Multi-Technology Cooperative Driving: An Analysis Based on PLEXE

Michele Segata, *Member, IEEE*, Renato Lo Cigno, *Senior Member, IEEE*, Tobias Hardes, *Student Member, IEEE*, Julian Heinovski, *Student Member, IEEE*, Max Schettler, *Student Member, IEEE*, Bastian Bloessl, *Member, IEEE*, Christoph Sommer, *Member, IEEE*, and Falko Dressler, *Fellow, IEEE*

Abstract—Cooperative Driving requires ultra-reliable communications, and it is now clear that no single technology will ever be able to satisfy such stringent requirements, if only because active jamming can kill (almost) any wireless technology. Cooperative driving with multiple communication technologies which complement each other opens new spaces for research and development, but also poses several challenges. The work we present tackles the fallback and recovery mechanisms that the longitudinal controlling system of a platoon of vehicles can implement as a distributed system with multiple communication interfaces. We present a protocol and procedure to correctly compute the safe transition between different controlling algorithms, down to autonomous (or manual) driving when no communication is possible. To empower the study, we also develop a new version of PLEXE, which is an integral part of this contribution as the only Open Source, free simulation tool that enables the study of such systems with a modular approach, and that we deem offers the community the possibility of boosting research in this field. The results we present demonstrate the feasibility of safe fallback, but also highlight that such complex systems require careful design choices, as naïve approaches can lead to instabilities or even collisions, and that such design can only be done with appropriate in-silico experiments.

Index Terms—Cooperative driving, vehicular networking, vehicle-to-vehicle communication, platoon controller, simulation techniques

1 Introduction

The development and testing of cooperative driving applications heavily relies on simulations. Given the field of study, this is very rational, as one of the main drivers for vehicular networks is enhancing safety, and to do this we need to analyze dangerous situations. So, safety must be enforced at any step of the development and analytic models are too coarse to capture all details; on the other hand, real world Field Operational Tests (FOTs) are very demanding, both in terms of funding, man-power, and time.

The simulation of cooperative driving applications is very demanding, as it entails several orders of magnitude in time scale, and different modeling techniques depending on the subsystem considered, from transmissions to networking, from motion prediction to vehicles control and dynamics. Given the complexity and non-linearity of the entire system, the traditional *divide-et-impera* engineering approach that analyzes each sub-system separately with local performance evaluation and assumes that the entire system will work if all subsystems work, cannot be applied. The impact of communications on the safety of cooperative driving must be evaluated on this latter, not through communications metrics like packet losses or generic reliability.

As the communication requirements of cooperative driving are very strict and a single technology might not be able to meet them [1], it is envisioned that heterogeneous communications will be employed simultaneously [2]. In addition, the concept of cooperative driving is intertwined with the concept of automatic control. Classic mobility simulators such as SUMO [3] do not model control systems or vehicle dynamics, but rather try to mimic human driving behavior through car-following models such as the Intelligent Driver Model (IDM) [4] or Krauss [5]. For the evaluation of such systems, the possibility of understanding the effects of the network on the performance of the autonomous control system and the dynamics of the vehicle is fundamental to complete the puzzle.

The main goal of this work is the study of realistic strategies to safely introduce cooperative driving technologies on the road. To achieve this goal, it is mandatory to propose and analyze fallback strategies that can give the control of the vehicle back to the driver when automation fails (especially in case of communication failures), but to carry out this analysis it is necessary to have a proper simulation framework that allows credible and replicable performance evaluation. Thus, the contribution of this paper, is twofold. In the first part of the paper, we propose a fallback mechanism called Safe Autonomous Switchover Algorithm (SafeSwitch) that, in the reasonable assumption that future vehicles will have multiple communication technologies, empowers the safe change of platooning control algorithms as a function of the available communication technologies, and eventually the fall back to a simple radar-based Adaptive Cruise Control

- *M. Segata is with the Faculty of Computer Science, Free University of Bolzano, Italy, E-mail: michele.segata@unibz.it.*
- *R. Lo Cigno is with DII, University of Brescia, Italy, E-mail: renato.locigno@unibs.it.*
- *T. Hardes is with TU Dresden, Faculty of Computer Science, Germany and the Software Innovation Campus Paderborn (SICP), Paderborn University, Germany, E-mail: tobias.hardes@upb.de.*
- *J. Heinovski, M. Schettler, and F. Dressler are with the School for Electrical Engineering and Computer Science, TU Berlin, Germany, E-Mail: {heinovksi, schettler, dressler}@ccs-labs.org.*
- *C. Sommer is with TU Dresden, Faculty of Computer Science, Germany, E-mail: sommer@cms-labs.org.*
- *B. Bloessl is with the Secure Mobile Networking Lab, TU Darmstadt, Germany, E-mail: mail@bastibl.net.*

(ACC) system in case of complete communication failure. SafeSwitch is derived heuristically. In the second part of the paper, we describe a simulation tool that we developed to achieve the first goal, but whose capabilities and potential go well beyond what we exploit in this paper, thus making it a contribution on its own. In the paper we highlight the main features of the framework, while in the additional material we give a more in-depth description of its functionalities, how it can be tuned to flexibly address different problems, and discuss its scalability.

The simulation tool, presented in Section 4, is named PLEXE and builds on the prototype that we proposed in [6]. PLEXE provides the support for heterogeneous communication technologies, such as IEEE 802.11p, Visible Light Communication (VLC), and LTE-based Cellular V2X (C-V2X). One of its key features is the easy coupling with additional communication technology models, that makes PLEXE ready for the evaluation of future systems (e.g., 5G New Radio (NR), 5G mmWave, RADar-based COMmunication (RADCOM) [7], and beyond). In addition, it ships with control algorithms (namely Cooperative Adaptive Cruise Controllers (CACCs) [8]–[11]) taken from the literature and enables users to implement and evaluate additional ones very easily. Together with control algorithms, it implements vehicle dynamics models to include the effects of actuation delays, i.e., the control delay that incurs due to engine/brake dynamics and inertia, which heavily influence the stability and the performance of the system. Last but not least, PLEXE and all the required software is Open Source and freely available online[1]. PLEXE is modular, which greatly improves maintainability (including the usage of upgraded version of software modules) and eases the process of integrating new simulation modules.

Section 3 presents SafeSwitch that we envisage based on IEEE 802.11p, LTE, and VLC, and builds on the notion of safety through redundancy, designing a protocol that enables the use of more aggressive control technologies when more communication technologies are available and relaxes the control requirements as communications degrade, until no communications are available and vehicles must return to autonomous and non-cooperative driving. Section 4 describes the PLEXE framework, while Section 5 evaluates the performance of the system and its ability to guarantee safety. In the remainder of the paper we assume the reader familiar with the concepts of platooning, CACC, and string-stability. A concise primer is reported in the additional material.

2 Related Work

The works that influenced and are related to our contribution fall in three different areas of research: Heterogeneous communication technologies to enhance vehicular networks resilience; fallback mechanisms and protocols to modify cooperative driving behavior; and frameworks for the simulation of cooperative driving. The following subsections analyze the state of the art in these three fields.

2.1 Heterogeneous Communication Technologies

Vehicle to Everything (V2X) technologies abounds, but it is now clear that a single technology cannot meet the strict requirements of cooperative driving in terms of reliability and dependability. This self-evident observation has spawned research that tries to understand the benefits of mixing together multiple communication technologies in a single V2X network with sufficient resilience to support dependable operation and safe fallback in case of some technology failure.

One approach that has often been used is the combination of IEEE 802.11p and VLC. Ishihara et al. [12] developed a protocol that forwards all beacons (leader and follower) of a platoon via the IEEE 802.11p and VLC interface. Simulations show that the number of lost packets and the end-to-end delay can be strongly reduced while the platoon is exposed to a jamming attack. Segata et al. [13] investigated the benefits and drawbacks use of VLC for platoons compared to an IEEE 802.11p approach. Simulations were performed to evaluate a combination of VLC and IEEE 802.11p in a traffic jam situation. The results showed an increased delay for VLC, but a substantial improvement in terms of scalability to hundreds of communicating cars. However, the work uses simplistic assumptions regarding the physical layer and the VLC channel model.

Another approach, which also analyzes platooning under the influence of jamming attacks, was presented by Ucar et al. [14]. Using simulations, the authors showed a reduction regarding the impact of such an attack by exploiting the directionality and opacity of light.

Schettler et al. [15] analyzed different protocols that utilize IEEE 802.11p and VLC together in a freeway scenario. The proposed protocols were evaluated using simulations in challenging environments such as emergency braking maneuvers at high vehicle densities. Results showed that an additional VLC channel can reduce the IEEE 802.11p channel load and still maintain a safe distance of close to 5 m between vehicles in a platoon.

Hardes and Sommer [16] investigated communication strategies for platooning with VLC and IEEE 802.11p in an urban environment. They proposed different approaches that are combining IEEE 802.11p and VLC depending on the road topology. Simulation results showed that a situation-aware usage of the IEEE 802.11p channel can strongly reduce the number of lost beacons.

Besides the use of VLC, the use of 3GPP C-V2X is also being considered together with other technologies. Sybis et al. [17] propose a dynamic spectrum management mechanism that uses C-V2X and IEEE 802.11p based communication. Based on context-aware databases, sensing nodes, and spectrum allocation, they propose to transmit critical information using a primary radio, whereas other types of data are transmitted at an additional frequency band that is selected from the spectrum white space. The scheme has been evaluated with field tests. The use of C-V2X together with other technologies is still rare, but recent works indicate that to reach the dependability required by cooperative driving even this technology alone is not enough [18], [19].

Similarly, mmWave technology has recently attracted the attention of the research community. While so far there is little research regarding its use in heterogeneous V2X, initial works indicate that it will be a valuable network technology complementing the other alternatives [20]. For its additional capabilities and bandwidth mmWave can be utilized in cooperative driving applications similarly to VLC.

1. http://plexe.car2x.org

2.2 CACC Fallback Mechanisms

One of the biggest problems in cooperative driving is dealing with network failures. A communication failure, let this be due to congestion or to active attacks [21] might have serious consequences because vehicles would switch from their communication-enabled enhanced perception to the limited, sensor-only perception. If this happens, vehicles need to switch to a different control/coordination mode to maintain passengers' safety, which is not a trivial task.

Congestion or interference-based communication degradation can be handled even with a single communication technology with appropriate protocols. Segata et al. [22] propose a communication protocol that adapts the transmission rate to vehicle dynamics with the aim of minimizing network resource utilization to the minimum. While this approach proved successful, it cannot cope with jamming, attacks, or simply technology failure because it assumes the use of IEEE 802.11p only. Similarly, Giordano et al. [23] design a CACC using a joint control/network approach that allows setting theoretical bounds on safety, meaning that the inter-vehicle distance is guaranteed to be always greater than a certain bound provided some network constraints are met. Briefly speaking, such constraints are expressed as a maximum number of possible consecutive losses; in this case the use of multiple technologies reduces the probability of such an event, enabling the possibility to lower the inter-vehicle distance at no expenses of safety.

To ensure safety under unreliable network conditions some works perform the fallback to ACC using data obtained from the radar instead of data received through communication. One example is the work by Ploeg et al. [24]. The authors consider a predecessor-following CACC that obtains the intended acceleration (i.e., prior engine actuation) of the preceding vehicle through communication and uses it as an input for the control system, and they extend the CACC to fall back to the radar-estimated front vehicle acceleration when data is lost. The authors show that this approach dampens oscillations better than simply falling back to an ACC, but it still shows a slightly string-unstable behavior for small time headways. In addition, the approach can be employed with predecessor-following CACCs only. In leader- and predecessor-following CACCs obtaining leader's acceleration from the radar is not possible. Wu et al. [25] tackle the same problem, but in their solution, they predict missing data using a Kalman filter, which is suitable only under some regularity assumptions.

Simply switching to ACC might have a negative impact on safety, as shown by Tu et al. [26] and Qin et al. [27]. Tu et al. [26] show the safety consequences of abruptly switching from the California PATH CACC [8] to an ACC in case of communication impairments indicating, as a potential solution, to limit the size of the platoon, which is clearly undesirable. The authors highlight the need for further research in this topic, not only with respect to safe degradation from CACC to ACC but also on how to switch back to CACC once communication is reestablished.

2.3 Cooperative Driving Simulation

The analysis is limited to tools that could support our contribution, since we do no aim at a full survey on vehicular networking simulators.

The possibility of investigating cooperative driving as a whole requires either the integration or the bi-directional coupling of network and vehicular mobility simulators, enabling the mobility to influence the network behavior and, most important, vice-versa.

Examples of such frameworks include Veins [28], Eclipse MOSAIC [29], and iTetris [30]. What differs between these frameworks are the tools they employ for simulating communication and mobility. With respect to mobility they all resort to SUMO [3], even though MOSAIC can support different simulators as well. With respect to communications, instead, Veins relies on OMNeT++ [31], iTetris uses ns-3 [32], while MOSAIC supports both of them.

In platooning-related control theoretic works, the classic evaluation tool is MATLAB, due to its versatility when it comes to handling mathematical systems. The downside of MATLAB-based evaluation is the lack of proper network modeling. Some studies using MATLAB treat the network as a simple constant delay [9], [33], but this clearly does not faithfully depict the behavior of communication standards such as IEEE 802.11p [34], where the broadcast and period nature of Vehicle to Vehicle communication (V2V) messages results in either an immediate delivery (that is, within a few hundred microseconds) or in a loss.

Modern network simulators interfacing with mobility ones, indeed put a lot of effort into capturing all the subtleties of complex wireless networks. This includes Medium Access Control (MAC) algorithms, transmission technology details, and propagation models to account for reflections, building and vehicle shadowing. Conscious of such needs, we published a concept version of PLEXE [6] in 2014[2]. This version featured an ACC algorithm and a CACC (namely the California PATH CACC [8]), a simple vehicle dynamics model (a first order lag), and the realistic simulation of the IEEE 802.11p stack thanks to the integration with Veins [28]. Despite its prototypical nature, PLEXE has been very successful, empowering many studies beyond platoon control: from heterogeneous communication e.g., with VLC, to platoon maneuvering and formation control, to vehicular network security and congestion control, to new simulation tools [10], [13], [21], [35]–[39], and to many others we cannot mention here.

Another successful Open Source framework for platooning proposed in the literature is VENTOS [40]. VENTOS also offers access to ACC and CACC models but, just like our previously-presented prototype version of PLEXE, it is monolithically built on top of Veins and restricted to the evaluation of single technology solutions (namely IEEE 802.11p). Like PLEXE, VENTOS is highly appreciated by the community for its online availability, but it has not been actively maintained in the last two years[3].

We can also find other sophisticated tools which include platooning models, such as VISSIM [41][4]. In [42], VISSIM is coupled with ns-3 and MATLAB to study platooning systems. The main issue is clearly that VISSIM is a proprietary software and, in addition, only runs on Windows. Ramezani et al.

2. PLEXE is actively maintained. Version 3.1, the one presented in this paper, is actively used and fully integrated with SUMO.

3. VENTOS home page: https://maniam.github.io/VENTOS/

4. VISSIM home page: http://www.vissim.com/

[43] develop a sophisticated framework for the simulation and the analysis of truck platooning, with special focus on traffic flow analysis using the proprietary software Aimsun[5], thus sharing the same limitations as VISSIM.

