Chapter 7
Towards Predictable Vehicular Networks

Elad Michael Schiller

Abstract Communication primitives consider information delivery with different guarantees regarding their reliability. The provision of reliability and predictability needs to overcome a number of challenges with respect to failures and a number of known impossibility results. This chapter covers a number of these challenges in the context of vehicular systems and networks. We start by showing the medium access control (MAC) protocol for wireless mobile ad hoc networks can recover from timing failures and message collision and yet provide a predictable schedule in a time-division fashion without the need for external reference, such as commonly synchronized clock. We then consider the case of transport layer protocols and show how to deal with settings in which messages can be omitted, reordered and duplicated. We also consider how mobile ad hoc networks and vehicular networks can organize themselves for emulating virtual nodes as well as emulating replicated state-machines using group communication. In this context, we discuss the different alternatives for overcoming well-known impossibilities when considering cooperative vehicular applications. Finally, we exemplify applications and discuss their validation.

7.1 Introduction

Recent algorithmic developments provide enablers for designing and demonstrating cyber-physical vehicular systems for safety-critical applications that base their decisions on (uncertain) sensory information and yet function safely in presence of failures, such as (unbounded) communication delays. In this chapter, we review these recent developments and discuss their applications.

Cyber-physical vehicular systems require positioning information about the road layout, obstacles, hazards, nearby vehicles, road users to name a few. According to this information, the system cooperatively decides on its joint vehicular

E.M. Schiller (✉)
Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
e-mail: elad@chalmers.se

M. Alam et al. (eds.), *Intelligent Transportation Systems*,
Studies in Systems, Decision and Control 52,
DOI 10.1007/978-3-319-28183-4_7

manoeuvres, which is often based on inter-vehicle exchange of sensory information and the combination between onboard and remote sources in order to reduce uncertainty, achieve completeness, provide dependability, as well as the integrate of perspectives and multi-dimensional viewpoints. The resulting (fused) sensory information is often better than what would be possible when merely using onboard sources. Recent developments in this field as well as sensory technology have opened the door for the automotive industry to consider using affordable sensors and inter-vehicle communications for cooperative safety-critical applications. Unfortunately, sensory information that originates from affordable onboard sensors increases significantly the position information uncertainty. Moreover, since inter-vehicle communications are prone to unbounded delays, their use implies arbitrary disconnection from remote information sources. Consequently, the current practice mostly relies on onboard sources rather than leveraging on remote sensory sources.

There are known methods for estimating the quality sensor information sources as well as their fused result. Based on these estimations, the cooperative vehicular systems are to choose, within a real-time constraint, exactly one functionality out of a set of (distributed) vehicular functionalities. We call this selected functionality the *operation level* of the (cooperative) vehicular system. Operational deadlines are imperative constraints of any safety-critical (cooperative) system. The functionality set could be, for example, the implementation of vehicular platooning. Here the system has to decide on the headings (or the inter-vehicle distances) and whether the vehicles are allowed to accelerate as well as to what extent so that a safe and comfort ride is guaranteed. The choice could be based, for instance, on the common uncertainty bound that all system vehicles can support. We call this bound the *information validity level* and point out that the selection of a single functionality can encode a complex formations, manoeuvres and behaviour in which each vehicle can anticipate the behaviour of nearby vehicles, with respect to the bounds that they use for estimating the uncertainty of the sensory information.

Some of the existing cooperative vehicular systems are based on *implicit communication* methods in which the vehicles use onboard sensors for anticipating the approaching vehicle intentions and by that aim at making sure that all vehicles indeed agree on who has the right to cross the intersection. Note that both implicit and explicit (inter-vehicle) communications are prone to interferences. However, the latter approach allows each vehicle to inform nearby vehicles about its (immediate) intensions. Virtual Traffic Lights (VTLs), for example, can use explicit communication for dynamically scheduling their green light phases. The challenges here include the need to facilitate (successive and) coherent decisions in the presence failures, such as (unbounded) communication delays. For instance, VTLs can only give a green light for vehicles coming from one direction after it gave red to all conflicting directions. Moreover, intermediate yellow periods are required between red and green periods. One of the key obstacles that we plan to overcome is the impossibility to decide uniformly in the presence of failures, such as (unbounded) communication delay [1, 2]. In other words, the problem to circumvent here is how not to allow communication failures to create a split brain phenomena in which, in the case of virtual traffic lights, different vehicles decides on different traffic light schedules.

One may seek the solution for such problems by using infrastructure-based services, such as group communication systems. These services can help to follow the presence of vehicles, i.e., location and membership. Note that such systems were also proposed for ad hoc networks [3]. In this chapter, we consider enables that follow both approaches.

We review recent results about new ways to circumvent the impossibility to choose uniformly a single value, i.e., the operation level of the cooperative system (Sect. 7.5). By that we can to significantly simplify the design of complex cooperative functionalities, for example, when scheduling lane changes, intersection crossing and going through roundabouts. The challenge referred to here is how to design (distributed) synchronization (control) mechanisms that can support successive cooperative decisions that are based on uncertain sensory information and perform coherently in presence failures, such as (unbounded) inter-vehicle communication delay.

