged information. Note that the local Control Algorithms depend on the (in general vectorial) variable $s$. Thus, $s$ is a common piece of information that all vehicles share in order to establish a correct cooperation. For instance, in a vehicle platoon application $s$ might include the maximum acceleration levels imposed to all vehicles by the limited braking capabilities of one of them [1]. Clearly, loss of messages when a new value of $s$ is established may lead to an inconsistent value in one or more vehicles, thus leading to an unsafe operation of all the cooperating vehicles. It is then necessary an additional layer, shown in Figure 1 right, between the Communication Layer and the Control Algorithm. Such additional layer that implements a Timed Protocol for Cooperation with Disagreement Correction for resolving disagreements on the variable $s$ among the vehicles is the main focus of this paper.

Specifically, we address the following research question: How can cooperative systems be used to attain the highest performance without compromising the safety despite communication failures? We consider applications in which individual vehicles estimate their ability to cooperate according to the sensory information quality and communicate their maximum supported cooperative level [2, 3]. Vehicles then decide on their cooperative level that all vehicles support based only on the received information. However, due to communication failures, vehicles can decide to operate on distinct levels. It is a key issue to guarantee that the uncertainty time along vehicles occurs only in short time periods. Formally the problem that we address is:

**Problem 1.** [Minimum Longest Uncertainty Time.] Is there an upper-bound on the longest period of time that cooperative vehicles spend with the uncertainty about the operation level of other vehicles?

Let us consider a system with discrete time (or rounds) of a given length. One can attempt to solve Problem [1] using distributed consensus algorithms. In the consensus problem, every component (vehicle) proposes a value and the objective is to agree on a common one. However, it is known that the consensus problem is not deterministically solvable in unreliable synchronous networks and any $r$-round algorithm has probability of disagreement of at least $\frac{1}{r+1}$ [4] (Theorem 5.1 and Theorem 5.5). Therefore, when the
communication failures are too frequent or too severe, the uncertainty time can be unbounded since they can disagree for unbounded number of protocol executions. In this paper, we present a communication protocol that guarantees that the uncertainty time is constant despite the communication failures.

Our solution is based on a communication protocol that collects values from all system components. Once this proposed set $s$ is delivered to all the components, the protocol employs a deterministic function to decide on a single value from $s$ that all system components are to use. The protocol identifies the periods in which there is a clear risk for disagreement due to temporary communication failures, i.e., a period in which $s$ was not delivered to the entire system by the due time. Once such risk is identified, the protocol triggers a correction strategy against the risk of having disagreement for more than a constant number of rounds. Namely, after the occurrence of communication failures that jeopardize safety, all system components will rapidly start a period to reestablish their system confidence by returning to the default value. Once the network becomes stable and no communication failures occur, within a constant time, the protocol behaves normally.

The correctness proof and its validation show that the proposed solution provides a trade-off between the uncertainty time (in the order of milliseconds) and the occurrence of such events. In other words, if the round length is short and, consequently, a short uncertainty time, the vehicles will experience low service level more frequently, and if round length is longer, the vehicles will experience low service level less frequently. However, the longer the round length, the longer the time that vehicles spend on disagreements and therefore, the risk of having accidents increases.

This paper also discusses a safety-critical application that facilitates cooperation using the proposed protocol. We assume a baseline adaptive cruise control (ACC) that does not require communication. Then, we extend it to a cooperative one that attains higher vehicle performance, but relies on higher confidence level about the position and speed of nearby vehicles. We explain how the protocol can provide a timed and distributed mechanism for facilitating decisions about when the vehicles should plan their trajectories according to the baseline application and when according to the extended one.

1.1 Related work

The distributed consensus problem considers the selection of a single value from a set of values proposed by the members. The solution is required to terminate within a bounded time by which all system components have decided on a common value. There is a number of impossibility results related to distributed consensus in general (see [5, 6, 7]). In [4], the author shows that the presence of communication failures makes impossible to deterministically reach consensus (Theorem 5.1) and any $r$-round algorithm has probability of disagreement of at least $\frac{1}{r+1}$ (Theorem 5.5). This implies that there is no guarantees that vehicles can reach consensus on bounded time since vehicle-to-vehicle communication is prone to failures. Moreover, when the communication failures are too frequent or too severe, vehicles can fail to reach consensus for an unbounded number of consecutive times. We therefore abandon consensus-based decision algorithms, and prefer to focus on solutions that offer early fall-back strategies against the risk of having disagreement for more than a constant number of rounds.