3 Multi-technology CACC to ACC Fallback

We assume vehicles are equipped with multiple communication technologies, and use them constantly and simultaneously to send control information. Technologies are assumed to be independent of one another, i.e., each packet generated by a vehicle is sent through all interfaces. Our Safe Autonomous Switchover Algorithm (SafeSwitch), the fallback mechanism we propose, relies on the active monitoring of the state of the communication interfaces. The basic idea is to monitor the Packet Delivery Ratio (PDR) of each technology through the others, i.e., if we receive a frame via a certain technology but not via another we can automatically and immediately infer that the latter has lost information. Obviously also interface failure and jamming indicators can be used, but these local monitors cannot identify failures, errors, and other impairments happening at the other communication parties. We describe the details on how this is implemented in Section 3.1.

When the PDR monitoring procedure detects a failure, the system modifies the behavior of the vehicles. The actual action depends on the severity of the issue (e.g., failure of a single or multiple interfaces), but includes increasing the inter-vehicle distance and switching to a different control algorithm, eventually falling back to a standard ACC when all communications fail. The algorithm performs this in complete safety as the other backup technologies ensure the continuous flow of information while the fallback procedure is in place. It is worth mentioning that in here we only consider distributed control approaches and not centralized approaches such as Multi-access Edge Computing (MEC)-based solutions [44], because such approaches do not rely on V2V but on Vehicle to Infrastructure communication (V2I). Any other distributed CACC approach (including for example Model Predictive Control (MPC) [45]) can be considered by SafeSwitch. Regardless of the control mechanism, vehicles need to exchange data for the system to properly work and exploiting redundancy is a natural way to improve reliability. Section 3.2 describes the details of the mechanism, while Section 3.3 describes the gap control procedure that adapts the inter-vehicle distance before switching to a different control algorithm to ensure safe and stable transition.

3.1 Monitoring of Communication Technologies

The technologies we consider are IEEE 802.11p, VLC, and LTE C-V2X (Mode 3, under coverage, thus subject to handovers), but the approach is generic and can be modified to account for different, additional, or fewer interfaces. Details on the communication technologies are out of the scope of this contribution. The monitoring process exploits a ring buffer to store information about frames being received or lost, such as sequence number, arrival time, etc. The ring buffer is used to estimate the PDR, thus the methodology we propose is completely technology independent. With respect to the communication topology we assume broadcast links because of the control algorithms being considered (PATH and Ploeg). PATH requires all followers to receive information from the leader, while both require information of the preceding vehicle. This information includes acceleration and speed, while relative distance is obtained through a radar sensor. As VLC works only in Line of Sight (LOS) conditions, vehicles propagate leader messages towards the tail of the platoon. All vehicles send data with a rate of 10 Hz.

Listing 1 Ring buffer management.

```
1:  procedure INIT(nInterfaces, bufferSize)
2:      latestSeq ← −1
3:      B ← RingBuffer(nInterfaces, bufferSize)
4:  procedure ONFRAME(seq, intf)
5:      isNewFrame ← seq > latestSeq
6:      if isNewFrame then
7:          for s = latestSeq + 1 to seq do
8:              for all interfaces i do
9:                  B_{i,s} ← LOST
10:         B_{intf,seq} ← RECEIVED
11:         latestSeq ← seq
12:     else
13:         B_{intf,seq} ← RECEIVED
```

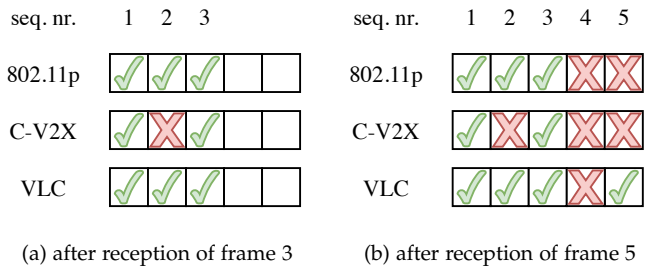
Listing 1 shows the pseudo-code that manages the ring buffer when receiving frames from a specific neighbor. Each vehicle runs an instance of the algorithm for every vehicle whose communications are monitored (e.g., two instances to monitor the platoon leader and the preceding vehicle). An interface is considered failed if either of the two monitors "declares" it as failed. The algorithm keeps track of the latest known frame sequence number, which is initialized to -1 assuming sequence numbers start from zero[6]. In addition, the ring buffer B keeps track of *bufferSize* frames for all the *nInterfaces* interfaces.

Each time a frame with sequence number *seq* is received from interface *intf*, the algorithm checks the value of the sequence number against the one of the most recently received frame. If it is greater, the received frame is a new one and the algorithm sets all the frames in between (if any) as lost for all interfaces. The received frame is set as received for the interface *intf* and lost for all the others, and the variable *latestSeq* is updated accordingly. If the sequence number of the received frame is smaller or equal than the latest known one, the frame is marked as received for the specific interface if *seq* falls inside the ring buffer, otherwise it is simply discarded as a duplicate frame. To compute the PDR for each technology it is sufficient to count the frames marked as "received" over the size of the ring buffer.

Figure 1 depicts the graphical evolution of the ring buffer for a better understanding. Figure 1a shows the ring buffer potential status after receiving frames up to frame number 3. In particular, the buffer indicates that frame number 2 has not been received from the C-V2X interface, while all others have been correctly received. Figure 1b shows how the state of the ring buffer changes when receiving frame number 5

5. Now a division of Siemens, Aimsun has also been extensively used beyond the specific paper we analyzed; see https://www.aimsun.com/

6. This is just for the sake of clarity in the paper. The actual implementation does not depend on this assumption.



(a) after reception of frame 3 (b) after reception of frame 5

Figure 1. Graphical evolution of the ring buffer monitoring frame receptions.

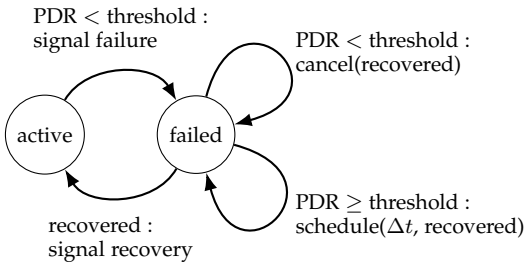


Figure 2. Finite state machine driving the PDR monitor of an interface.

from the VLC interface after frame number 3. The algorithm assumes that frame number 4 is lost on all interfaces while 5 is lost on the 802.11p and the C-V2X interfaces. This view is clearly temporary. If the vehicle then receives frame 5 from the 802.11p interface it will update the buffer accordingly.

To inform SafeSwitch about communication failures, we continuously monitor the PDR of each communication technology and, when this crosses a certain threshold, we notify the fallback application following the Finite State Machine (FSM) in Figure 2. The application is immediately notified as soon as the PDR falls below the threshold but, to signal its recovery, the PDR must be higher than the threshold for a certain period of time (Δt in the picture).

3.2 SafeSwitch Fallback System

SafeSwitch is designed through an FSM whose transitions are driven by network failures and recoveries signaled by the PDR monitor defined in Section 3.1 plus additional events. Figure 3 depicts a sample FSM that considers three simultaneous communication interfaces. The approach can easily be extended (or reduced) to additional (or to fewer) communication technologies. We assume each follower in the platoon to implement such an FSM and the leader to be controlled independently, e.g., driven by a classic ACC, as commonly assumed in the literature [46].

The FSM considers two different numbered states, i.e., F_i (follow) and G_i (gap control). F_i states indicate that a vehicle is currently in a stable following state, driven by a certain control algorithm. The index i accounts for the number of actively working communication interfaces in that state. Each state F_i is associated with a specific control algorithm C_i which, in turn, is associated with a desired inter vehicle gap $g_i = h_i \cdot v + d_i$, where h_i is the time headway, v is the cruising speed, and d_i is the stand-still distance.

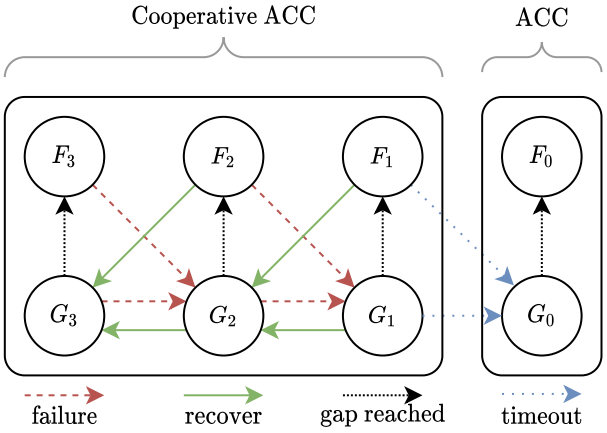


Figure 3. Finite state machine driving SafeSwitch.

G_i states describe an intermediate phase where vehicles switch to a different control mode (we call it *gap control*, Section 3.3) where vehicles increase (or decrease) their inter-vehicle distance before switching to a different control algorithm.

Assuming the vehicle is with N communication interfaces, the state transitions in Figure 3 fall into seven possible categories:

1) $F_i \rightarrow G_{i-1},\ i = 2, \ldots, N$: failure of a communication interface in a stable following state. The vehicle shifts from a follow state to a gap control state, where the inter-vehicle distance is progressively increased preparing the switch to a different control algorithm;
2) $G_i \rightarrow G_{i-1},\ i = 2, \ldots, N$: failure of a communication interface while the gap control algorithm is adapting the inter-vehicle distance. The vehicle simply changes the target distance accordingly. The gap control algorithm brings the vehicle to the new target distance;
3) $F_i \rightarrow G_{i+1},\ i = 1, \ldots, N-1$: recovery of a communication interface in a stable following state. The gap control algorithm reduces the inter-vehicle distance preparing the switch to a different control algorithm;
4) $G_i \rightarrow G_{i+1},\ i = 1, \ldots, N-1$: recovery of a communication interface while the gap control algorithm is adapting the inter-vehicle distance. The target distance is changed accordingly and the gap control algorithm runs on the new distance;
5) $G_i \rightarrow F_i,\ i = 0, \ldots, N$: the target distance is reached and the vehicle can safely switch to the new control algorithm, or to the ACC, or even to manual drive if foreseen by the fallback procedure;
6) $F_1 \rightarrow G_0$: failure of the last available communication interface. This event can only be measured by means of a timeout, because there is no information from other interfaces. The gap control algorithm brings the vehicle to the correct distance to activate a standard ACC or to commute to manual driving;
7) $G_1 \rightarrow G_0$: failure of the last available communication interface while performing a gap control procedure. Again, this event can only be measured by means of a timeout and the gap control algorithm brings the vehicle to the correct distance to activate a standard

ACC or to commute to manual driving. Note that states G_0 and F_0 are deemed an irrecoverable state of the system, requiring platoon formation to re-start, hence no transitions to higher states exist in the state machine.

There is an additional case we do not consider here, i.e., a complete failure of the electronics that cause all the communication interfaces to fail simultaneously. How to tackle such a catastrophic situation is more similar to handling engine or brake failure, and it is outside the scope of our work. A similar case occurs when interfaces fail almost immediately one after the other. In such a case, the protocol would still handle the failures by transitioning between G_i states. The behavior of the system would depend on the severity of the failure though, because a failure is triggered when the PDR falls below a certain threshold. If the technologies together can still provide a minimum number of necessary packets, the protocol can safely bring the vehicles to a different state.

Listing 2 formally describes the actions undertaken during the navigation of the FSM of Figure 3. The system considers the variable i, which indicates the number of currently active interfaces, initialized to N. Upon initialization (Line 1), the system sets the active state to F_N, which corresponds to driving using the control algorithm C_N (and thus an inter-vehicle gap g_N). In addition, the leader of the platoon is set to be the first vehicle (vehicle 0) and we initialize some variables that we require to temporarily change the leader in some cases. setControllerGap(h, d) is a generic function used to set the inter-vehicle distance to $h \cdot v + d$ for all CACCs.

When an interface fails (Line 9), the system starts the gap control to change the inter-vehicle distance to g_{i-1}, switches to state G_{i-1}, and decreases the number of active interfaces. In addition, if we are about to switch from a leader-following to a predecessor-following controller (e.g., from the PATH CACC to an ACC), the vehicle elects itself as temporary leader. This flag is included within each beacon periodically sent by the vehicle.

Upon recovery of an interface (Line 15), the system starts the gap control to increase the distance to g_{i+1}, switches to state G_{i+1} and sets to $i+1$ the number of active interfaces. It also immediately changes the active control system to C_{i+1} setting the target distance to the current one. This is required because when switching to a state with a higher number of interfaces we might be reducing the inter-vehicle distance down to a value that can cause the current control system to become string-unstable (e.g., switching from ACC to CACC).

When the gap control procedure ends bringing the distance up (or down) to the target (Line 27), the system disables the gap control and switches to the control algorithm C_i used within state F_i. In addition, when switching to a leader-following controller, the vehicle stops advertising itself as temporary leader.

The remaining case occurs when the last interface fails (Line 33), which we simply handle as a standard failure switching to G_0. We can improve safety by choosing C_1 and C_0 to be the same algorithm (in particular a classic ACC which requires no communication) or by having C_1 be a CACC using the same spacing policy as C_0. This increases the chances that vehicles are already traveling at the proper inter-vehicle distance and that the state switch is purely logical, only indicating a complete communication failure, but without consequences on safety.

Listing 2 SafeSwitch actions on events.

```
1:  procedure INIT
2:      i ← N
3:      state ← F_i
4:      activeController ← C_i
5:      setControllerGap(h_i, d_i)
6:      currentLeader ← 0
7:      tempLeaders ← {0}
8:      becomeTemporaryLeader ← false
9:  procedure FAILURE
10:     startGapControl(h_{i-1}, d_{i-1})
11:     state ← G_{i-1}
12:     if leaderBased(C_i) and not leaderBased(C_{i-1}) then
13:         becomeTemporaryLeader ← true
14:     i ← i − 1
15: procedure RECOVER
16:     if i = 0 then
17:         return
18:     if usesTimeHeadway(C_{i+1}) then
19:         h ← (g_i − d_{i+1})/v
20:         setControllerGap(h, d_{i+1})
21:     else
22:         setControllerGap(0, g_i)
23:     activeController ← C_{i+1}
24:     startGapControl(h_{i+1}, d_{i+1})
25:     state ← G_{i+1}
26:     i ← i + 1
27: procedure GAP REACHED
28:     state ← F_i
29:     activeController ← C_i
30:     setControllerGap(h_i, d_i)
31:     if leaderBased(C_i) then
32:         becomeTemporaryLeader ← false
33: procedure TIMEOUT
34:     FAILURE()
35: procedure ONTEMPORARYLEADER(veh, tempLeader)
36:     if not isAhead(veh) then
37:         return
38:     if tempLeader = true then
39:         tempLeaders = tempLeaders ∪ {veh}
40:     else
41:         tempLeaders = tempLeaders \ {veh}
42:     currentLeader ← max(tempLeaders)
```

The last procedure (Line 35) is not part of the FSM, but describes how vehicles handle the reception of a frame from a vehicle having the *temporary leader* flag enabled (or disabled). Basically, the transmitting vehicle is added to or removed from the list of temporary leaders (depending on the value of the flag) and the one closer to the receiving one is chosen as its current leader.

3.3 Gap Control System

The last subsystem we need to define is the gap control procedure. The idea is to progressively modify the ACC/CACC desired gap to smoothly approach the new target one.