7.1.1 The Self-stabilization Design Criteria

Large and dynamic networks are hard to control and it is challenging to provide network protocols with predictable behaviour. Very important design criteria for the implementation of communication services are their fault tolerance and robustness. However, the distributed algorithms that implement these communication protocols often assume a particular set of possible failures, such as crash failures, link failures, or message loss. The correctness of these implementing algorithms is proved by assuming a predefined initial state and considering every possible execution that involves the assumed possible set of failures. This abstraction, which limit the set of possible failures allows a more convenient correctness demonstration. However, it is also too restrictive. Communication protocols are often long-lived, on-line services for which it is hard to predict in advance the exact set of possible faults. Moreover, when communication delays are unbounded, and the chances for packet drop are high, it may be the case that due to the occurrence of an unexpected fault the system reaches a state that is not attainable from the initial state by steps of the algorithm and the occurrence of the assumed fault model. Therefore, self-stabilizing systems [4, 5] can be started in any arbitrary state, and they will exhibit the desired behavior after a convergence period.

Another important benefit of the self-stabilizing design criteria is the system ability to offer automatic recovery from unexpected transient failures, such as a temporary violation of the assumptions that were made by the system designers. As an illustrative example, let us consider the use of probabilistic error detection codes for ensuring that the arriving packets are identical to the ones sent. It can happen that the error detection eventually fails to detect a corrupted message. The system then regards a corrupted message as a legitimate one. This might bring the system to an arbitrary state for which, in the case of non-self-stabilizing systems, there are no guarantees for service functionality and availability (without human intervention).

7.1.2 Chapter Roadmap

This chapter covers a number of challenges in the context of vehicular systems and networks. We start by showing that medium access control (MAC) protocols for wireless mobile ad hoc networks can recover from timing failures and message collision and yet provide a predictable schedule in a time-division fashion without the need for external reference, such as commonly synchronized clock (Sect. 7.2). We then consider the case of transport layer protocols and show how to deal with networks in which messages can be omitted, reordered and duplicated (Sect. 7.3). We also consider how mobile ad hoc networks and vehicular networks can organize themselves for emulating virtual nodes as well as emulating replicated state-machines using group communication services (Sect. 7.4). In this context, we discuss the different alternatives for overcoming well-known impossibilities when considering cooperative vehicular applications (Sect. 7.5). Finally, we exemplify applications (Sect. 7.6) and discuss their validation (Sect. 7.7).

7.2 Self-stabilizing MAC for Wireless Ad Hoc Networks

One of the key enablers of predictive communication is having a media access control (MAC) layer, with predictable behaviour. Namely, after a convergence period and in the absence of external interferences, each node should be able to access the network within a bounded communication delay. We discuss several recent development that allows the system to increase their predictability degree when using wireless ad hoc networks.

Mustafa et al. [6], Leone et al. [7–9] and Petig et al. [10] suggest algorithmic designs for stabilizing MAC algorithms with an emphasis is on providing resilience and predictability. Such algorithms are required for ad hoc vehicular networks.

In the context of predictable vehicular ad hoc network (VANET), which have frequent topological changes, MAC protocols need to be self-stabilizing, have low communication delays and high bandwidth utilization. We propose a self-stabilizing MAC algorithm that guarantees to satisfy these severe timing requirements.

In the context of TDMA, the timing alignment of packet transmissions helps to avoid transmission interferences. Existing VANET implementations often assume the availability synchronized clocks, e.g., GPS signals. Mustafa et al. [6] consider autonomic design criteria, and present a (probabilistic) self-$\star$ broadcasting timeslot alignment for ad hoc wireless networks that follow the time division multiple access (TDMA) approaches. In these networks, the radio time is divided into timeslots, wherein each such timeslot a subset of the network node are allowed to transmit, such that the interference degree among of concurrent transmissions among the nodes in these subsets in kept low. The algorithm by Mustafa at al. [6] can make sure that these timeslots are well aligned.

Leone et al. [8] assumes such timeslot alignment and present a (probabilistic) self-stabilizing algorithm for scheduling transmissions among nodes, such that no two neighbouring nodes transmit concurrently. The authors prove a rapid stabilization, and by that, the algorithm allows greater degree of predictability, while maintaining high throughput and low communication delays.

During the stabilization period, several nodes can have the same assigned timeslot. The algorithm solves such timeslot allocation conflicts via a (listening/signalling) competition in which node p_i and node p_j participate before transmitting in their broadcasting timeslots. The competition requires p_i and p_j to select one out of n listening/signalling periods their timeslots. Note that among all the nodes that aim at broadcasting in a particular timeslot, the ones who win and access the communication media are the ones that select the earliest listening/signalling period. Prior to accessing their timeslots, the winners inform to their neighbourhoods about their win by broadcasting beacons during their selected signalling periods. Upon beacon reception, a node defer from transmitting during that timeslot, since it lost the competition. After a back-off period, the losing nodes compete on the next broadcasting round for their new timeslots by selecting them randomly.

Petig et al. [10] present protocols for dynamic wireless ad-hoc networks that allocates the timeslot while considering the transmission timing aspects of the problem. They show that the solution existence depends on the ratio, τ/δ, between the frame size τ (which is the number of timeslot in each TDMA frame) and the node degree δ in the communication graph. They prove that $\tau/\delta \geq 2$ is required for any (eventually) collision-free TDMA algorithms and present a (probabilistic) algorithm for the case of $\tau/\delta \leq 4$.