The existing literature on consensus algorithms with real-time requirements does consider processor failures. However, it often assumes timed and reliable communication. For example, in [8] the authors give an algorithm that reaches agreement in the worst case in time that is sublinear in the number of processors and maximum message delay. In [9], the authors provide a time optimal consensus algorithm that reaches consensus in time $O(D(f+1))$ in the worst case where $D$ is the maximum message delay and $f$ the maximum number of processors that can crash. In this paper, we do not assume reliable communication. Thus, message drops can occur independently among processors at any time.
Group communication systems [10] treat a group of participants as a single communication endpoint. The group membership service [11, 12] monitors the set of recently live and connected participating system components whereas the multicast service delivers messages to that group under some delivery guarantees, such as delivery acknowledgment. In this paper we assume the existence of a membership service and a best-effort (single round solution) dissemination (multicast) protocol that has no delivery acknowledgment.

There exists literature on adaptive cruise control [13, 14] as well as vehicle platooning [15, 16]. In [17], the author considers vehicle platooning and lane merging, and bases his construction on distributed high level communication primitives. We consider a different failure model for which there is no deterministic implementation for these communication primitives.

The studied problem is motivated by the KARYON project [3]. The KARYON project aims to provide a predictable and safe coordination of intelligent vehicles that interact in inherently uncertain environments. It proposes the use of a safety kernel that enforces the service level that the vehicle can safely operate. A cooperative service level can ensure that vehicles follow the same performance level. In this paper, we study a communication protocol that implements the KARYON’s cooperative service level evaluator. In [2], we present the architecture that considers the interactions between the safety kernel, a local dynamic map and the cooperative service level evaluator. Unlike the earlier abstract presentation of the cooperative service level evaluator, this paper provides in detail, the design and analysis of the communication protocol.

1.2 Our contribution

We study an elegant solution for cooperative vehicular systems that have to deal with communication uncertainties. We base the solution on a communication protocol that, we believe, can be well understood by designers of safety-critical, automated and cyber-physical systems. We explain how the designers of fault-tolerant cooperative applications can use this solution to deal with communication failures.

We consider cooperative applications that must periodically decide on a shared values $s$. Since the consensus problem cannot be deterministically solved in the presence of communication failures, the system is doomed to disagree on the value of $s$. We bound the period in which the vehicles can be unaware of such disagreements with respect to $s$. We prove and validate that this bound is no more than one communication round. We also study the percentage of time during which the system avoids disagreement on $s$ using ns-3 simulations.

We exemplify how the proposed solution helps to guarantee the safety. We consider vehicles that operate on a cooperative operational mode as long as they are aware that all the nearby vehicles are also in the same mode. However, if at least one vehicle is suspecting that another vehicle is not, all vehicles switch to the autonomous operational mode to maintain high safety levels.

1.3 Document structure

We list our assumptions and define the problem statement (Section 2), before providing the timed protocol for cooperation with disagreement correction (Section 3) and its correctness proof (Section 4). As protocol validation study, we consider computer simulation (Section 5). The cooperative vehicular application is presented in Section 6.

2 System Settings

We consider a message passing system that includes a set members of $n$ communicating prone-resilient vehicles. We refer to the vehicles with id $i$ as $p_i$. We assume that all vehicles have access to a common
global clock, e.g., from GPS systems, by calling the function clock() which provides the global time with sub-microsecond offsets \[18\]. Hence, we assume that the maximum time difference along vehicles is at most $\text{syncBound}$. We consider that the system runs on top of a timed and fault-tolerant, yet unreliable, dissemination protocol, such as \[19\,20\], that uses $\text{gossipSend}_i(m)$ to broadcast message $m$ from vehicle $p_i \in \text{members}$ to all vehicles in $\text{members}$. We assume that end-to-end message delay is at most $\text{messageDelay}$ time. Thus, messages are either delivered within $\text{messageDelay}$ time or omitted. The constant $\text{messageDelay}$ depends on distinct factors such as the MAC protocol that is used, vehicle speed, interference, etc. For example, This bound can be set to 100ms or less using, for example, dedicated short-range communications (DSRC) \[21\].

Vehicle $p_j$ receives $m$ from $i$ by raising the event $\text{gossipReceive}_j(i,m)$. We consider a fully connected network topology. However, the network can arbitrarily decide to omit messages, but not to delay them for more than $\text{messageDelay}$ time. These assumptions allow the protocol to run in a synchronous round based fashion. We consider rounds of time $\text{roundLength}$ where $\text{roundLength} \geq \text{messageDelay} + 2\times \text{syncBound}$.

Every vehicle $p_i$ executes a program that is a sequence of (atomic) steps. An input event can be either the receipt of a message or a periodic timer going off triggering $p_i$ to start a new iteration of the do forever loop.

We define the uncertainty time as the period of time that vehicles can disagree. We say that there was a communication failure at round $r$ if there exists a vehicle that has not received the messages from all vehicles during round $r$.

### 2.1 The task

The system’s task is a set of executions in which requirements 1 to 3 hold.

**Requirement 1** (Certainty Time). No two vehicles use different values during the certainty time.

**Requirement 2** (Disagreement Correction). All vehicles use the default return value while communication failures occur and for a bounded time when communication reestablishes.