Switching from the current gap to the target one immediately might cause uncomfortable accelerations. Instead, we define a gap adaptation rate Δg in m/s and periodically update the ACC/CACC gap every Δt seconds. We can directly use the gap adaptation rate Δg to update the inter-vehicle distance for CACCs using a constant-spacing gap policy. For controllers using a constant time-headway spacing policy, we need to adapt the time headway with a rate Δh (in s/s) that results in an actual distance change rate equal to Δg solving the following equation for Δh:

$$((h + \Delta h \Delta t)v + d) - (hv + d) = \Delta g \Delta t. \quad (1)$$

In Equation (1), h is the time headway, v is the current speed, while d is the stand-still distance. The left side of the equation describes how much the distance changes over the time period Δt, which must match the rate Δg over the same time period. Solving the equation, we obtain

$$\Delta h = \frac{\Delta g}{v}. \quad (2)$$

Listing 3 describes the gap control procedure. SafeSwitch invokes the procedure indicating a target time headway and stand-still distance. For a constant-gap controller, $h_t = 0$. Using these values, it computes the target distance g_t. Depending on whether a time-headway or a constant-gap spacing policy is used, the procedure initializes the variable h and g in a different way. When using a time-headway spacing policy, the procedure adapts the time headway h using the rate Δh: it thus initializes h with the current time headway, while g is used to store the stand-still distance (which is kept constant). In case of a constant-gap spacing policy, instead, the procedure adapts the fixed distance g using the rate Δg: it thus initializes g with the current distance, while h is set equal to h_t (0). In both cases, the procedure keeps track of whether it is trying to increase or decrease the gap to decide whether the procedure is completed or not.

The algorithm progressively adapts the distance by periodically calling the *updateGap* procedure (Line 16). The procedure first checks whether the values h or g have reached their target value (function *isGapControlCompleted*); if this is the case, the gap is set to the final value and the procedure waits for the vehicle to actually reach the target distance (*isGapReached*). When this happens, SafeSwitch is notified the end of the procedure, thus triggering the switch between a G_i and an F_i state.

If the gap control procedure is not completed, either h or g are increased or decreased using the rates Δh or Δg, respectively (Lines 31 and 33).

Listing 3 Gap control algorithm.

```
1: procedure STARTGAPCONTROL(h_t, d_t)
2:     v ← curSpeed
3:     g_t ← h_t · v + d_t
4:     increasingGap ← true
5:     if using time headway then
6:         g ← d_t
7:         h ← (curDistance − d_t)/v
8:         if h_t < h then
9:             increasingGap ← false
10:    else
11:        g ← curDistance
12:        h ← h_t
13:        if g_t < g then
14:            increasingGap ← false
15:    updateGap()
16: procedure UPDATEGAP
17:    v ← curSpeed
18:    Δh ← Δg/v
19:    g_t ← h_t · v + d_t
20:    if isGapControlCompleted() then
21:        if using time headway then
22:            h ← h_t
23:        else
24:            g ← g_t
25:        setControllerGap(h, g)
26:        if isGapReached() then
27:            gapReached()
28:            return
29:    else
30:        if using time headway then
31:            h ← h + sgn(h_t − h) · Δh · Δt   ▷ see footnote
32:        else
33:            g ← g + sgn(g_t − g) · Δg · Δt
34:        setControllerGap(h, g)
35:    schedule(Δt, updateGap())
36: procedure ISGAPCONTROLCOMPLETED
37:    if using time headway then
38:        if increasingGap then
39:            return h ≥ h_t
40:        else
41:            return h ≤ h_t
42:    else
43:        if increasingGap then
44:            return g ≥ g_t
45:        else
46:            return g ≤ g_t
47: procedure ISGAPREACHED
48:    if increasingGap then
49:        return curDistance ≥ g_t
50:    else
51:        return curDistance ≤ g_t
```

* The sign function is defined as $\mathrm{sgn}(x) = 1$ if $x \geq 0, -1$ otherwise.

4 Simulation: Requirements and Design

The study of CACC fallback and dynamic management described in Section 3 requires a comprehensive and modular simulation framework. As discussed in Section 2.3 none of the free Open Source frameworks available can support this research, thus we decided to enhance the PLEXE prototype presented in [6] and make it an integral part of this contribution. We start our discussion from Figure 4, which shows the structure of the framework and its components. The figure highlights PLEXE core functionalities with green-filled boxes (or light gray in case of BW printing), while white boxes indicate external modules.

With respect to mobility, the top box shows the main

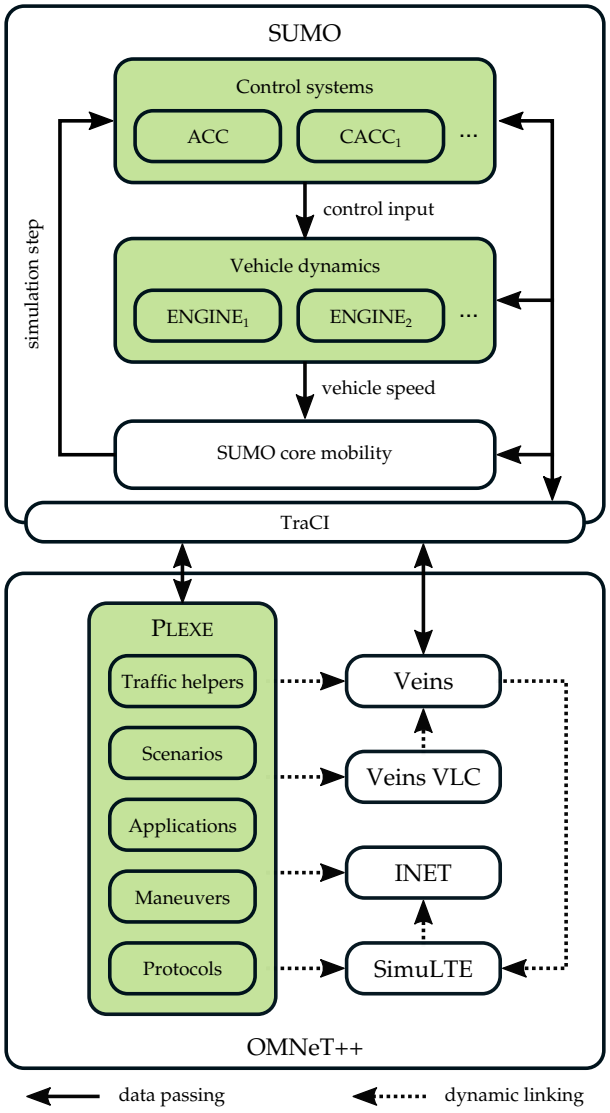


Figure 4. Structure of the PLEXE simulation framework.

components of SUMO. What PLEXE adds to SUMO are control algorithms and vehicle dynamics models. Control systems are implemented within a new car-following model, alternative to the ones that implement human-behavioral models such as IDM or Krauss. Car-following models are invoked each time step by the SUMO mobility logic for each vehicle to compute the speed it will follow in the next simulation step depending on the state of the vehicle itself, the surrounding environment, and additional external variables. The state of the vehicle includes information such as the current speed, acceleration, and position, the surrounding environment is represented by other vehicles or traffic lights, while external variables (especially for CACC) include data received via wireless communications. PLEXE offers one ACC algorithm [47, Chapter 6] and four CACCs [8], [9], [11], [48]. Research based on PLEXE lead to the implementation of additional ones. We plan to release these algorithms once their implementation is stable.

Independently of the chosen control algorithm, the output is the control input, i.e., the desired acceleration of the vehicle. The desired acceleration is passed to the vehicle dynamics module, which emulates the response of the vehicle to the control action. The simplest possible model available in PLEXE is a first order lag with a time constant τ. A first order lag, however, does not consider the physical limits of the vehicles. For example, a vehicle might have a maximum acceleration that depends on engine power, current gear ratio, aerodynamic drag, etc. For this reason, PLEXE also offers an engine model that considers mechanical specifications of the vehicle to compute its maximum acceleration, deceleration, and speed. We do not detail the model here but just give a brief description, as the focus of the work is on the communication-related insights that PLEXE enables. The output of the car-following model is a speed that is passed to the mobility core to update vehicle's speed and position.

With respect to network modeling, all the involved software modules are based on the OMNeT++ Discrete Event Simulation (DES) framework [31]. For coupling the mobility and the network simulation, SUMO offers the Traffic Control Interface (TraCI). Through this interface it is possible to control the simulation, obtain information about any object (vehicles, traffic lights, pedestrians, ...), and change their attributes. In particular, Veins exploits the interface to track the presence and position of all vehicles and makes this information available to its own and other networking models. PLEXE exploits and extends the TraCI interface as well, enabling the exchange of state information with the road traffic simulation models in SUMO. This includes, for example, transferring data to the control system (e.g., information about surrounding vehicles used in the computation of the control action) or retrieving vehicle data (e.g., information to be sent to neighboring vehicles). The website lists all the APIs that PLEXE offers to the users, which we clearly do not detail here. By itself, as shown in Figure 4, PLEXE offers tools to implement maneuvers, to control traffic and platoons, to define scenarios, protocols, and applications. These blocks characterize PLEXE and distinguish it from all other tools, thus we explain them in detail in Section 4.1.

PLEXE can interface external tools to provide additional functionalities, for example, to investigate the impact of a particular communication technology on the performance of the control system. In its prototype design PLEXE extended Veins, which provides the coupling with SUMO and models for the IEEE 1609.4 and IEEE 802.11p vehicular networking standards. When a user downloaded PLEXE he/she in fact downloaded Veins with additions to the original code base. This negatively affected both maintainability, as upgrades to the main Veins release were difficult to integrate, and the integration of additional tools.

In the new design, components are dynamically linked against each other and so, for example, PLEXE simply uses Veins as an external library, and this is the case for all the additional modules that PLEXE uses. The Veins VLC module [49], which provides propagation and communication models for LED-based V2V communications, is one of these models, fundamental because the interest for this communication technology is continuously increasing. There is no need to stress the role of cellular networks in cooperative driving applications. PLEXE thus couples with SimuLTE [50], which provides models for 4G LTE. This includes both the uplink/downlink transmission standard, typically used in cloud-based or MEC applications, and the

new C-V2X Mode 3 that supports direct V2V communication under the coordination of the eNodeB (eNB) in the release 14 of the standard [51].

The library-like structure of the whole framework makes it easy to plug-in new tools as they become available. As an example, to test the ease of integration, we combined PLEXE with a tool for simulating C-V2X Mode 4 (OpenCV2X [52]), which provides direct V2V communication through cellular networks without the coordination of an eNB. It will thus be easy to integrate OpenCV2X as soon as a stable version is released, or newly released frameworks such as Simu5G [53]. Here we describe its main features, while in the additional material we discuss its usage and flexibility.

4.1 Implementation details

As shown in Figure 4, far from just gluing all the network and mobility modules together, PLEXE offers core functionalities for the modeling and the analysis of cooperative driving and platooning systems. In the next subsection we describe core functionalities from a high-level perspective, while implementation-specific details can be found in the additional material.

4.1.1 Core functionalities

PLEXE defines a set of modules dedicated to set up cooperative driving simulations. One core feature is provided by the traffic helpers block, which is composed of *traffic managers* and *position helpers*.

Traffic managers are responsible to initialize the simulation from a traffic perspective (injecting vehicles into the simulation) while position helpers are used by vehicles to understand their status and the role they have within platoons. In particular, a position helper can be queried by a vehicle to know whether it is within a platoon or not, if it is a leader or a follower, which position it occupies inside the platoon, which are the other members, what is the id of the platoon, etc.

Traffic managers are particularly useful to insert preformed platoons in a steady-state configuration and speed-up simulations. While inserting the platoons, information about them is stored in a data structure, which is later accessed by the position helpers for initialization. Once initialized, each position helper (there is one for each vehicle) becomes independent from the others to represent the local knowledge of the vehicle. This is fundamental for studying cooperative driving maneuvers, where the knowledge of a vehicle depends on the information received from the communication protocol. Packet losses can lead to inconsistent views and the simulator must be capable of reproducing such inconsistencies. As an example, imagine the scenario in which two consecutive platoons merge together. At the end of the maneuver, the followers of the second platoon will need to change their leader upon receiving an update message. If such message is lost for some vehicles and the protocol has not been designed to handle such faults, some vehicles will follow the wrong leader. PLEXE enables the analysis of such inconsistencies, allowing also protocol verification.

The second building block is represented by *scenarios*. Scenarios model the high-level behavior of vehicles. Within its examples, PLEXE offers some classical ones that include emergency braking and changing the speed pattern of a platoon leader.

This is in contrast with *applications*, where vehicles' behavior is influenced by messages received from the network. The simplest form of application implemented in PLEXE is the one that processes incoming frames to check whether they encapsulate data to feed to the CACCs. Clearly this is a kind of message influencing vehicle's behavior.

Another particular type of application is represented by *maneuvers*. Although maneuvers are effectively applications, they deserve a dedicated spot due to the vast amount of research in the field. In the end, cooperative driving deals with the coordination of groups of vehicles. Maneuvers in PLEXE are hierarchically organized, meaning that they all derive from a base class which offers the same methods but different implementations. PLEXE offers two sample implementations: a join-at-back and a merge-at-back maneuver. The join at back maneuver enables a free vehicle to become part of an already existing platoon, joining as the last one. The merge at back is conceptually similar, but, in this case, it is the leader of a platoon that joins a platoon at the back and then instructs its followers to change their leader to be the one of the front platoon.

The final building block consists of *protocols*. In PLEXE, a protocol identifies a specific way of exchanging beacons that encapsulate control data. This is another active research field, as control information should be frequently and timely delivered to the control system in order to guarantee safety. Yet, in highly congested scenarios, packet losses might hinder correct frame delivery. PLEXE offers two sample implementations on top of IEEE 802.11p. A classic protocol sending frames with a frequency of 10 Hz and a slotted protocol proposed in [1] where vehicles within a single platoon divide each beacon interval in a number of slots equal to the number of vehicles in the platoon, each one sending its beacon in the slot number corresponding to their position inside the platoon. For the latter protocol, they exploit leader's beacons for synchronization.

4.1.2 Validation Example

The full validation of a tool like PLEXE is a constant (and never ending) endeavor. We offer here a simple sanity check based on Ploeg's CACC [9] coupled with IEEE 802.11p and VLC, showing the difference induced in the platoon behavior by the two technologies. The vehicles travel with an average speed of 100 km/h with the leader changing its speed between 95 and 105 km/h with a frequency of 0.2 Hz. Setting the CACC parameters to have a time headway of 0.5 s and a stand-still distance of 2 m, the distance between platoon vehicles oscillates around 15.9 m. Table 1 lists simulation parameters.

Figure 5 shows the inter-vehicle distances measured during the simulation. With respect to IEEE 802.11p, the graph shows the theoretically expected behavior of Ploeg's algorithm [9], as with only 8 vehicles 802.11p causes no losses and the control system works in perfect conditions. The CACC progressively smoothens the oscillations and it is possible to observe the delay in the oscillation introduced by the time-headway spacing policy. When using VLC, instead, due to the unreliability of the interface, several packets can be consecutively lost, causing instability and large distance

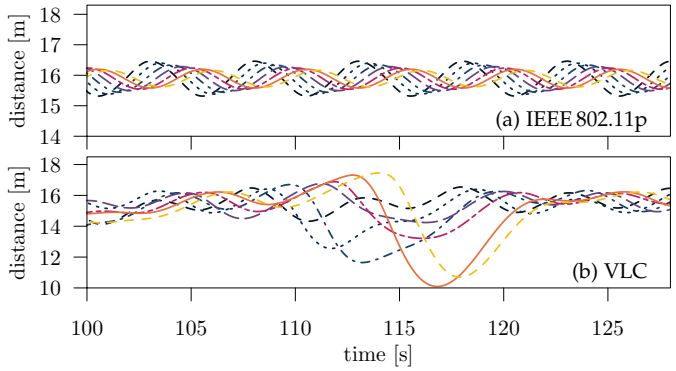
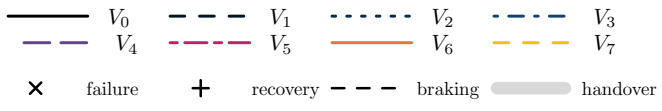


Figure 5. Inter-vehicle distances for a platoon of 8 cars using Ploeg's CACC under a sinusoidal disturbance and different communication technologies, following conventions of Table 1. We use different y-axis ranges on purpose to better show the behavior of the CACC in different conditions.

