

Distributed Computing for Functional Safety of Automotive Embedded Systems

Guoqi Xie¹, Renfa Li¹, Keqin Li^{1,2}

- 1) College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410008, China
- 2) Department of Computer Science, State University of New York, New Paltz, New York 12561, USA.

Abstract: The architectures of modern automobiles are heterogeneous distributed integrated architectures that integrate multiple heterogeneous processing units and network buses with a central gateway. Modern automotive embedded systems combine the related characteristics of real-time, cyber-physical, mixed-criticality, and heterogeneous distributed systems; moreover, such systems must meet specific functional safety requirements based on the ISO 26262 standard that was issued in 2011. The safe operation of automobiles must also be guaranteed and the lives of civilians inside and outside vehicles protected; thus, how to coincide with the functional safety requirements from the point of computing is a challenge. The main backbone of this chapter will discuss the distributed computing for functional safety of automotive embedded systems. We first describe the architectures of automobiles and then introduce their distributed functions and systems. We also propose distributed computing models of automobiles for the aforementioned architectures and systems. For the functional safety requirements provided in ISO 26262, we discuss the corresponding issues with distributed computing for schedulability analysis, real-time scheduling, reliability, and fault tolerance.

1. Introduction

1.1 Background

People have continuously pursued driving safety since the invention of automobiles. Safety belts, air bags, and other passive safety functions have saved the lives of millions of people. Subsequently, active safety functions were developed, including the anti-lock braking system (ABS) and the electronic stability program (ESP); these functions have greatly improved the safety of automobiles. The automotive industry has benefited significantly from the rapid development and wide application of electronic and information technologies. Nonetheless, the cost of an

automotive electronic system accounts for 30% to 40% of the total cost of an automobile. In addition, 90% of the innovations in the automotive industry depend heavily on the development of electronics and software [1].

Automotive electronics systems are safety-critical, high-end embedded systems. The safety issues that arise from systematic failures and random hardware failures have attracted attention as these problems often lead to accidents in this respect. The risk of such accidents increases with the complexity and the integration of automotive embedded systems. These accidents emphasize the importance of building safe and reliable automotive embedded systems that minimize risk and loss to protect drivers, passengers, pedestrians, and property. The Road Vehicles Functional Safety standard (ISO 26262) was published on 11 November 2011 to address automotive risks as well as to avoid unacceptable safety hazards and functional risks [2]. Meeting this standard has become an important requirement for automotive embedded systems.

The ISO 26262 standard covers a wide range of automotive electronics and functions, including almost all traditional and new energy automobiles. The safety of these automotive functions must be guaranteed; that is, their potential safety risk should be controlled within an acceptable range. The ISO 26262 standard requires designers to assess and to eliminate all potential risks in advance and as soon as possible, respectively, such that the goals of safety-related automotive functions, especially active safety functions, can be achieved. Therefore, the core tasks in the design of modern automotive embedded systems are ensuring the normal operation of safety-related automotive functions under various harsh conditions and guaranteeing the safety of drivers, passengers, and pedestrians.

1.2 Motivation

Automotive embedded systems have developed heterogeneous distributed architectures over time because of the size, weight, and power consumption (SWaP) for cost and high performance benefits [3]. These architectures consist of many heterogeneous electronic control units (ECUs) that are distributed on multiple network buses. These buses are interconnected through a central gateway. At present, a luxury car comprises at least 70 heterogeneous ECUs with approximately 2,500 signals [4]. This number is expected to increase in future automotive embedded systems.

In the aforementioned distributed architecture, the number of distributed functions (also called functionalities or applications in a few studies) increases with precedence-constrained tasks in automotive embedded systems. An example of an active safety function is the brake-by-wire function [5]. The integration of multiple functions in the same architecture forms an “integrated architecture” in which multiple functions can be supported by one ECU and one function can be distributed over multiple ECUs [6]. Integrated architectures are essential for addressing SWaP problems and reducing costs; however, these architectures require new models and methods [6].

An integrated architecture incorporates several levels of safety-criticality and non-safety-criticality functions into a single platform; this architecture also introduces criticality levels and mixed-criticality systems. Criticality level is represented in ISO 26262 by the automotive safety integrity level (ASIL), which refers to a classification of inherent safety goals that are required by the standard to ensure the accomplishment of objectives in the system. ASILs D and A represent the highest and the lowest criticalities, respectively [2]. Mixed-criticality systems are new systems that combine multiple functions with different criticality levels on a single platform. Moreover, automotive cyber-physical systems (ACPS) design arises in the context of automotive architectures and software [7].

Automotive electronic and electrical safety-related systems must adhere to ISO 26262 throughout their lifecycles, including in terms of specification, design, implementation, integration, verification, validation, production, and release. However, this standard does not clearly specify the quantitative requirements for functional safety. At present, research on functional safety mainly focuses on product development and management, risk analysis and assessment, as well as the testing, verification, and confirmation of software and hardware systems. Modern automotive embedded systems are heterogeneous distributed embedded computing systems that link the related characteristics of real-time, mixed-criticality, cyber-physical and heterogeneous distributed systems. Hence, functional safety must be studied from the perspective of distributed computing; from this point of view, meeting functional safety requirements is challenging.

1.3 Organization

This work discusses the use of distributed computing to ensure the functional safety of automotive embedded systems. Our analysis is not exhaustive because we focus only on certain critical aspects and issues that tend to become significant in the near future. In particular, we discuss the functional safety issues that are specified in ASIL of ISO 26262 by applying distributed computing for schedulability analysis, real-time scheduling, reliability, and fault tolerance.

The rest of this chapter is organized as follows. Section 2 describes the architectures of automotive embedded systems. Section 3 discusses the distributed functions and systems. Section 4 constructs related models for automotive embedded systems. Section 5 discusses the distributed computing for functional safety. Section 6 concludes this study.