**Requirement 3** (Bounded Uncertainty Time). The uncertainty time is bounded.

We show that any execution of our proposed solution fulfill the task properties. More specifically we show in Section 3 the following theorem.

**Theorem 1.** Protocol \[\text{Protocol}1\] fulfills Properties \[1\,2\] and \[3\] where the uncertainty time is bounded by one round. Moreover, if vehicles do no experience communication failures, the disagreement correction holds for at most one round.

### 3 Disagreement Correction Protocol

We present the communication protocol in which the participants exchange messages until a deadline (Algorithm \[1\]). These messages can include information, for example, about nearby vehicles as well as the confidences that each vehicle has about its information. Once everybody receives the needed information from each other, the participants can locally and deterministically decide on their actions. In case of a communication failure, each participant that experiences a failure imposes the default return value for one round.

Each vehicle $p_i \in \text{members}$ executes the protocol that appears in Algorithm \[1\]. It uses a do forever loop for implementing a round base solution. It accesses the global clock (line \[20\]) and checks whether the vehicle
should send the information of the current round (line 21). A vehicle starts sending messages at syncBound time from the beginning of each round and syncBound + messageDelay before of the end of each round using the gossipSend() interface (Line 22). Recall that syncBound is the maximum time difference over the vehicles and messageDelay is the longest time that a message can live in the network. Next, it tests whether the current round number myRound points to the current round in time (line 23). A new round starts when myClock \div roundLength is greater than myRound.

At the beginning of every round, the protocol first keeps a copy of the collected data and the received information, and updates the round counter, as well as nullifying data and ack (line 24). Then it tests whether it has received all the needed information for the previous round (line 25). If a communication failure occurred in the previous round, the protocol sets the data to be sent to the default return value \perp (line 26) and writes to the control loop interface the received information as well as the default return value \perp (line 27). However, if all messages of previous round arrived on time, the system reads the application information using readState() interface and writes to controlLoop() interface the received information as well as the value that the deterministic function decide() returns (line 30).

The proposed protocol interfaces with the gossip (dissemination) protocol by sending messages (gossipSend()) periodically, and receiving them (gossipReceive()). The protocol locally stores the arriving information from \( p_j \in members \) on each round in data\([k]\) and waits for the end of the round before it finishes to accumulate all arriving information. More specifically, for each message that is reported with the same round, the protocol stores the data from \( p_k \) and sets the acknowledgment variable to true if the message comes directly from \( p_j \), \( (k = j) \), or transitively from \( p_j \) without considering its own values (ack\([k]\) and \( k \neq i \)).

## 4 Correctness

In this section we prove that protocol depicted in Algorithm 1 follows the three requirement tasks. First we introduce some definitions that we use during the proofs. Throughout this section, we consider an execution of Protocol 1.

**Definition 1** (stable Communication Period). A stable communication period \( X[r_1, r_2] \) is the period of \( r_2 - r_1 \) rounds where vehicles do not experience communication failures. Otherwise, it is called an unstable communication period, denoted by \( Y[r_1, r_2] \).

We say that a stable communication period \( X[r_1, r_2] \) is maximal if the rounds \( r_1 - 1 \) and \( r_2 + 1 \) are in unstable communication periods. Analogously, we define a maximal unstable communication period \( Y[r_1, r_2] \).

**Lemma 1.** Let \( Y[r_1, r_2] \) be any maximal unstable communication period followed by a maximal stable communication period \( X[r_2 + 1, r_3] \). The following three statements hold.

1. **Bounded Uncertainty Time.** Vehicles may have disagreements at round \( r_1 + 1 \).
2. **Disagreement Correction.** All vehicles use the default return value during \([r_1 + 2, r_2 + 1]\).
3. **Certainty Time.** Vehicles use the same value during \([r_1 + 2, r_3 + 1]\).

**Proof.** Let \( s_i(r) \) be the set of messages that vehicle \( p_i \) receives from all the vehicles \( p_j \in P \), either directly or indirectly, and that \( p_j \) has sent during round \( r \). Observe that each vehicle decides the value to be used on round \( r + 1 \) based on the received information at round \( r \) (lines 27 and 30). We claim that \( s_i(r) = s_i(r) \) for \( \forall p_k, p_l \in P \) and \( \forall r \in [x_2 + 1, x_3] \). Note that this implies that no two vehicles use different values when processing round \( r \), because vehicle \( p_l \) determines its output value according to the deterministic function decide(s_i).
Algorithm 1 Timed Protocol for Cooperation with Disagreement Correction (code for \( p_i \))

1. **Constant:** \( \text{members} = \{ p_1, p_2, \ldots, p_k \} \): The system vehicles.

2. **Constant:** \( \bot \): Denotes a void (initialized) entry, as well as the default return value.

3. **Constant:** \( \text{syncBound} \): The maximum time difference among vehicle clocks.