Table 1
Road traffic simulation parameters and legend used in all the plots. Failure and recovery event symbols are shown in black here but are colored in the plots to indicate the vehicle involved.

	Parameter	Value
mobility	Cruising speed	100 km/h
	Braking vehicle deceleration	7 m/s^2
	Post-braking speed	30 km/h
	Platoon size	8 cars
	Simulation time step	10 ms
	Min/max acceleration	−9 to 2.5 m/s^2
controllers	Engine lag τ	0.5 s
	ACC λ	0.1
	PATH C_1	0.5
	PATH ω_n	0.2
	PATH ξ	1
	PLOEG k_p	0.2
	PLOEG k_d	0.7



errors that can be observed around 115 s of simulation. These simple results show the proper implementation of the CACC algorithms and the correct coupling between vehicle and communication dynamics as well as the importance thereof.

5 Performance Evaluation

We analyze the performance of SafeSwitch in different scenarios, which include artificially-generated and realistic failures, as well as emergency braking events. In the following, for the sake of space, all graphs will refer to the same legend, depicted in Table 1 together with mobility and control parameters common to all scenarios. Besides colored lines, which represent quantities of a specific value, we also show failure and recovery events marked by a × and a + symbol, respectively. The color of the symbol indicates on which vehicle the event occurred. Vertical dashed lines indicate the occurrence of an emergency braking event, while vertical gray bands indicate a handover period of the platoon, i.e., from the time the first vehicle begins the handover to the time the last one terminates it. In general, we assume that

Table 2
Configuration of the state machines (scenarios 0 to 3).

	SafeSwitch			Naïve Fallback		
i	C_i	h_i	d_i	C_i	h_i	d_i
0	ACC	1.2 s	2 m	ACC	1.2 s	2 m
1	PLOEG	1.2 s	2 m	PATH	0 s	5 m
2	PLOEG	1.2 s	2 m	—	—	
3	PATH	0 s	5 m	—	—	

vehicles run PATH in F_3, Ploeg with different parameters in F_2 and F_1 and a standard ACC in F_0.

We compare the full system with the three communication technologies defined in Section 3 against a naïve baseline approach (the only possible approach if a single communication technology is available): When a failure occurs the system immediately falls back to ACC with a transition $F_1 \rightarrow F_0$ (and vice-versa on recovery). We maintain the temporary leader mechanism for a fair comparison.

5.1 Benchmark with single failures

We start the performance evaluation with the following basic scenarios based on crafted single technology failures:

Scenario 0: single failure and recovery on a single vehicle in the middle of the platoon;

Scenario 1: single failure on all members of the platoon;

Scenario 2: single failure and recovery on a single vehicle in the middle of the platoon with an emergency braking occurring after the failure;

Scenario 3: single failure on all members of the platoon with an emergency braking occurring after the failure.

To simulate the emergency braking event, we inject a vehicle in front of the platoon that, at a certain instant, abruptly decelerates down to a lower speed. Table 2 lists the parameter of the two fallback mechanisms. For SafeSwitch, the failure will trigger the transitions $F_3 \rightarrow G_2 \rightarrow F_2$, while for the naïve fallback the transition is obviously $F_1 \rightarrow F_0$. For the sake of comparison with the naïve approach, we set $h_2 = 1.2\,\text{s}$ and $d_2 = 2\,\text{m}$ for SafeSwitch to match ACC values. The choice of using PATH in F_3 and Ploeg in F_2 and F_1 in SafeSwitch comes from the fact that PATH is more "demanding" in terms of communication requirements as it needs data from both leading and preceding vehicles and it employs a constant-spacing policy. This choice, however, can be customized as SafeSwitch can simply be reconfigured if a controller assignment is found to be more appropriate.

Figures 6 to 9 show the acceleration and the inter-vehicle distance measured in the four scenarios. Starting with Figure 6 (single vehicle failure) it is immediately clear that our proposed approach behaves more gently, with unperceivable accelerations. This is not only good for passengers' comfort but also for fuel efficiency. With the naïve approach, the acceleration is larger but not so uncomfortable, as the controller parameters normally used in the literature, that we replicate here, limit the impact of large distance error on the acceleration, but this can also have negative impacts on safety as we will see shortly.

When one technology fails for all vehicles (Figure 7) the situation is similar but PLEXE highlights a problem in both

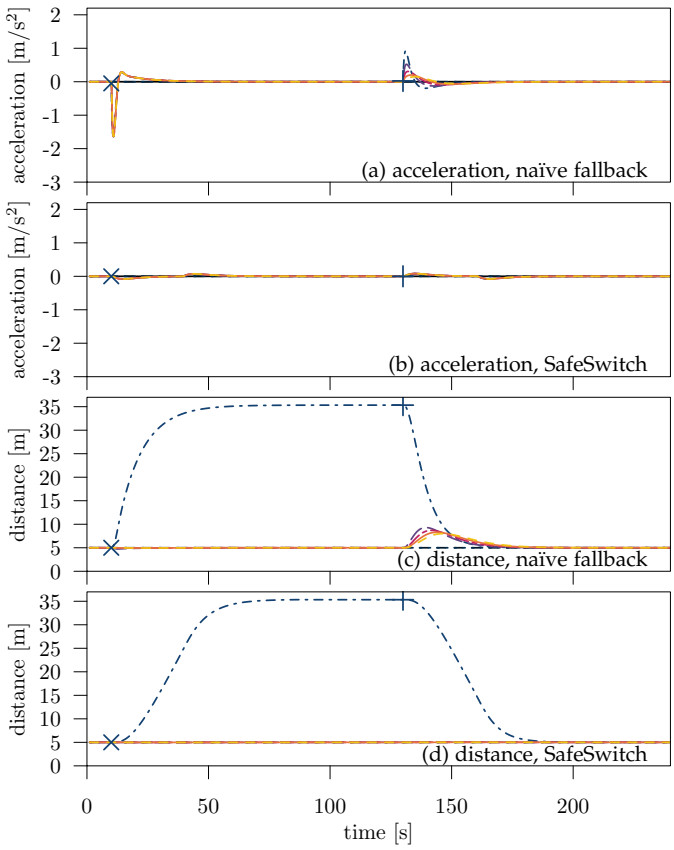


Figure 6. Comparison of acceleration and inter-vehicle distances for scenario 0 (single vehicle failure and recovery), following conventions of Table 1.

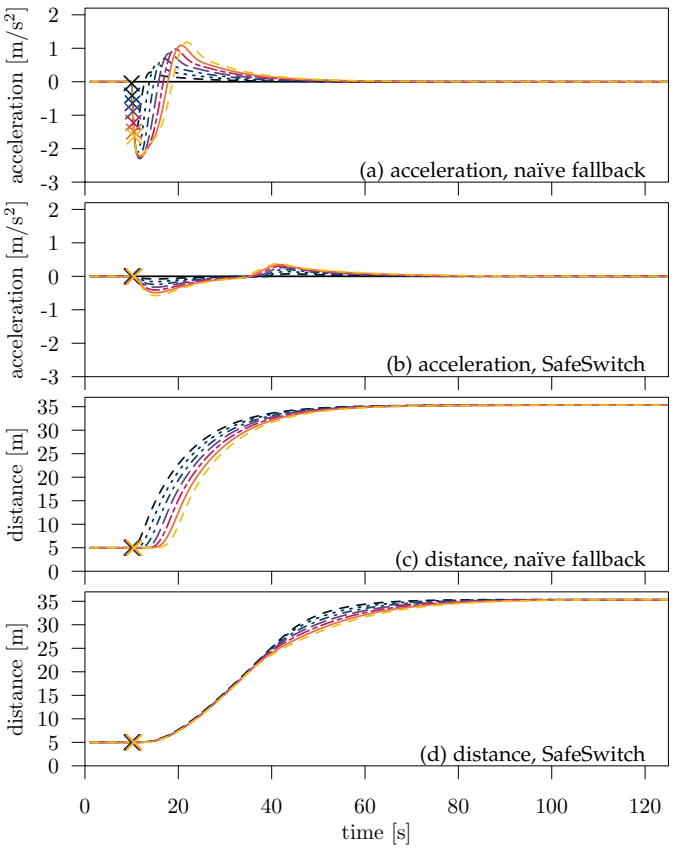


Figure 7. Comparison of acceleration and inter-vehicle distances for scenario 1 (multiple vehicles failure), following conventions of Table 1.

procedures. Looking at the acceleration profiles, it can be seen that the deceleration becomes larger and larger towards the tail of the platoon. This is due to the fact that each vehicle is increasing the gap to its predecessor, so it needs to compensate for its own error plus the error induced by its predecessors. In a large string of vehicles, this might cause unsafe situations. Moreover, there is an overshoot (vehicles accelerate again) before reaching stability. The sophisticated analysis empowered by PLEXE allows spotting the issue and thinking about possible solutions (e.g., as the problem here is caused by each vehicle independently increasing their inter-vehicle gap, the problem might be solved with a more coordinated approach exploiting remaining communication technologies). In any case the behavior of SafeSwitch is much better than the naïve fallback, highlighting the need for multi-technology resilience and sound fallback mechanisms for safety. Both in this case and in Figure 6, it is important to notice that the behavior of SafeSwitch is *qualitatively* different from the naïve one, highlighting that probably the abrupt fallback from cooperative to autonomous driving is not safe.

When considering an emergency braking immediately after a failure (Figures 8 and 9) the behavior of the two fallback mechanisms changes dramatically. With respect to the naïve approach, regardless of whether the failure is for a single vehicle or for multiple vehicles, a collision occurs, thus we do not report the relative graphs as they are useless. This is the consequence of using a small λ for the ACC[7] which, in

7. The λ parameter weights the distance error in the control formula.

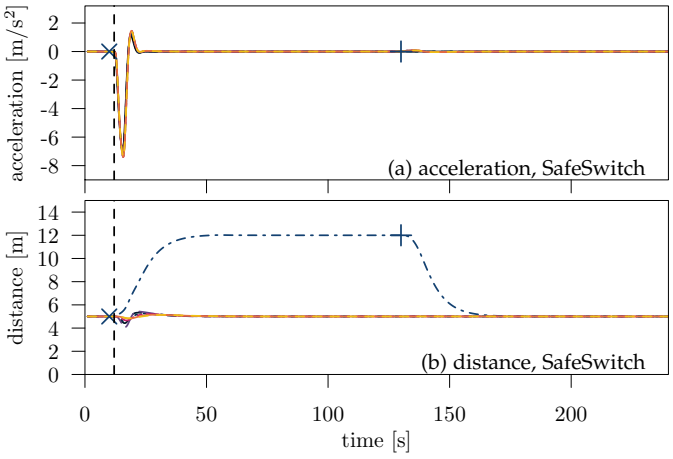


Figure 8. Acceleration and inter-vehicle distances for scenario 2 (single vehicle failure and recovery with emergency braking) for SafeSwitch, naïve fallback results in crashes (see the additional material), following conventions of Table 1.

the first two scenarios, results in smooth deceleration when switching controllers, but induces a slow reaction to the sudden deceleration of the platoon leader. With SafeSwitch, instead, the vehicles are capable of decelerating avoiding the collision and reaching the target inter-vehicle gap. The graphs again show the problem highlighted by the first two scenarios, i.e., vehicles at the tail of the platoon are braking more than the one at the head. While the heading vehicle is braking with a $7\,\mathrm{m/s^2}$ deceleration, the scenario 3 reaches a maximum of about $8\,\mathrm{m/s^2}$.

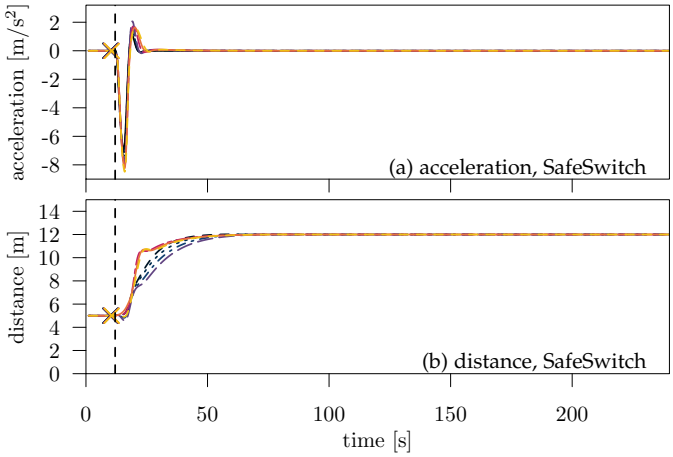


Figure 9. Acceleration and inter-vehicle distances for scenario 3 (multiple vehicles failure with emergency braking) for SafeSwitch, naïve fallback results in crashes (see the additional material), following conventions of Table 1.

5.2 Multiple artificial failures

We now focus on SafeSwitch only, still considering artificial failures, in particular multiple consecutive ones. We maintain the parameters defined in Table 1 but we use the parameters in Table 3 to configure the FSM, reducing the headway time for the Ploeg controller to 0.5 s in F_2 as commonly considered when communications work. We define the following scenarios to analyze:

Scenario 4: multiple failures and recoveries to a single vehicle in the middle of the platoon;
Scenario 5: multiple failures to all members of the platoon;
Scenario 6: multiple failures and recoveries to a single vehicle in the middle of the platoon with an emergency braking occurring after the second failure;
Scenario 7: multiple failures to all members of the platoon with an emergency braking occurring after the second failure.

We consider two variants for scenario 7. In the first one, failures occur spaced in time: vehicles have enough time to adapt to the new gap, performing the $G_2 \rightarrow F_2$ transition before the occurrence of the second failure. In the second, instead, the second failure occurs 1 s after the other: vehicles thus perform the $G_2 \rightarrow G_1$ transition. We do not consider the additional variant with three almost simultaneous failures as the behavior would simply correspond to the one observed with the naïve fallback, i.e., immediately switching to ACC-only control. We report here the results for scenarios 4 and 7, while those for scenarios 5 and 6, all positive, are included in the additional material to avoid repetitive patterns.

Figures 10 and 11 show the results, which are all similar to the ones observed for single failures. After the second failure in Figure 10 the acceleration pattern is slightly different from the first failure but this is due to a different fallback CACC. In the top two plots of Figure 11 (failure of different interfaces spaced in time), instead, the amplification phenomenon observed in Figure 9 disappears and the reason is the combination of the spacing policy and the braking event. As the Ploeg controller uses a time-headway spacing policy, the target distance depends on the speed, but as the vehicles slow down their target distance decreases, automatically

Table 3
Configuration of SafeSwitch (scenarios 4 to 7).

i	C_i	h_i	d_i
0	ACC	1.2 s	2 m
1	PLOEG	1.2 s	2 m
2	PLOEG	0.5 s	2 m
3	PATH	0 s	5 m

Parameter	Value
PDR thr. (802.11p)	0.8
PDR thr. (C-V2X)	0.8
PDR thr. (VLC)	0.6
recover time Δ_t	10 s
ring buffer size	10 packets

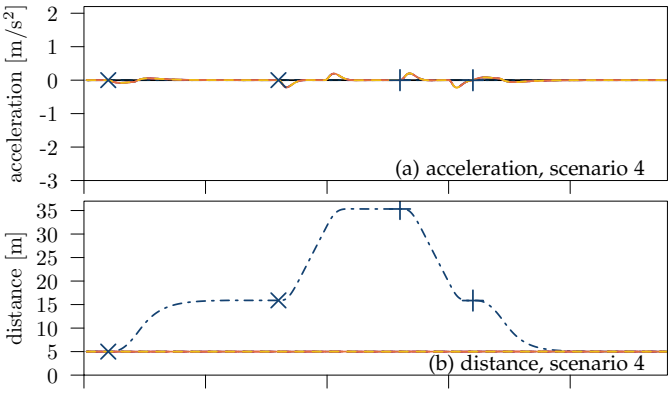


Figure 10. Acceleration and inter-vehicle distances for scenarios 4: Multiple failure and recovery on a single vehicle, following conventions of Table 1.

reducing the spacing errors and requiring less deceleration effort to correct it. Again, this is a unique insight provided by PLEXE, which highlights its importance and versatility in such complex analyses. When the failures of different interfaces occur almost simultaneously (bottom two plots of Figure 11), the behavior is comparable to the one in scenario 3 (Figure 9) because PATH is used to reduce the gap down to the value required for driving in the state F_1. Still, the system manages to bring the vehicles to the desired inter-vehicle distance without causing collisions.

