Fault Tolerant Services for Safe In-Car Embedded Systems

Nicolas Navet, Françoise Simonot-Lion

Fault Tolerant Services For Safe In-Car Embedded Systems

26th October 2004

Nicolas Navet (LORIA UMR 7503 - INRIA)
Phone: +33 3 83 58 17 63
Françoise Simonot-Lion (LORIA UMR 7503 - INPL)
Phone: +33 3 83 58 17 62
Campus Scientifique - BP 239 - 54506 - Vandoeuvre-lès-Nancy - France
Fax: +33 3 83 58 17 01
Email: {Nicolas.Navet, Francoise.Simonot}@loria.fr

Contents

Abstract

Due to the increasing criticality of the functions in terms of safety, embedded automotive systems must now respect stringent dependability constraints despite the faults that may occur in a very harsh environment. In a context where critical functions are distributed over the network, the communication system plays a major role. First, we discuss the main services and functionalities that a communication system should offer for easying the design of fault-tolerant applications in the automotive context. Then, we review the features of the protocols that are currently considered for being used and, finally, we highlight areas where developments are still needed.

1 Introduction

In the next decade, most features of a car will be supported by an electronic embedded system. This strategy is already used for functions like light management, window management, door management, etc. as well as for the control of traditional functions like braking, steering, etc. Moreover, the planned deployment of X-by-Wire technologies is leading the automotive industry in the world of safety-critical applications. Therefore, such systems must, obviously, respect their *functional requirements*, obey the properties of *performance* and *cost* and furthermore, guarantee their *dependability* despite the possible faults (physical or design) that may occur. More precisely, the design of such systems must take into account the dependability of two kinds of requirements. On the one hand, safety, the absence of catastrophic consequences, for the driver, the passengers and the environment, has to be ensured and on the other hand, the system has to provide reliable service and be available for the sollications of its users. This section introduces the emerging standards that are likely to influence the certification process for in-vehicle embedded systems and describes the general concepts of dependability and the means by which dependability can be attained. The communication system is a key point for an application: it is in charge of the transmission of critical information or events between functions that are deployed on distant stations (Electronic Control Units - ECUs) and it is a means for the OEM (carmakers) to integrate functions provided by different suppliers. So, in this chapter, we pay special attention to in-vehicle embedded networks and to the services that enhance the dependability of the exchanges and the dependability of the embedded applications. Note that a classical means, that is sometimes imposed by the regulatory policies in domains close to those in automotives, consists of introducing mechanisms that enable a system to tolerate faults. The purpose of section 2 is to present the main services, provided by a protocol, that allow an application to tolerate certain faults. These services generally provide fault detection and, for some of them, are able to mask fault occurrences from upper layer and to prevent the propagation of faults. In section 3, we compare some classes of protocols with respect to their ability to ensure services for increasing the dependability of an application. For each class, we will discuss the effort needed at the middleware level or application level for reaching the same quality of system.

1.1 The issue of safety-critical systems in the automotive industry

In some domains recognized as critical (for example, nuclear plants, railways, avionics), safety requirements in computer-based embedded systems are very rigorous and the manner of specification and the management of dependability / safety requirements is an important issue. These systems have to obey regulatory policies that require these industries to follow a precise certification process. At the moment, nothing similar exists in the automotive industry for certifying electronic embedded systems. Nevertheless, the problem is crucial for car-makers as well as for suppliers and, so, several proposals, are presently under study. Among the existing certification standards [32], RTCA/DO-178B [14], used in avionics, or EN 50128 [8], applied in the railway industry, provide stringent guidelines for the development of a safety-critical embedded system. But, these standards are hardly transposable for in-vehicle software-based systems: partitioning of software (critical / non critical), multiple versions, dissimilar software components, use of active redundancy, hardware redundancy. In the automotive sector, the Motor Industry Software Reliability Association (MISRA), a consortium of the major actors of automotive products in UK, proposes a loose model for the safety-directed development of vehicles with software on-board [22]. Finally, the generic standard IEC 61508 [18], applied to Electrical / Electronic / Programmable Electronic systems is a good candidate for supporting a certification process in the automotive industry. In Europe, in particular, in the transport domain, the trend is to move from "rule based" to "risk based" regulation [29]. So, the certification process will certainly be based on the definition of safety performance levels that characterize a safety function regarding the consequences of its failures defined as catastrophic, severe, major, minor or insignificant. The IEC 61508 standard proposes, in addition to other requirements on the design, validation and testing processes, 4 integrity levels, termed "Safety Integrity Levels" (SIL) and a quantitative safety requirement for each (see table 1). The challenge is therefore to prove that each function realized by a computer-based system, reaches the requirements imposed by its Safety Integrity Level. "Dependability", "safety", "failure", etc. are terms used in standard documents. So, we evoke, in the next section, definitions admitted in the context of dependability.

1.2 Generic concepts of dependability

Dependability is defined in [3] as "the ability of a system to deliver service that can justifiably be trusted". The service delivered by a system is its behavior as it is perceived by another system (human or physical) interacting with it.

A service can deviate from its desired functionality. The occurrence of such an

Integrity Level	Probability of dangerous failure occurrence / hour
SIL 4	$P \leq 10^{-8}$
SIL 3	$10^{-8} < P \leq 10^{-7}$
SIL 2	$10^{-7} < P \leq 10^{-6}$
SIL 1	$10^{-6} < P \leq 10^{-5}$

Table 1: Relationship between integrity levels and quantitative requirements for a system in continuous operation (IEC-61508)

event is termed a *failure*. An *error* is defined as the part of the system state that may cause a failure. A *fault* is the determined or hypothesized cause of an error. It can be active, when it produces an error and dormant otherwise. A system fails according to several *failure modes*. A failure mode characterizes a service that does not fit with its desired functionality according to three parameters: the failure domain (value domain or time domain, see section), the perception of the failure by several users of the system (consistent or inconsistent) and the consequences of the failures (from insignificant to catastrophic). As we will see in section 2 at the communication level, services are available to contend with the occurrences of failures in the value or time domain and to preserve the consistency, as well as the possibility, of the perception of a failure by several stations. The consequence of a failure at the communication level is the responsability of the designer of the embedded system and its assessment is a difficult issue.