4. **Constant:** \( \text{maximumDelay} \): The maximum time that a message time can live in the network.

5. **Constant:** \( \text{roundLength} > 2 \cdot \text{syncBound} + \text{maximumDelay} \): The length of a round.

6. **Variable:** \( \text{myRound} \leftarrow 0 \): Current communication round.

7. **Variable:** \( \text{data}[n] = \{ \ldots \} \): Application data where \( \text{data}[k] \) is the data received at round \( \text{myRound} \) from member \( p_k \).

8. **Variable:** \( \text{ack}[n] = \{ \text{false, \ldots} \} \): Acknowledge for data reception where \( \text{ack}[k] \) is true if \( p_i \) has received (directly or indirectly) the message from \( p_k \) of the current round.

9. **Interface** \( \text{gossipSend}() \): Disseminate information to the system members.

10. **Interface** \( \text{gossipReceive}() \): Dispatch arriving messages.

11. **Interface** \( \text{readState}() \): Return a datum to be sent.

12. **Interface** \( \text{controlLoop}() \): Write decided output where \( \bot \) is the default return value.

13. **Interface** \( \text{decide}(s) \): Deterministically determines an item from \( s \). We assume that whenever \( \bot \in s \), then \( \bot = \text{decide}(s) \).

14. **Upon** \( \text{brcv}(j, \text{< round } j, \text{data } j, \text{ack } j >) \)

15. if \( (\text{myRound} = \text{round } j) \) then

16. for all \( p_k \in \text{members} \) do

17. if \( (\text{ack}[k] \text{ and } i \neq k) \text{ or } (k = j) \) then

18. \( (\text{data}[k], \text{ack}[k]) \leftarrow (\text{data}_j[k], \text{true}) \)

19. **loop**

20. let \( \text{myClock} = \text{clock}(\) \)

21. if \( \text{myClock} \in \{ \text{roundLength} \cdot \text{myRound} + \text{syncBound}, \text{roundLength} \cdot (\text{myRound} + 1) - (\text{syncBound} + \text{maximumDelay}) \} \) then

22. \( \text{gossipSend}(i, \text{< myRound, data, ack >}) \)

23. if \( \text{myRound} < \text{myClock} \div \text{roundLength} \) then

24. \( (s, i, \text{myRound}, \text{data}, \text{ack}[k]) \leftarrow (\text{data}, \text{ack}, \text{myClock} \div \text{roundLength}, \{ \bot, \ldots \}, k = i : \forall p_k \in \text{members}) \)

25. if \( \text{false} \in \{ r[k] : p_k \in \text{members} \} \) then

26. \( \text{data}[i] \leftarrow \bot \)

27. \( \text{controlLoop}(s, \bot) \)

28. **else**

29. \( \text{data}[i] \leftarrow \text{readState}() \)

30. \( \text{controlLoop}(s, \text{decide}(s)) \)

---

**Figure 2:** Maximal unstable communication period \( Y[r_1, r_2] \) followed by a maximal stable communication period \( X[r_2 + 1, r_3] \).

**Claim 1.** \( s_k(r) = s_i(r) \) for \( \forall p_k, p_i \in P \) where \( r \in [x_2 + 1, x_3] \).

**Proof of the Claim.** First we show that each vehicle maintains consistent its own information over each round. Observe that lines between \([23]\) and \([30]\) are executed once during round \( \text{myRound} \) since \( \text{myRound} \) is set to \( \text{clock}() \div \text{roundLength} \) and \( \text{clock}() \) always returns larger values. Therefore, each vehicle \( p_i \) loads its message on the register \( \text{data}[i] \) once during \( \text{myRound} \). Thus, assume that vehicle \( v_i \) overwrites its \( \text{data}[i] \)
when receiving a message from vehicle \( v_j \). Since the condition ensures that it loads \( data[k] \) only if either \( i \neq k \) or \( k = j \), we conclude that \( i = j = k \). Thus, \( data[i] \) is consistent on \( p_i \) during round \( myRound \).

We say that a message \( m_k \) is sent transitively, if \( p_j \) receives \( m_k \) from \( p_j \) where \( j \neq k \). We show that the message transitivity maintains the consistency of the messages during a stable communication period. We argue by contradiction. Assume that there are two messages, \( m_i \in s_k \) and \( m'_i \in s_j \) such that \( m_i \neq m'_i \). Consider the first time that \( m_i, m'_i \) were sent. Observe that \( p_i \) sent the two messages. A contradiction since \( p_i \) maintains consistent its own information over each round.

The claim follows by showing that at the end of the current round \( myRound \), it holds that \( s_k(myRound) = s_l(myRound) \). Indeed, since messages of each round are sent \( (syncBound + maximumDelay) \) time units before the end of \( myRound \) and after \( syncBound \) time units after the beginning of \( myRound \), vehicles receive messages only from the current round. Recall that \( syncBound \) is the maximum difference time among vehicle clocks and \( maximumDelay \) is the maximum time that a message can live in the network.

