The KARYON Project: Predictable and Safe Coordination in Cooperative Vehicular Systems *

António Casimiro
University of Lisboa
casim@di.fc.ul.pt

Jörg Kaiser
Otto-von-Guericke Univ. Magdeburg
kaiser@ivs.cs.uni-magdeburg.de

Elad M. Schiller
Chalmers Univ. of Tech.
elad@chalmers.se

Pedro Costa
GMVIS SKYSOFT
pedro.costa@gmv.com

José Parizi
EMBRAER SA
parizi@embraer.com.br

Rolf Johansson
SP AB
rolf.johansson@sp.se

Renato Librino
4S SRL
renato.librino@4sgroup.it

***Abstract*—KARYON, a kernel-based architecture for safety-critical control, is a European project that proposes a new perspective to improve performance of smart vehicle coordination. The key objective of KARYON is to provide system solutions for predictable and safe coordination of smart vehicles that autonomously cooperate and interact in an open and inherently uncertain environment. One of the main challenges is to ensure high performance levels of vehicular functionality in the presence of uncertainties and failures. This paper describes some of the steps being taken in KARYON to address this challenge, from the definition of a suitable architectural pattern to the development of proof-of-concept prototypes intended to show the applicability of the KARYON solutions. The project proposes a safety architecture that exploits the concept of architectural hybridization to define systems in which a small local safety kernel can be built for guaranteeing functional safety along a set of safety rules. KARYON is also developing a fault model and fault semantics for distributed, continuous-valued sensor systems, which allows abstracting specific sensor faults and facilitates the definition of safety rules in terms of quality of perception. Solutions for improved communication predictability are proposed, ranging from network inaccessibility control at lower communication levels to protocols for assessment of cooperation state at the process level. KARYON contributions include improved simulation and fault-injection tools for evaluating safety assurance according to the ISO 26262 safety standard. The results will be assessed using selected use cases in the automotive and avionic domains.**

I. Introduction

The constantly increasing traffic density on the roads puts substantial challenges to society. Because it is not possible to just build new roads at the same pace as traffic increases or extend the airspace, traffic throughput has to be improved. One way to increase throughput is to enable the cooperation between vehicles to allow information sharing, prospective planning and tight manoeuvre synchronization between vehicles beyond human reaction capabilities. The importance of the problem finds echo in large programs for car-to-car and car-to-roadside communication that have been launched on the national and European level by the car industry[1]. The benefit from this vehicle cooperation firstly comes from extending the perception of the environment dramatically. E.g. an obstacle warning system can look miles ahead, prospective trajectory planning can detect possible conflicts far ahead using the information from other planes or a central data repository as in SESAR[2]. Additionally, new opportunities can be explored, such as extensive driver assistance systems that today are based on autonomous environment perception (and allowed only in very constraint mission contexts), which may greatly benefit from extended cooperation options.

However, there is a basic safety problem when moving to these cooperative scenarios. It should be noted that all the individual vehicles are composed from dozens (and more) of electronic control units (ECUs) connected by a handful of networks [29]. The safety problem of these on-board systems has been tackled through strict design rules, massive redundancy and synchronous models of operation [22]. The assumptions, policies and mechanisms made for the safety-critical on-board control systems can hardly be transferred to a scenario of cooperating vehicles communicating via a wireless network. Thus there exists a safety problem in the cooperative scenario that needs fairly different solutions. We face an extremely difficult to solve problem: on the one hand, the benefits of exploiting information coming from remote sources are substantial and obvious. They extend the range and quality of environment perception. On the other side, incorporating this information to control the mobile entities raises severe safety problems because of the inherently less predictable wireless communication, the difficulty to assess trustworthiness and age of this information and other uncertainties emerging from such a cooperative scenario. In essence, we thus face a subtle performance-safety trade-off that we are exploring in the KARYON project [5]. The availability of solutions to effectively manage this trade-off will be crucial for a wide acceptance of autonomous and semi-autonomous operation of vehicles by the society and is a prerequisite for safety certification by the respective certification bodies.

* This work was partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol).

[1] A list of current project is available on the CAR 2 CAR Communication Consortium (http://www.car-to-car.org/index.php?id=6)

[2] http://www.sesarju.eu/about

The key objective of KARYON is to provide system solutions for predictable and safe coordination of smart vehicles that autonomously cooperate and interact in an open and inherently uncertain environment. As illustrated in the discussion above, this is a challenging objective due to the increased safety risks introduced by increasingly complex control components and wireless communication, which would allow improving performance. To achieve this main goal, KARYON encompasses a set of objectives with a more specific scope, which are driving the work in the project. In this paper we provide an overall view of the work being developed, focusing on the following fundamental issues:

- **Architectural solution:** We propose a system architecture for safety and performance management, leveraging on the existence of components providing improved functionality and on a safety kernel for ultimate safety provision.
- **Abstract sensor model:** We define a fault semantics for complex sensor faults, for achieving some form of validity assessment or trustworthy perception of the environment while using sensor data in a large-scale distributed application and while exploring the actuation-perception loop.
- **Mechanisms for improved perception:** We also propose improvements in the reliable and trustworthy environment perception based not only on adequate fault models for complex sensor faults, but also on solutions for increased communication predictability, namely solutions for network inaccessibility control at lower communication levels and protocols for assessment of cooperation state.
- **Proof of concept prototypes:** We aim at demonstrating KARYON innovations via computer simulations with fault injection support to experimentally evaluate safety assurance according to the ISO 26262 safety standard [2]. Moreover, we will provide proof of concept prototypes and a simulation-based demonstration that evaluate our results in the context of automotive and aircraft applications.

The paper is organized as follows. In the next section we describe the progress beyond the state of the art, referring to related work in the main focal areas of KARYON. Then, in Section III we introduce fundamental abstractions considered in our solution and we describe the generic KARYON architecture. Then, sections IV and V address, respectively, the approaches for dealing with sensor faults and for improving the predictability of communication. In Section V we also refer to the middleware solution that is proposed in KARYON to deal with the dynamic integration of remote sensor systems. Then, Section VI describes the use cases considered in KARYON, which will serve to demonstrate the project achievements. Finally, Section VII concludes the paper.

II. Progress Beyond the State-of-the-art

State-of-the-art approaches to deal with safety in vehicular applications are typically based on worst-case analysis and pessimistic allocation of resources to achieve the intended functionality. This has a strong impact on the final cost of the solutions. Most often, for instance when considering automotive systems, even a slight increase in cost is not affordable. This is an obstacle to achieve safe systems with improved functionality. The scientific and technical contributions of KARYON will advance the state-of-the-art in this direction, allowing more efficient and functional systems to be developed, while ensuring the required safety goals.

A. Architectural Support for Safety-critical Systems

Safety-critical systems call for predictability, that is to say, real-time operation. This includes real-time properties as a crucial requirement. Therefore, system solutions for safety-critical systems have traditionally been based on synchronous system models, which ensure that fundamental timing variables, such as processing and transmission delays, are known and bounded.