Dependability is a concept that covers, in fact, several attributes. From a quality point of view, *reliability*, or the continuity of a correct service, and *availability*, expressing the readiness for a correct service, are important for automotive embedded systems. Note that the on-line detection of a low level of the reliability or availability of a service supported by an embedded system can lead to the “non-availability” of the vehicle and consequently affect the quality of the vehicle as perceived by the customer.

Safety is the reliability of the system regarding critical failure modes, or failure modes leading to catastrophic, severe or major consequences [2]. This attribute characterizes the ability of a system to avoid the occurrences of catastrophic events that may be very costly in terms of monetary loss and / or human suffering.

One way to reach the safety objective is, first, to apply a safe development process in order to prevent and / or remove any design faults. As presented in [3], this method has to be completed, in the design step, with an evaluation of the embedded system's behavior (fault forecasting). This can be achieved through a qualitative analysis (identification of failure modes, component failures, environmental conditions leading to a system failure) and a quantitative analysis (the probability evaluation applied to some parameters for the verification of dependability properties). The last means for reaching dependability is to apply a fault tolerant approach. This technique is mandatory for in-car embedded systems because the environment of the system is partially known and the reliability of the hardware components cannot be fully guaranteed.

Note that, the problem, in the automotive industry, is not only to be compliant to standards whose purpose mainly concerns the safety of the driver, the passengers, the vehicle and its environment but also to ensure a level of performance, comfort, and, more generally, the quality of the vehicle. The specification, in a quantitative way, of the properties required by an electronic embedded system, and the proof that this system meets these requirements are the principal challenges in the automotive industry.

2 Safety-relevant communication services

In this section, we discuss the main services and functionalities that the communication system should offer for easying the design of fault-tolerant automotive applications. In order to reduce the development time and increase quality through the re-use of validated components, these services should, as much as possible, be implemented in layers below the applicative level software. More precisely, some services such as the global time are usually provided by the communication controller, while others, such as redundancy management, are implemented in the middleware software layer (e.g., OSEK Fault-Tolerant Layer [31] or the middleware described in [44]). As suggested in [24], solutions where the middleware is running on a dedicated CPU, will enhance the predictability of the system by reducing the interactions between the middleware layer and the application level software. In particular, it will prevent conflicts in accessing the CPU, which may induce temporal faults such as missed deadlines.

2.1 Reliable communication

The purpose of this section is to discuss the main services and features related to the data exchange one can expect for safety-critical automotive applications. On the one hand, these services serve to hide the occurrence of faults from higher levels. For example, a shielded transmission support will mask some EMIs (Electro-Magnetic Interferences), considered as faults. On the other hand, other services are intended to detect the occurrence of errors and to avoid their propagation in the system (e.g., a CRC will prevent corrupted data from being used by an applicative process).

2.1.1 Robustness against EMIs

Embedded automotive systems suffer from environmental perturbations such as α particles, temperature peaks or EMIs.. The latter type of perturbations has been identified for a long time [30, 50] as being a serious threat to the correct behavior of an automotive system. EMIs can either be radiated by some in-vehicle electrical devices (switches, relays, etc.) or come from a source outside the vehicle (radio, radar, flashes of lightning, etc.). EMIs could affect the correct functioning of all the electronic devices but the transmission support is a particularly “weak link”. The whole problem is to ensure that the system will behave according to its specification whatever the environment.

In general, the same Medium Access Control (MAC) protocol can be implemented on different types of physical layers (e.g., unshielded pair, shielded twisted pair or plastic optical fiber) which exhibit significantly different behavior with regards to EMIs (see [5] for more details on the electro-magnetic sensitivity of different types of transmission support). Unfortunately, the use of an all-optical network, which offers very high immunity to EMIs, is not generally feasible because of the low-cost requirement imposed by the automotive industry.

Besides using a resilient physical layer, another means to alleviate the EMI problem is to replicate the transmission channels where each channel transports its own copy of the same frame. Although an EMI is likely to affect both channels in quite a similar manner, the redundancy provides some resilience to transmission errors.

The two previous approaches are classical means for hiding as well as possible a fault due to EMIs that can occur at the physical layer level. Nevertheless, when a frame is corrupted during transmission (i.e., at least one bit has been inverted), it is crucial that the receiver be able to detect it in order to discard the frame. This is the role of the Cyclic Redundancy Check (CRC) whose so-called Hamming distance indicates the number of inverted bits below which the CRC will detect the corruption. It is worth noting that if the Hamming distance of the MAC protocol CRC is too small with regards to the dependability objectives, a middleware layer can transparently insert an additional CRC in the data field of the MAC level frame. This will reinforce the ability of the system to detect errors happening during the transmission.

2.1.2 Time-triggered transmissions

One major design issue is to ensure that at run-time no errors will jeopardize the requirements imposed on the temporal behavior of the system; for data exchanges, these temporal requirements can be imposed on response times of frames or jitter upon reception. Among communication networks, one distinguishes time-triggered (TT) protocols where transmissions are driven by the progress of time (i.e., frames are transmitted at predefined points in time) and event-triggered (ET) protocols where transmissions are driven by the occurrence of events. Major representatives of ET and TT protocols considered for use in safety-critical in-vehicle communications will be discussed in section 3. Both types of communication have advantages and drawbacks but it is now widely admitted that dependability is much easier to ensure using a TT bus (see, for instance, [41, 13, 15, 1]), the main reasons being that

- access to the medium is deterministic (i.e., the order of the transmissions is defined statically at the design time and organized in "rounds" that repeat in cycles), and thus the frame response times are bounded and there is no jitter at reception,

- it simplifies the composability, which is the ability to add new nodes without affecting existing ones[1], as well as partitioning, which is the property that assures that a failure occurring in one sub-system cannot propagate to others,

- the behavior of a TT communication system is predictable, which makes it easier to understand its behavior and verify if the temporal constraints have been respected,

- message transmissions can be used as "heartbeats" which allow a very prompt detection of station failures,

- finally, the medium access scheme does not limit the network bandwidth, as is the case with the arbitration on message priority used by CAN, and thus large amounts of data can be transferred between nodes.