5.3 Realistic failures

As a final analysis, we present the behavior of SafeSwitch under realistic failure models for the three communication technologies. We simulate an oval-shaped (two long straights connected by long 180-degree curves) circuit where the platoon travels continuously. The circuit has a length of roughly 3.2 km. At the two curves we install two LTE base stations that manage the resource allocation for the sidelink (vehicle to vehicle) communications. As vehicles continuously pass from the coverage of one base station to the other, the vehicles perform handovers which might cause packet losses.

Close to one base station, two static IEEE 802.11p nodes continuously exchange messages, causing interference when the vehicles are in their proximity. The nodes use a physical bitrate of 6 Mbit/s and generate a traffic load of roughly 4 Mbit/s, almost saturating the channel. Finally, with respect to VLC, vehicles implement a forwarding mechanism for frames to the head and the tail of the platoon, as communication is only possible between adjacent vehicles and strictly in line of sight. If a frame is lost, its content

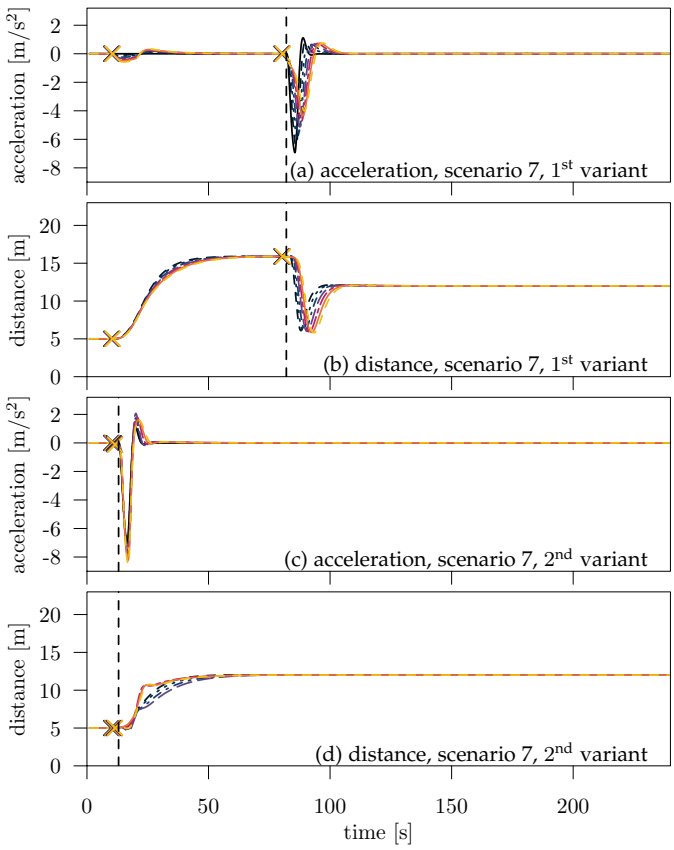


Figure 11. Acceleration and inter-vehicle distances for scenario 7: progressive complete failure for all vehicles with an emergency brake, following conventions of Table 1. The top two plots refer to the first variant (failures of different interfaces spaced in time) while the bottom two refer to the second variant (almost simultaneous failure of multiple interfaces).

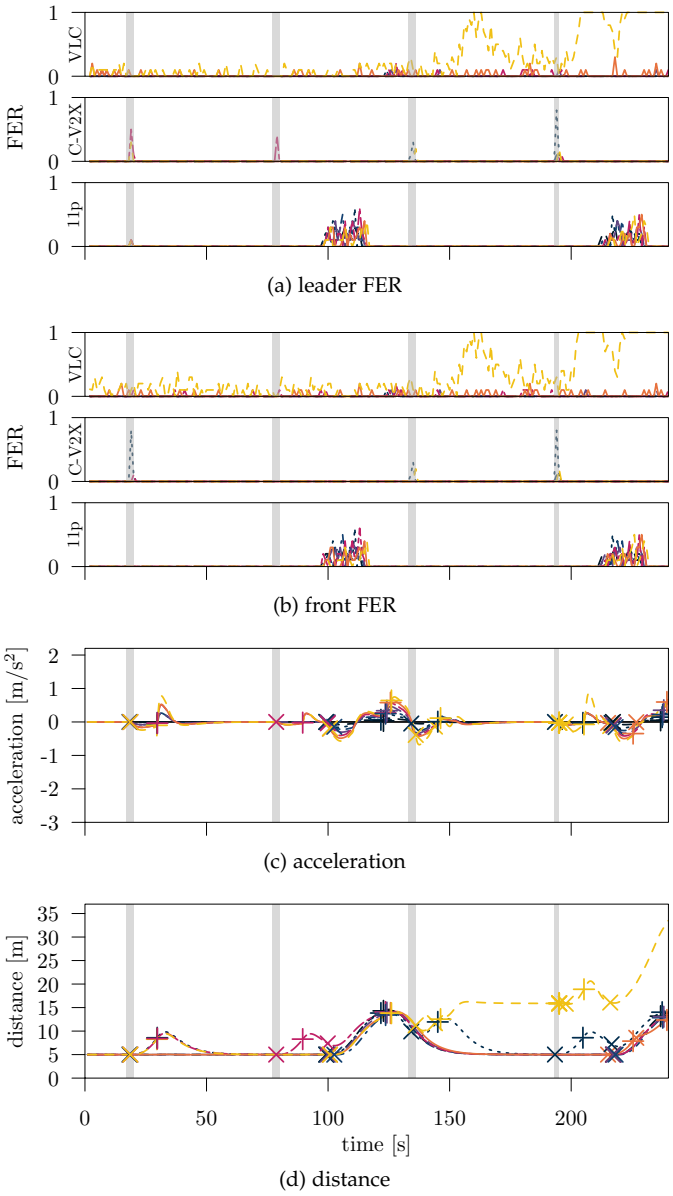


Figure 12. Vehicle dynamics induced by SafeSwitch and frame error rates for the scenario with realistic failures, following conventions of Table 1.

is lost for all the subsequent vehicles in the direction of propagation. We do not list the complete set of parameters for all the technologies as we use the default ones provided by SimuLTE, PLEXE/Veins, and Veins VLC: the aim of our work is showing the performance of SafeSwitch and the unique insights PLEXE can provide rather than performing a parameter tuning study. Finally, the right-side of Table 3 lists the parameters of the PDR monitor (Section 3.1). We repeat the simulations 10 times to obtain different random loss patterns but, for the sake of brevity, we show just one scenario; two more are included in the Additional Material.

Figure 12 shows the results for the selected simulation. Together with the dynamics, we also show the measured Frame Error Rate (FER) for all the technologies, for the leader and the front vehicle frames. With respect to FERs, for IEEE 802.11p vehicles suffer packet losses when traveling close to the static nodes that generate channel interference. When this occurs, all nodes start increasing their gap and then, when out of the interference zone, re-compact once again. For C-V2X, handovers cause very brief losses which are quickly recovered. Vehicles thus begin the fallback procedure and then quickly return to the initial gap.

Regarding VLC, the simulator shows that the technology is less reliable compared to the other two, but in the first part of the simulation this does not trigger failures. It is interesting to notice, however, the correlation between VLC's FER and the dynamics of vehicle V_7 starting from simulation time 150 s. In particular, the distance of V_7 from V_6 increases due to losses in IEEE 802.11p and LTE and this causes an increase in the FER for the VLC interface as it is very sensitive to distance. When an additional failure occurs, the gap is increased even further causing additional losses. Above a certain inter-vehicle distance (roughly 20 m) no VLC communication is possible and the vehicle is forced to increase the gap to the maximum under CACC with the next burst of IEEE 802.11p losses. This indicates that SafeSwitch, developed to ensure the overall safety of the system, can itself induce a degradation of communication performance due to the increase in the safety distance, meaning that the design of the protocol must carefully take into account the intrinsic characteristics of the lower communication means. This is again a unique insight that PLEXE provides: the interaction between communication technologies, cooperative driving protocols, and vehicle dynamics which would simply be impossible using standard cooperative driving simulators.

6 Conclusion

Cooperative driving is a complex topic where many scientific fields have to merge to achieve meaningful results, and where performance must be evaluated mainly through credible in-silico experiments, as building real systems to test is too expensive and dangerous. This paper has presented our Safe Autonomous Switchover Algorithm (SafeSwitch), an interdisciplinary protocol to appropriately handle fallback procedures that allow the smooth transition between different algorithms for platooning longitudinal control and, if needed, the return to a non-cooperative ACC or even full manual control. We have shown that to study such procedures it is necessary to exploit sophisticated simulation tools that allow the study of vehicles dynamics together with multiple communication interfaces to empower the safe handling of conceivable failures of communications systems, including active jamming that completely *shuts off* one technology for the entire platoon.

Besides SafeSwitch and its analysis, one integral contribution of our work is PLEXE, the simulation tool we developed having in mind not only our present study, but to enable future work in the area of cooperative driving. PLEXE is Open Source, freely downloadable, and actively maintained to help the study and development of innovative cooperative driving applications based on many networking technologies, from IEEE 802.11p, to C-V2X, to VLC, and many others.

Acknowledgments

Michele Segata would like to thank Giovanni Nardini for the long discussion and troubleshooting sessions on SimuLTE and the time he spent fixing the issues inside the codebase.

References

[1] M. Segata et al., "Towards Communication Strategies for Platooning: Simulative and Experimental Evaluation," *TVT*, vol. 64, no. 12, pp. 5411–5423, Dec. 2015.

[2] F. Dressler, F. Klingler, M. Segata, and R. Lo Cigno, "Cooperative Driving and the Tactile Internet," *Proceedings of the IEEE*, vol. 107, no. 2, pp. 436–446, Feb. 2019.

[3] P. Alvarez Lopez et al., "Microscopic Traffic Simulation using SUMO," in *IEEE ITSC 2018*, Maui, HI: IEEE, Nov. 2018, pp. 2575–2582.

[4] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations," *PRE*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000.

[5] S. Krauß, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *APS Physical Review E*, vol. 55, no. 5, pp. 5597–5602, May 1997.

[6] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Lo Cigno, "PLEXE: A Platooning Extension for Veins," in *IEEE VNC 2014*, Paderborn, Germany: IEEE, Dec. 2014, pp. 53–60.

[7] C. D. Ozkaptan, E. Ekici, O. Altintas, and C.-H. Wang, "OFDM Pilot-Based Radar for Joint Vehicular Communication and Radar Systems," in *IEEE VNC 2018*, Taipei, Taiwan: IEEE, Dec. 2018.

[8] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of Integrated Longitudinal and Lateral Control for the Operation of Automated Vehicles in Platoons," *TCST*, vol. 8, no. 4, pp. 695–708, Jul. 2000.

[9] J. Ploeg, B. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and Experimental Evaluation of Cooperative Adaptive Cruise Control," in *IEEE ITSC 2011*, Washington, D.C.: IEEE, Oct. 2011, pp. 260–265.

[10] S. Santini, A. Salvi, A. S. Valente, A. Pescapè, M. Segata, and R. Lo Cigno, "Platooning Maneuvers in Vehicular Networks: a Distributed and Consensus-Based Approach," *T-IV*, vol. 4, no. 1, pp. 59–72, Mar. 2019.

[11] A. Ali, G. Garcia, and P. Martinet, "The Flatbed Platoon Towing Model for Safe and Dense Platooning on Highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 58–68, Jan. 2015.

[12] S. Ishihara, R. V. Rabsatt, and M. Gerla, "Improving Reliability of Platooning Control Messages Using Radio and Visible Light Hybrid Communication," in *IEEE VNC 2015*, Kyoto, Japan: IEEE, Dec. 2015, pp. 96–103.

[13] M. Segata, R. Lo Cigno, H.-M. Tsai, and F. Dressler, "On Platooning Control using IEEE 802.11p in Conjunction with Visible Light Communications," in *IEEE/IFIP WONS 2016*, Cortina d'Ampezzo, Italy: IEEE, Jan. 2016, pp. 124–127.

[14] S. Ucar, S. Coleri Ergen, and O. Ozkasap, "IEEE 802.11p and Visible Light Hybrid Communication based Secure Autonomous Platoon," *TVT*, vol. 67, no. 9, pp. 8667–8681, Sep. 2018.

[15] M. Schettler, A. Memedi, and F. Dressler, "Deeply Integrating Visible Light and Radio Communication for Ultra-High Reliable Platooning," in *IEEE/IFIP WONS 2019*, Wengen, Switzerland: IEEE, Jan. 2019, pp. 36–43.

[16] T. Hardes and C. Sommer, "Towards Heterogeneous Communication Strategies for Urban Platooning at Intersections," in *IEEE VNC 2019*, Los Angeles, CA: IEEE, Dec. 2019, pp. 322–329.

[17] M. Sybis, P. Kryszkiewicz, and P. Sroka, "On the Context-Aware, Dynamic Spectrum Access for Robust Intraplatoon Communications," *Hindawi Mobile Information Systems*, Jun. 2018.

[18] A. Bazzi, C. Campolo, A. Molinaro, A. O. Berthet, B. M. Masini, and A. Zanella, "On Wireless Blind Spots in the C-V2X Sidelink," *TVT*, vol. 69, no. 8, pp. 9239–9243, Aug. 2020.

[19] M. Segata, P. Arvani, and R. Lo Cigno, "A Critical Assessment of CV2X Resource Allocation Scheme for Platooning Applications," in *IEEE/IFIP WONS 2021*, Virtual Conference: IEEE, Mar. 2021.

[20] B. Coll-Perales, J. Gozalvez, and M. Gruteser, "Sub-6GHz Assisted MAC for Millimeter Wave Vehicular Communications," *COMMAG*, vol. 57, no. 3, pp. 125–131, Mar. 2019.

[21] R. W. van der Heijden, T. Lukaseder, and F. Kargl, "Analyzing Attacks on Cooperative Adaptive Cruise Control (CACC)," in *IEEE VNC 2017*, Turin, Italy: IEEE, Nov. 2017, pp. 45–52.

[22] M. Segata, F. Dressler, and R. Lo Cigno, "Jerk Beaconing: A Dynamic Approach to Platooning," in *IEEE VNC 2015*, Kyoto, Japan: IEEE, Dec. 2015, pp. 135–142.

[23] G. Giordano, M. Segata, F. Blanchini, and R. Lo Cigno, "The joint network/control design of platooning algorithms can enforce guaranteed safety constraints," *Elsevier Ad Hoc Networks*, vol. 94, Nov. 2019.