In the synchronous system model, the mechanisms to meet reliability and timeliness requirements are well understood, both in terms of distributed systems theory and in real-time systems design principles. As examples, we mention reliable real-time communication [8, 22, 36], real-time scheduling [9, 34] and real-time distributed replication management [32]. However, when moving to distributed, large-scale, wireless and possibly complex infrastructures, which is the case considered in KARYON, it is necessary to recognize that these infrastructures do not provide the timeliness guarantees required by the synchronous system model. Therefore, designing applications using the synchronous model would cause incorrect system behavior due to the violation of assumptions, and would defeat any safety requirements.

An important architecture that has been employed over the last decade in safety-critical computing systems and, in particular, in the vehicular area, is the Time-Triggered Architecture (TTA) [17, 22]. TTA provides the basis for inter-operable embedded systems, connected through a TTP network, and ensuring reliability levels that are adequate for safe drive-by-wire systems. However, this is also an example of an architecture that assumes a fully synchronous system model, and is inadequate when trying to incorporate new services and adding functionality based on distributed sensor data.

The GENESYS project proposed a generic architecture for component-based development of distributed real-time systems, providing core services that may be used to integrate higher-level components. Interestingly, this architecture acknowledges the hybrid nature of systems, where different modules can be executing in different environments and be subject to different synchrony properties. But differently from the KARYON safety architecture, the main goal of the GENESYS architecture is not to manage the trade-off between the functionality achievable with

complex control components and the safety needs of the overall application.

KARYON considers a hybrid system model and explores the concept of architectural hybridization [37]. Some previous work that also exploited an hybrid architecture was carried out in the HIDENETS Project [28]. The objective in that project was to address reliability concerns, but the results are nevertheless relevant for KARYON. In KARYON we consider that control algorithms and components implementing the functionality are separated from a safety kernel, which is only concerned with managing safety constraints defined in design time. With the safety kernel it is possible to guarantee functional safety based on timely management of operational settings. The approach separates the (safety-critical) reconfiguration and adaptation that is necessary to meet safety constraints, from the functionality itself. This will be the basis for achieving the intended balance between functionality and safety.

B. Supporting Services for Sensor-based Safe Coordination

Advanced control systems are composed by multiple cooperating components. The important characteristic of these systems is their reliance on a correct perception of the environment and of the system state. In computer science, a large body of research in fault-tolerant distributed systems provides solutions for the second aspect, the consistent view on the system state in the presence of faults and concurrency [40]. Results in this field address synchrony and replication issues but often assume correct information at its origin, and replicas are all the same. If reliable operation of sensors and actuators requires dealing with the environment perception and actuation on it, these methods have to be extended. Reliable operation has to cope with peculiarities of sensors resulting in value failures. Here, redundancy mechanisms have to be different. These issues have been addressed in the control community. Analysis and fault detection are based on mathematical models of the sensor-to-actuator chain. This is referred to as fault detection and isolation (FDI) [16] or analytical redundancy methods [4]. However, because mainly developed for industrial control systems, the mechanisms do not sufficiently care about the system impacts of largely varying network latencies or dynamically varying sensor information beyond mere statistical effects. The work in KARYON extends and complements the state-of-the-art by combining results from both research directions.

Treating environment events and the system in a generic and uniform way will enable to relate them, put them into order and to detect and monitor the causal and temporal dependencies between them. The Generic Events Architecture (GEAR) [6] was a first step in this direction. GEAR elegantly deals with the communication between systems that is substantiated by an actuation and the observation of a related sensor event. It allows dealing with "hidden channels" that are a problem in conventional control system design because they are only weakly reflected in the control system itself [23].

One key concept that we pursue is keeping environment models in an appropriate form for run-time assessment. This has major advantages, such as relating actuation and subsequent sensing events, assessing the temporal uncertainty of information arriving via a network with low predictability, and supporting the formulation and detection of a safety critical state. In consequence, the "hidden channels" that are not explicitly covered by any approach in distributed system research will now be represented in an environment model. Hidden channels are understood as physical communication channels and as an opportunity rather than impairment, because they allow detecting unsafe states even when the network is down. Modelling these channels appropriately opens the opportunity of detecting unsafe states even when the network is not working as specified.

III. The KARYON Architecture

In KARYON we exploit the concept of architectural hybridization in the definition of the KARYON architecture, in particular to realize the separation of the overall system in parts that have different properties. In this way, we are able to identify the components that constitute the safety kernel, which are in charge of guaranteeing that the intended functionality is provided in a safe way despite faults and uncertainties.

Recalling the KARYON main objective, which is to provide system solutions for predictable and safe coordination of smart vehicles that autonomously cooperate and interact in an open and inherently uncertain environment, the need for reconciling predictability with uncertainty is evident. Let us reason in terms of the synchrony dimension. Should we consider an asynchronous system model to homogeneously characterize the overall system, we would have no way of addressing timeliness requirements and providing timeliness guarantees for the temporal behaviour of the developed systems. In essence, ensuring functional safety would not be possible, given that even simple hazards require some (temporally) bounded system reaction, something that cannot be handled when considering an asynchronous model. On the other hand, despite the technology improvements in computing and communication, we should also not use a synchronous system model for the entire system because some activities, either processing or communication, are inherently uncertain and have temporal bounds that are hardly known in advance with sufficient certainty (unless these would be unreasonably high). For example, to deal with uncertain wireless communication delays, a synchronous model would either postulate a very high bound for the message delivery delay, which could be unacceptable for performance, or else, by postulating a lower bound the risk of violating the assumption could be too high and unacceptable.

In contrast with homogeneous models, a hybrid system model can state assumptions that only hold during some periods of time (hybridization in the time dimension), or it can state different kinds of assumptions for different parts of the system (hybridization in the space dimension). Then, provided it is possible to find a mapping between some hybrid model and a correspondingly hybrid architecture that is feasible in reality (considering some

concrete networking and computational environment), it will be possible to exploit the increased expressiveness of the hybrid model to design improved solutions and, in particular, to address the conflicting goals of predictability and uncertainty. A detailed discussion of the theoretical and practical advantages of hybrid models when compared to homogeneous models can be found in [37].

In KARYON we apply architectural hybridization to exploit the better properties of a restricted part of the system in the achievement of the desired safe behaviour. Therefore, in the generic KARYON architecture, illustrated in Figure 1, we draw an "hybridization line" to clearly separate the components that behave in a predictable way and for which it will be possible to validate safety properties in design time, from the components that might be affected by run-time uncertainties (uncertain temporal behavior, faults not covered by the design). Note that in real systems, the latter might be just a few components, for instance those that realize the coordination with remote systems through wireless networks, or components realizing complex operations that may take uncertain amounts of time.