These reasons explain that, currently, only time-triggered communication systems are being considered for use in safety critical applications such as Steer-by-Wire [49, 48] or Brake-by-Wire.

2.1.3 Global time

Some control functions need to know the occurrence order among a set of events that happened in the system; some functions, such as diagnosis, even need to be able to precisely date them. This can be achieved by forming a global synchronized time base.

The second reason why a global time is needed comes from the TT communication scheme. In TT communications, as time drives the transmissions, all nodes of the network must have a coherent notion of time and a clock synchronization algorithm is required. This clock synchronisation algorithm is, in fact, a service that tolerates faults that can affect local clocks. In fact, the local clocks tend to drift apart since oscillators are not perfect; this imposes periodic resynchronization. For instance, on TTP/C, each node periodically adjusts its clock according to the difference between its own clock

[1] Adding new nodes requires that some bandwidth has been reserved for their transmission at design time. For instance, in TTP/C, some "slots" can be left free for future use.

and the average value of those from other nodes (the clocks with the highest value and lowest value are discarded).

A crucial performance metric for a clock synchronization algorithm is the maximum difference that can be observed among all local clocks. This value directly impacts the network's throughput in TT buses since the length of a transmission window, in addition to the actual transmission time of the frame, has to include some extra time to compensate for the skew between local clocks (i.e., a frame transmitted at the right point in time must not be rejected because the clock of a receiver diverges from the clock of the sender). Other criteria of major interest are the number and the type of faults (e.g., wrong clock value or no value received) that can be tolerated by the algorithm. For example, the TTP/C algorithm can tolerate a single fault on a network composed of at least 4 nodes (see [40] for a detailed analysis).

2.1.4 Atomic broadcast and acknowledgement

At a same point in time, it is mandatory that some functions distributed on the network have the same understanding of the state of the system in order to interoperate in a satisfactory manner. This implies that the information on the state of the system must be consistent throughout the whole network (this property is termed "spatial consistency" or "exact agreement"). The requirement of spatial consistency is particularly important for active redundancy [2], which is the basic strategy for ensuring fault-tolerance, i.e., the capacity of a system to deliver its service even in the presence of faults. To be able to compare the output results, it is crucial that the set of all replicated components process the same input data, which, in particular, implies that the values obtained from local sensors are exchanged over the network. All non-faulty nodes must thus receive the messages in the same order and with the same content. This property, which is called "atomic broadcast" or "interactive consistent broadcast" (see [9, 41]), enables distributed processes to reach common decisions or "consensus" despites faults, for instance, using majority voting.

[2] Active redundancy means that a set of components realizing the same functions in parallel enable the system to continue to operate despite the loss of one or more units. In passive redundancy, additional components are only activated when the primary component fails.

In practice, it may happen that all or a subset of nodes do not receive a message because of an incorrect signal shape due to EMIs or because nodes are temporarily faulty. The communication system usually provides, through the use of a CRC for detecting corrupted frames, a weak form of atomic broadcast that ensures that all stations that successfully receive a frame get the same value. This alone is however not sufficient for constructing fault-tolerant applications and, in addition, at least the acknowledgement of the reception of a message is needed because the sender, and possibly other nodes, may have to adapt their behavior according to this information (e.g., reschedule the transmission of the information in a subsequent frame). This latter requirement is important, in the automotive context, for distributed functions such as steering, braking or active suspension.

2.1.5 Avoiding “babbling-idiots”

As already said before, it is crucial that the system does not deviate from the temporal behaviour defined at design time. If a node does not behave in the specified manner, it has to be detected and masked at the communication system level in order to prevent the failure from propagating.

It may happen that a faulty ECU transmits outside its specification, e.g., it may send at a wrong point in time or send a frame larger than planned at design time. When communications are multiplexed, this will perturb the correct functioning of the whole network, especially the temporal behavior of the data exchanges. One well-known manifestation is the so-called "babbling idiots" [46, 45] nodes that transmit continuously (e.g., due to a defective oscillator). To avoid this situation, a component called the "bus guardian", restricts the controller’s ability to transmit by allowing transmission only when the node exhibits a specified behavior. Ideally, the bus guardian should have its own copy of the communication schedule, should be physically separated from the controller, should possess its own power supply and should be able to construct the global time itself. Due to the strong pressure from the automotive industry concerning costs, these assumptions are not fulfilled in general, which reduces the efficiency of the bus guardian strategy.

If the network has a star topology, with a central interface - called the “star” - for interconnection, instead of the classical bus topology, then the star can act as a central bus guardian and protect against errors that cannot be avoided by a local bus guardian. For instance, a star topology is more resilient to spatial proximity faults (e.g., temperature peaks) and to faults due to the desynchronization of an ECU (i.e., the star can disconnect a desynchronized station). To avoid a single point of failure, a dual star topology should be used with the drawback that the length of the wires is significantly increased.

2.2 Higher-level services

In this paragraph, we identify services that provide fault-tolerant mechanisms belonging conceptually to layers above the Medium Access Control in the OSI reference model.