2. Heterogeneous Distributed Architectures

Traditional designs are based on the concept of a federated architecture, which involves the independent development of processing units in self-contained hardware ECUs that are connected by several buses as well as the exchange of information and coordination signals [6]. Such architectures sharply increase the cost of embedded electronics because the enhanced complexity of functionality hinders the usage of a single ECU to implement a single function, generates uneven computational needs that lead to the underutilization of some units, and poorly controls the emergent behavior of subsystems that can no longer be considered to be loosely connected. Therefore, these architectures do not satisfy the needs of modern automotive embedded systems.

2.1 Heterogeneous Processing Units

The processing units of the modern automotive electronics are composed of more than 100 heterogeneous ECUs, sensors, actuators, gateways, and other physical devices. For example, Fusion Energi, a plug-in hybrid electric vehicle that was launched by Ford Motors in 2003, comprises 145 actuators, 74 sensors, and 70 ECUs. Fig. 1 shows a sub-system of an automotive in-vehicle network that consists of four ECUs, three sensors, and eight actuators to which five software functions can be

assigned [8]. These ECUs are interconnected by two low-speed controller area network (CAN) buses [15] with a bandwidth of 125 kbit/s. The network buses CAN 1 and CAN 2 are interconnected through a gateway.

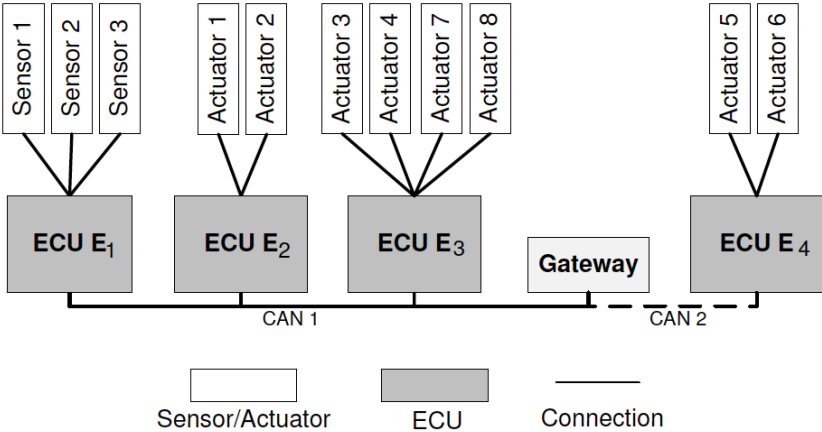


Fig.1 Interconnected ECUs and connected sensors/actuators

The heterogeneity of processing units is mainly reflected in two aspects. First, different types of such units can perform only specific types of tasks (i.e., sensors for sensing data, actuators for implementing particular instructions, and ECUs for calculating real-time tasks). Second, a single task may be executed by different processing units with varying processing capabilities.

2.2 Heterogeneous Networks

Several network technologies have been applied in modern automobiles to meet the specific requirements for the different functions and subsystems of in-vehicle networks. The network topology structure is increasingly complex; for instance, the power control subsystem is mainly used in high-speed CANs (HS-CANs), the chassis control subsystem in high-speed CANs and FlexRay, and the body control subsystem in CANs. For example, the BMW-7 series containing more than 70 ECUs is divided into seven sub-networks. Fig. 2 shows the network architecture of this series, which includes three HS-CANs (500 kbps), one low-speed CAN (100 kbps), one FlexRay, one media oriented system transport (MOST), one Ethernet, and a central gateway [9]. We can see that the automotive networks are no longer mere bus topologies but are mixtures of various network topologies.

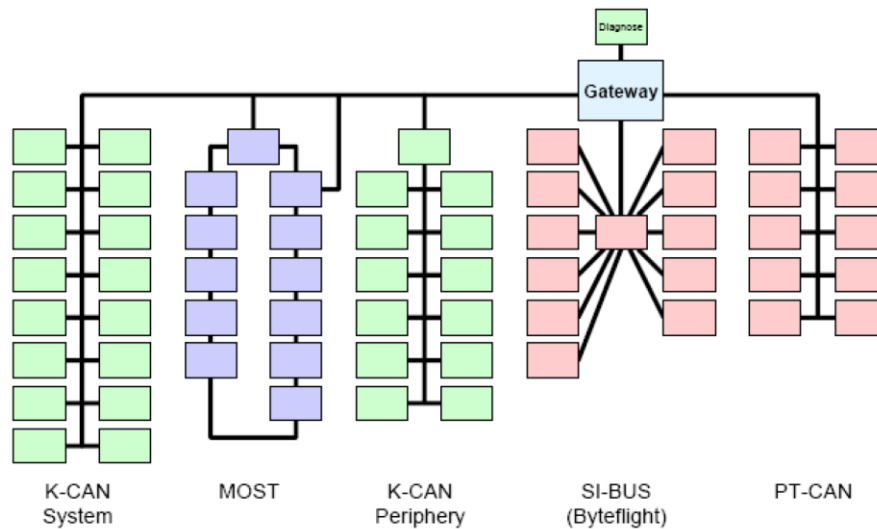


Fig.2 In-vehicle network topology a BMW-7 series

2.3 The Central Gateway

Given the increasing numbers of sensors, actuators, and ECUs, modern control concepts require devices to support the cross linking of these channels. This interconnection is realized with a gateway that connects several buses among sub-networks at different speeds. Gateways are important nodes that connect and facilitate the transmission of messages from one bus to another; however, gateways can become bottlenecks in message transmission because they need to copy thousands of messages from one bus to the other bus(es) to fulfill distributed cross-bus functionalities.