[24] J. Ploeg, E. Semsar-Kazerooni, G. Lijster, N. van de Wouw, and H. Nijmeijer, "Graceful Degradation of Cooperative Adaptive Cruise Control," *TITS*, vol. 16, no. 1, pp. 488–497, Feb. 2015.

[25] C. Wu, Y. Lin, and A. Eskandarian, "Cooperative Adaptive Cruise Control With Adaptive Kalman Filter Subject to Temporary Communication Loss," *IEEE Access*, vol. 7, pp. 93 558–93 568, Jul. 2019.

[26] Y. Tu, W. Wang, Y. Li, C. Xu, T. Xu, and X. Li, "Longitudinal safety impacts of cooperative adaptive cruise control vehicle's degradation," *Journal of Safety Research*, vol. 69, pp. 177–192, Jun. 2019.

[27] Y.-Y. Qin, Z.-Y. He, and B. Ran, "Rear-End Crash Risk of CACC-Manual Driven Mixed Flow Considering the Degeneration of CACC Systems," *IEEE Access*, vol. 7, pp. 140 421–140 429, Sep. 2019.

[28] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *TMC*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[29] B. Schünemann, "V2X Simulation Runtime Infrastructure VSimRTI: An Assessment Tool to Design Smart Traffic Management Systems," *COMNET*, vol. 55, no. 14, pp. 3189–3198, Oct. 2011.

[30] M. Rondinone et al., "iTETRIS: A Modular Simulation Platform for the Large Scale Evaluation of Cooperative ITS Applications," *Simulation Modelling Practice and Theory*, vol. 34, pp. 99–125, May 2013.

[31] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *ACM/ICST SIMUTools 2008*, Marseille, France: ACM, Mar. 2008.

[32] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds., Springer, 2010, pp. 15–34.

[33] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative Adaptive Cruise Control in Real Traffic Situations," *TITS*, vol. 15, no. 1, pp. 296–305, Feb. 2014.

[34] IEEE, "Wireless Access in Vehicular Environments," IEEE, Std 802.11p-2010, Jul. 2010.

[35] M. Segata, B. Bloessl, S. Joerer, F. Dressler, and R. Lo Cigno, "Supporting Platooning Maneuvers through IVC: An Initial Protocol Analysis for the Join Maneuver," in *IEEE/IFIP WONS 2014*, Obergurgl, Austria: IEEE, Apr. 2014, pp. 130–137.

[36] J. Mena-Oreja and J. Gozalvez, "PERMIT – A SUMO Simulator for Platooning Maneuvers in Mixed Traffic Scenarios," in *IEEE ITSC 2018*, Maui, HI: IEEE, Nov. 2018, pp. 3445–3450.

[37] J. Heinovski and F. Dressler, "Platoon Formation: Optimized Car to Platoon Assignment Strategies and Protocols," in *IEEE VNC 2018*, Taipei, Taiwan: IEEE, Dec. 2018.

[38] T. Hardes and C. Sommer, "Dynamic Platoon Formation at Urban Intersections," in *IEEE LCN 2019, Poster Session*, Osnabrück, Germany: IEEE, Oct. 2019.

[39] N. Lyamin, B. Bellalta, and A. Vinel, "Age-of-Information-Aware Decentralized Congestion Control in VANETs," *IEEE Networking Letters*, vol. 2, no. 1, pp. 33–37, Mar. 2020.

[40] M. Amoozadeh, B. Ching, C.-N. Chuah, D. Ghosal, and H. M. Zhang, "VENTOS: Vehicular Network Open Simulator with Hardware-in-the-Loop Support," *Procedia Computer Science*, vol. 151, pp. 61–68, Jan. 2019.

[41] M. Fellendorf and P. Vortisch, "Microscopic Traffic Flow Simulator VISSIM," in *Fundamentals of Traffic Simulation*, J. Barceló, Ed., Springer, 2010, pp. 63–93.

[42] A. Choudhury et al., "An integrated V2X simulator with applications in vehicle platooning," in *IEEE ITSC 2016*, Rio de Janeiro, Brazil, Nov. 2016.

[43] H. Ramezani, S. E. Shladover, X.-Y. Lu, and O. D. Altan, "Micro-Simulation of Truck Platooning with Cooperative Adaptive Cruise Control: Model Development and a Case Study," *Transportation Research Record*, vol. 2672, no. 19, Dec. 2018.

[44] C. Ayimba, M. Segata, P. Casari, and V. Mancuso, "Closer than Close: MEC-Assisted Platooning with Intelligent Controller Migration," in *ACM MSWiM 2021*, Alicante, Spain, Nov. 2021.

[45] R. Kianfar, P. Falcone, and J. Fredriksson, "A Receding Horizon Approach to String Stable Cooperative Adaptive Cruise Control," in *14th IEEE International Conference on Intelligent Transportation Systems (ITSC 2011)*, Washington, D.C.: IEEE, Oct. 2011, pp. 734–739.

[46] T. Robinson, E. Chan, and E. Coelingh, "Operating Platoons On Public Motorways: An Introduction To The SARTRE Platooning Programme," in *17th World Congress on Intelligent Transport Systems (ITS 2010)*, Busan, South Korea, Oct. 2010.

[47] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. Springer, 2012.

[48] S. Santini, A. Salvi, A. S. Valente, A. Pescapè, M. Segata, and R. Lo Cigno, "A Consensus-based Approach for Platooning with Inter-Vehicular Communications and its Validation in Realistic Scenarios," *TVT*, vol. 66, no. 3, pp. 1985–1999, Mar. 2017.

[49] A. Memedi, C. Tebruegge, J. Jahneke, and F. Dressler, "Impact of Vehicle Type and Headlight Characteristics on Vehicular VLC Performance," in *IEEE VNC 2018*, Taipei, Taiwan: IEEE, Dec. 2018.

[50] A. Virdis, G. Stea, and G. Nardini, "SimuLTE - A Modular System-level Simulator for LTE/LTE-A Networks based on OMNeT++," in *International SIMULTECH 2014*, Vienna, Austria, Aug. 2014.

[51] "GSM; UMTS; LTE; 5G; Release description; Release 14," 3GPP, Sophia Antipolis, France, TS 21.914 v14.0.0, Jun. 2018.

[52] B. McCarthy and A. O'Driscoll, "OpenCV2X Mode 4: A Simulation Extension for Cellular Vehicular Communication Networks," in *IEEE CAMAD 2019*, Limassol, Cyprus, Sep. 2019.

[53] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G – An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, Jan. 2020.

Michele Segata is assistant professor at the Faculty of Computer Science of the Free University of Bolzano. His research focuses on cooperative driving systems.

Renato Lo Cigno is full professor of Computer Networks at the Information Engineering Department of the University of Brescia, Italy. His research interests cover many areas in networking, including vehicular networks and cooperative driving.

Tobias Hardes is a PhD candidate at TU Dresden, Germany. His research focus is on wirelessly connected mobile systems, especially on platoons in urban areas and UAV networks.

Julian Heinovski is a PhD candidate at the School of Electrical Engineering and Computer Science, TU Berlin, Germany. His research interest is in cooperative driving systems, mainly focusing on platooning.

Max Schettler is a PhD candidate at the School of Electrical Engineering and Computer Science, TU Berlin, Germany. His research interest is in heterogeneous wireless communication and edge computing.

Bastian Bloessl is a postdoctoral researcher at the Secure Mobile Networking Lab, TU Darmstadt, Germany. His research focuses on software-defined wireless communication systems.

Christoph Sommer is a full professor and holds a chair at TU Dresden, Germany, heading the Networked Systems Modeling group. His research is focused on protocol and system designs of wirelessly connected mobile systems exhibiting high topology dynamics.

Falko Dressler is full professor and Chair for Telecommunication Networks at the School of Electrical Engineering and Computer Science, TU Berlin, Germany. His research objectives include adaptive wireless networking and wireless-based sensing with applications in ad hoc and sensor networks, the internet of things, and cyber-physical systems.

Multi-Technology Cooperative Driving: An Analysis Based on PLEXE

Michele Segata, *Member, IEEE*, Renato Lo Cigno, *Senior Member, IEEE*, Tobias Hardes, *Student Member, IEEE*, Julian Heinovski, *Student Member, IEEE*, Max Schettler, *Student Member, IEEE*, Bastian Bloessl, *Member, IEEE*, Christoph Sommer, *Member, IEEE*, and Falko Dressler, *Fellow, IEEE*

We report in this Additional Material some background and results that may be useful to fully appreciate and exploit the contribution we provide in the main paper. Section 1 reports a short primer on platooning and longitudinal vehicles' control. The material is all available in the original publications, but we think that having it all available at hand with a uniform notation makes the paper easier to read and appreciate. Section 2 presents the results and plots that are not included in the main paper because they are somewhat repetitive, but can be useful to build further contributions beyond our work. Finally, Section 3 adds some details on PLEXE and show some use cases that highlight the potentialities of the tool.

1 PLATOONING CONTROL

The goal of this short Section is simply to provide the reader with all the information needed to easily follow the paper without the need to browse in the literature and with a uniform notation. All the equations we provide here are reported from the cited original works, and correspond to the implementation in PLEXE. We assume the reader has some familiarity with control theory and distributed systems as this is not meant to be a primer or a tutorial on cooperative driving and platooning. Symbols are described in the text as soon as they are introduced, but we summarize them in Table 1 for reader's convenience. In the following x_i, v_i and a_i indicate the position (relative to the vehicle in front), the speed, and the acceleration of the vehicle in position i in the platoon. A dot on a symbol, as in $\dot{a}$, indicate a time derivative.

- *M. Segata is with the Faculty of Computer Science, Free University of Bolzano, Italy, E-mail: michele.segata@unibz.it.*
- *R. Lo Cigno is with DII, University of Brescia, Italy, E-mail: renato.locigno@unibs.it.*
- *T. Hardes is with TU Dresden, Faculty of Computer Science, Germany and the Software Innovation Campus Paderborn (SICP), Paderborn University, Germany, E-mail: tobias.hardes@upb.de.*
- *J. Heinovski, M. Schettler, and F. Dressler are with the School for Electrical Engineering and Computer Science, TU Berlin, Germany, E-Mail: {heinovksi, schettler, dressler}@ccs-labs.org.*
- *C. Sommer is with TU Dresden, Faculty of Computer Science, Germany, E-mail: sommer@cms-labs.org.*
- *B. Bloessl is with the Secure Mobile Networking Lab, TU Darmstadt, Germany, E-mail: mail@bastibl.net.*

Table 1
List of symbols.

Symbol	Meaning
$\dot{\square}$	first derivative of $\square$
$\square_i$	value of a variable $\square$ for the i-th vehicle in a platoon
u	control input (desired acceleration)
a	acceleration
v	speed
x	position
l	length of the vehicle
τ	first-order lag time constant
Δ_t	simulation time step
H	time headway for inter-vehicle gap (ACC and Ploeg)
λ	weight factor between spacing and speed errors (ACC)
k_p	distance error gain (proportional gain, Ploeg)
k_d	speed error gain (derivative gain, Ploeg)
d_d	fixed, desired gap (PATH)
C_1	leader and preceding vehicle acceleration weight (PATH)
ξ	controller damping ratio (PATH)
ω_n	controller bandwidth (PATH)

The first requirement to implement an automatic control on a vehicle is knowing (and modeling) it's dynamic behavior, or, in other words, understand how the vehicle responds to an input or command normally given in terms of desired acceleration.

The simplest possible model available in PLEXE is a first order lag characterized by the following differential equation for the acceleration a and the control input u (which is the command computed by the control law or, equivalently, a desired acceleration):

$$\dot{a} = -\frac{1}{\tau}a + \frac{1}{\tau}u. \tag{1}$$

In Equation (1) τ indicates the time constant of the lag: the higher the value, the slower the response of the engine and the braking system. A typical value found in the literature is 500 ms [1]. Within the simulator, Equation (1) is implemented using the following discrete update rule

$$a[k+1] = \alpha \cdot u[k] + (1-\alpha) \cdot a[k], \quad \alpha = \frac{\Delta_t}{\tau + \Delta_t}, \tag{2}$$

where k is the simulation step and Δ_t the sampling time of the simulator. PLEXE includes more sophisticated models of vehicles' dynamics that include the role or air-drag, the engine power, the gears and so forth. As these models are

brand-and-model specific, they are not suited to derive general results, but they can be used to verify general hypotheses onto specific vehicles, and can also be used as templates to include other models and possibly to state minimum requirements that vehicles have to meet to be part of a cooperative driving system. The interested reader can find a complete description of the model in [2, Section 2.3].

In our contribution we assume that when communications are not available the vehicle falls back to a standard, radar-based ACC. ACC implementations may differ slightly one another, but they can all be modeled by the control law in Equation (3) reported in [3, Chapter 6], which assumes a constant headway-time between vehicles, as mandated with human drivers, and thus a distance between platoon vehicles that increases with speed.

$$u_i = -\frac{1}{H}\left(\dot{\varepsilon}_i + \lambda\delta_i\right) \tag{3}$$

$$\delta_i = x_i - x_{i-1} + l_{i-1} + Hv_i \tag{4}$$

$$\dot{\varepsilon}_i = v_i - v_{i-1} \tag{5}$$

In Equation (3), H is the time headway and l_i is the length of vehicle i.

We note incidentally that this control law assumes that the radar can efficiently estimate the speed difference of the own vehicle from the vehicle in front, which is not necessarily true, thus this model can be somewhat optimistic in the evaluation of ACC performance. The parameter λ controls how aggressive the ACC is with respect to the distance from the vehicle in front. We used a constant value often reported in the literature in the experiments, which, as we have shown, can easily lead to accidents in case of abrupt transition from PATH Cooperative Adaptive Cruise Control (CACC) (see Equation (7)) to ACC. A larger value of λ, or better, a dynamic λ can be used to reduce the probability of rear-end collisions, but this is clearly beyond the scope of this paper. Also exploring what happens when the control of the vehicle is given back to the human driver instead of an ACC is interesting, but to do this analysis the Intelligent Driver Model (IDM) [4] and Krauss [5] car-following models already available in PLEXE are not enough, because they do not account for the "surprise" of a driver that all of a sudden is requested to be in control of a vehicle they were not driving till the moment before.

Integral parts of our study are the Ploeg's and PATH CACC [1], [6] that we use when all three communication interfaces are available (PATH) or when only one or two are available (Ploeg).

Ploeg's CACC has been designed to mimic an advanced ACC system, thus it follows the constant headway-time model:

$$\begin{aligned}\dot{u}_i = \frac{1}{H}\left(-u_i + k_p\left(x_{i-1} - x_i - l_{i-1} - Hv_i\right)\right. \\ \left. + k_d\left(v_{i-1} - v_i - Ha_i\right) + u_{i-1}\right)\end{aligned} \tag{6}$$

Ploeg's CACC allows a smaller headway-time compared to standard ACC because it can exploit the knowledge of the front vehicle input u_{i-1}, thus discounting the actuation lag of the vehicle in front. Given its design goals Ploeg's CACC is the ideal transition controller toward an ACC, as it can smoothly increase the vehicles' headway-time to the one which is safe for the ACC system.