2.2.1 Group membership service

As discussed in section 2.1.4, atomic broadcast ensures that all non-faulty stations possess the same variables describing the state of the system at a particular point in time. Another property that is required for implementing fault tolerance at a high level is that all non-faulty stations know the set of stations that are operational (or “non-faulty”). This service, which is basically a consensus on the set of operational nodes, is provided by the group membership and it is generally highly recommended for X-by-Wire applications. A classical example detailed in [24] is a brake-by-wire system where four ECUs, interconnected by a network, control the brakes located at the four wheels of the car. As soon as a wheel ECU is no longer functioning, the brake force applied to its wheel has to be redistributed among the remaining three wheels in such a way that the car can be safely parked. As pointed out in [24], for a brake-by-wire application, the time interval between the dysfunctioning of the wheel ECU and the knowledge of this event by all other stations has an impact on the safety of the application and thus it has to bounded and taken into account at design time.

A membership service implemented at the communication system level assumes that all nodes that are correctly participating in the communication protocol are non-faulty. In TT systems, as transmissions are perfectly foreseeable, the decisions regarding membership can be taken at points in time where frames should have been received. In a very simplified way, a missing or "faulty" frame indicates to the receivers that the sending node is not functioning properly. In addition, a node that is unable to transmit must consider itself as faulty and stops operating. Since it takes some time to detect faulty nodes, there can be faulty stations in the membership list of a node during some time intervals. The maximum number of such undetected faulty nodes, the maximum duration it takes to discover that a node is faulty, the maximum number of faulty stations and the types of faults than can be detected are major performance criteria of a membership algorithm. Other criteria include: the time needed for a "repaired" node to rejoin the membership list, how well the different nodes agree on the membership list at any point in time (are "cliques", i.e., sets of stations that disagree on the state of the system, possible? and how long can these cliques co-exist?) and the implementation overheads mainly in terms of CPU load and network bandwidth.

Group membership algorithms are complex distributed algorithms and formal methods are of great help in analyzing and validating them; the reader can refer to [9, 33, 40, 34] as good starting points on this topic.

2.2.2 Management of nodes' redundancy

A classical way for ensuring fault tolerance is to replicate critical components. We saw, in section 2.1.1, that the redundancy of the bus can hide faults due to EMIs. To achieve fault-tolerance, certain nodes are also replicated and clustered into so-called Fault-Tolerant Units (FTUs). A FTU is a set of several stations which perform the same function and each node of a FTU possesses its own slot in the round so that the failure of one or more stations in the same FTU can be tolerated. Actually, the role of FTUs is two-fold. First, they make the system resilient in the presence of transmission errors (some frames sent by nodes of the FTU may be correct while others are corrupted). Second, they provide a means to fight against measurement and computation errors occurring before transmission (some nodes may send the correct values while others may make errors).

Fail-silence property In the fault-tolerance terminology, a node is said fail-silent if 1.a) it sends frames at the correct point in time (correctness in the time domain), and 1.b) the correct value is transmitted (correctness in the value domain), or 2.) it sends detectably incorrect frames (e.g., wrong Cyclic Redundancy Check - CRC) in its own slot or no frame at all. A communication system such as TTP/C provides very good support for the requirements 1.a) and 2) (whose fulfillment provides the so-called "fail-silence in the temporal domain") especially through the bus guardian concept (see §2.1.5), while the value domain is the responsibility of higher level layers.

The use of fail-silent nodes greatly decreases the complexity of designing a critical application since data produced by fail-silent nodes is always correct and thus can be safely consumed by the receivers. Tolerating one arbitrary failure can be achieved with FTUs made of two nodes whereas three are necessary if the nodes are not fail-silent. However, in practice, it is difficult to ensure the fail-silent assumption, especially in the value domain. Basically, a fail-silent node has to implement redundancy plus error detection mechanisms and stop functioning after a failure is detected. Self-check mechanisms can be implemented in hardware or, more usually, in software on commercial off-the-shelf hardware [7]. An example of such mechanisms is the "double execution" strategy, which consists of running each task twice and to compare the output. However, both executions can be affected in the same way by a single error; a solution that provides some protection against so-called "common-mode faults" is to perform a third execution with a set of reference input data and to compare the output of the execution with pre-computed results that are known to be correct. This strategy is known as "double execution with reference check".

The reader is referred to [7, 17, 44] for good starting points on the problem of implementing fail-silent nodes.

Message agreement From an implementation point of view, it is usually preferable to present only one copy of data to the application in order to simplify the application code (considering possible divergences between replicated message instances is not needed) and to keep it independent from the degree of redundancy (i.e., the number of nodes composing a FTU).

The algorithm responsible for the choice of the value that will be transmitted to the application is termed "the agreement algorithm". Many agreement strategies are possible: pick-any (replicated messages are coming from a FTU made of fail-silent nodes), average-value, pick-a-particular-one (the selected value has been produced by the best sensor), majority vote, etc. OSEK/VDX consortium [31] has proposed a software layer responsible for implementing the agreement strategy. Two other important services of the OSEK FTCom (Fault-Tolerant Communication layer) are 1) to manage the packing of signals (elementary pieces of information such as the speed of the vehicle) into frames according to a pre-computed configuration, which is needed if the use of network bandwidth has to be optimized (see, for instance, [28, 42] for frame-packing algorithms), and 2) to provide message filtering mechanisms for passing only "significant" data to the application. Another fault-tolerant layer that offers the agreement service is described, as well as the set of associated tools, in [44].

2.2.3 Support for functioning mode

A functioning mode is a specific operational phase of an application. Typically, several functioning modes, that are mutually exclusive, are defined in a safety-critical application. For a vehicle, possible modes include factory mode (e.g., download of calibration parameters), pre-run mode (after doors are unlocked and before the engine is started - pre-heating is possible for some components), post-run mode (engine was shut-off but, for example, cooling can still be necessary), park mode (most ECUs are powered off) and even show-room mode. Besides these “normal” functioning modes, the occurrence of a failure can trigger the switching to a particular mode that will aim to bring the system back to a safe state again.