The exposure period of a packet is the period during which it may be co-transmitted with other packets from nearby nodes. In the absence of an external reference, the TDMA algorithm has to concurrently align timeslots while allocating them. Petig et al. [10] show that $\tau/\delta \leq 4$ is sufficient for guaranteeing zero exposure period with respect to a single timeslot, s, and a single receiver, rather than all neighbouring transmitters. The algorithm considers nodes that transmit data and control packets. Data packets are sent by active nodes during their data packet timeslots while passive nodes listen to the actives ones and do not transmit. Both active and passive nodes use control packets, which include frame information about the recently received packets. The algorithm uses this information for avoiding collisions, acknowledging packet transmission and resolving hidden node problems.

7.3 Self-stabilizing End-to-End Protocols

End-to-end communication protocols is an important reliable communication enabler for any type of network, including (vehicular) mobile ad hoc networks. The end-to-end protocol handles the exchange of packets between pair of network nodes, which do not necessary communicate directly and thus need the assistant of relay nodes. Where as the relay nodes perform the packet forwarding operations, it is up to the

nodes at the forwarding path ends to make sure that the packet are delivered to their destination in exactly the same order in which the source has sent them.

Network protocols use techniques, such as retransmissions and multi-path routing, to increase their robustness and over packet omissions. These techniques can create anomalies, such as packet reordering and duplication. Dolev at el. [11] present end-to-end algorithms for dynamic networks that makes sure that the receiver at the destination node delivers the same sequence of (high level) messages. The algorithm can be applied to networks for which there is a bound on the number of packets that can reside in them their capacity that omit, duplicate and reorder packets.

We outline the algorithm's basic ideas while sketching an algorithm with a large message overhead. The detailed algorithm uses error correction codes and has a smaller overhead, see [11] for details and [12] for a self-stabilizing end-to-end protocol in the presence of Byzantine nodes.

7.3.1 The Algorithm Sketch

Let us consider a sender, p_s, and receiver p_r nodes. Node p_s needs to fetch messages and send them to p_r, which in turn needs to deliver m the order in which it was sent. Once p_s fetches m, it starts transmitting $2 \cdot capacity + 1$ copies of it, and p_r acknowledges them, where $capacity$ is the network capacity. These transmissions use labels that are distinct from each copy. The sender does not stop retransmitting till it receives from p_r $(capacity + 1)$ different labeled acknowledgments, where the majority of them are copies of m. Namely, p_s maintains an alternating index, $AltIndex \in [0, 2]$, which is a three state counter that is incremented upon fetching a new message. Moreover, p_s transmits a set of packets, $\langle ai, lbl, dat \rangle$, where $ai = AltIndex$, and lbl are packet labels. This transmission ends once p_r receives a packet set, $\{\langle 0, \ell, dat \rangle\}_{\ell \in [1, 2 \cdot capacity + 1]}$, that is distinctly labeled by ℓ with respect to the alternating index. After recovering from any transient failure, the set of received packets includes a majority of packets that have the same value of dat. When that happens, p_r delivers m and updates the value of the last delivered alternating index.

The correct packet transmission depends on the synchrony of m's alternating index at the sending-side, and $LastDeliveredIndex$ on the receiver side, as well as the packets that p_r accumulates in $packet\_set_r$. Node p_s repeatedly sends its packet set until it receives $(capacity + 1)$ distinctly labeled acknowledgment packets, $\langle ldai, lbl \rangle$, for which it holds that $ldai = AltIndex$. Node p_r acknowledges each incoming packet, $\langle ai, lbl, dat \rangle$, using acknowledgment packet $\langle ldai, lbl \rangle$, where $ldai$ refers to the value of the last alternating index, $LastDeliveredIndex$. That that the receiver also delivers this packet.

On the other-side, node p_r delivers $m = \langle dat \rangle$, from one of the $(capacity + 1)$ distinctly labeled packets that have identical dat and ai values. Then, p_r assigns ai to $LastDeliveredIndex$, empties its packet set and restarts accumulating packets, $\langle ai', lbl', dat' \rangle$, for which $LastDeliveredIndex \neq ai'$.

7.4 Self-stabilizing Group Communication

Group communication systems provide high level communication primitives that enable nodes that share a collective interest, to identify themselves as a single logical communication endpoint. Each such endpoint is named a group, and each group has a unique group identifier. Nodes may join a leave the group and it is up to the membership service to provide the current group view that is uniquely identified. The view, thus, includes the group membership set and the view identifier. The group multicast service can the multicast messages the group and collect its acknowledgement after its delivery to all of the view members. After reviewing the relevant self-stabilizing literature on group communication, we present solutions that are dedicated to mobile ad hoc (vehicular) networks and vehicular networks that also include communication infrastructure.

7.4.1 Infrastructure-Based Approaches

The first algorithmic design of self-stabilizing group communication system includes the ones for undirected [13] and directed networks [14]. The proof of correctness demonstrates convergence after the last transient fault, such as a crash failure or any other topological change to the network graph. In the presence of failures, such as fail-stop crashes and unbounded communication delays, it is not possible to guarantee message delivery to all members of the sending-view. The property of virtual synchrony [15] allows all system events, view changes and multicast messages, to be delivered in the same order. The virtual synchrony property requires that any two nodes that are members of two consecutive views of communicating groups shall deliver the set of system events, e.g., multicast messages. This property makes it easier to vehicular applications that, for example, are based on state-machine replication [16–18].