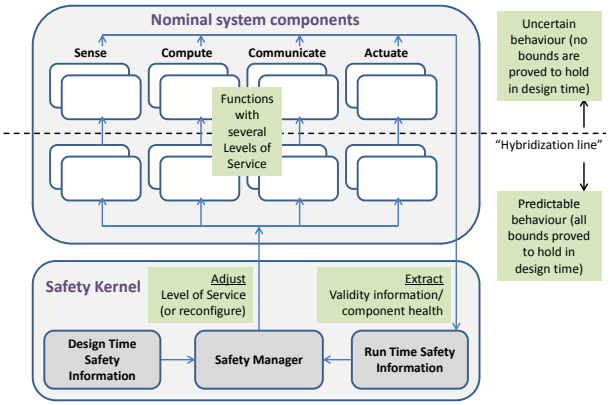

Fig. 1. Generic KARYON architecture.

In the generic architecture represented in Figure 1, the *nominal system* corresponds to the set of components that realize the several local and cooperative functionalities. These components include sensors, actuators, computing and communication components. They are in charge of getting information from the environment, processing it, sending and receiving information from other vehicles (which will have the same architecture), executing control algorithms and sending control commands to actuators. In order to understand the role of the Safety Kernel part, let us introduce the concept of *Level of Service (LoS)* and summarize the considered fault model.

Given that we are interested in managing the trade-off between performance and safety, we consider that functionality can be performed with possibly several LoS. The idea is to allow the cooperative functionality to be performed in different ways, each with its own set of safety requirements imposed on every local system and each allowing a certain maximum performance level. Then, in run-time it will be possible to select the LoS that will allow the highest performance for the functionality while making sure that all unacceptable risks are avoided and thus that functional safety is achieved. In design time it is necessary to perform hazard analysis and derive the set of conditions on the system components and data (the design time safety information shown in Figure 1) that, for each LoS, need to hold in order to ensure functional safety. In run time it is necessary to evaluate which conditions (safety rules) hold and then determine and enforce the adequate LoS.

We consider that there is always one LoS that will meet all the conditions for functional safety. This LoS may correspond to a non-cooperative mode of operation, in which safety analysis and safety solutions are the same that are used when considering non cooperative systems. Moreover, in this LoS the functionality is realized only using components below the hybridization line. We note that when some conditions are met (namely the possibility to interact, within some temporal bounds, with other vehicles) it will be possible to realize the cooperative functionality in a higher LoS. Finally, we also note that we consider the existence of a *scope* for the realization of cooperative functionality, and that this scope is consistently perceived by all involved actors.

It only makes sense to consider several LoS because faults may happen, leading to the violation of safety rules and disabling the provision of cooperative functionality with higher performance levels. Sensor components can experience various faults affecting their output. For remote sensors (i.e., sensors providing remote information received through the wireless network) failures may occur in the time and in the value domain. Failures are abstracted in a data centric way and expressed by a *validity estimate*. A subset of the sensors will be reliable, where a reliable sensor is understood as an abstract sensor exploiting redundancy (analytic, component, time) and fusion techniques to achieve the required reliability (see Section IV-B). Computing components above the hybridization line can fail by crashing or doing timing faults. They may also experience failures in the value domain, but only if the same data centric approach that is used for sensor to express a the failure through a validity estimate can be applied. A subset of computing components will be reliable (to the extent needed as dictated by the safety analysis performed in design time), including Safety Kernel components. Communication components above the hybridization line can experience crash or timing faults, but do not corrupt data. Actuators are assumed not to fail (in fact we consider that they are all below the hybridization line).

The *Safety Kernel (SK)* is the part of the system in charge of controlling the current LoS. It includes the *Safety Manager* component and associated *Design Time Safety Information* and *Run Time Safety Information* components. There is logically only one SK per vehicle, although it may be possible to make it distributed in some implementation.

The Design Time Safety Information component holds a set of predefined safety rules establishing the conditions for functional safety assurance in each LoS. A certain functionality will only be safe in a given LoS (excluding the lower one), if the associated set of safety rules

is satisfied at run time. These safety rules express the needed validity of (sensor) data and integrity of components (e.g., timeliness requirements). The periodically collected information is represented in the architecture by the Run Time Safety Information component, which also abstracts the concrete mechanisms that must be put in place to do this information collection (which will include, for instance, failure detectors for detecting timing faults). Finally, the Safety Manager is the component that triggers changes in the operation of the nominal system components in order to adjust the LoS as necessary. There will be a predefined configuration of the nominal system for all the possible combinations of LoS of the functionalities. The safety manager will periodically check the run time safety data against safety rules and make the necessary adjustments in the nominal system components. Upper bounds on the time needed to perform each cycle will be known at design time, since this is necessary to perform the safety analysis. In particular, arguing about safety can only be done if the time needed to switch between any two LoS of some functionality is known and bounded.

IV. Dealing with Sensor Faults

The primary motivation for expending particular effort on the analysis and abstraction of failure modes of system components originates from the distribution, mobility and complexity of the control systems envisaged in KARYON. We argue that this firstly requires fault models that abstract from the subtle and diverse behaviours of faulty components and provide a well-defined failure semantics at the component's interface. This allows defining a control algorithm without the detailed knowledge of individual component failures. The basic idea is to derive a validity of the information by analyzing and classifying the failure modes of components. This section firstly introduces the notion of an abstract sensor that encapsulates the complex failure modes of physical sensors and, secondly, provides and overview of the architecture of an abstract reliable sensor that we consider in KARYON.

A. Failure Semantics in Sensor-based Systems

We propose a failure semantics that describes the observable behaviour of a component, i.e. the service that a component delivers, in case of an internal failure [7] at the component's interface. The failure semantics provides a higher-level model for a failing component that hides a more complex behaviour inside. As a result, it is by far easier to develop fault-tolerant applications. Figure 2 illustrates the main idea.

The component C is the nominal component that delivers the respective function. C may suffer from specific failures. The goal is to map these failures to a well-defined failure mode at the interface. F comprises the needed redundancy for this mapping. The user of the component's function only has to deal with the failure mode that is externally visible.

The most important difference between distributed computing in general and sensor-driven computations is the nature of failures and the required redundancy. A sensor

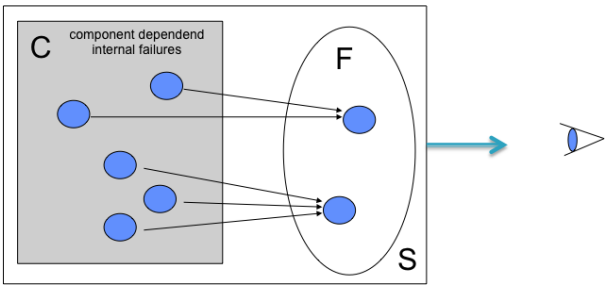

Fig. 2. Illustrating failure semantics.

delivers continuous valued data and the sensor reading is inherently affected by a measurement error. In an early paper, Marzullo described a replication concept dealing with such failures inspired by his work on fault-tolerant clock synchronization [26].

In [14] and [31] the authors derive a validity value for each measurement. The validity value represents the probability of a fault occurrence during the sensor data acquisition phase. The validity value shifts the decision about acceptance or rejection of sensor information to a higher system level. It may be useful, e.g. for a fusion algorithm to use even low validity data rather than just drop the sensor reading. The authors of [14] describe a scheme that distinguishes 16 validity levels. Kaiser and Piontek [31] introduce a continuous scale to define the validity of a measurement. Although these schemes are suggesting the use of such a validity value, they put less emphasis on the problem, how this validity values are derived.