The CAN clusters (more than four or five CAN buses are interconnected by a gateway [10]) are taken as an example. Three types of gateway architectures can be employed in automotive applications, namely, the discrete channel, complex channel, and modular gateways [11]. At present, the first two architectures are the most popular gateways provided by semiconductor manufacturers. Although the discrete channel gateway requires a high-performance host CPU to facilitate real-time operation at high bus load, this gateway is flexible in terms of the number of bus channels. By contrast, the complex channel gateway supports the transfer of messages to other networks without increasing the load on the host CPU; however, this gateway is inflexible with respect to the number of CAN channels. Fig. 3(a) depicts the architecture of the discrete channel gateway that consists of three components, namely, a CPU, CPU peripheral bus, and several single-channel CAN modules. The gateway processing steps are described as follows: the CPU reads all necessary

control information and receives messages over the peripheral bus from one CAN module. Subsequently, the CPU writes the same data over the peripheral bus to (an) other CAN module(s).

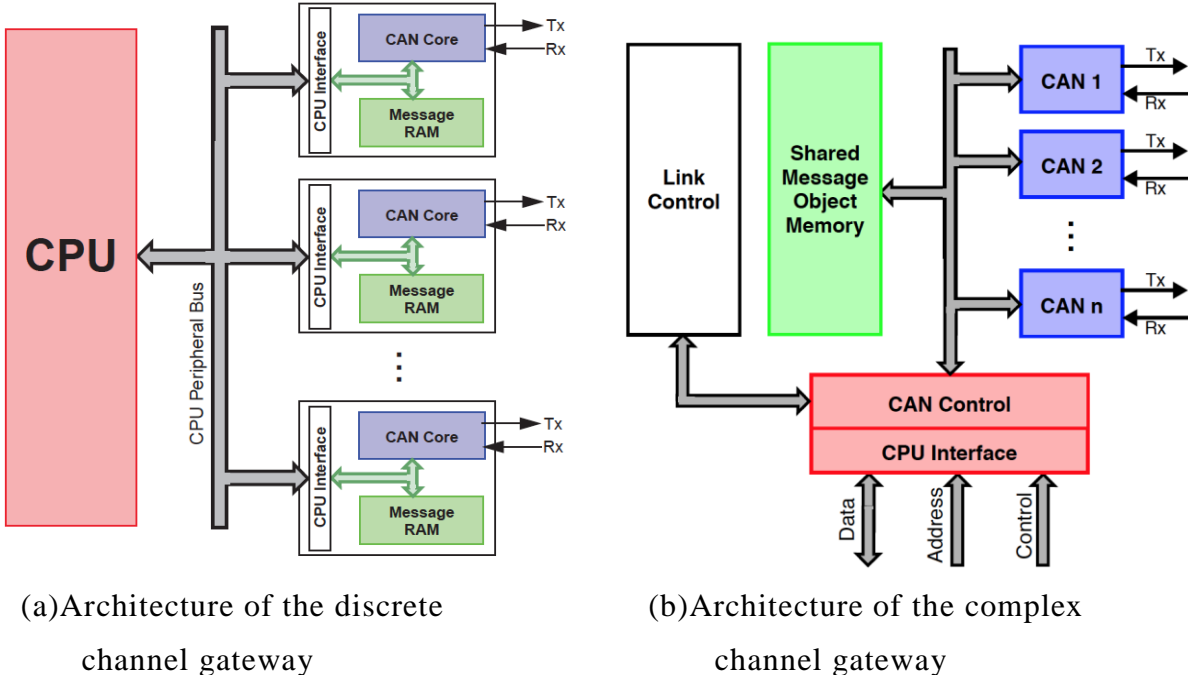


Fig.3 Architecture of the gateway

The complex channel gateway is more elaborate than the discrete channel gateway; the former consists of a shared message RAM, several CAN cores, and an internal control unit (CAN control). In some cases, a link control is added to provide basic gateway functionality without generating any CPU load. The message RAM is implemented as a shared memory, and the messages for all CAN channels are combined in the same RAM block to reduce the need for internal data transfer and lighten CPU load in the process.

3. Distributed Functions and Systems of Automobiles

3.1 Distributed Functions

In the future, automobiles are required to support an ever-increasing number of complex, distributed, and interdependent functions [6]. Many complex active and passive safety functions are introduced into automobiles; furthermore, the implementation process depends on the interaction, feedback, and coordination of multiple processing units through networks. A series of high-end luxury cars (e.g., Lexus and Mercedes Benz) contain more than 800 functions that comprise more than

100 ECUs, and a single function covers various tasks and communication messages. The tasks of different functions share many heterogeneous ECUs, sensors, and actuators. Cross-domain message communication among different buses may be required to complete the information interaction procedure; however, this process increases the volume of data ECU communication traffic. For example, the brake-by-wire function is a distributed function that consists of multiple precedence-constrained tasks and messages (as displayed in Fig. 4) [12]. Upon stepping on the brake pedal, information on the surrounding physical environment is collected by the sensors. The contact signal is then transported to the ECU units by various types of buses. These ECU cells receive and transmit the corresponding signals to other ECUs, thus initiating the relevant bearing mechanism. The cells issue instructions and open the brake signals to the brake actuators; then, the actuators receive the signal to perform the braking operation. This entire computation and communication process can be accomplished in a millisecond. The tasks involved in the calculation procedure, from the collection of data to the execution of the brake command, are subject to significant precedence constraints. The entire process must be allocated to five ECUs, two actuators, one sensor, and one gateway. Furthermore, different communication signals are generated in various tasks and packaged into corresponding messages that are transmitted through network buses.