Recently, it was shown how to design a self-stabilizing group communication system and how that system can emulate state-machine replication [19] using a self-stabilizing emulator of multi-reader multi-writer registers over message passing systems, similar to the ones by [20, 21]. The emulation of a replicated state-machine is a way to let the nodes to periodically exchange their current state and their current input by sending multicast messages. Once all multicast messages and their acknowledgements are delivered, the nodes can verify that they all share the same state, apply the new input to the current state and by that get the new state.

7.4.2 Infrastructure-Less Approaches

Self-stabilizing state-machines replication is also in the heart of high-level communication primitives, such as virtual (mobile) nodes [22–25]. The idea is to emulate replicates in geographic regions, say, by tilling the area, such that each tile has its

own replicated stationary automata [26]. Any nodes that enters a tile starts emulating the tile's automata in the manner of state machines replication. Virtual mobile nodes consider the case in which the tiles are moving according to a deterministic function that is based on time, as in [24, 25] or also the environment input [22, 23]. The design of this virtual infrastructure has inspired the idea about virtual traffic light in a junction as well as other applications, which we discuss in Sect. 7.6.

We note that the first algorithmic design for (self-stabilizing) group communication systems for ad hoc networks [27] was mobile agents, collecting and distributing information, during their many random walks. They eventually elect a single agent that is the basis of the group membership and multicast services, i.e., it collects and distributes information. Several systems were developed based on this approach, such as RaWMS [28] and Pilot [29].

7.5 Vehicular Coordination in the Presence of Failures

Vehicular networks are the basis for providing high-performmance cooperative vehicular systems while assuring safety standards. The vehicles determine their maneuver strategies according remote sensory information together with its quality, i.e., information validity. Since radio communications are prone to failure, it is unclear how to assume a joint awareness of timely message reception. Using such joint awareness, the planning of conflict-free trajectories becomes earlier. Morales et al. [30, 31] present a timed deterministic communication protocol that facilitate cooperative vehicular functionality in the presence of failures. After reviewing their protocol and its related properties, in this paper we discuss the protocol application for cooperative (vehicular) systems. Such applications, can facilitate the development of automated driving system, which have to work around the (communication) uncertainties that failure-prone communications bring in.

7.5.1 Problem Description

Vehicle-to-vehicle communications are the way to allow automated driving system to become affordable by raising the confidence level on the sensory information. This information confederate is at the heart of advanced cooperative (vehicular) functionalities, such as lane changing and intersection crossing, as well as busting the road capacity [32]. It is imperative that such cooperative system are able to deal with communication failures while maintaining safety in all hazardous situations.

Morales et al. [30, 31] consider a communication protocol for exchanging messages among vehicles. When planning the vehicle trajectories, a control algorithm uses this communication protocol together with the remote and onboard sensory information. The process of trajectory planning becomes simple when all vehicles can use the (in general vectorial) variable LoS (level of service). The correctness

of the cooperation algorithm depends on this common information resource, LoS. Let us consider an example in which vehicular platooning, LoS might include the maximum acceleration, which is due to the vehicle's braking limits [33].

The problem of distributed (uniform) consensus is a related problem to that one studied by Morales et al. [30, 31]. Both problems consider a set of values proposed by the nodes, i.e., the system vehicles, and the uniform selection of a single value from such sets. Since vehicular systems is a safety critical one, this section must terminate within a time constraint that is more severe than the bounded achievable for different versions of the uniform consensus problem. The safety analysis is greatly simplified when using exact and deterministic solutions, in contrast to the approximate consensus [34].

It is well-known that unbounded communication delays, say, due to packet drop, can defer reaching a uniform consensus about LoS's value, and the literature includes a several related negative results [2, 35, 36]. Lynch [37] shows that communication failures can prevent the system from reaching consensus deterministically. Moreover, any probabilistic algorithm that takes no more than r rounds reaches a non-uniform decision with a probability of at least $\frac{1}{r+1}$. Thus, there are no assurance to reach uniform consensus within a deadline, because radio communications are prone to failures. Therefore, we cannot hope for achieving a solution that is based on protocols that provide uniform consensus. Thus, when there is a need to establish a new value of s, it is imperative not to let the communication network, which may present non-uniform values of S to different vehicles, to lead the cooperative vehicular system an unsafe operation.

Uniform consensus algorithms with real-time constraints often assume timed and reliable communication [38, 39]. Morales et al. [30, 31] do not assume reliable communication. They present the problem of minimum longest uncertainty period. The problem considers deterministic solutions for eventually deciding on the LoS. Unlike the uniform consensus problem, the LoS needs to repeatedly decided on; once in every synchronous round. Moreover, there could be a period, called the uncertainty period, during which different nodes output different LoS values. This problem asks what is the longest period in which the different vehicles can consider different performmance levels. Note that by Lynch [37], this bound cannot be zero. Namely, the vehicles are at risk to disagree when, for example, some nodes miss receiving the required information by the deadline of a synchronous round. However, at the end of the uncertainty period, all nodes need to agree on the same value.