Particular functions corresponds to each functioning mode, which means a different set of tasks and messages as well as different schedules. If mode changes provide flexibility, great care must be taken that changes happen at the right points in time and that all nodes agree on the current mode. The communication system can provide some support in this area by ensuring that mode changes take place only at predefined points in time, are triggered by the authorized nodes and that the message schedule is changed simultaneously for all nodes. For example, TTP/C [25, 47] offers services for immediate mode changes (i.e., the change is performed at the end of the transmission window where it was requested) as well as deferred mode changes (i.e., the change is performed at the end of the current message schedule or cluster cycle in the TPP/C terminology).

3 Fault-tolerant communication systems

Among communication protocols that are considered for being used in safety-critical automotive systems, one can distinguish three main types:

- Protocols that have been designed from scratch to provide all the main fault-tolerant services. The prominent representative of this class is the TTP/C protocol [47].
- Protocols which offer the basic functionalities for fault-tolerant systems among which global time and bus guardians. The idea is to allow a scalable dependability on a per network or even on a per node basis. Missing features are to be implemented in software layers above the communication controllers. The representative of this class in the automotive context is FlexRay [11].
- Protocols not initially conceived with the objective of fault-tolerance to which missing features are added. This is the case with CAN [19], current de-facto standard in production cars, which is being considered for use in safety-critical applications (see, for instance, [15]) with the condition of additional features.

3.1 Dependability from scratch : TTP/C

The TTP/C protocol, which is specified in [47], was designed and extensively studied at the Vienna University of Technology. TTP/C is a central part of the Time-Triggered Architecture (TTA - see [23]) which is a complete framework for building fault-tolerant distributed applications according to the time-triggered paradigm. Hardware implementations of the TTP/C protocol, as well as software tools for the design of the application, are commercialized by the TTTech company and available today.

On a TTP/C network, transmission support is replicated and each channel transports its own copy of the same message. TTP/C can be implemented with a bus topology or a more resilient single star or dual star topology. At the Medium Access Control level, the TTP/C protocol implements a synchronous TDMA scheme: the stations (or nodes) have access to the bus in a strict deterministic sequential order and each station possesses the bus for a constant period of time called a “slot” during which it has to transmit one frame. The sequence of slots such that all stations have accessed the bus one time, is called a “TDMA round”. The size of the slot is not necessarily identical for all stations in the TDMA round, but a slot belonging to one station is the same size in each round. Consecutive TDMA rounds may differ according to the data transmitted during the slots, and the sequence of all TDMA rounds is the “cluster cycle” which repeats itself in a cycle.

TTP/C possesses numerous features and services related to dependability along with time-triggered communication. In particular, TTP/C implements a clique avoidance algorithm (the stations that belong to a “minority” in their understanding of the state of the system will eventually be excluded) and a membership algorithm that also provides data acknowledgment (one knows after a bounded time whether a station has received a message or not). Bus guardian, global clock and support for mode changes are also parts of the specification.

The algorithms used in TTP/C are by themselves intricate and interact in a very complex manner but most of them have been formally verified (see [6, 33, 40]). The fault hypothesis used for the design of TTP/C is well specified, but also quite restrictive (two successive faults such as transmission errors must occur at least two rounds apart). Situations outside the fault hypothesis are treated using “never give up” (NUP) strategies which aim to continue operating in a degraded mode. From the point of view of the set of available services, TTP/C is a mature solution. In our opinion, future research should investigate whether the fault hypothesis considered in the TTP/C design are pertinent in the context of automotive embedded systems where the environment can be very harsh (e.g., bursts of transmission errors may happen). This can be done starting from measurements taken on board of prototypes which would help to estimate the relevance of the fault-hypothesis. Other research could study the behavior of the communication system outside the fault-hypothesis and the impact on the application; this could be undertaken using fault-injection.

3.2 Scalable dependability : FlexRay

A consortium of major companies from the automotive field is currently developing the FlexRay protocol. The core members are BMW, Bosch, Daimler-Chrysler, General Motors, Motorola, Philips and Volkswagen. The first publicly available specifications of the FlexRay Protocol have already been released [11].

The FlexRay network is very flexible with regard to topology and transmission support redundancy. It can be configured as a bus, a star or multi-star and it is not mandatory that each station possess replicated channels nor a bus guardian, even though this should be the case for critical functions. At the MAC level, FlexRay defines a communication cycle as the concatenation of a time-triggered (or static) window and an event triggered (or dynamic) window. In each communication window, whose size is set statically at design time, a different protocol is applied. The communication cycles are executed periodically. The time-triggered window uses a TDMA MAC protocol; the main difference with TTP/C is that a station might possess several slots in the time-triggered window, but the size of all the slots is identical.

In the event-triggered part of the communication cycle, the protocol is FTDMA (Flexible Time Division Multiple Access): the time is divided into so-called mini-slots, each station possesses a given number of mini-slots (not necessarily consecutive) and it can start the transmission of a frame inside each of its own mini-slots. The bus guardian is not used in the dynamic window to control wether transmissions take place as specified. A mini-slot remains idle if the station has nothing to transmit. An example of a dynamic window is shown in Figure 1: on channel B, frame m has started being transmitted in mini-slot n while mini-slots n+1 and n+2 have not been used. It is

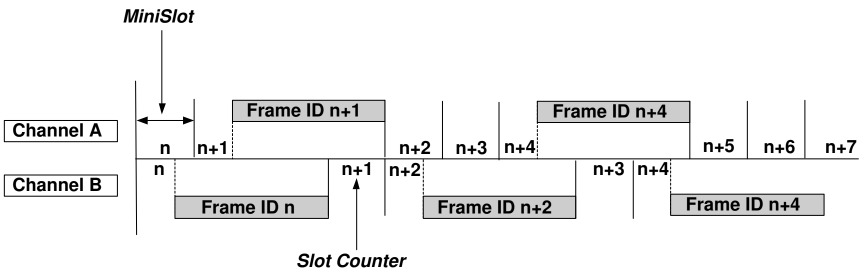

Figure 1: Example of message scheduling in the dynamic segment of the FlexRay communication cycle

noteworthy that frame n+4 is not received simultaneously on channels A and B since, in the dynamic window, transmissions are independent in both channels.