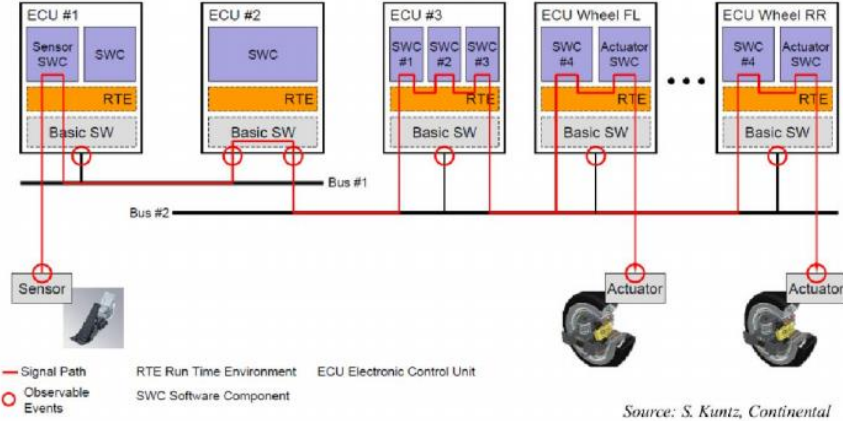


Fig.4 Brake by wire function

3.2 Automotive Cyber-Physical Systems

The principal challenges in embedded systems stem from interactions with physical processes and not from limited resources. The term “cyber-physical systems” (CPS) refers to the integration of computations with physical processes; in these systems, embedded computers and networks usually monitor and control physical processes with feedback loops. These processes affect the computations and

vice versa [13].

Automotive embedded systems are also called automotive CPS (ACPS) because automobiles support distributed functions with end-to-end computations that collect and transfer physical-world data from the 360° sensors to the actuators. These systems comprise throttle, brake, and steering subsystems as well as advanced human-machine interface devices. Active safety functions include adaptive cruise control and lane departure warning or lane keeping systems [6]. In an active cruise control system (Fig. 5), a set of radars (or of other sensors) scan the road in front of a car to ensure that no other cars or objects are moving at a low speed or have stopped in the middle of a lane. If such an object is detected, the system lowers the target speed of cruise control to match the speed of the detected obstacle. These functions are deployed together in automotive electronics system and possess the same sensing and actuation layers. In addition, active safety functions may share intermediate processing stages, such as sensor fusion, object detection functions, and actuator arbitration layers. Thus, a complex graph of functions (programmed as tasks) is generated with a high degree of communication dependency and deadlines on selected pairs of endpoints.

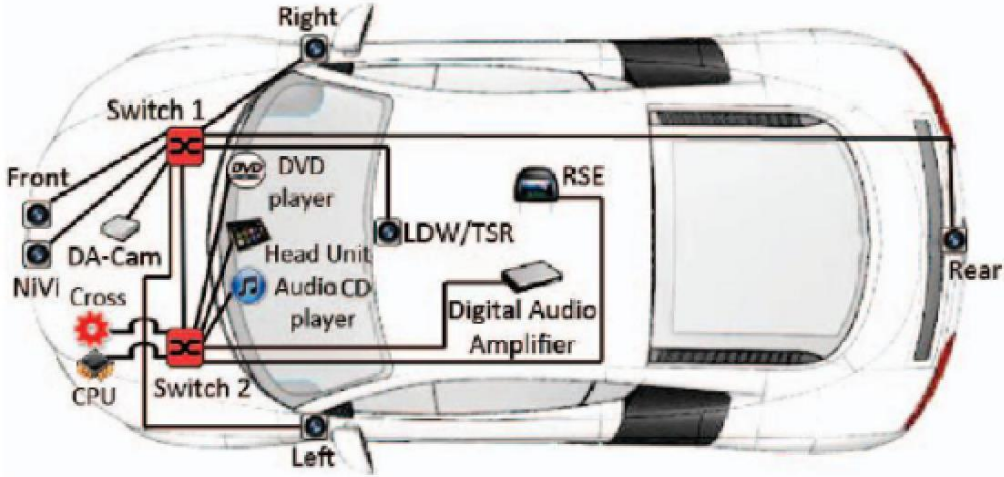


Fig. 5 Cruise control system

As pointed out in the preface of [13]: “In CPS, many things happen at once; physical processes are compositions of many parallel processes; concurrency is intrinsic in CPS; in addition to dealing with temporal dynamics, CPS designs invariably face challenging concurrency issues”. When we treat automotive embedded systems as ACPS, the inherent properties are dynamics and concurrency. Therefore, ACPS is not only a useful supplement to the current functional safety

standard but is also an urgent and inevitable trend in automotive functional safety research.

3.3 Automotive Mixed-criticality Systems

Automotive embedded systems are systems that mix power control, chassis control, safety control, and vehicle control subsystems. Each subsystem often contains multiple distributed functions; however, the network traffic of the segments is not always isolated. For instance, certain automotive applications require messages to be exchanged among different segments. For example, the engine controller of the powertrain subsystem obtains its input from climate control of the body subsystem, and the dashboard of the body subsystem displays information from other subsystems [14]. These functions are developed using different design methods and are provided by various levels of auto parts suppliers. As a result, diverse functional safety requirements are established.

Since their introduction in 2007 [15], mixed-criticality systems have become important topics in the embedded computing literature. A mixed-critical system integrates the hardware operating systems, middleware services, and application software that support the execution of safety-critical, mission-critical, and non-critical software within a single, secure computing platform [16].

Given their cost and other considerations, embedded systems are increasingly used to implement multiple functionalities in a single shared computing platform. The mixed-criticality concept is quickly becoming important to such systems and has been identified as a core foundational idea in the emerging CPS discipline. Mixed criticality postulates that certain functions tend to become more significant (critical) than others to the overall welfare of a platform that supports multiple functionalities. For instance, correcting the behavior of an automotive control system to enhance the effectiveness of ABS operation is more important than correcting the behavior of an on-board radio [16]. Hence, automotive embedded systems are also typical mixed-criticality systems, and are called automotive mixed-criticality systems.