7.5.1.1 The Solution Approach and Key Concepts

Morales et al. [30, 31] present a communication protocol that collects LoS proposals from all system vehicles. When a vehicle receives proposals from all vehicles in the system, the protocol can decide deterministically on a single proposal. The protocol identifies the risky periods due to transient failures, i.e., a period in which not all LoS proposals were received in due time by every node in the system. Upon risk

identification, the protocol triggers a strategy that deals with possible disagreement about the LoS value.

The system settings assume the availability of a membership service that returns the set of all system nodes (vehicles), $P = \{p_i\}_{i \in \{1,\ldots,n\}}$, such as [13, 14, 19, 27]. They assume the availability of a common clock and the division of the execution to communication rounds. They also assume the availability of an unreliable dissemination protocol, such as [40, 41], that its messages can reside in the network for a bounded duration.

The system also considers vehicular applications that are based on a set of cooperative functionalities out of which the protocol is to select one, which is the LoS. The assumption here is that this set of cooperative functionalities always include a baseline functionality that is always safe. For example, in the case of Adaptive Cruise Control (ACC) and Vehicular Platooning, the application has to adjust the inter-vehicular distance by controlling the velocity of the different vehicles. ACC uses its sensory information for keeping that distance while considering on the vehicles that are in its direct line-of-sight. A Vehicular platooning application can be seen as an advanced (cooperative) ACC that is based not only on providing sensory information from remote sensors but also having a joint control strategy that allows shorter inter-vehicle distance, as long as communication is available. Morales et al. [30, 31] show how to deal with communication failures and assumes the existence of a baseline application such as ACC in the above set of cooperative functionalities.

The Morales et al. [30, 31] protocol let the system nodes to gossip their LoS proposals until the deadline at the end of each communication round k. Once all nodes receive by all node, they can locally and deterministically select a single LoS proposal for round k. In case of a communication failure during round k, each node p_i that experience a failure, outputs the baseline application as its LoS for round k and reports about the failure during the next communication round, $k + 1$. During that round each other node, p_j, either receives p_i's report about the failure during round k or experience a failure during round k. In both cases, p_j outputs the baseline application as its LoS for round $k + 1$. The correctness proof demonstrates that, in the presence of at least one communication round, there could be at most one disagreement round in which different vehicles follow different LoS values. Moreover, once the network become stable and all communication rounds allow any pair of nodes to exchange messages in a timely manner, the system returns to select an LoS value of the sent proposals.

7.6 Example Applications

We discussed a number of functionalities that are based on the protocol by Morales et al. [30, 31]. We consider both functionalities that support key system enables and applications of vehicular coordination algorithms.

7.6.1 Supporting Functionalities

Casimiro et al. [42, 43] present the *safety kernel* as an architectural concept that allows a cooperative vehicular system to decide on a common performmance level based on the information validity level of the different vehicles. The selection of the performmance level is based on a common service level value, LoS. The protocol by Morales et al. [30, 31] can be the basis of the joint LoS selection and by that implement an architectural component that is called *cooperative evaluator of service level* [42, 43]. At each round, each vehicle proposes the maximum service level that it can support. The cooperative evaluator of service level uses the protocol for collecting and selecting a joint LoS value. This value is the basis for selecting a common performmance level during the next communication round. Note that the possibility to disagree on the joint LoS value (and hence the performmance level) could last for a bounded period of time, which is just one round in the common case of a single hop network.

Once can imagine an infrastructure-based support for the safety kernel on the network level. Stations for intelligent transportation system (ITS-stations) and their global and local dynamic maps [44, 45] can be the providers of critical information that is needed for cooperative vehicular systems. In particular, one can consider an extension of the ETSI standard for ITS stations that will also include the position of nearby vehicles. Moreover, using concepts similar to the ones of safety kernel and cooperative evaluator of service level the ITS station would be able to facilitate a joint choice of the operation level in a safe manner.

7.6.2 Cooperative Vehicular Functionalities

Adaptive Cruise Control and Vehicle Platooning are two applications that are part of a set of applications in which vehicles control their headway (inter-vehicle distance) by adjusting their velocity according to their join performmance level. Casimiro et al. [42, 43] propose how to base the choice of the performmance level using a safety kernel architecture [46–48]. Namely, the safety kernel indirectly decides on the headway and chooses the performmance level that all vehicles can support. For example, the safety kernel creates a longer larger inter-vehicle distance whenever at least one vehicles cannot support the current performmance level. In contrast, whenever the performmance level that all vehicles can support recovery to a higher level, the safety kernel shorten the headway.

Casimiro et al. [42, 43] also propose how to use the safety kernel for coordinating intersection crossing. In this application, vehicles from conflicting direction need to schedule their exact arrival time to the intersection boarder so that they can safely enter and leave the intersection. Here, at the performmance highest level, the vehicles cross the intersection with minimal waiting time while in the lowest level the vehicles take caution by, say, always give way to the vehicle coming from the right. Note that

there is no need for explicit agreement on an ad hoc schedule, but rather to base the coordination on the needed safe headway and some predefined rules, such as direction priority, although both approaches can work.

Casimiro et al. [42, 43] also propose the application of coordinated lane-change. The coordination algorithm adjusts the inter-vehicle distances at the target lane using the safety kernel. Whenever the performmance level is high, the application performers the maneuver while keeping a shorter inter-vehicle distance than it maintains in lower performmance levels.