The FlexRay MAC protocol is more flexible than the TTP/C MAC since in the static window nodes are assigned as many slots as necessary (up to 4095 for each node) and since the frames are only transmitted if necessary in the dynamic part of the communication cycle. Compared to TTP/C, the structure of the communication cycle is not statically stored in the nodes, it is indeed revealed during the startup phase. However, unlike TTP/C, mode changes with a different communication schedule for each mode are not possible.

From the dependability point of view, FlexRay specifies solely time-triggered communication with bus guardian and clock synchronization algorithm on dual wires (shielded or unshielded - see [10] for the specifications of the physical layer). Should we consider the example of Brake-by-Wire in §2.2.1, the protocol offers no way offered for a node to know that one of the wheel ECUs is no longer operational, which would be needed to take the appropriate decision (e.g., redistribution of the brake force). Features that can be necessary for implementing fault-tolerant applications, such as membership and acknowledgment services or mode management facilities, will have to be implemented in software or hardware layers on top of FlexRay with the drawback that efficient implementations might be more difficult to achieve above the Data Link Layer level. There is indeed in literature individual solutions for each of the missing services but these protocols might have very complex interactions when used jointly, which requires that the whole communication profile is carefully validated by tests, simulation, fault-injection and formal proof under a well defined fault-hypothesis.

In automotive systems, critical and non-critical functions will increasingly co-exist and interoperate. In the FlexRay specification ([11] page 8), it is argued that the protocol provides scalable dependability i.e., the “ability to operate in configurations that provide various degrees of fault tolerance”. Indeed, the protocol allows for mixing single and dual transmission supports (interconnected though a star) on the same network, sub-networks of nodes without bus-guardians or with different fault-tolerance capability with regards to clock synchronization, nodes that do not send or receive time-triggered messages, etc. This flexibility can prove to be efficient in the automotive context in terms of cost and re-use of existing components if missing fault-tolerance features are provided in a middleware layer such as OSEK FTCom (see introduction of section 2 and reference [31]) or the one currently under development within the automotive industry project AUTOSAR (see `http://www.autosar.org`).

3.3 Adding missing features to an existing protocol: CAN

Controller Area Network has proved to be a very cost and performance effective solution for data exchange in automotive systems during the last 15 years. However, as specified by the ISO standards [19, 20], CAN lacks almost all the features and services identified in section 2 as important for the implementation of fault-tolerant systems: no redundant medium, no time-triggered communication, no global time, no atomic broadcast (even in the “weak form” described in §2.1.4, due to the well-known inconsistent message omission [39]), no reliable acknowledgment, no bus-guardian, no group membership, no functioning mode management services, etc.

Some authors advocate that “CAN can be used as a base and missing facilities can be added as needed” [15] and, over the last years, there was in fact a number of studies and proposals aimed at adding fault-tolerant features to CAN (see, for instance, [21, 13, 26, 27, 37, 12, 38, 4, 43, 35]). In the rest of this section, we discuss some such proposals of possible interest for automotive systems.

3.3.1 TTCAN : Time-triggered communications on top of CAN

Two main protocols were proposed to enable TT transmissions over CAN: TT-CAN (Time-Triggered Controller Area Network - see [21, 36]) and FTT-CAN (Flexible Time-Triggered CAN - see [13]). In the following, we consider TT-CAN, which has received much attention in the automotive field since it was proposed by Robert Bosch GmbH, a major actor in the automotive industry.

TT-CAN was developed on the basis of the CAN physical and data-link layers. The bus topology of the network, the characteristics of the transmission support, the frame format, as well as the maximum data rate - 1Mbits/s - are imposed by CAN protocol [36]. In addition to the standard CAN features, TT-CAN controllers must have the possibility to disable automatic retransmission and to provide the application with the time at which the first bit of a frame was sent or received [36]. Channel redundancy is possible, but not standardized, and no bus guardian is implemented in the node. The key idea is to propose, as with FlexRay, a flexible time-triggered/event-triggered protocol. TTCAN defines a basic cycle (the equivalent of the FlexRay communication cycle) as the concatenation of one or several time-triggered (or “exclusive”) windows and one event-triggered (or "arbitrating") window. Exclusive windows are devoted to time-triggered transmissions (i.e., periodic messages) while the arbitrating window is ruled by the standard CAN protocol: transmissions are dynamic and bus access is granted according to the priority of the frames. Several basic cycles, that differ in their organization (exclusive and arbitrating windows) and in the messages sent inside exclusive windows, can be defined. The list of successive basic cycles is called the system matrix and the matrix is executed in loops. Interestingly, the protocol enables the master node, the node that initiates the basic cycle through the transmission of the “reference message”, to stop functioning in TTCAN mode and to resume in standard CAN. Later, the master node can switch back to TT-CAN mode by sending a reference message.

3.3.2 Improving error confinement

CAN protocol possesses fault confinement mechanisms aimed at differentiating between short disturbances caused by electromagnetic interferences (EMI) and permanent failures due to hardware dysfunctioning. The scheme is based on error counters that are increased and decreased according to particular events (e.g., successful reception of a frame, reception of a corrupted frame, ...). The relevance of the algorithms involved is questionable (see [16]) but the main drawback is that a node has to diagnostic itself which can lead to the non detection of some critical errors such as the node transmitting continuously a dominant bit (one manifestation of the "babbling idiot" fault known as "stuck-at-dominant", see §2.1.5 and reference [4]). Furthermore, other faults such as the partitioning of the network into several sub-networks may prevent all nodes from communicating due to bad signal reflection at the extremities.