Dealing with this crucial problem for continuous valued data starts with a careful classification of sensor failures. In KARYON we performed a failure mode analysis for different sensors and identified several fault modes that were categorized along five main dimensions: delay faults, sporadic offset faults, permanent offset faults, stochastic offset faults and stuck-at faults. The details of these results are provided in [42].

In KARYON we strive for deriving validity measures for sensor data that are provided with the respective data. This validity estimates can be exploited by remote nodes that fuse multiple sensor sources for assessment and selection. Estimating the validity of a single sensor reading is only one part, although an important one, towards assessing the overall impact of failures to the system. We consider systems that combine and fuse individual sensor data to generate higher level, application relevant information. Thus, there is a chain of filters, estimators, and fusion components which finally produce the desired output to the control functions. The MOSAIC architecture, which we overview in the next section, is intended to cope with all these aspects.

B. Architecture of an Abstract Reliable Sensor

Reliable sensors require a failure detection strategy that indicates the occurrence of a fault and assesses its impact on the output. The detection mechanisms are based

on a comparison of measurements with a reference that may be uncertain. Redundant information can be derived in three different ways. First of all the system may be extended by additional sensors. This is not possible for all sensor types and may also cause an increased effort in energy, weight and installation space. An alternative to component redundancy is analytical redundancy, based on a mathematical model. A prerequisite of this approach is a detailed knowledge about the system, to derive an appropriate mathematical model. Additional computational resources may be needed for execution. The third option is based on a series of samples and some comparison or averaging. This can be interpreted as some form of temporal redundancy.

When developing an application the programmer has to cope with the diversity of detection methods. To ease this task we propose the MOSAIC architecture for reliable sensors integrated in a programming concept for smart devices. MOSAIC provides methods for dynamic data acquisition, processing and communication, etc. [42, 43]. Figure 3 illustrates the basic structure of a node combining the abstract sensor input layer, application modules, an abstract communication layer and the associated crosscutting fault management. A MOSAIC component disseminates typed message objects called events, including the respective sensor data and additional attributes like position, timestamps, validity estimation, etc. Static properties and information of a MOSAIC component are described in an electronic data sheet stored on the node.

This allows the programmer to describe an application as a network of independent components, exchanging sensor data, fusion results and actuator commands.

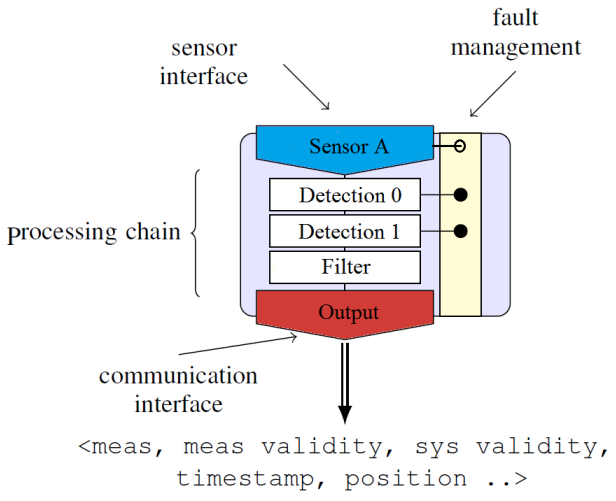

Fig. 3. Structure of a sensor node in MOSAIC.

In Figure 3, two of the three application modules (Detection 0 and Detection 1) as well as the input layer (Sensor A) generate an individual failure detection result. The input layer may monitor the delays or omissions of the transducer output. All tests are connected to the fault management module that combines the individual fault estimations and calculates a general validity value between 0 and 100%.

MOSAIC distinguishes between two types of failure detectors: a) dominant detectors that render a result invalid (i.e. a validity of 0) if they detect a failure, and b) other detectors that lead to a certain continuous validity estimate. In this case, the fault management unit derives a validity from the internal knowledge about the process and other detection results. The Detection 0 and Detection 1 modules belong to the first class and are represented by a solid dot in Figure 3. The dot that is not filled represents a detector of the second category.

The validity result, which we call (sensor) *data validity*, is assigned as an additional attribute to each data set. Hence, the data validity attribute represents an abstract estimation of the reliability of the exchanged information. It provides the possibility to compare sensor data without an explicit knowledge of underlying fault models and implemented fault detection strategies.

V. Dealing with Communication Uncertainty

KARYON devotes particular attention to the problems caused by communication uncertainty. In this section we overview the approaches that are taken in KARYON to improve the predictability and resilience of communication, to define middleware that is suitable to deal with heterogeneity and different QoS attributes of networks in distributed control, and to address the problem of achieving a consistent view among cooperative entities.

A. Predictability and Resilience in Embedded Networks

Providing predictability like temporal guarantees in a wireless communication system may follow two different but complementary approaches depending on the dynamic properties of the network. One approach strives for monitoring the network, predicting inaccessibility times and providing means to put bounds on these inaccessibility times. The second approach to predictable communication is based on avoiding any arbitration conflicts.

1) Network Inaccessibility: Disturbances induced in the operation of MAC protocols may create temporary partitions in the network, derived of the time required to detect and recover from these situations. These disturbances can be produced by external interferences or by some glitches in the operation of the MAC layer. These temporary network partitions are called periods of network inaccessibility [39, 41].

Since the periods of network inaccessibility may have durations much higher than the normal worst case network access delay, inaccessibility incidents do represent a source of unpredictability. The effects of network inaccessibility may propagate to the higher layers of a communication stack – usually a collapsed version of the Open Systems Interconnection (OSI) reference model – potentially disrupting the operation of services and applications. As a consequence, the overall dependability, predictability and timeliness properties of the system may be at risk, being compromised at the communication service.

KARYON proposes an innovative extensible component architecture, dubbed R2T-MAC, which surrounds the standard MAC level with additional components designed to extend and enhance its native characteristics. The extra components allow to improve MAC level predictability and resilience against accidental disturbances in medium and medium access levels, providing a service layer interface with enhanced predictability and timeliness properties intended to simplify the development of networked real-time protocols and applications.

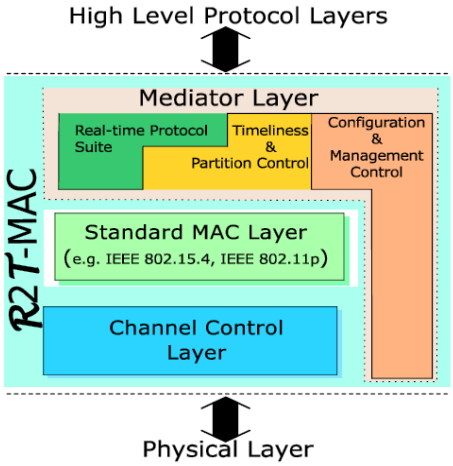

Fig. 4. The KARYON R2T-MAC architecture.