ASIL refers to an abstract classification of inherent safety risks (as defined by ISO 26262) to identify the level of risk reduction required to prevent a specific hazard. The ASIL assessed for a specific hazard is then assigned to the safety goal set to address the risk in question. Subsequently, this level is referenced in the safety

requirements derived from that goal. ASIL is an adaptation of the safety integrated level that was introduced in IEC 61508 for the automotive industry. This classification helps define the safety requirements that must be satisfied to accord with ISO 26262. ASIL is established by analyzing the severity, exposure, and controllability of a vehicle under a hazard scenario. In turn, the safety goal for that particular hazard includes the ASIL requirements. The standard identifies four ASILs, namely, ASILs A, B, C, and D. ASILs D and A reflect the highest and lowest integrity requirements of the product, respectively.

Each hazardous events (e.g., missing the deadlines of functions) are classified according to the severity (S) of expected injuries. Severity involves four criticality levels, namely, S0, S1, S2, and S3, where S0 and S4 represent the lowest criticality level (i.e., no injuries) and the highest criticality level (i.e., life-threatening to fatal injuries), respectively.

Exposure indicates the relative expected frequency of the operational conditions under which injury may occur. This factor is classified in the ascending sequence of E0, E1, E2, E3, and E4; these classes represent incredible probability, very low probability (injury can be incurred only in rare operating conditions), low probability, medium probability, and high probability (injury can be inflicted under most operating conditions), respectively.

Controllability is an important safety metric in ISO 26262; this aspect indicates the relative likelihood that a driver can prevent injuries. Nonetheless, this metric focuses on the controllability of the driver rather than that of the system. Controllability is classified in the ascending order of C0, C1, C2, and C3; these classes reflect general controllability, simple controllability, normal controllability (most drivers can prevent injury), and difficult or impossible controllability, respectively.

On the basis of these classifications, ASIL D is defined as an event that has a reasonable possibility of causing life-threatening (survival uncertain) or fatal physical injury in most operating conditions. In this event, the driver has little chance of preventing such an injury. ASIL D combines the S3, E4, and C3 classifications; that is, ASIL D is the combination of S3, E4, and C3 classifications.

the level is reduced from D for every deterioration in the maximum values of any of these classifications (excluding the decline from C1 to C0). Table 1 shows the combination of ASILs. For example, a hypothetically uncontrollable (C3) fatal injury (S3) hazard can be classified as ASIL A if the hazard has a very low probability (E1) of happening. Quality management (QM) is the lowest ASIL (below A); according to the standard, this level is irrelevant to safety and requires only standard quality management processes.

Table 1 Automotive Safety Integrity Level

Severity class	Probability class	Controllability class		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Given that ASIL is the combination of severity, exposure, and controllability classifications, the safety requirement of an automotive function is actually the combination of the real time requirement, the reliability requirement of systems, and the controllability requirement of drivers.

4. Distributed computing Models of Automobiles

4.1 Function Model

Many studies for mixed-criticality systems have been developed in the past years, but such studies are mainly based on periodic and sporadic task models. As mentioned earlier, the distributed functions of automotive embedded systems have apparent

precedence constraints among tasks. A few models, such as time chains and task chains, have been employed in automobiles; however, these models are only suitable for simple distributed functions. With the increasing complexity and parallelization of automotive functions, a model that accurately reflects the distributed characteristics of automotive functions is desirable. In heterogeneous distributed systems, a distributed function with precedence-constrained tasks at a high level is described as a directed acyclic graph (DAG), in which the nodes represent the tasks and the edges represent the communication messages between the tasks [17]. The DAG-based model has also been applied to automotive embedded systems [18].

4.2 Synchronization Model

Besides the representation of a function, a more critical aspect is synchronization of tasks and messages in a function, such that timing constraint with end-to-end response time is determined [17]. End-to-end response time means a unified end-to-end execution and transmission path combined with tasks and messages to determine whether it can meet the real-time requirements. There has been increasing interest and importance in end-to-end synchronization of distributed automotive functions. However, although some approaches refer to synchronization, they do not take full advantage of DAG of parallel and distributed computing, and also do not create accurate and efficient schedules.

Communication is concurrent, and it is isolated with computation in the classical model. To obtain a more realistic description of more complex architectures, communication contention was introduced for messages in most studies. However, most of them ignore the heterogeneity of network. Modern automotive networks are no longer mere bus topologies but are mixtures of various network topologies, including buses, stars, rings, trees, and mesh types. For example, CANs and FlexRay are usually configured with a bus topology, but they can be divided by gateways to form other topologies over different domains. The media MOST in automotive networks is generally configured with a ring topology [17]. Therefore, compared with many different types of single networks or bus networks, the topology of modern automotive network is a mixed network topology, which consists of different network technologies with different topologies.

4.3 Mixed-criticality Systems Model

An automotive mixed-criticality system consists of multiple distributed mixed-criticality functions. An important property in mixed-criticality systems is the criticality level ASIL. ISO 26262 identifies four criticality levels denoted by ASILs (i.e., A, B, C, and D) mentioned earlier. Hence, the systems comprise more than two criticality levels (hence the name multiple-criticality systems) and are thus different from dual-criticality systems, which comprise only two criticality levels in many studies.

Two types of ASILs exist in automotive mixed-criticality systems, namely, the ASIL of systems and the ASILs of functions. Each function has an initial ASIL. The ASIL of systems can be changed to higher-criticality levels and back to lower-criticality levels. A change of the ASIL of systems indicates a switch in system mode. A function can only be executed on the modes in which its ASIL is higher than or equal to the ASIL of systems. Notice that different functions may come at different time instants on automotive mixed-criticality systems due to the dynamics of ACPS.