PATH's CACC goal is instead the maximization of performance in terms of road usage and fuel consumption: both goals require the minimum possible vehicle inter-distance independently from the speed. To achieve this goal it uses information also from the leader of the platoon and not only from the vehicle in front as described by the following equations:

$$\begin{aligned}u_i = \alpha_1 u_{i-1} + \alpha_2 u_0 + \alpha_3(v_i - v_{i-1}) \\ + \alpha_4(v_i - v_0) + \alpha_5(x_i - x_{i-1} + l_{i-1} + d_d)\end{aligned} \tag{7}$$

where

$$\alpha_1 = 1 - C_1; \quad \alpha_2 = C_1; \quad \alpha_5 = -\omega_n^2 \tag{8}$$

$$\alpha_3 = -\left(2\xi - C_1\left(\xi + \sqrt{\xi^2 - 1}\right)\right)\omega_n \tag{9}$$

$$\alpha_4 = -C_1\left(\xi + \sqrt{\xi^2 - 1}\right)\omega_n. \tag{10}$$

Differently from Equations (3) and (6), we have no headway time H but a fixed desired distance d_d regardless of the cruising speed. C_1, ξ, and ω_n are control parameters regulating the weight between leading and preceding vehicle accelerations, the damping ratio, and controller bandwidth, respectively.

PLEXE includes also other CACC models [7]–[9] that we have not used in this study, and adding others is simple given the modularity of the simulation framework.

Table 2
Network parameters.

	Parameter	Value
	Packet payload size	200 Byte
	Beacon frequency	10 Hz
802.11p	Transmission power	20 dBm
	Bit rate	6 Mbit/s
	Noise floor	–95 dBm
	Path-loss model	Free-space, $\alpha = 2$
	Frequency	5.89 GHz
VLC	Transmission power	10 dBm
	Bit rate	1 Mbit/s
	Headlight max tx range	100 m (LOS)
	Taillight max tx range	30 m (LOS)
	Headlight max tx angle	45°
	Taillight max tx angle	60°
C-V2X	Transmission power (UE)	26 dBm
	Transmission power (eNB)	40 dBm
	Frequency	2.1 GHz
	Mode	C-V2X Mode 3 (D2D, eNB assisted)
	Channel configuration	Urban macrocell (SimuLTE provided)

2 Additional Results

This section shows additional results that are not included in the main manuscript. In addition, in Table 2, we list all the main network parameters used in the simulations. Figures 1 and 2 show the acceleration and distance behavior for the naïve fallback mechanisms for scenarios 2 and 3 respectively. It is clear in both cases that the crashes we observe are not a remote chance, but they are intrinsic to an abrupt change from a cooperative driving situation, where vehicles exploit the knowledge of each other dynamics and intentions, to

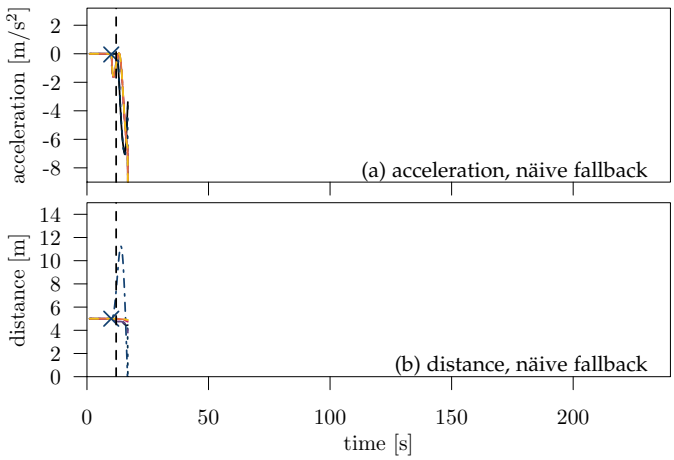


Figure 1. Acceleration and inter-vehicle until the crash for the näive fallback in scenario 2.

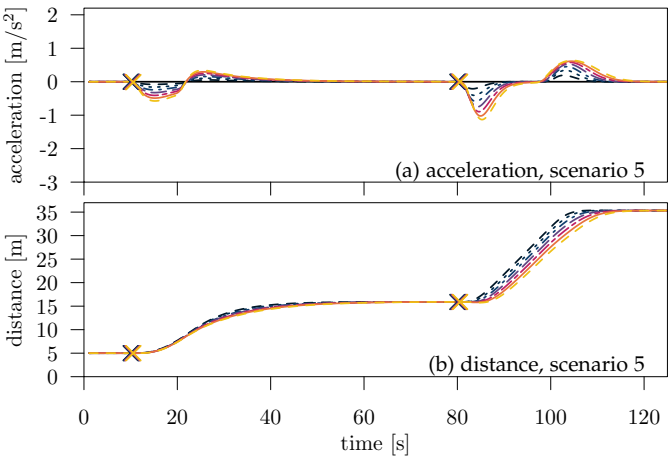


Figure 3. Acceleration and inter-vehicle distances for scenarios 5 (multiple failures).

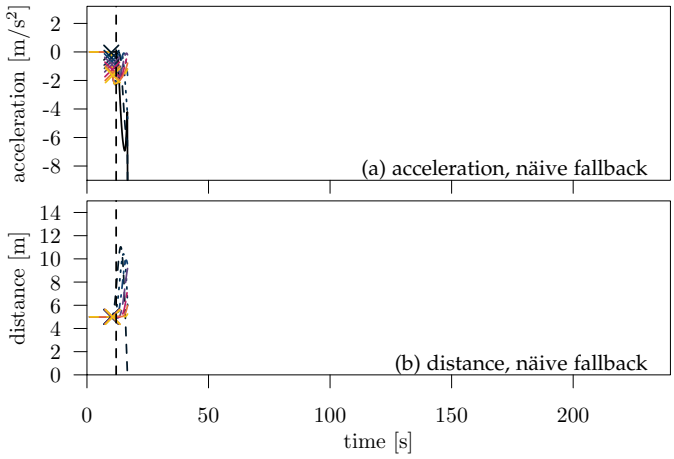


Figure 2. Acceleration and inter-vehicle until the crash for the näive fallback in scenario 3.

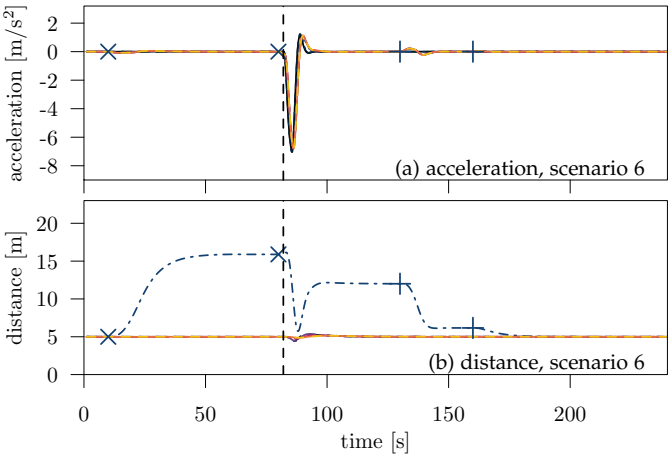


Figure 4. Comparison of acceleration and inter-vehicle distances for scenarios 6 (multiple failures).

one of autonomous driving, where vehicles can rely only on the local sensors. The crashes occur immediately after the communication failure with the emergency braking; in practice none of the vehicles attempt a reaction to V_0 abrupt braking, and collisions occur. The experiments stop as soon as one vehicle collides.

Figure 3 presents the performance of the proposed Safe Autonomous Switchover Algorithm (SafeSwitch) for scenario 5, i.e., when a technology fails for all the vehicles in the platoon at the same time and with failure of two technologies. This can happen in rare cases, for instance if there is a jamming attack on IEEE 802.11p, or because an LTE base station fails, thus stopping to serve the entire platoon. Clearly the occurrence of both cases is extreme. SafeSwitch works as intended, safely distancing vehicles without abrupt accelerations. PLEXE highlights the same amplification of the deceleration observed in scenario 1 (main paper). The amplification is however more severe and this is due to the fact that a different controller (i.e., Ploeg) is used, so the behavior is slightly different.

The situation of scenario 6, multiple failures and recoveries at the same vehicle with an emergency brake after the second failure, leads to the results presented in Figure 4. The deceleration due to the emergency braking is abrupt, but this is unavoidable, and during the braking maneuver the distance of V_3, the one with failing communications, reduces drastically, but remains safe, and then quickly returns to the distance dictated by Ploeg's controller in state F_1, after the communication recovery the platoon returns to normal cruising with PATH's controller.

Finally, Figures 5 and 6 present two additional runs relative to the scenario with realistic communication failures. It is evident that the stochastic pattern of losses affects the quantitative behavior of the platoon, as the dynamic patterns observed in the lower plots of these figures are different and are also different from those reported in the main paper. However, from a qualitative point of view the behavior is consistent, with IEEE 802.11p and LTE losses driving the performance, with Visible Light Communication (VLC) ones often induced by the increased distance between vehicles, a situation that can be recovered only if the other two technologies work well for a long enough period of time, as can be observed in Figure 6. We also observe that handovers sometimes result in burst of losses and sometimes not, hinting that handover procedures can still be improved. This is very important, in light of 5G advent, where with a high density of gNodeB and very frequent handovers, it is important that these latter are loss-free for cooperative

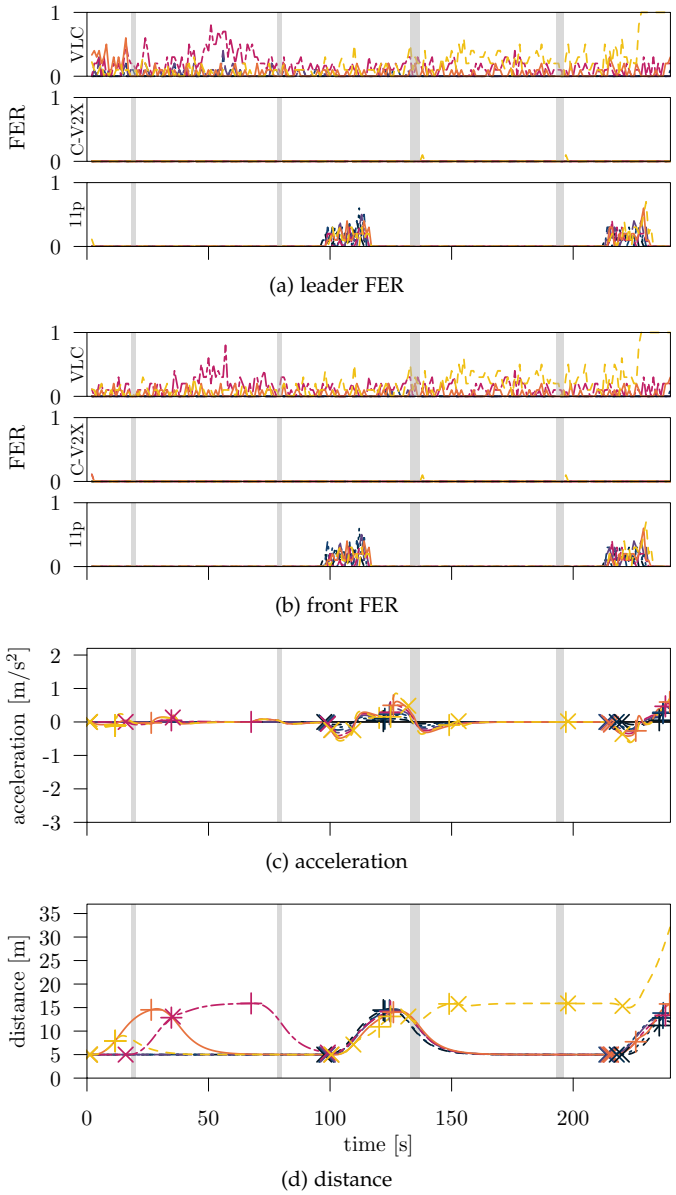


Figure 5. Vehicle dynamics induced by the fallback FSM and frame error rates for the scenario with realistic failures, second replica.

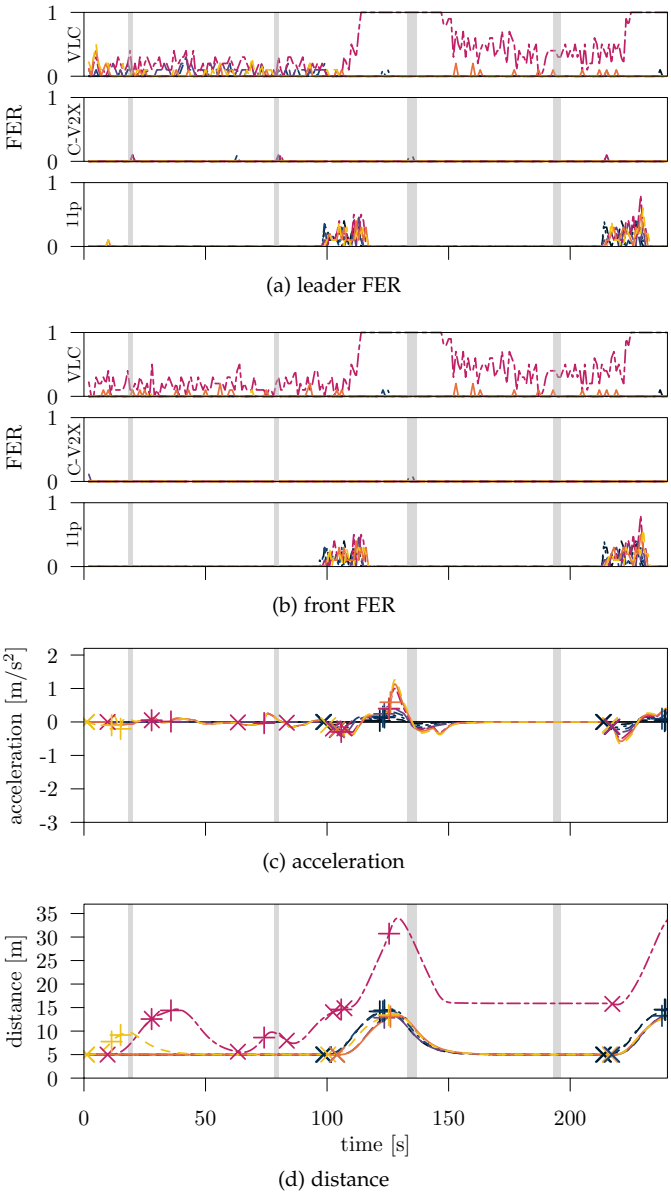


Figure 6. Vehicle dynamics induced by the fallback FSM and frame error rates for the scenario with realistic failures third replica.

driving applications.

3 Sample use cases and subprojects

This section describes the structure of PLEXE's code base for future users, together with the default scenarios it provides. The following description assumes the user to be familiar with the OMNeT++ ecosystem. PLEXE is divided in three main sub-folders: main source code, examples, and subprojects. The implementation of all the functionalities is found under the `src/plexe` folder, starting from the root, and this is where the core of PLEXE resides. The `examples` folder includes all the sample simulations users can play with to get acquainted with the frameworks. The `subprojects` folder, instead, includes the source code and examples that involve external libraries such as VLC or Cellular V2X (C-V2X) modules. By default, PLEXE depends only on SUMO and *Veins*, and the other components are optional.

3.1 Flexibility through dynamic linking

Optional components are managed through a `configure` script, which is present in each subproject. The role of such script is to configure and check the dependencies of the subproject (that is, other frameworks such as *Veins VLC* or *SimuLTE*), generate the `Makefile` needed for building, and the files required to run simulations. To better explain this, we describe the `veins_vlc` subproject, which includes example simulations like the ones provided by basic PLEXE, but its vehicles are using VLC to communicate. We are going to consider the file system structure in listing 1, which shows a partial view of the folder structure and the most relevant files. Listing 2 shows a portion of the `configure` script. Lines 2 to 18 define the dependencies which, in this specific case, are PLEXE, *Veins*, and *Veins VLC*. For each dependency the user needs to provide:

- `name`: a name of the dependency;

- `library`: the name of the shared library which the subproject will be linked against;
- `default_path`: the default root path where the library is located. The user can override this via command line arguments when running the script;
- `versions`: a list of accepted versions of the library;
- `source_folder`: the location (starting from the root path of the library) where the source files are located;
- `lib_folder`: the location (starting from the root path of the library) where the compiled shared library is located;
- `images_folder`: the located (starting from the root path of the library) where additional images provided by the library are located. It is possible to make use of such images (e.g., a car or a pedestrian) when running the OMNeT++ simulation in graphical mode;
- `version_script`: name of a script that prints (or a file that contains) the version of the library.