The architecture illustrated by Figure 4 is composed by two different layers: MLA, the Mediator Layer and the Channel Control Layer, surrounding a standard MAC layer. This means, such a solution can be incorporated in Commercial Off-The-Shelf (COTS) components without fundamental modifications in the standard MAC level protocol.

The Mediator Layer intermediates the communication and provides error isolation between the MAC and higher layers, minimizing the negative effects caused by disturbances in the medium and medium access control protocols. This is a standard-compliant solution which extends MAC layer services with additional features and guarantees, enhancing the predictability and timeliness of wireless communications. The Mediator Layer may include components to provide services such as reliable and real-time frame transmissions, node failure detection and membership, control of temporary network partitions (inaccessibility), control of resilience of communications, and management of MAC layer and its configurations.

The Channel Control Layer is designed to allow the control of channel state, and also to improve the network resilience by profiting from the control of the diversity of radio channels used on the communication medium.

2) Self-stabilizing MAC algorithms: We study communication fundamentals that are related to medium access control. We show a way to increase the degree predictability and resilience of wireless embedded networks. We focus on advance provision with respect to enforcing predictability and resilience in a relevant set of wireless network settings by suggesting our design for self-stabilizing MAC algorithms that provides a greater predictability degree than existing ones [25] in addition for an algorithmic design for TDMA alignment [27]. Such algorithms are required for autonomous implementation of [25], i.e., without the use of external time sources, such as GPS. We also study protocols that directly use the MAC protocol, i.e., the data link layer and the end-to-end communications in dynamic networks [12].

MAC protocols for VANETs need to be autonomous and robust as well as have high bandwidth utilization, high predictability degree of bandwidth allocation, and low communication delay in the presence of frequent topological changes to the communication network. We propose a self-stabilizing MAC algorithm that guarantees satisfying these severe timing requirements [25]. Besides the contribution in the algorithmic front of research, we expect that our proposal can enable quicker adoption by practitioners and faster deployment of VANETs, such as the IEEE 802.11p.

The problem of local clock synchronization is studied in the context of TDMA protocols for dynamic and wireless ad hoc networks. In the context of TDMA, local pulse synchronization mechanisms let neighboring nodes align the timing of their packet transmissions, and by that avoid transmission interferences between consecutive timeslots. Existing implementations for VANETs assume the availability of common (external) sources of time, such as base-stations or GPS time sources. We are the first to consider autonomic design criteria, which are imperative when no common time sources are available, or preferred not to be used, due to their cost and signal loss and use self-$\star$ pulse synchronization strategies [27]. Their algorithms consider the effects of communication delays and transmission interferences. We demonstrate the algorithms via extensive simulations in different settings including node mobility. We also validate these simulations in the MicaZ platform, whose native clocks are driven by inexpensive crystal oscillators. The results imply that the studied algorithms can facilitate autonomous TDMA protocols for VANETs.

End-to-end communication over the data link layer (or overlay networks) is one of the most important communication tasks in every communication network, including mobile ad hoc networks, and VANETs. We study data link layer and end-to-end algorithms that exchange packets to deliver (high level) messages in FIFO order without omissions or duplications [12]. We present a self-stabilizing end-to-end algorithm that can be applied to networks of bounded capacity that omit, duplicate and reorder packets. The algorithm is network topology independent, and hence suitable for always changing dynamic networks with any churn rate.

B. Adaptive Middleware for Advanced Control Systems

In KARYON we aim for predictable and safe coordination of smart vehicles. This requires a spontaneous communication system in which communication end-points may dynamically need to dynamically use information from other vehicles or from the available infrastructure.

Another challenging property results from the system-of-systems property in a KARYON scenario. This means that we have to deal with heterogeneous networks concerning data formats and addressing schemes. With some effort, heterogeneity can be made transparent by the respective middleware abstractions. However, Quality of Service (QoS) will be a problem, because latencies, jitter and other inherent uncertainties cannot be removed easily.

The publish/subscribe communication model is well known to support spontaneous, many to many communication relations and reflect autonomy of communicating entities [15, 21, 33]. This approach prevents control flow dependencies between the communication participants. However, QoS is a major problem. In a dynamic communication scenario the quality of service will change over time and requirements need to be checked dynamically whenever the communication link is established and during run-time. AUTOSAR enables communication by using the publish/subscribe interface in a local system where QoS issues can be solved statically. However, AUTOSAR [1] does not address spontaneous communication across system boundaries.

Because this property is crucial for KARYON, we will use the FAMOUSO communication middleware [20, 43], which builds on basic concepts that have been developed in the context of the IST project CORTEX [38] as COSMIC middleware [19]. The middleware has been considerably improved and extended with respect to adaptability and configurability during all stages of the development process and the support for heterogeneous systems and programming languages. Key to this adaptability are machine exploitable descriptions of the service requirements and the provided functionality and performance of the underlying processors and the communication system.

FAMOUSO provides event-based communication that is explicitly designed for dynamic, distributed control. We propose the concept of event channels that address the problem of assessing and maintaining QoS in such a cooperative system. Figure 5 sketches the channel concept.

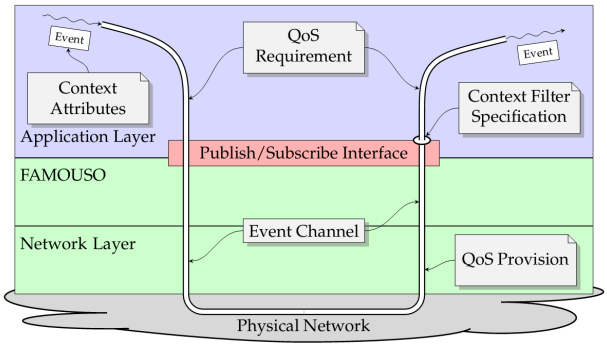

Fig. 5. Channel concept implemented in FAMOUSO.

In FAMOUSO all disseminated information is encapsulated in typed message objects called events. An event is composed from three parts:

- a *subject*,
- *attributes*, and
- *content*.

A *subject* identifies to the content of an event and is represented by a unique identifier (UID). The UIDs span a global name space across all networks. Subjects are used to route an event to the interested subscribers. The binding between a publisher and a subscriber is performed dynamically over network boundaries.

Attributes specify quality requirements and the context of an event. Quality attributes provide information like timeliness and dependability parameters. Context attributes supply information like location or time. As illustrated in Figure 5, the publisher may add a context attribute to an event. The subscriber may specify a set of context attributes by using the context filter specification. The subscriber will only get those events which pass the context filter. As an example, a subscriber is interested in events from a specific location.