5. Distributed Computing for Functional Safety

Given that the safety requirement of an automotive function is actually the combination of the real time requirement, the reliability requirement of systems, and the controllability requirement of drivers, this section discusses the related functional safety issues by applying the distributed computing, namely, schedulability analysis, real-time scheduling, and reliability and fault-tolerance.

5.1 Distributed Computing for Schedulability Analysis

Determining the end-to-end delay bounds of distributed functions are extremely important for a safe and reliable design phase of automotive embedded systems. Timing verification techniques are employed to obtain the end-to-end delays. Three main verification techniques exist for real-time systems, namely, simulations, exhaustive exploration, and schedulability analysis (also referred as the worst case response time (WCRT) analysis) [10]. Simulations are relative easier approaches for developing and validating, but obtained results cannot be guaranteed to be safe, because it only explores partial combinations of the state space. Exhaustive exploration can find the exact result by exhausting all possible combinations, but it is only useful for very small-scale cases because the time complexity accelerates

exponentially. Schedulability analysis is an extensive approach by building a mathematical model of the worst possible situations that can be encountered at run-time. Schedulability analysis allows us to compute the WCRT bound, and provide firm and deterministic guarantees that are needed in safety-critical functions. Hence, schedulability analysis is a reasonable approach from the point view of safety. However, many problems still exist in automotive embedded systems.

First, most studies on the schedulability analysis of real-time systems are based on the classical Liu and Layland's (L&L) task model, in which critical instants are proposed [19]. The classical schedulability analysis method for a single CAN bus can be applied to segmented networks, but it can only obtain quite pessimistic results as it overlooks the influence of the gateway. To reduce the pessimism, the influence from the gateway for the WCRTs of messages of dual-buses was first systematically summarized [20]. However, their obtained WCRTs are still pessimistic. Recently, end-to-end schedulability analysis for CAN messages has attracted a few researches on CAN buses interconnected by a gateway. In the aforementioned research, the "busy period" is defined for messages to obtain a safe but pessimism upper bounds. However, existing approaches on schedulability analysis for messages only focus on two buses interconnected by a gateway.

Second, the gateway is an important node that is required to copy thousands of CAN messages from one bus to the other bus(es) to complete these distributed cross-bus functionalities. Hence, the end-to-end schedulability analysis for CAN messages and gateway tasks is an important research topic. However, existing end-to-end time analysis methods do not consider the effects of gateways. To achieve a safe and reliable design of in-vehicle networks, we should analyze the WCRT of gateway tasks [21].

Third, existing schedulability analysis methods either explicitly enumerate combinations (i.e., exhaustive exploration), which lead to combinatorial explosion problems, or mainly calculate safe upper bounds (i.e., approximate response times) to improve efficiency at the cost of precision loss. In many practical cases, especially in complex architectures with gateways and heterogeneous communication, the approximate response time cannot capture the complexity of practical systems.

Therefore, an approximate and pessimistic result cannot be predicted and quantified, and unnecessarily conservative design choices may be obtained. Hence, an exact WCRT analysis becomes necessary for in-vehicle networks.

Fig. 7 shows the probability distribution of interfering delays for the task under analysis [21]. The starting and ending points of the distribution curve are the best and worst case delays, respectively. Determining the exact worst case delay is a combinatorial explosion problem. As such, several existing studies aim to estimate a pessimistic but safe upper bound delay within a pseudo-polynomial computational time.

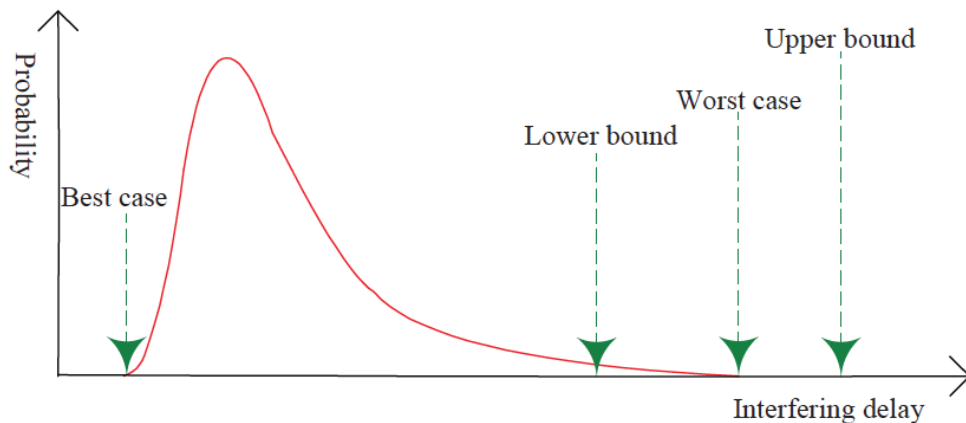


Fig. 7 Distribution of interfering delays

As depicted in Fig.7, a large body of interfering delays is close to the best case. In general cases, interfering delays rarely get close to the worst case. If we can find a sufficiently large lower bound delay which is larger than or equal to the lower bound delay and is less than or equal to the worst case delay in an extremely short time, then we can obtain the exact worst case delay belonging to the small scope of the lower bound and the upper bound delays.

5.2 Distributed Computing for Real-time Scheduling

Severity is caused by a hazardous event, and one of the parts of ASIL of ISO 26262. Each function has different severity levels and missing the deadlines of certain high-criticality functions may cause fatal injuries to people under this situation.

The mixed-criticality scheduling problem was first identified and formalized by Vestal [15], whose work has been extended and has inspired further substantial

investigations [3,16]. However, the models of these works are only periodic and sporadic tasks models. Hence, these works only considered mixed-criticality from the "task level" perspective and cannot reflect the distributed characteristics of functions in automobiles.