(1) **Bounded Uncertainty Time.** Consider round \( r_1 \). Since \( Y[r_1,r_2] \) is an unstable communication period, there exists a vehicle \( p_i \) that did not receive the message from all vehicles, i.e., \( \perp \in \text{ack} \). Observe that vehicle \( p_j \) is unaware that \( p_i \) had experienced a communication failure during round \( r_1 \). Let us assume that \( p_j \) did not experience any communication failure. Therefore, \( p_j \) uses the deterministic value that \( \text{decide()} \) returns on round \( r_1 + 1 \). However, \( p_i \) imposes the default return value (line [27]) since it has experienced a communication failure. Thus, during round \( r_1 + 1 \), \( p_i \) sends the default return value by setting \( data[i] \) to \( \perp \) and uses it (lines [26] and [27], respectively). Therefore, the first default return value of \( p_j \) arrives (if a vehicle does not miss it) along round \( r_1 + 1 \). Thus, during round \( r_1 + 1 \), \( p_j \) uses a distinct value than \( p_i \).

(2) **Disagreement Correction.** We show that all vehicles use the default return value in round \( r \in [r_1 + 2, r_2 + 1] \). It is sufficient to show that there exists at least one default return value in \( s_j(r) \) in each round \( r \in [r_1 + 1, r_2] \). Assume that at round \( r \in [r_1, r_2] \), some vehicle \( p_k \) experienced a communication failure. Therefore, at round \( r + 1 \) all other vehicles either receive the default value of \( p_k \) or miss the message from \( p_k \). Thus, from definition of the function \( \text{decide()} \) (line [13]) and the fact that each vehicle writes the default return value if it experiences a communication failure, all vehicles use the same value (default return value) in each round \( r \in [r_1 + 2, r_2 + 1] \) (lines [26] and [30]).

(3) **Certainty Time.** We show that during \( [r_1 + 2, r_3 + 1] \) all vehicles use the same values. Indeed, from the previous point, every vehicle uses the default return value in every round \( r \in [r_1 + 2, r_2 + 1] \). It remains to show that they use the same value in each round \( r \in [r_2 + 2, r_3 + 1] \). From the claim, \( s_i(r) = s_k(r) \) for each pair \( p_i, p_k \in P \) during \( [r_1, r_3] \) since all vehicles received the information from each other vehicle. The lemma follows since vehicles decide the value to be used on round \( r + 1 \) based on the received information at round \( r \) (lines [27] and [30] using the deterministic function \( \text{decide()} \).

\[ \text{Theorem 2} \] it follows directly from Lemma 1.

5 Evaluation

We consider a cooperative system that has two service levels where the lowest one is the default service level to which the system falls-back to in the presence of communication failures. This, for example, can be a vehicular system in which the cooperative service level is the highest, and the autonomous service level is the lowest (default) one. Since we focus on communication failures, the experiments assume that every system component can always support the highest service level, and thus read input \( \text{readState} \) always returns the highest service level. We use computer simulation to validate the protocol as well as its efficiency.

8
For the efficiency, we consider the *reliability* performance measure which we define as the percentage of communication rounds during which the protocol allows the system to run at its highest service level. First, we validate that the disagreement period is of at most one round and next the reliability of the protocol.

We simulate the protocol using ns-3. We choose IEEE 802.11p as the communication channel with a log-distance path loss model and Nakagami fading channel model. Since DSRC technologies support end-to-end message delay of less than 100ms [21], we fix the message delay to 100ms. We consider a synchrony bound of 5ms, say, using GPS [18] or a distributed clock-synchronization protocol. We implement a straightforward gossip protocol in which every node retransmits message every 50ms.

We validate that the disagreement period is of at most one round. We plot in Figure 3 the decision that 4 vehicles took independently during 25 rounds using the protocol under frequent communication failures. We set the round length to 160ms so that messages can be transmitted twice in each round. Observe that at round 20 vehicles 1 and 2 reduce the service level due to a communication failure, but vehicles 3 and 4 still continue in the highest level of service. However, at round 21, they lower their service level. Although vehicles do not operate on distinct service levels for more than one round, the service level of some vehicles may be oscillating. We can reduce this effect by increasing the round length. However, the uncertainty time also increases.

![Figure 3: Vehicle behavior under frequent communication failures. The plot shows the decision that four vehicles took among two service levels during 25 rounds using the protocol.](image)

Note the trade-off between the upper bound on the disagreement period, which is one communication
round, and the success rate of the gossip protocol, which decreases as the round length becomes shorter. The type of gossip protocol as well as the number of system components also influences this success rate. We use computer simulation to study how these trade-offs work together and present the reliability.