An event channel provides a unidirectional communication channel connecting multiple publishers to multiple subscribers. Before a publisher can disseminate an event, it has to announce the respective event channel that is identified by the subject of events that will be disseminated. The notion of an event channel allows specifying and enforcing QoS attributes. The publisher may specify the QoS that is needed, e.g. a maximal latency, a bandwidth, a rate of events or a delivery guarantee. It is obvious, that in a static system these requirements can be checked against what the network is able to provide e.g. at configuration time, i.e. before the system is actually in operation. In a system-of-systems in which spontaneous communication is needed, the information about the underlying network properties have to be acquired dynamically during run-time. Nevertheless, any guarantee involves some assessment and subsequent resource reservation before communication can start. The dynamic assessment of the underlying network properties is part of the announcement process when a publisher creates a new event channel. The monitoring and dynamic adaptation concepts in wireless networks, such as those described in the previous section, acquire the knowledge that is needed to check whether the requirements match what the networks can provide.

FAMOUSO is able to support a broad variety of different hardware platforms ranging from low-end 8-Bit micro-controllers up to high-end 64-Bit server systems and enables interaction over different communication media like the CAN field-bus [19], Wireless Sensor Networks like IEEE 802.15.4,Wireless Mesh Networks [18] and Ethernet like UDP broad- and multicast. FAMOUSO can be used from different programming languages (C/C++, Python, Java, .NET) as well as from engineering tools (LabVIEW, MATLAB/Simulink) simultaneously [20].

C. Reliable Assessment of Cooperation State

Solutions for reliable cooperation between mobile nodes should have a consistent view about the operational state of cooperating entities and their intentions. We look into protocols that can learn about the distributed system

state of the vehicular system and its network and by that facilitate application at the higher level. Agreement protocols are needed as building blocks for application at the higher level. For example, Le Lann [24] considers the vehicle platooning and lane change maneuvers. For these applications he proposes to use high-level communication primitives that are based on group communication and concurrent transactions. We study additional approaches that can allow agreement on the ongoing maneuvers with the vehicles in close proximity. We look into the necessary building blocks for wireless implementation of communication primitives that are needed for the reliable assessment of the distributed system state and its network. One of these approaches is based on virtual nodes that maintain shared finite state machines that tile the plane [10]. These state machines can monitor the activity in a given region, such as intersections, or a cluster of vehicles that cruise on the highway by consider mobile virtual nodes [11]. The next steps are about the study of reliability aspects of virtual nodes that are related to reliable broadcast, distributed fault detection and agreement [3].

From the theoretical point of view, we are also working towards understanding what may not be possible to achieve within a predictable time bound because it heavily relies on the inherent uncertainties of the wireless network. Therefore, we consider Byzantine nodes in addition to the above benign system settings. We study the problem of topology discovery that is needed as a building block for the problem of Byzantine agreement. We study the possibilities of algorithmic solutions that have constant costs [13]. Traditional Byzantine resilient (agreement) algorithms use $2f+1$ vertex-disjoint paths to ensure message delivery in the presence of up to f Byzantine nodes. The question of how these paths are identified is related to the fundamental problem of topology discovery. Distributed algorithms for topology discovery cope with a never ending task, dealing with frequent changes in the network topology and unpredictable transient faults. Therefore, algorithms for topology discovery should be self-stabilizing to ensure convergence of the topology information following any such unpredictable sequence of events.

VI. Use Cases

We consider the implementation of KARYON's concepts in two vehicle-based scenarios of avionic and automotive. One of the factors used in Level of Service determination is the distance between vehicles. This refers to a given spatial scale and will be further described in the next section. However, distance per se, is not a sufficient factor to ascertain if a hazardous situation may occur. We need to consider the predictability of such a hazard to happen and factor it, not only in preventive functions but also in corrective functions. This leads to the necessity to define a temporal scale of events and from it deriving the recommended and minimum allowed time to reactive functionalities. These times and the associated hazard analysis are mainly connected to the level of service of each situation.

A. Automobile use cases

We study a set of Advanced Driver Assistance Systems (ADASs) for coordinating vehicles and propose a set of solutions that increase their safety. In particular, we examine scenarios in which vehicles cooperate while: (1) going on the road and keeping their distance from other vehicles, (2) cruising in their lanes and coordinating when lane changes are needed and (3) crossing intersections in a coordinated way. We consider a set of ADASs covers basic operations that allow the drivers to safely pilot their vehicles on the road and we plan to demonstrate some of these ADASs on the Gulliver test-bed [30].

1) Adaptive Cruise Control Systems: ACCs allow vehicles to slow when approaching other vehicle and to accelerate to their cruising speed when possible. These systems are important for accident prevention as well as for reducing energy consumption, because they smoothly adjust the vehicle speed and by that reduce the stop-and-go phenomena when the traffic contention is high. ACCs often incorporate with several other subsystems, such as Lane Keep Assist Systems (LKASs), Lane Change Assistance Mechanisms, Electronic Stability Control (ESC) and Real-time Traffic Information Systems (RTISs). Each of these subsystems relies on other subsystems and enabling technologies, such as Global Positioning System (GPS) and Vehicle-to-Vehicle Communication (V2V). In such a complex system of systems, the ability for monitoring the system wellbeing is essential.

The level of service for this use case is mainly the needed time margin between vehicles for meeting the safety goals. Higher level of service means a lower time margin between vehicles. For each level of service, and for each speed interval, the safety goals are different with respect their attributes of Automotive Software Integrity Levels (ASIL). This means that depending on the vehicles judgement of the integrity level possible to guarantee at a certain moment, the level of service can be determined. The integrity includes health status of sensors both on the actual vehicle and the vehicles in front as well as communication channels and computing resources.

2) Crossing road intersections using ITSs' traffic lights: One of the most fundamental components in contemporary Intelligent Transport Systems (ITSs) are the traffic lights that coordinate and monitor the crossing of intersections. When a traffic light system detects a critical failure in its components, it signals to the arriving vehicles that it is in an inoperative mode (i.e., blinking the orange light). While the traffic light is in failure mode, the drivers coordinate the crossing of the intersection by themselves. Future traffic light systems will periodically broadcast I-am-alive messages to the arriving vehicles. The arriving vehicles will monitor the reception of the I-am-alive messages. When the traffic light system is in an inoperative mode, the vehicles will switch to the use of a backup system: a virtual traffic light that relies on vehicle-to-vehicle communications for coordinating the intersection crossing. It is unclear whether a virtual traffic light that relies entirely on mobile ad hoc networking can provide the same dependability level as a traffic light system that uses stationary infrastructure. How-

ever, virtual traffic light can be literally deployed anywhere without the need for stationary infrastructure. Therefore, virtual traffic lights are likely to emerge as an important ITS technology. The KARYON project develops a means that would facilitate the assertion of safety constraints that are related to virtual traffic lights.

3) Coordinated lane change manoeuvres on highways: Unintentional lane departure is one of the highest risk factor on the road. The idea here it to provide a distributed mechanism for assuring that at any time and any region there is at most one vehicle that is changing its lane and that the nearby vehicles allow it to safely complete the manoeuvre. This concept can be, of course, extended to platoons of cars that can change lanes in a coordinated manner.