Lines 20 to 24, instead, define the properties of the `plexe_vlc` subproject itself, i.e., the name of the shared library (being built), and the parameters required to run simulations (the shared library itself and where OMNeT++ will find the `.ned` files).

The script automatically checks for the presence of all the components and that the versions correspond to the required ones, generating an error for the user if such task fails. Upon a successful execution, the script generates a `Makefile` for building the subproject and the script required to run the simulation, which automatically indicates to OMNeT++ where to find all the dependencies. The user can now develop custom OMNeT++ modules (in `src/plexe_vlc`) and simulations (in `examples/platooning_vlc` or any other folder). For the former step, it is important to remember that if the user adds new C++ source files, the `configure` scripts needs to be run again to update the `Makefile`. For example, `PlatoonVlcCar.ned` defines an OMNeT++ communication node representing a vehicle which, differently to standard PLEXE, uses a VLC interface. As the configure script automatically resolves the dependencies, the user can now import the modules provided by *Veins VLC* and sobstitute the 802.11p interface with VLC. The user has clearly the possibility to add a new interface to study heterogeneous communication systems.

To run a simulation, after defining all the classic OMNeT++ configuration files such as `omnetpp.ini` in a folder of choice, the user can simply exploit the script generated by `configure`. For example, running `../../bin/plexe_vlc_run` from the `examples/platooning_vlc` folder will start the OMNeT++ GUI, permitting the user to choose which simulation to run. The user can also specify to run through command line arguments, as in standard PLEXE.

This brief description shows how flexible PLEXE is even considering its complexity, and that integrating a new communication technology or any other OMNeT++ framework simply requires to add a dependency to a project.

3.2 Base scenarios

Base PLEXE scenarios are located in the `examples` folder and the main one resides inside the platooning subfolder. This example simulates two scenarios using the available ACC and CACC algorithms. The scenarios include a sinusoidal speed profile, a classical test of the stability of control systems, and an emergency braking scenario, which is of obvious interest for safety reasons. Obtaining simple performance metrics of control systems starting from the examples is just a matter of changing the parameters. For example, by reconfiguring the emergency braking scenario and introducing an artificial frame error rate ranging from 0 % to 80 %, we obtain the results in Figure 7. In the scenario we have a platoon of 8 vehicles with the leader performing an emergency braking maneuver with a deceleration of $8\,\mathrm{m/s^2}$. We repeat each simulation point 10 times and we compute the minimum inter-vehicle distance between each pair of vehicles. We then plot the average over the 10 repetitions with the relative 95 % confidence intervals. In the graph, we only show the results for the PATH [6] and the Ploeg [1] CACCs.

```
1  /Users/user/src/
2  |-- plexe/
3  |   |-- print-plexe-version
4  |   |-- src/
5  |   |   |-- libplexe.dylib
6  |   |   |-- plexe/
7  |   |-- subprojects/
8  |       |-- plexe_vlc/
9  |           |-- configure
10 |           |-- bin/
11 |               |-- plexe_vlc_run
12 |           |-- examples/platooning_vlc/
13 |           |   |-- omnetpp.ini
14 |           |-- src/plexe_vlc/
15 |               | PlatoonVlcCar.ned
16 |               | VlcRepropagationProtocol.cc
17 |               | VlcRepropagationProtocol.h
18 |               | VlcRepropagationProtocol.ned
19 |-- veins/
20 |   |-- print-veins-version
21 |   |-- src/
22 |       |-- libveins.dylib
23 |       |-- veins/
24 |-- veins-vlc/
25     |-- print-veins_vlc-version
26     |-- src/
27         |-- libveins-vlc.dylib
28         |-- veins-vlc/
```

Listing 1. File system structure (only most relevant folders and files shown).

The main differences between the two is that the PATH CACC employs a leader- and predecessor-following control topology with a fixed inter-vehicle spacing policy (which in our simulation is set to 5 m), while Ploeg's CACC employs a predecessor-following control topology with a time-headway spacing policy of the form

$$d = d_0 + Hv, \tag{11}$$

where $H = 0.5\,\mathrm{s}$ is the time-headway and v is the speed in m/s. The $d_0 = 2\,\mathrm{m}$ term is defined as the stand-still distance, which avoids the vehicles colliding with each other when the speed goes to zero.

What the figure shows is, first of all, the robustness of CACCs to packet losses. The performance is unaffected for losses up to 20 %, and for vehicles to become dangerously close we need to have packet loss rates higher than 50 %.

This is a very basic scenario, but it is of extreme interest in the cooperative driving community, and obtaining such

```
[...]
plexe = Library(name="Plexe", library="plexe", default_path="../../",
                versions=["3.0"], source_folder="src/plexe",
                lib_folder="src", images_folder="images",
                version_script="print-plexe-version")
veins = Library(name="Veins", library="veins", default_path="../../../veins",
                versions=["5.1"], source_folder="src/veins", lib_folder="src",
                images_folder="images", version_script="print-veins-version")
veins_vlc = Library(name="Veins␣VLC", library="veins-vlc",
                    default_path="../../../veins_vlc", versions=["1.0"],
                    source_folder="src/veins-vlc", lib_folder="src",
                    images_folder="images",
                    version_script="print-veins_vlc-version")

libraries = LibraryChecker()
libraries.add_lib(plexe)
libraries.add_lib(veins)
libraries.add_lib(veins_vlc)

makemake_flags = ["-f", "--deep", "--no-deep-includes", "--make-so", "-I", ".", "-o", "plexe_vlc",
                  "-O", "out", "-p", "PLEXE_VLC"]
run_libs = [join("src", "plexe_vlc")]
run_neds = [join("src", "plexe_vlc")]
run_imgs = []

libraries.check_libraries(makemake_flags, run_libs, run_neds, run_imgs)
[...]
```

Listing 2. Partial content of the configure file of the veins_vlc subproject.

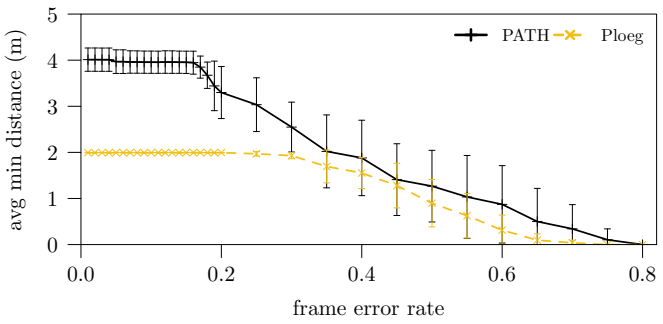


Figure 7. Average minimum inter-vehicle distance with 95 % confidence intervals in an emergency braking scenario as function of the frame error rate for the PATH and the Ploeg CACCs.

results is a matter of a few minutes.

Another sample scenario of interest is the aforementioned join at back maneuver. In this example, a lone vehicle joins an existing platoon of 4 cars by first requesting permission to join, then being instructed to approach the platoon, and finally joining the platoon.

Other examples include a scenario where a human vehicle is inserted into the simulation, to show how to simulate mixed scenarios where human-driven vehicles can interfere from a mobility and a communication perspective. In addition, we find a simple scenarios showing the behavior of the realistic engine model by simulating a drag race between three different types of vehicle.

3.3 Heterogeneous scenarios

PLEXE includes sample scenarios for the subprojects, which deals with VLC and C-V2X. In the VLC subproject, we have the same main scenario (sinusoidal plus emergency braking) but using VLC as the underlying communication technology as previously described. In addition, there is a modified communication protocol that performs re-propagation of the beacon messages as VLC works only in direct line-of-sight.

In the C-V2X subproject, instead, we have two platoons running on a ring-like freeway that are located far apart. The two platoons use the LTE uplink/downlink standard to communicate with a centralized Traffic Authority (TA) server. One platoon queries the TA for other platoons and, upon receiving the reply, requests the TA guidance to approach such platoon with the aim of merging. Once the approaching platoon is close enough to the other, the TA demands the former to contact the latter and performs the merge maneuver autonomously using direct Vehicle to Vehicle communication (V2V) communication via IEEE 802.11p. In addition, V2V beacons with control information are sent redundantly using C-V2X Mode 3. This example is particularly important as it shows how vehicles can be coordinated on different levels (locally with V2V communication and remotely through the infrastructure) and how easily PLEXE enables cooperative driving studies with heterogeneous networking technologies.

The final subproject (named `plexe_hetnet`) includes a ring road scenario with a platoon of vehicles using multiple communication technologies simultaneously to communicate in a V2V fashion. This is the scenario on top of which we developed the evaluation of SafeSwitch, and it can be extremely useful to users to understand how to integrate multiple communication technologies into their cooperative driving scenarios or even integrate additional simulation frameworks.

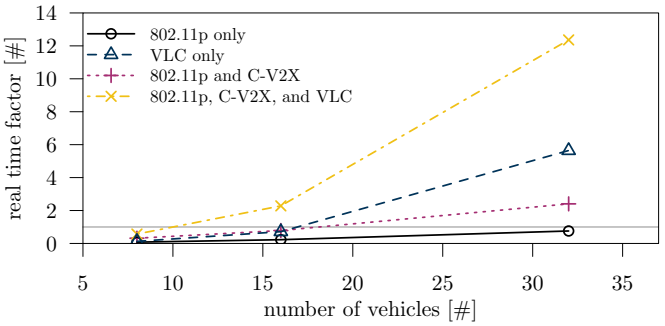
One fundamental aspect which always represents a parameter of choice when selecting a simulation framework is scalability. The scalability of PLEXE heavily depends on the communication models required for the analysis. Some of them can be very demanding in terms of computation. For example, *Veins VLC* employs geometrical models to compute the effect of vehicle shadowing, which clearly cannot be

neglected. In the past, when considering IEEE 802.11p only, we could easily simulate hundreds of vehicles [10], but such a large number of vehicles would be difficult to handle when using very detailed communication models. To give the reader an idea, we run a set of simulations with a single platoon composed by 8, 16, and 32 cars, using different communication technologies, and measuring the execution time. In the simulation, each vehicle sends 10 broadcast frames per second. Figure 8 plots the real time factor of the simulation, which is defined as follows. Let t_s and t_w the time elapsed in the simulation and in the real world, respectively. The real time factor is defined as $f = t_w/t_s$. For example, if $f = 0.1$, it means that simulating 1 s requires 100 ms in the real world. On the contrary, if $f = 10$ the user needs to wait 10 s for each second within the simulation. Simplistically speaking, the lower the value of f, the better. Clearly, the real time factor heavily depends on the hardware. The results in Figure 8 are obtained on a 2018 MacBook Pro with an Intel i9 processor. Running the simulations on a different hardware would quantitatively change the results, but qualitatively they should be, in general, the same.

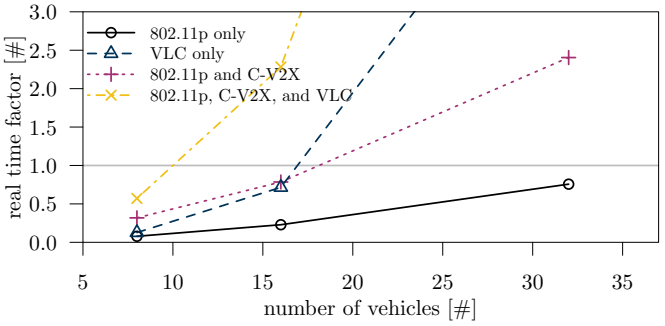
First of all, the plot highlights a well-known fact about network simulations, i.e., that the real time factor increases more than linearly in the number of network nodes. This is true when all nodes communicate with each other because, disregarding optimizations, the communication model needs to compute the probability of reception for each node in the simulation, so the complexity intuitively increases quadratically. Overall, in the worst case, i.e., with 32 cars and 3 simultaneous communication technologies, the real time factor is roughly 12, meaning that it requires 120 s in the real world to simulate a scenario of 10 s in PLEXE.

The graph finally highlights the impact of communication models: the more the technologies, the higher the simulation time. The IEEE 802.11p model provided by *Veins* is the most efficient one. When using only this technology, even a simulation with 32 vehicles runs faster than real time. What it is interesting to observe is the huge impact of the VLC model. For 8 and 16 vehicles, the VLC model is faster than the simulations using IEEE 802.11p and C-V2X, but then its real time factor drastically increases for the simulation with 32 cars. As mentioned before, the VLC channel layer requires geometrical computations to calculate the effects of shadowing. For a small number of vehicles, the computational effort is limited, but this quickly grows as we increase the nodes in the simulation, dominating over all the other algorithmic components.

These results show that PLEXE can potentially scale but there is clearly a limitation induced by the models that researchers need to consider. It is thus necessary to select the communication models and the scale of the simulation depending on the required granularity. If a study focuses on the physical layer and requires very low level channel details (e.g., ray tracing), it will certainly be unfeasible to run a simulation with hundreds of cars. On the other hand, if the user is interested in traffic-related metrics for which a large number of vehicles is required (e.g., throughput), considering multiple communication technologies might not be necessary. It is thus needed to find the right balance depending on the requirements but, as we have clearly shown, the flexibility of PLEXE enables its users to easily tune and configure the simulator for their purposes.



(a) full y axis



(b) zoomed y axis

Figure 8. Simulation real time factor as function of the number of vehicles, for different communication models. The gray line highlights a real time factor of 1.

REFERENCES

[1] J. Ploeg, B. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and Experimental Evaluation of Cooperative Adaptive Cruise Control," in *IEEE ITSC 2011*, Washington, D.C.: IEEE, Oct. 2011, pp. 260–265.

[2] M. Segata, "Safe and Efficient Communication Protocols for Platooning Control," PhD Thesis, University of Innsbruck, Innsbruck, Austria, Feb. 2016.

[3] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. Springer, 2012.

[4] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations," *PRE*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000.

[5] S. Krauß, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *APS Physical Review E*, vol. 55, no. 5, pp. 5597–5602, May 1997.

[6] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of Integrated Longitudinal and Lateral Control for the Operation of Automated Vehicles in Platoons," *TCST*, vol. 8, no. 4, pp. 695–708, Jul. 2000.

[7] A. Ali, G. Garcia, and P. Martinet, "The Flatbed Platoon Towing Model for Safe and Dense Platooning on Highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 58–68, Jan. 2015.

[8] S. Santini, A. Salvi, A. S. Valente, A. Pescapè, M. Segata, and R. Lo Cigno, "A Consensus-based Approach for Platooning with Inter-Vehicular Communications and its Validation in Realistic Scenarios," *TVT*, vol. 66, no. 3, pp. 1985–1999, Mar. 2017.

[9] G. Giordano, M. Segata, F. Blanchini, and R. Lo Cigno, "The joint network/control design of platooning algorithms can enforce guaranteed safety constraints," *Elsevier Ad Hoc Networks*, vol. 94, Nov. 2019.

[10] M. Segata, B. Bloessl, S. Joerer, et al., "Towards Communication Strategies for Platooning: Simulative and Experimental Evaluation," *TVT*, vol. 64, no. 12, pp. 5411–5423, Dec. 2015.