We consider three round lengths between 160ms and 360ms with intervals of 100ms so that vehicles can transmit 2, 4, and 6 messages in each round, respectively. We variate the number of vehicles between two and eight. The reliability of the system is plotted in Figure 4. We run each experiment for 360 simulation seconds. During the simulations, we observe a packet drop average of 0.1436347. The packet drop rate per number of vehicles is presented in Table 1. Further, the percentage of time that all vehicles agree on the highest service level is greater than 98% with round lengths of at least 260ms with at least four vehicle. Observe that the reliability is higher with more vehicles than with less. This is because of the transitivity property.

Figure 4: The percentage of time that all vehicles agree on the highest service level (number of vehicles vs the round length in milliseconds).

6 Cooperative Vehicular Application

Autonomous vehicles have great capabilities to safely respond to unexpected events and keep short headways. However, keeping short inter-vehicle distances without considering the nearby vehicles can result in hazardous situations. For example, rear-end crash as well as near-crash events usually involve an action of the lead vehicle [22]. Cooperative vehicular systems have the potential to mitigate these events and
Number of Vehicles | Packet Drop Rate  
---|---  
2 | 0.1605357  
3 | 0.1436347  
4 | 0.159418  
5 | 0.141237  
6 | 0.1426173  
7 | 0.138037  
8 | 0.1713623  

Table 1: Packet Drop Rate.

improve the vehicle performance by exchanging information periodically as well as their confidence level (validity) about their own information. However, due to communication failures, vehicles may have inconsistent information and, therefore, low confidence level. We present a cooperative vehicular application that exchanges periodically information and uses Algorithm 1 for dealing with communication failures. Our design demonstrates that, even though the presence of communication failures can lead to disagreement about joint validity values, this only happens for a period of at most one round. We exemplify our approach using the adaptive cruise control and vehicle platooning in an environment with communication uncertainties. In this cooperative system, the vehicles decide which application to use according to the system service level $s$, which Algorithm 1 decides on its value.

We do not aim to design a new vehicular system, but exemplify how the proposed solution helps to guarantee the safety in cooperative vehicular applications in environment with communication uncertainties. In our approach, while the vehicles are aware that nearby vehicles have a high level of certainty, they perform a fully cooperative operational mode to improve their performance. However, if they cannot deterministically determine it, they switch to the autonomous operational mode to maintain high safety levels.

Adaptive Cruise Control and Vehicular Platooning adjust the vehicle speed so that they keep a predefined and safe headway. Adaptive cruise control (ACC) sets the headway according to the vehicles in its direct line-of-sight. In other words, this application relies merely on on-board sensors. Vehicular platooning applications (or cooperative adaptive cruise control applications), however, do exchange information among vehicles and jointly aim at reducing air friction and energy consumption. They achieve such cooperative objectives by keeping shorter inter-vehicle distance than the autonomous ACC application. We show how to use the protocol solution for cooperative vehicle platooning with ACC as a base-line application.

In vehicular platooning, vehicles exchange the vectorial variable $s$ that contains their localization, speed and the highest level service that they support. The service level provides known bounds on the information error (see table 2) as well as the operational parameters such as headway and acceleration bounds where unbounded error means that the vehicle cannot determine it, for example because a component related to the parameter is in a faulty state (possible broken). We notice that our list of three service levels can be extended to include more levels. We assume that vehicles have the capability to determine the errors with high level of confidence. We also assume that vehicles can determine the relative position of the vehicle ahead using on-board sensors within a known error.

In Algorithm 2, we execute an instance of Protocol 1 and implement the interface functions readState, controlLoop and decide. The function readState returns the local level of service of $P_i$ (see Table 2) as well as the operational information (localization, heading, speed, etc.). The function decide returns the minimum local level of service in the data structure $s$ so that all vehicles can meet the constrains. The main functionality is implemented in controlLoop function. It uses the information of all the vehicle in data to determine the speed/acceleration for the next round according to the cooperative level of service using the parameters in
| Level of Service | Loc. Err. ($P_e$) | Speed Error ($S_e$) |
|------------------|------------------|--------------------|
| High             | $P_e \leq L$     | $S_e \leq S$       |
| Medium           | Unbounded $P_e$  | $S_e \leq S$       |
| Low              | Unbounded $P_e$  | Unbounded $S_e$    |

Table 2: $L$ and $S$ are constant values known by all participants.

| Level of Service | Headway | Acc Bound |
|------------------|---------|-----------|
| High             | $H_1$   | $\mathcal{A}_1$ |
| Medium           | $H_2$   | $\mathcal{A}_2$ |
| Low              | $H_3$   | $\mathcal{A}_3$ |

Table 3: $H_{i}$ are constant values and $\mathcal{A}_{i}$ are constant acceleration bounds such that $H_1 < H_2 < H_3$ and $\mathcal{A}_1 \subset \mathcal{A}_2 \subset \mathcal{A}_3$.