The scheduling of a single DAG-based distributed function is the basis of the scheduling of multiple DAG-based distributed functions. List scheduling includes two phases: the first phase orders tasks according to the descending order of priorities (task prioritizing), whereas the second phase allocates each task to a proper processor (task allocation). Scheduling tasks for a single DAG-based function with the fastest execution is a well-known NP-hard optimization problem [17].

In recent years, the scheduling of multiple DAG-based distributed mixed-criticality functions in heterogeneous distributed architectures has been studied extensively [22,23]. The authors considered functions to be separated and used a temporal and spatial-partitioning scheme, in which safety-criticality functions are scheduled using static-cycling scheduling and non-safety-criticality functions are scheduled using fixed-priority preemptive scheduling [22]. The major problems are as follows: 1) they only considered two criticality levels based on dual-criticality systems and 2) the communication times of messages for connecting precedence-constrained tasks are ignored. In [23], the authors considered processors interconnected by the time-triggered Ethernet protocol for mixed-criticality systems. Although these works used the DAG-based model for distributed functions, they still adopted the improved earliest deadline first (EDF) and rate monotonic (RM) scheduling algorithms of real-time systems. Distributed architectures and functions should be combined with the methods of distributed embedded systems. For the functional safety of automobiles, scheduling should be considered at the "function level" and not at the "task level".

Some studies for dynamic multiple DAG-based distributed functions scheduling, which means that they arrive at different time instants, are proposed to minimize the overall scheduling length (i.e., makespan). However, two important factors can be improved for the high-performance of systems. One factor is heterogeneity. Most algorithms use the upward rank value for ordering tasks and the earliest finish time (EFT) for assigning processors. These two criteria are derived from single DAG-based

function scheduling algorithms and can be improved to permit the creation of accurate and efficient schedules on heterogeneous distributed systems. Another factor is fairness. Slowdown-driven and round-robin strategies are suitable choices for fairness. However, existing algorithms fail to make full use of the fairness in dynamic environments; and cause obvious unfairness to longer-makespan functions or shorter-makespan functions (longer-makespan functions have longer makespans than shorter-makespan functions if they use the same single DAG-based function scheduling algorithm).

Systems and functions in automotive mixed-criticality systems involve considerable conflicts. Overall makespan is the main concern in system performance, whereas deadlines are the major timing constraints of functions. The deadlines of all functions cannot be met in heterogeneous distributed embedded systems, particularly in resource-constrained distributed embedded environments. A high-criticality function (i.e., a function with high criticality level) has a considerably important and strict timing constraint for a given deadline. Missing the deadlines of high-criticality functions would result in fatal injuries to people. Most algorithms use fairness policies to reduce the overall makespan of systems in heterogeneous distributed systems; however, these policies could lead to the failure to meet the deadlines of high-criticality functions. Therefore, both performance and timing constraints should be considered to achieve a good overall makespan and low deadline miss ratio (DMR).

5.3 Distributed Computing for Reliability and Fault-tolerance

As mentioned earlier, exposure is an important part of ASIL of ISO 26262. Exposure means the relative expected frequency of the operational conditions in which the injury can possibly happen. However, exposure only reflects the basic range of qualitative probability; if this factor is considered in the computation, then the probability of failure must be quantified. In computer science, reliability is usually described the exposure.

Reliability is defined as the probability that the schedule is successful, i.e., succeeds to complete its execution. Fault-tolerant techniques are common way to improve the reliability, and replication is an important software fault-tolerant technique [24]. Many researches about reliability of heterogeneous distributed systems have been presented with replication. The widely accepted reliability model is that each hardware component (processor or communication link) is fail-silent and

is characterized by a constant failure rate per time unit. The occurrences of the failures follow a constant parameter Poisson law [24]. This law is also known as the exponential distribution model. The most representative of replication is the primary-backup replication, which is implemented by performing multiple replicas to improve the reliability [25]. However, many different characteristics and new problems still exist in automotive embedded systems.

First, a distributed function is integrated with precedence constrained tasks, which are closely related and dependent from each other. A predecessor task's failure might influence its successor tasks and easily cause failure propagation [26]. Moreover, whether the exit task of a function is successfully completed or not is the major concern of both functions and systems. Hence, how to construct reliability models of distributed functions including such failure propagation is desirable.

Second, adding more replicas could increase both the reliability and the makespan of a function intuitively. That is, the two criteria (makespan and reliability) are conflicting. Bi-objective scheduling for optimizing makespan and reliability is a bi-criteria optima problem [27]. However, distributed automotive functions do not need to strictly deal with the above bi-criteria because of the following reasons: 1) meeting the reliability requirements of functions is the main objective, whereas higher reliability for a function means higher cost rather than the bicriteria optimization; 2) any function cannot be 100% reliable, but only if the systems can meet the reliability requirement of the function, then we recognize the function is reliable. Hence, the reliability problem is that meeting the reliability requirement while still reducing the resource consumption as soon as possible.

Third, two typical primary-backup replication methods exist in existing research, namely, the one backup replica for each primary replica [27], and the fixed ε replicas for each primary replica [28]. The problem of the first method is that it only tolerates at most one failure, and cannot cope with potential multiple (more than two) failures. Although the second method may improve the reliability of the function to provide better service, it will cause higher redundancy and resource consumption with higher economic cost and longer makespan. Both of the above two methods are infeasible for automotive embedded systems. To reduce the resource consumption under meeting the reliability requirement of a function, replicas

quantifying for different tasks of a function should be considered.