7.7 Conclusions

We have revised some of the recent advances in the area of distributed vehicular systems. The review considers different communication later, design criteria, such as self-stabilization, as well as infrastructure and ad hoc approaches. These vehicular system and networks require extensive validation. Pahlavan et al. [49] and Berger et al. [50] propose combining digital simulation together with the use of a cyber-physical platform that include scaled vehicles. This way, the system designer can gradually demonstrate the system properties. In particular, in can demonstrate the system in a relevant environment before requiring the more costly demonstration using full-scale vehicles in the representative environment.

References

1. N.A. Lynch, *Distributed Computing* (Morgan Kaufmann Publishers, 1996). ISBN: 1-55860-348-4
2. M.J. Fischer, N.A. Lynch, and M.Paterson (ed.), Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)
3. Friedhelm Meyer auf der Heide, C.A. Phillips (eds.), *Best-Effort Group Service in Dynamic Networks*, in *SPAA 2010: Proceedings of the 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures, Thira, Santorini, Greece, June13-15, 2010*, (ACM, 2010), pp. 233–242. ISBN: 978-1-4503-0079-7
4. E.W. Dijkstra, Self-stabilizing systems in spite of distributed control. ACM Commun. **17**(11), 643–644 (1974). doi:10.1145/361179.361202
5. S. Dolev, *Self-Stabilization* (MIT Press, Cambridge, 2000)
6. *Autonomous TDMA Alignment for VANETs*, in *Proceedings of the 76th IEEE Vehicular Technology Conference, VTC Fall 2012, Quebec City, QC, Canada, September 3-6, 2012*, (IEEE, 2012), pp. 1–5. ISBN: 978-1-4673-1880-8. doi:10.1109/VTCFall..6399373
7. P. Leone, M. Papatriantafilou and E.M. Schiller, *Relocation Analysis of Stabilizing MAC Algorithms for Large-Scale Mobile Ad Hoc Networks*, Lecture Notes in Computer Science, vol. 5804 (Springer, 2009), pp. 203–217. ISBN: 978-3-642-05433-4. doi:10.1007/978-3-642-05434-1_21
8. P. Leone, E. Schiller, Self-stabilizing TDMA algorithms for dynamic wireless ad hoc networks. Int. J. Distrib. Sens. Netw. **2013** (2013). doi:10.1155/2013/639761
9. P. Leone, M. Papatriantafilou, E. M. Schiller, and G. Zhu (eds.), *Chameleon-MAC: Adaptive and Self-* Algorithms for Media Access Control in Mobile Ad Hoc Networks*, in *Stabilization,*

Safety, and Security of Distributed Systems - 12th International Symposium, SSS 2010, New York, NY, USA, September 20-22, 2010. Proceedings, Lecture Notes in Computer Science, vol. 6366 (Springer, 2010), pp. 468–488. ISBN: 978-3-642-16022-6. doi:10.1007/978-3-642-16023-3_37

10. T. Petig, E. Schiller, and P. Tsigas *Self-stabilizing TDMA Algorithms for Wireless Ad-hoc Networks without External Reference*, in *13th Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET 2014, Piran, Slovenia, June 2-4, 2014*, (IEEE, 2014), pp. 87–94. ISBN: 978-1-4799-5258-8. doi:10.1109/MedHocNet.2014.6849109
11. S. Dolev, A. Hanemann, E. M. Schiller, and S. Sharma (eds.), *Self-stabilizing End-to-End Communication in (bounded capacity, omitting, duplicating and non-fifo) Dynamic Networks*, in *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium, SSS 2012, Toronto, Canada, October 1-4, 2012. Proceedings*, Lecture Notes in Computer Science, vol. 7596 (Springer, 2012), pp. 133–147. ISBN: 978-3-642-33535-8. doi:10.1007/978-3-642-33536-5_14
12. S. Dolev, O. Liba, and E. M. Schiller (eds.), *Self-stabilizing Byzantine Resilient Topology Discovery and Message Delivery*, in *Networked Systems - First International Conference, NETYS 2013, Marrakech, Morocco, May 2-4, 2013, Revised Selected Papers*, Lecture Notes in Computer Science, vol. 7853 (Springer, 2013), pp. 42–57. ISBN: 978-3-642-40147-3. doi:10.1007/978-3-642-40148-0_4
13. S. Dolev, E. Schiller, Communication adaptive self-stabilizing group membership service. IEEE Trans. Parallel Distrib. Syst. **14**(7), 709–720 (2003). doi:10.1109/TPDS.2003.1214322
14. S. Dolev, E. Schiller, Self-stabilizing group communication in directed networks. Acta Informatica **40**(9), 609–636 (2004). doi:10.1007/s00236-004-0143-1
15. K.P. Birman, R. van Renesse et al., *Reliable Distributed Computing with the Isis Toolkit*, vol. 85 (IEEE Computer Society Press, Los Alamitos, 1994)
16. A. Bartoli, Implementing a replicated service with group communication. J. Syst. Architect. **50**(8), 493–519 (2004). doi:10.1016/j.sysarc.2003.11.003
17. K. Birman (ed.), *A History of the Virtual Synchrony Replication Model*, in *Replication: Theory and Practice*, Lecture Notes in Computer Science, vol. 5959 (Springer, 2010), pp. 91–120. ISBN: 978-3-642-11293-5. doi:10.1007/978-3-642-11294-2_6
18. R. Khazan, A. Fekete, and N. A. Lynch (eds.), *Multicast Group Communication as a Base for a Load-Balancing Replicated Data Service*, in *Distributed Computing, 12th International Symposium, DISC '98, Andros, Greece, September 24-26, 1998, Proceedings*, Lecture Notes in Computer Science, vol. 1499 (Springer, 1998), pp. 258–272. ISBN: 3-540-65066-0. doi:10.1007/BFb0056488
19. S. Dolev, C. Georgiou, I. Marcoullis, and E. M. Schiller (eds.), *Practically Stabilizing Virtual Synchrony*, in *Stabilization, Safety, and Security of DistributedSystems - 17th International Symposium, SSS 2015, Edmonton, Canada, August 18-21, 2015. Proceedings*, Lecture Notes in Computer Science (Springer, 2015)
20. S. Dolev, T. Petig, and E. M. Schiller (eds.), *Brief Announcement: Robust and Private Distributed Shared Atomic Memory in Message Passing Networks*, in *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, (ACM, 2015), pp. 311–313. ISBN: 978-1-4503-3617-8. doi:10.1145/2767386.2767450
21. R. Fan and N. A. Lynch (eds.), *Efficient Replication of Large Data Objects*, in *Distributed Computing, 17th International Conference, DISC 2003, Sorrento, Italy, October 1-3, 2003,Proceedings*, Lecture Notes in Computer Science, vol. 2848 (Springer, 2003), pp. 75–91. ISBN: 3-540-20184-X. doi:10.1007/978-3-540-39989-6_6
22. S. Dolev, S. Gilbert, E. Schiller, A. A. Shvartsman, and J. L. Welch (eds.), Autonomous Virtual Mobile Nodes, in *Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, (2005), pp. 62–69. ISBN: 1-58113-986-1. doi:10.1145/1073970.1074004
23. S. Dolev, S. Gilbert, E. Schiller, A. A. Shvartsman, and J. L. Welch (eds.), Autonomous Virtual Mobile Nodes, in *SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallelism*