**Algorithm 2** Cooperative vehicle platooning with ACC as a base-line application (code for vehicle $p_i \in \text{members}$).

1: Execute Protocol$^1$
2: function $\text{decide}(s)$
3: return $\min_{j \in s}(s[j].localLoS)$
4: function $\text{readState}()$
5: Let $V$ and $localLoS$ be the operation information and maximum local service level that it supports, respectively, of $p_i$
6: return ($localLoS$, $V$)
7: function $\text{controlLoop}(data, LoS)$
8: if $p_i$ is the platoon leader then
9: Use $data$ and acceleration bounds provided in Table$^2$ according to $LoS$ to maintain the cruise speed if possible
10: else
11: Use $data$ and acceleration bounds provided in Table$^2$ according to $LoS$ to maintain the headway given in Table$^3$

Table$^3$: Notice that in the autonomous application, vehicles use only their own sensory information. We assume that each operation mode is proven to be safe provided that the information meets the requirements, i.e., the errors are within the bounds given in Table$^3$. For the worst case scenario the behavior of the platoon can be influenced by vehicles that are not part of the platoon. This is because some events can cause cascade effects if they occur during the communication failures. We observe that switching from the highest service level to the lowest service level is a critical time.

The safety of Algorithm$^2$ depends directly on the mechanical constraints and the parameters’ election. From the previous section, it is reasonable to consider rounds of length at least 260 milliseconds. Thus, the headway can be determined from the round length and the error bounds on the information. We observe that Algorithm$^2$ can reduce the risk of having accidents by enforcing the vehicles to operate in a common service level that has been proven to be safe according to the information quality.
7 Example

In this section we consider an example with two scenarios to exemplify the proposed solutions. We consider a worst case scenario with three vehicles: (1) vehicle platoon where vehicles apply a back-off strategy while communication failures but without considering the proposed solution and (2) vehicle platoon running Algorithm 2.

Let \( v_1, v_2, v_3 \) be the vehicles such that \( v_1 \) is leading the platoon followed by \( v_2 \) and \( v_3 \) as depicted in Figure 5. Assume that vehicles are driving on platooning with operational parameters given by the medium service level in Table 3. Therefore, they keep a headway of \( H_2 \). Suppose that at time \( t \), \( v_2 \) starts losing the messages from \( v_3 \) for \( \delta \) time. Further, assume that at time \( t + \epsilon \), \( v_2 \) loses the messages from \( v_1 \) for \( \delta - \epsilon \) time and at the same time \( v_1 \) requires to decelerate due to an obstacle, for example a pedestrian. Let us assume that \( \epsilon \) and \( \delta \) are at least two times the round length.

**Platooning with back-off strategy that does not include the proposed solutions.** Since \( v_2 \) does not receive the messages from \( v_3 \) during \([t, t + \delta]\), it is unaware whether \( v_3 \) continues operating on platooning. Thus, \( v_2 \) continues operating on platooning and assumes that it is the last vehicle in it. At time \( t + \epsilon \), \( v_2 \) starts losing messages from \( v_1 \) and consequently switches to the back-off strategy in the next round. However, since \( v_1 \) requires to brake, \( v_2 \) uses the acceleration bounds in \( A_3 \). But \( v_3 \) continues operating on platooning during \([t, t + \delta]\), since it is unaware that \( v_2 \) is not receiving messages from \( v_1 \) and \( v_3 \). By definition, the system is not safe during \([t + \epsilon, t + \delta]\), since the platoon has only been proved to be safe when the headway is at most \( H_2 \) and acceleration bounds are in \( A_2 \).

**Platooning using Algorithm 2.** From the algorithm property that the uncertainty does not hold for more than one round, \( v_1 \) and \( v_3 \) will be aware that at least one vehicle has a communication failure in the next round. Therefore, all switch to the lower service level and start opening space to keep a headway of \( H_3 \). Thus, at time \( t + \epsilon \) they have larger inter-vehicle distances which reduces the cascade effects. Observe that for an \( \epsilon \) less than two round lengths, the problem also occurs in this approach. Indeed, every cooperative vehicular application that relies on communication suffers from this problem. However, we believe that our approach minimizes the effects.
8 Conclusion

We have proposed an efficient protocol that can be used in safety-critical cooperative vehicular applications that have to deal with communication uncertainties. The protocol guarantees that all vehicles will not be exposed for more than a constant time to risks that are due to communication failures. We demonstrate correctness, evaluate performance and validate our results via ns-3 simulations. We also showed how vehicular platooning can use the protocol for maintaining system safety.

The proposed solution can be also extended to other cooperative vehicular applications, such as intersection crossing, coordinated lane change. Moreover, we have considered the simplest multi-hop communication primitive, i.e., gossip with constant retransmissions. However, that communication primitive can be substitute with a gossip protocol that facilitate greater degree of fault-tolerance and better performance. This work opens the door for the algorithmic design and safety analysis of many cooperative applications that use different high-level communication primitives.