6. Conclusion

Modern automotive embedded systems join the related characteristic of real-time systems, cyber-physical systems, mixed-criticality systems, and heterogeneous distributed systems. This study discusses the functional safety problems of automotive embedded systems according to the ISO 26262 standard. We discuss the schedulability analysis, real-time scheduling, and reliability and fault-tolerance of automotive embedded systems by applying the distributed computing. We believe that our study could help understanding the related functional safety issues of automotive embedded systems.

References

- [1] Furst S. Challenges in the design of automotive software. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010IEEE, 2010:256-258.
- [2] Sinha P. Architectural design and reliability analysis of a fail-operational brake-by-wire system from ISO 26262 perspectives. Reliability Engineering & System Safety, 2011, 96(10):1349–1359.
- [3] Li H, Baruah S. Load-based schedulability analysis of certifiable mixed-criticality systems. Proceedings of the tenth ACM international conference on Embedded Software ACM, 2010:99--108.
- [4] Schmidt E G, Schmidt K. Message Scheduling for the FlexRay Protocol: The Dynamic Segment. Vehicular Technology IEEE Transactions on, 2009, 58(5):2160 - 2169.
- [5] Zhu Q, Zeng H, Zheng W, et al. Optimization of task allocation and priority assignment in hard real-time distributed systems. ACM Transactions on Embedded Computing Systems, 2012, 11(4):386-424.
- [6] Natale M D, Sangiovanni-Vincentelli A L. Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools. Proceedings of the IEEE, 2010, 98(4):603-620.
- [7] Chakraborty S. Keynote Talk: Challenges in Automotive Cyber-physical Systems Design. 2012 25th International Conference on VLSI Design IEEE, 2012:9-10.
- [8] Zeller M, Prehofer C, Weiss G, et al. Towards Self-Adaptation in Real-Time, Networked Systems: Efficient Solving of System Constraints for Automotive

- Embedded Systems. Proceedings of the 2011 IEEE Fifth International Conference on Self-Adaptive and Self-Organizing Systems, IEEE, 2011:79-88.
- [9] Keskin U. In-vehicle communication networks: a literature survey. Technische Universiteitindhoven, 2009.
- [10] Navet, Nicolas, et al. Timing verification of automotive communication architectures using quantile estimation. Embedded Real-Time Software and Systems (ERTS 2014). 2014.
- [11] Taube, Jan, Florian Hartwich, and Helmut Beikirch. Comparison of CAN gateway modules for automotive and industrial control applications. Proceedings of the 10th international CAN Conference (ICC 2005). 2005.
- [12] Kuntz S. The TIMMO Project and Perspectives for Timing in AUTOSAR R4.0. The 2nd SymTA/S News Conference, 2009-10-09.
- [13] Lee E A, Seshia S A. Introduction to embedded systems: A cyber-physical systems approach, 2011.
- [14] Azketa E, Gutierrez J J, Palencia J C, et al. Schedulability analysis of multi-packet messages in segmented CAN. Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on IEEE, 2012:1 - 8.
- [15] Vestal S. Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance. RTSS IEEE Computer Society, 2007:239-243.
- [16] Baruah S, Li H, Stougie L. Towards the Design of Certifiable Mixed-criticality Systems. 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium IEEE Computer Society, 2010:13-22.
- [17] Xie G, Li R, Li K. Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems. Journal of Parallel & Distributed Computing, vol. 83, pp. 1-12, 2015.
- [18] Zeng H, Natale M D, Ghosal A, et al. Schedule Optimization of Time-Triggered Systems Communicating Over the FlexRay Static Segment. Industrial Informatics IEEE Transactions on, 2011, 7(1):1 - 17.
- [19] Liu C L, Layland J W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM, 1973, 20(3):179–194.
- [20] Xie Y. Worst Case Response Time Analysis for Messages in Controller Area Network with Gateway. IEICE Transactions on Information & Systems, 2013, E96-D(7):1467-1477.

- [21] Xie G, Zeng G, Kurachi R, et al. Gateway modeling and response time analysis on can clusters of automobiles, in *Embedded Software Systems (ICESS)*, 2015 IEEE 12th Conference on. IEEE, 2015, pp. 1147–1153.
- [22] Tamas-Selicean D, Pop P. Design Optimization of Mixed-Criticality Real-Time Applications on Cost-Constrained Partitioned Architectures. 2011 32nd IEEE Real-Time Systems Symposium IEEE Computer Society, 2011:24-33.
- [23] Tamas-Selicean D, Marinescu S O, Pop P. Analysis and optimization of mixed-criticality applications on partitioned distributed architectures. *System Safety*, incorporating the Cyber Security Conference 2012, 7th IET International Conference on IET, 2012:1-6.
- [24] Chtepen M, Claeys F H A, Dhoedt B, et al. Adaptive Task Checkpointing and Replication: Toward Efficient Fault-Tolerant Grids. *IEEE Transactions on Parallel & Distributed Systems*, 2009, 20(2):180-190.
- [25] Zheng Q, Veeravalli B. On the design of communication-aware fault-tolerant scheduling algorithms for precedence constrained tasks in grid computing systems with dedicated communication devices. *Journal of Parallel & Distributed Computing*, 2009, 69(3):282-294.
- [26] P. Fenelon, J. A. McDermid, M. Nicolson, D. J. Pumfrey, Towards integrated safety analysis and design, *ACM SIGAPP Applied Computing Review* 2 (1) (1994) 21–32.
- [27] Girault A, Kalla H. A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate. *IEEE Transactions on Dependable & Secure Computing*, 2009, 6(4):241-254.
- [28] Benoit A, Hakem M, Robert Y. Fault Tolerant Scheduling of Precedence Task Graphs on Heterogeneous Platforms. *Parallel and Distributed Processing, IPDPS*, IEEE International Symposium on IEEE, 2007:1 - 8.