in Algorithms and Architectures, July 18-20, 2005, Las Vegas,Nevada, USA (ACM, 2005), p. 215
24. S. Dolev, S. Gilbert, N. A. Lynch, E. Schiller, A. A. Shvartsman, and J. L. Welch (eds.), *Brief Announcement: Virtual Mobile Nodes for Mobile Ad Hoc Networks*, in *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, (ACM, 2004), p. 385
25. S. Dolev, S. Gilbert, N. A. Lynch, E. Schiller, A. A. Shvartsman, and J. L. Welch (eds.), *Virtual Mobile Nodes for Mobile Ad Hoc Networks*, in *Distributed Computing, 18th International Conference, DISC 2004, Amsterdam, The Netherlands, October 4-7, 2004, Proceedings*, Lecture Notes in Computer Science, vol. 3274 (Springer, 2004), pp. 230–244. ISBN: 3-540-23306-7. doi:10.1007/978-3-540-30186-8_17
26. S. Dolev, S. Gilbert, L. Lahiani, N. A. Lynch, and T. Nolte (eds.), *Timed Virtual Stationary Automata for Mobile Networks*, in *Principles of Distributed Systems, 9th International Conference, OPODIS 2005, Pisa, Italy, December 12-14, 2005, RevisedSelected Papers*, Lecture Notes in Computer Science, vol. 3974 (Springer, 2005), pp. 130–145. ISBN: 3-540-36321-1. doi:10.1007/11795490_12
27. S. Dolev, E. Schiller, J.L. Welch, Random walk for self-stabilizing group communication in ad hoc networks. IEEE Trans. Mob. Comput. **5**(7), 893–905 (2006). doi:10.1109/TMC.2006.104
28. Z. Bar-Yossef, R. Friedman, G. Kliot, RaWMS—random walk based lightweight membership service for wireless ad hoc networks. ACM Trans. Comput. Syst. **26**(2) (2008). doi:10.1145/1365815.1365817
29. J. Luo, P.T. Eugster, J.-P. Hubaux, Pilot: probabilistic lightweight group communication system for ad hoc networks. IEEE Trans. Mob. Comput. **3**(2), 164–179 (2004) doi:10.1109/TMC.2004.12
30. O. Morales-Ponce, E. M. Schiller, and P. Falcone (eds.), Cooperation with Disagreement Correction in the Presence of Communication Failures, in (IEEE, 2014), *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 1105–1110
31. O. Morales Ponce, E.M. Schiller, P. Falcone, Cooperation with Disagreement Correction in the Presence of Communication Failures. In: CoRR abs/1408.7035 (2014). arXiv:1408.7035
32. B. Kulcsar, O. Morales-Ponce, M. Papatriantafilou, E.M. Schiller and P. Tsigas (ed.), Cooperative Driving for Best Road Network Capacity, *Nationella Konferens i Transportforskning* (2013)
33. R. Kianfar, P. Falcone, J. Fredriksson, Safety verification of automated driving systems. IEEE Intell. Transp. Syst. Mag. **5**(4), 73–86 (2013). doi:10.1109/MITS.2013.2278405
34. J.H. Lala, R.E. Harper, S. Alger, A design approach for utrareliable real-time systems. IEEE Comput. **24**(5), 12–22 (1991). doi:10.1109/2.76283
35. A. Fekete et al., The impossibility of implementing reliable communication in the face of crashes. J. ACM **40**(5), 1087–1107 (1993)
36. M.J. Fischer, N.A. Lynch, M. Merritt, Easy impossibility proofs for distributed consensus problems. Distrib. Comput. **1**(1), 26–39 (1986)
37. N.A. Lynch, *Distributed Algorithms* (Morgan Kaufmann Publishers, 1996). ISBN:1-55860-348-4
38. J.-F. Hermant, G. Le Lann, Fast asynchronous uniform consensus in real-time distributed systems. IEEE Trans. Comput. **51**(8), 931–944 (2002)
39. M. K. Aguilera, G. L. Lann, and S. Toueg (eds.), *On the Impact of Fast Failure Detectors on Real-Time Fault-Tolerant Systems*, in *Distributed Computing, 16th International Conference, DISC 2002, Toulouse, France, October 28-30, 2002 Proceedings*, Lecture Notes in Computer Science, vol. 2508 (Springer, 2002), pp. 354–370. ISBN: 3-540-00073-9
40. S.P. Boyd et al., Randomized gossip algorithms. IEEE Trans. Inf. Theory. **52**(6), 2508–2530 (2006)
41. C. Georgiou, S. Gilbert, D.R. Kowalski, Meeting the bdeadline: on the complexity of fault-tolerant continuous gossip. Distrib. Comput. **24**(5), 223–244 (2011)
42. A. Casimiro et al., A Kernel-Based Architecture for Safe Cooperative Vehicular Functions, *Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on*, June 2014, pp. 228–237. doi:10.1109/SIES.2014.6871208