To address these problems, several solutions were proposed among which the variant of RedCAN discussed in [43] and CANcentrate [4]. The latter proposal is an active star that integrates some fault-diagnosis and fault-confinement mechanisms that can in particular prevent a stuck-at-dominant behaviour. The former proposal relies on a ring architecture where each node is connected to the bus through a switch that possesses the ability to exclude a faulty node or a faulty segment from the communication. These two proposals are promising but developments are still needed (e.g., test implementation, fault-injection, formal proofs) before they can be actually used in safety-critical applications. Furthermore, some faults such as a node transmitting correct frames more often than specified at design time are not covered by these proposals.

Many other mechanisms were proposed for increasing the dependability on CAN-based networks (see [26, 27, 37, 12, 38, 35]), but as pointed out in [37], if each proposal solves a particular problem, they have not been thought to be combined. Furthermore, the fault-hypothesis used in the design are not necessarily the same and the interactions between protocols remains to be studied in a formal way.

4 Conclusion

In the current state of practice, automotive embedded systems make widely use of fault-prevention (e.g., shielded ECU or transmission support), fault-detection (e.g., watchdog ECU that monitors the functioning state of the engine controller, check whether a data is obsolete or out-of-range) and fault confinement techniques (e.g., missing critical data are reconstituted on the basis of other data and more generally, specification and implementation of several degraded functioning modes). Redundancy is used at the sensor level (e.g., for the wheel angle) but seldom at the ECU level because of cost pressure and because the criticality of the functions does not absolutely impose it. Some future functions, such as brake and steer-by-wire, are likely to require active redundancy in order to comply with the acceptable risk levels and the design guidelines that could be issued by certification organisms.

For critical functions that are distributed and replicated throughout the network, the communication system will play a central role by providing the services that will simplify the implementation of fault-tolerant applications. The networks that are candidates are TTP/C, FlexRay and CAN-based time-triggered solutions. TTP/C is a mature technology that provides the most important services for supporting fault-tolerant applications. Moreover, TTP/C was designed under a well specified fault-hypothesis and the committees of most of its algorithms was formally proven. In our opinion, future research should investigate the relevance of the TTP/C fault-hypothesis in the context of automotive embedded systems and the behavior of the protocol outside the fault-hypothesis. At the time of writing, FlexRay, which is developed by the major actors of the European automotive industry, seems in a strong position for becoming a standard in the industry. The main advantage of FlexRay is its flexibility; in particular, it provides both time-triggered and event-triggered communications and nodes with different fault-tolerance capabilities can co-exist on the same network. The services provided by FlexRay do not fulfill all the needs for fault-tolerance and higher level protocols will have to be developed and validated before FlexRay can be used in very demanding applications. The major issue is that higher level implementations tend to be less efficient (e.g., bandwidth overhead for acknowledgment, maximum time needed for detecting faulty nodes). Finally, the solutions based on the TT-CAN protocol will require additional low-level mechanisms for fault-confinement as well as higher-level services such as atomic broadcast and membership. Many proposals exist for more dependability on CAN-based network but much work remains to be done to come up with a coherent and validated communication stack that includes all necessary services.

Acknowledgments. *We would like to thank Mr. Christophe Marchand, project leader in the field of diagnosis at PSA Peugeot Citroën, for helpful comments on an earlier version of this chapter.*

References

[1] A. Albert. Comparison of event-triggered and time-triggered concepts with regards to distributed control systems. In *Proceedings of Embedded World 2004*, Nürnberg, February 2004.

[2] ARTIST, Project IST-2001-34820. Selected topics in embedded systems design: Roadmaps for research, May 2004. Available at http://www.artist-embedded.org/Roadmaps/ARTIST_Roadmaps_Y2.pdf.

[3] A. Avizienis, J. Laprie, and B. Randell. Fundamental concepts of dependability. In *Proceedings of the 3rd Information Survivability Workshop*, pages 7–12, 2000.

[4] M. Barranco, G. Rodriguez-Navas, J. Proenza, and L. Almeida. CANcentrate: An active star topology for can networks. In *Proceedings of the 5th International Workshop on Factory Communication System*, 2004.

[5] J. Barrenscheen and G. Otte. Analysis of the physical CAN bus layer. In *4th international CAN Conference, ICC'97*, pages 06.02–06.08, Octobre 1997.

[6] G. Bauer and M. Paulitsch. An investigation of membership and clique avoidance in ttp/c. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems*, 2000.

[7] F. Brasileiro, P. Ezhilchelvan, S. Shrivastava, N. Speirs, and S. Tao. Implementing fail-silent nodes for distributed systems. *IEEE Transactions on Computers*, 45(11):1226–1238, 1996.

[8] CENELEC. Railway Applications - Software for Railway Control and Protection Systems, EN50128, 2001.

[9] T.D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.

[10] FlexRay Consortium. *FlexRay Communication System, Electrical Physical Layer, Version 2.0*, June 2004. Available at http://www.flexray.com.

[11] FlexRay Consortium. *FlexRay Communication System, Protocol Specification, Version 2.0*, June 2004. Available at http://www.flexray.com.

[12] J. Ferreira, L. Almeida, J. Fonseca, G. Rodriguez-Navas, and J. Proenza. Enforcing consistency of communication requirements updates in FTT-CAN. In *Proceedings of the 22nd Symposium on Reliable Distributed Systems*, 2003.

[13] J. Ferreira, P. Pedreiras, L. Almeida, and J.A. Fonseca. The FTT-CAN protocol for flexibility in safety-critical systems. *IEEE Micro, Special Issue on Critical Embedded Automotive Networks*, 22(4):46–55, July-August 2002.

[14] Radio Technical Commission for Aeronautics. RTCA DO-178B - Software Considerations in Airbone Systems and Equipment Certification, 1994.

[15] L.-B. Fredriksson. CAN for critical embedded automotive networks. *IEEE Micro, Special Issue on Critical Embedded Automotive Networks*, 22(4):28–35, July-August 2002.

[16] B. Gaujal and N. Navet. Fault confinement mechanisms on CAN: Analysis and improvements. *IEEE Transactions On Vehicular Technology*, 2004. Accepted for publication. Preliminary version available as INRIA Research Report at http://www.inria.fr/rrrt/rr-4603.html.