B. Avionics

The current considered pattern (see Figure 6) is for the Remote Piloted Vehicles (RPV) to begin a controlled climb into the boundary of non-segregated air space and take safety measures and a final 4D navigation plan. It will then commence its ascent to the target altitude and space, perform the scanning of the targeted area through a grid sweep pattern and then descend to the previously referred boundary. Once ground control has been reasserted at the boundary, the RPV will then proceed to the selected landing space and finalize its operation.

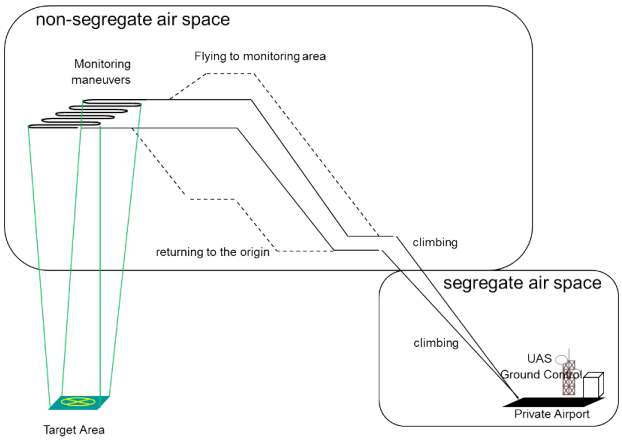

Fig. 6. Avionics base scenario.

A "safety state" for an aerial vehicle can be considered as a spatial volume around the vehicle where the possibility of entrance of others objects is minimal, see Figure 7. The approach or the eventual entrance of some other vehicle into this volume is defined as an "air traffic conflict." Usually this spatial volume is described in terms of a vertical and a lateral distance, called "separation minima."

In the future air traffic management systems (ATM), each aerial vehicle will sense its position and time clock based on satellite navigation information, sending its position information to the ATM on the ground and to other aerial vehicles flying in the same airspace. This way, the air traffic mapping of all vehicles will be available to the ATM

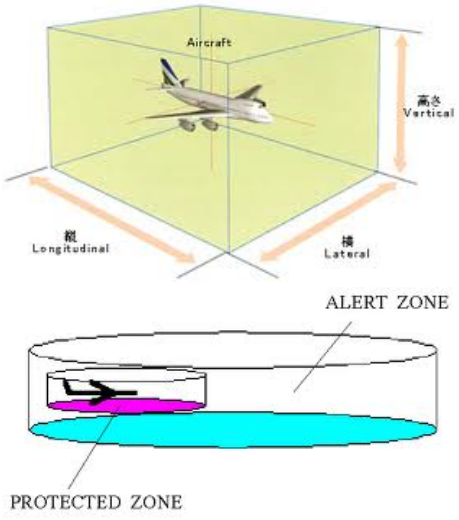

Fig. 7. Aerial Vehicle Safe State.

and also to each vehicle flying in that region. The dissemination of vehicle position information based on satellite technology shall allow the development of a collaborative air traffic management. It is expected that direct flights following optimal trajectories, the called 4D-Trajectories, shall be authorized. Complex flight procedures executed for safety purposes shall be eliminated and, mainly, it shall allow the integration of RPVs into the airspace shared by others piloted vehicles.

Aiming to perform a safety analysis of a shared airspace traffic including RPVs it is convenient to consider two special traffic scenarios involving: (1) a RPV and a collaborative aerial vehicle, and (2) a RPV and a non-collaborative aerial vehicle. A collaborative vehicle here means an aerial vehicle that knows its position and is able to diffuse it to other vehicles, as well as to the ATM center. A non-collaborative vehicle, i.e., not using ADS-B satellite based information, has a much less accurate estimative of its actual position, and only can transmit it to the ATM center by a voice channel.

In the sequel we present three avionic use cases that are related to the automotive domain use cases, which will allow to demonstrate KARYON concepts. Although requiring somewhat different safety conditions and having different control options, the scenarios are similar in nature in a form that allow us to extrapolate safety and performance measures in a confident nature. In each of the three use cases, the two traffic scenarios mentioned above will be considered.

1) Common trajectory traffic in the same direction: This is an aerial traffic situation analogous to the road traffic scenario with cars running Adaptive Cruise Control Systems (ACCS).

Two aerial vehicles fly a common optimal trajectory that connects a common origin and destination. A traffic conflict may appear when the rear vehicle is faster than the front one, or when both vehicles fly in the same speed.

A similar and more frequently situation occurs between two aerial vehicles during the climbing flight phase after departing from the same airport.

2) Leveled crossing trajectories: This is an aerial traffic situation analogous to a crossing road intersection. It is a conflict situation of frequent occurrence, where two aerial vehicles with similar performance would have optimal trajectories that cross in some airspace point.

3) Coordinated flight level change manoeuvres: This scenario considers flight level change for a RPV where it intersects the flight altitude of other vehicles. Difference between this scenario and the previous is that the cross is not directly in a collision path.

VII. Conclusions

The key objective of KARYON is to provide system solutions for predictable and safe coordination of smart vehicles that autonomously cooperate and interact in an open and inherently uncertain environment. This is a challenging objective since the same increasingly complex control components and wireless communication, which would allow improving performance, end up introducing new safety risks, which have to be mitigated or neutralized. Addressing this challenge requires innovative solutions concerning the overall system architecture, the middleware and the mechanisms to deal with faults and uncertainty.

In this paper we presented the work that is being developed in KARYON, the approaches that are being considered, and selected ideas on how to address the problems ahead. We also described the scenarios in the automotive and avionic domains, which will be considered for the demonstration of the technical achievements.

We expect that KARYON will open new perspectives on the use of available technology for safe cooperative systems with increased efficiency. In particular, we believe that the proposed approaches and solutions may be important contributions to improved vehicle density without driver involvement and increased traffic throughput to maintain mobility without a need to build new traffic infrastructures.

References

[1] AUTOSAR: AUTomotive Open System ARchitecture. Available online: http://www.autosar.org/, Oct 15, 2012.

[2] ISO 26262, Road vehicles - Functional safety, Part 1-9, 2011.

[3] M. K. Aguilera, G. L. Lann, and S. Toueg. On the impact of fast failure detectors on real-time fault-tolerant systems. In D. Malkhi, editor, *DISC*, volume 2508 of *Lecture Notes in Computer Science*, pages 354–370. Springer, 2002.

[4] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer, 2006.

[5] A. Casimiro, J. Kaiser, J. Karlsson, E. M. Schiller, P. Tsigas, P. Costa, J. Parizi, R. Johansson, and R. Librino. Brief announcement: KARYON: Towards safety kernels for cooperative vehicular systems. In Richa and Scheideler [35], pages 232–235.

[6] A. Casimiro, J. Kaiser, and P. Veríssimo. Generic-events architecture: Integrating real-world aspects in event-based systems. In R. de Lemos, C. Gacek, and A. B. Romanovsky, editors, *WADS*, volume 4615 of *Lecture Notes in Computer Science*, pages 287–315. Springer, 2006.

[7] F. Cristian. Understanding fault-tolerant distributed systems. *Commun. ACM*, 34(2):56–78, Feb. 1991.