References

[1] R. Kianfar, P. Falcone, and J. Fredriksson, “Safety verification of automated driving systems,” Intelligent Transportation Systems Magazine, IEEE, vol. 5, no. 4, pp. 73–86, winter 2013.

[2] A. Casimiro, J. Rufino, R. C. Pinto, E. Vial, E. M. Schiller, O. Morales-Ponce, and T. Petig, “A kernel-based architecture for safe cooperative vehicular functions,” in 9th IEEE International Symposium on Industrial Embedded Systems (SIES’14), 2014.

[3] A. Casimiro, J. Kaiser, J. Karlsson, E. M. Schiller, P. Tsigas, P. Costa, J. Parizi, R. Johansson, and R. Librino, “Brief announcement: Karyon: Towards safety kernels for cooperative vehicular systems,” in SSS, ser. LNCS, A. W. Richa and C. Scheideler, Eds., vol. 7596. Springer, 2012, pp. 232–235.

[4] N. A. Lynch, Distributed Algorithms. Morgan Kaufmann, 1996.

[5] A. Fekete, N. A. Lynch, Y. Mansour, and J. Spinelli, “The impossibility of implementing reliable communication in the face of crashes,” J. ACM, vol. 40, no. 5, pp. 1087–1107, 1993.

[6] M. J. Fischer, N. A. Lynch, and M. Paterson, “Impossibility of distributed consensus with one faulty process,” J. ACM, vol. 32, no. 2, pp. 374–382, 1985.
[7] M. J. Fischer, N. A. Lynch, and M. Merritt, “Easy impossibility proofs for distributed consensus problems,” *Distributed Computing*, vol. 1, no. 1, pp. 26–39, 1986.

[8] J.-F. Hermant and G. L. Lann, “Fast asynchronous uniform consensus in real-time distributed systems,” *IEEE Trans. Computers*, vol. 51, no. 8, pp. 931–944, 2002.

[9] M. K. Aguilera, G. L. Lann, and S. Toueg, “On the impact of fast failure detectors on real-time fault-tolerant systems,” in *DISC*, ser. Lecture Notes in Computer Science, D. Malkhi, Ed., vol. 2508. Springer, 2002, pp. 354–370.

[10] G. Chockler, I. Keidar, and R. Vitenberg, “Group communication specifications: a comprehensive study,” *ACM Comput. Surv.*, vol. 33, no. 4, pp. 427–469, 2001.

[11] B. Ducourthial, S. Khalfallah, and F. Petit, “Best-effort group service in dynamic networks,” in *SPAA*, F. Meyer auf der Heide and C. A. Phillips, Eds. ACM, 2010, pp. 233–242.

[12] M. Verma and D. Huang, “Segcom: Secure group communication in vanets,” in *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, Jan 2009, pp. 1–5.

[13] S. S. Stankovic, M. J. Stanojevic, and D. D. Siljak, “Decentralized overlapping control of a platoon of vehicles,” *Control Systems Technology, IEEE Transactions on*, vol. 8, no. 5, pp. 816–832, 2000.

[14] Y. Zhang, B. Kosmatopoulos, P. A. Ioannou, and C. Chien, “Using front and back information for tight vehicle following maneuvers,” *Vehicular Technology, IEEE Transactions on*, vol. 48, no. 1, pp. 319–328, 1999.

[15] S. E. Shladover, “Longitudinal control of automotive vehicles in close-formation platoons,” *Advanced automotive technologies, 1989*, 1989.

[16] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W. B. Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, “Automated vehicle control developments in the path program,” *Vehicular Technology, IEEE Transactions on*, vol. 40, no. 1, pp. 114–130, 1991.

[17] G. L. Lann, “Cohorts and groups for safe and efficient autonomous driving on highways,” in *VNC*, O. Altintas, W. Chen, and G. J. Heijenk, Eds. IEEE, 2011, pp. 1–8.

[18] D. W. Allan and M. A. Weiss, *Accurate time and frequency transfer during common-view of a GPS satellite*. Electronic Industries Association, 1980.

[19] S. P. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[20] C. Georgiou, S. Gilbert, and D. R. Kowalski, “Meeting the deadline: on the complexity of fault-tolerant continuous gossip,” *Distributed Computing*, vol. 24, no. 5, pp. 223–244, 2011.

[21] C. V. S. C. Consortium *et al.*, “Vehicle safety communications project: task 3 final report: identify intelligent vehicle safety applications enabled by dsrc,” *National Highway Traffic Safety Administration, US Department of Transportation, Washington DC*, 2005.

[22] S. Lee, E. Llaneras, S. Klauer, and J. Sudweeks, “Analyses of rear-end crashes and near-crashes in the 100-car naturalistic driving study to support rear-signaling countermeasure development,” *DOT HS*, vol. 810, p. 846, 2007.