[17] M. Hiller. Software fault-tolerance techniques from a real-time systems point of view - an overview. Technical report, Chalmers University of Technology, Göteborg, Sweden, November 1998.

[18] IEC. IEC61508-1, Functional Safety of Electrical/Electronic/ Programmable Safety-related Systems - Part 1: General requirements, IEC/SC65A, 1998.

[19] International Standard Organization. *ISO 11519-2, Road Vehicles - Low Speed serial data communication - Part 2: Low Speed Controller Area Network*. ISO, 1994.

[20] International Standard Organization. *ISO 11898, Road Vehicles - Interchange of Digital Information - Controller Area Network for high-speed Communication*. ISO, 1994.

[21] International Standard Organization. *11898-4, Road Vehicles - Controller Area Network (CAN) - Part 4: Time-Triggered Communication*. ISO, 2000.

[22] P. H. Jesty, K. M. Hobley, R. Evans, and I. Kendall. Safety analysis of vehicle-based systems. In *Proceedings of the 8th Safety-critical Systems Symposium*, 2000.

[23] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[24] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[25] H. Kopetz, R. Nossal, R. Hexel, A. Krüger, D. Millinger, R. Pallierer, C. Temple, and M. Krug. Mode handling in the time-triggered architecture. *Control Engineering Practice*, 6(1998):61–66, Mar. 1998.

[26] G. Lima and A. Burns. Timing-independent safety on top of CAN. In *Proceedings of the 1st International Workshop on Real-Time LANs in the Internet Age*, 2002.

[27] G. Lima and A. Burns. A consensus protocol for CAN-based systems. In *Proceedings of the 24th Real-time Systems Symposium*, pages 420–429, 2003.

[28] R. Santos Marques, N. Navet, and F. Simonot-Lion. Frame packing under real-time constraints. In *Proceedings of the 5th IFAC International Conference on Fieldbus Systems and their Applications - FeT'2003, Aveiro, Portugal*, pages 185–192, July 2003.

[29] J. A. McDermid. Trends in system safety: A european view? In *Proceedings of the 7th Australian Workshop on Safety Critical Systems and Software*, 2002.

[30] I.E. Noble. EMC and the automotive industry. *Electronics & Communication Engineering Journal*, pages 263–271, Octobre 1992.

[31] OSEK Consortium. *OSEK/VDX Fault-Tolerant Communication, Version 1.0*, July 2001. Available at `http://www.osek-vdx.org/`.

[32] Y. Papadopoulos and J.A. McDermid. The potential for a generic approach to certification of safety-critical systems in the transportation sector. *Journal of Reliability Engineering and System Safety*, 63:47–66, 1999.

[33] H. Pfeifer. Formal verification of the ttp group membership algorithm. In *Proceedings of FORTE/PSTV 2000*, 2000.

[34] H. Pfeifer and F.W. von Henke. Formal Analysis for Dependability Properties: the Time-Triggered Architecture Example. In *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2001)*, pages 343–352, October 2001.

[35] L.M. Pinho and F. Vasques. Reliable real-time communication in can networks. *IEEE Transactions on Computers*, 52(12):1594–1607, 2003.

[36] Robert Bosch GmbH. Time triggered communication on CAN. Available at `http://www.can.bosch.com/content/TT_CAN.html`, 2004.

[37] G. Rodriguez-Navas, M. Barranco, and J. Proenza. Harmonizing dependability and real time in CAN networks. In *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, 2003.

[38] G. Rodriguez-Navas and J. Proenza. Clock synchronization in CAN distributed embedded systems. In *Proceedings of the 3rd Intl Workshop on Real-Time Networks*, 2004.

[39] J. Rufino, P. Veríssimo, G. Arroz, C. Almeida, and L. Rodrigues. Fault-tolerant broadcasts in CAN. In *Proceedings of the 28th International Symposium on Fault-Tolerant Computing Systems*, pages 150–159, Munich, Germany, June 1998. IEEE.

[40] J. Rushby. An overview of formal verification for the time-triggered architecture. In *Proceedings of Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 83–105, 2002.

[41] J. Rushby. A comparison of bus architecture for safety-critical embedded systems. Technical report, NASA/CR, March 2003.

[42] R. Saket and N. Navet. Frame packing algorithms for automotive applications. Technical Report RR-4998, INRIA, 2003. Available at `http://www.inria.fr/rrrt/rr-4998.html`.

[43] H. Sivencrona, T. Olsson, R. Johansson, and J. Torin. RedCAN: Simulations of two fault recovery algorithms for CAN. In *Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 302–311, 2004.

[44] C. Tanzer, S. Poledna, E. Dilger, and T. Fuhrer. A fault-tolerance layer for distributed fault-tolerant hard real-time systems. In *Proceedings of the Annual IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems*, 1999.

[45] C. Temple. Avoiding the babbling-idiot failure in a time-triggered communication system. In *Proceedings of the 28th International Symposium on Fault- Tolerant Computing*, Jun. 1998.

[46] K. Tindell and H. Hansson. Babbling idiots, the dual-priority protocol, and smart CAN controllers. In *Proceedings of the 2nd International CAN Conference*, pages 7.22–7.28, 1995.

[47] TTTech Computertechnik GmbH. *Time-Triggered Protocol TTP/C, High-Level Specification Document, Protocol Version 1.1*, November 2003. Available at http://www.tttech.com.

[48] C. Wilwert, Y.Q. Song, F. Simonot-Lion, and T. Clément. Evaluating quality of service and behavioral reliability of steer-by-wire systems. In *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Lisbon, Portugal, 2003.

[49] X-by-Wire Project, Brite-EuRam 111 Program. X-By-Wire - safety related fault tolerant systems in vehicles, final report, 1998.

[50] E. Zanoni and P. Pavan. Improving the reliability and safety of automotive electronics. *IEEE Micro*, 13(1):30–48, 1993.