[8] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. Controller area network (can) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[9] Z. Deng and J. W.-S. Liu. Scheduling real-time applications in an open environment. In *IEEE Real-Time Systems Symposium*, pages 308–319. IEEE Computer Society, 1997.

[10] S. Dolev, S. Gilbert, L. Lahiani, N. A. Lynch, and T. Nolte. Timed virtual stationary automata for mobile networks. In J. H. Anderson, G. Prencipe, and R. Wattenhofer, editors, *OPODIS*, volume 3974 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2005.

[11] S. Dolev, S. Gilbert, E. Schiller, A. A. Shvartsman, and J. L. Welch. Autonomous virtual mobile nodes. In *DIALM-POMC*, pages 62–69, 2005.

[12] S. Dolev, A. Hanemann, E. M. Schiller, and S. Sharma. Self-stabilizing end-to-end communication in (bounded capacity, omitting, duplicating and non-fifo) dynamic networks - (extended abstract). In Richa and Scheideler [35], pages 133–147.

[13] S. Dolev, O. Liba, and E. M. Schiller. Self-stabilizing byzantine resilient topology discovery and message delivery. *CoRR*, abs/1208.5620, 2012.

[14] W. Elmenreich, S. Pitzek, and M. Schlager. Modeling distributed embedded applications on an interface file system. In *In Proceedings of the Seventh IEEE International Symposium on Object-Oriented RealTime Distributed Computing*, pages 175–182, 2005.

[15] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, June 2003.

[16] P. M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - a survey and some new results. *Automatica*, 26(3):459–474, May 1990.

[17] G. Heiner and T. Thurner. Time-triggered architecture for safety-related distributed real-time systems in transportation systems. In *Proceedings of the The Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*, FTCS '98, pages 402–, Washington, DC, USA, 1998. IEEE Computer Society.

[18] A. Herms, M. Schulze, J. Kaiser, and E. Nett. Exploiting publish/subscribe communication in wireless mesh networks for industrial scenarios. In *13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 48–655, Hamburg, Germany, 2008.

[19] J. Kaiser, C. Brudna, and C. Mitidieri. Cosmic: A

real-time event-based middleware for the can-bus. *J. Syst. Softw.*, 77(1):27–36, July 2005.

[20] J. Kaiser, L. Buss Becker, S. Zug, and M. Schulze. Supporting independent development, deployment and co-operation of autonomous objects in distributed control systems. In *9th International Symposium on Autonomous Decentralized Systems (ISADS 2009)*, 2009.

[21] J. Kaiser and M. Mock. Implementing the real-time publisher/subscriber model on the controller area network (can. In *In Proceedings of the 2nd International Symposium on Object-oriented Real-time distributed Computing (ISORC99*, 1999.

[22] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1997.

[23] H. Kopetz and P. Verissimo. Real-time and dependability concepts. In S. J. Mullender, editor, *Distributed Systems*, chapter 16, pages 441–446. Addison-Wesley, 2nd edition, 1993.

[24] G. L. Lann. Cohorts and groups for safe and efficient autonomous driving on highways. In O. Altintas, W. Chen, and G. J. Heijenk, editors, *VNC*, pages 1–8. IEEE, 2011.

[25] P. Leone and E. M. Schiller. Self-stabilizing tdma algorithms for dynamic wireless ad-hoc networks. In A. Bar-Noy and M. M. Halldórsson, editors, *ALGOSENSORS*, volume 7718 of *Lecture Notes in Computer Science*, pages 105–107. Springer, 2012.

[26] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Trans. Comput. Syst.*, 8(4):284–304, Nov. 1990.

[27] M. Mustafa, M. Papatriantafilou, E. M. Schiller, A. Tohidi, and P. Tsigas. Autonomous tdma alignment for vanets. In *VTC Fall*, pages 1–5. IEEE, 2012.

[28] I.-F.-. H. H. D. I.-B. NETworks and Services. http://www.hidenets.aau.dk/, 2013.

[29] T. Nolte, H. Hansson, and L. Lo Bello. Automotive communications-past, current and future. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, volume 1, pages 8 pp.–992, 2005.

[30] M. Pahlavan, M. Papatriantafilou, and E. M. Schiller. Gulliver: A test-bed for developing, demonstrating and prototyping vehicular systems. In *VTC Spring*, pages 1–2. IEEE, 2012.

[31] H. Piontek and J. Kaiser. Self-describing devices in cosmic. In *10th IEEE International Conference on Emerging Technologies and Factory Automation*, page 669 672, Catania, Italy, 2005.

[32] D. Powell. Distributed fault tolerance - lessons learned from Delta-4. In M. Banâtre and P. A. Lee, editors, *Hardware and Software Architectures for Fault Tolerance*, volume 774 of *Lecture Notes in Computer Science*, pages 199–217. Springer, 1993.

[33] R. Rajkumar, M. Gagliardi, and L. Sha. The real-time publisher/subscriber inter-process communication model for distributed real-time systems: Design and implementation. In *Design and Implementation, in First IEEE Real-time Technology and Applications Symposium*, pages 66–75, 1997.

[34] K. Ramamritham and J. Stankovic. Scheduling algorithms and operating systems support for real-time systems. *Proceedings of the IEEE*, 82(1):55–67, 1994.

[35] A. W. Richa and C. Scheideler, editors. *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium, SSS 2012, Toronto, Canada, October 1-4, 2012. Proceedings*, volume 7596 of *Lecture Notes in Computer Science*. Springer, 2012.

[36] K. Tindell, A. Burns, and A. J. Wellings. Analysis of hard real-time communications. *Real-Time Systems*, 9(2):147–171, 1995.

[37] P. Veríssimo. Travelling through wormholes: a new look at distributed systems models. *SIGACT News*, 37(1):66–81, 2006.

[38] P. Veríssimo, V. Cahill, A. Casimiro, K. Cheverst, A. Friday, and J. Kaiser. Cortex: Towards supporting autonomous and cooperating sentient entities. In *Proceedings of European Wireless 2002*, pages 595–601, Florence, Italy, Feb. 2002.

[39] P. Veríssimo and J. A. Marques. Reliable broadcast for fault-tolerance on local computer networks. In *In Proceedings of the Ninth Symposium on Reliable Distributed Systems*, pages 24–90. IEEE, 1990.

[40] P. Verissimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[41] P. Veríssimo, L. Rodrigues, and M. Baptista. Amp: a highly parallel atomic multicast protocol. *SIGCOMM Comput. Commun. Rev.*, 19(4):83–93, Aug. 1989.

[42] S. Zug, A. Dietrich, and J. Kaiser. An architecture for a dependable distributed sensor system. *IEEE Transactions on Instrumentation and Measurement*, 60 Issue 2:408 419, 2011.

[43] S. Zug, M. Schulze, A. Dietrich, and J. Kaiser. Programming abstractions and middleware for building control systems as networks of smart sensors and actuators. In *Proceedings of Emerging Technologies in Factory Automation (ETFA 10)*, Bilbao, Spain, 2010.