43. A. Casimiro, O. Morales Ponce, T. Petig, and E.M. Schiller (ed.), Vehicular Coordination via a Safety Kernel in the Gulliver Test-Bed, in *Distributed Computing Systems Workshops (ICDCSW), 2014 IEEE 34th International Conference on*, June 2014, pp. 167–176. doi:10.1109/ICDCSW.2014.25
44. C. Berger, O. M. Ponce, T. Petig, and E. M. Schiller (eds.), *Driving with Confidence: Local Dynamic Maps that Provide LoS for the Gulliver Test-Bed*, in *3rd Workshop on Architecting Safety in Collaborative Mobile Systems (ASCoMS), Florence, Italy, September 8-9, 2014. Proceedings*, Lecture Notes in Computer Science, vol. 8696 (Springer, 2014), pp. 36–45. ISBN: 978-3-319-10556-7. doi:10.1007/978-3-319-10557-4_6
45. J. Ibanez-Guzman, S. Lefevre, A. Mokkadem, and S. Rodhaim (ed.), Vehicle to Vehicle Communications Applied to Road Intersection Safety, Field Results, in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on* Sep 2010, pp. 192–197. doi:10.1109/ITSC.2010.5625246
46. A. Casimiro et al. (eds.), *KARYON: Towards Safety Kernels for Cooperative Vehicular Systems*, in *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium, SSS 2012, Toronto, Canada, October 1-4, 2012. Proceedings*, Lecture Notes in Computer Science, vol. 7596 (Springer, 2012), pp. 232–235. ISBN: 978-3-642-33535-8. doi:10.1007/978-3-642-33536-5_22
47. P.N.D. Costa, J. Craveiro, A. Casimiro, and J. Rufino (eds.), *Safety Kernel for Cooperative Sensor-Based Systems*, in *SAFECOMP 2013 - Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems) of the 32nd International Conference onComputer Safety, Reliability and Security, Toulouse, France, 2013*, (HAL, 2013). http://hal.archives-ouvertes.fr/SAFECOMP2013-ASCOMS/hal-00847903
48. E. Vial and A. Casimiro (eds.), *Evaluation of Safety Rules in a Safety Kernel-Based Architecture*, in *SAFECOMP 2014 - Workshop ASCoMS (Architecting Safety inCollaborative Mobile Systems) of the 33rd International Conference on Computer Safety, Reliability and Security, Florence, Italy, September 8-9, 2014. Proceedings*, Lecture Notes in Computer Science, vol. 8696 (Springer, 2014), pp. 27–35. ISBN: 978-3-319-10556-7. doi:10.1007/978-3-319-10557-4_5
49. M. Pahlavan, M. Papatriantafilou, and E. M. Schiller (ed.), Gulliver: A Test-Bed for Developing, Demonstrating and Prototyping Vehicular Systems, in *Proceedings of the 75th IEEE Vehicular Technology Conference, VTC Spring 2012, Yokohama, Japan, May 6-9, 2012*, (IEEE, 2012), pp. 1–2. ISBN: 978-1-4673-0989-9. doi:10.1109/VETECS.2012.6239951
50. C. Berger et al., Bridging physical and digital traffic system simulations with the gulliver test-bed, in *Proceedings of 5th International Workshop on Communication Technologies for Vehicles, Nets4Cars/Nets4Trains 2013*, Villeneuve d'Ascq, France, 14–15 May 2013, ed. by Marion Berbineau et al., vol. 7865. Lecture Notes in Computer Science (Springer, 2013), pp. 169–184. ISBN: 978-3-642-37973-4. doi:10.1007/978-3-642-37974-1_14