In-vehicle communication networks - a historical perspective and review*

Nicolas Navet[1], Françoise Simonot-Lion[2]

August 31, 2013

1: University of Luxembourg, FSTC/LASSY, 6 rue Coudenhove-Kalergi,
L-1359 Luxembourg
2: LORIA - Université de Lorraine, Campus Scientifique, BP 239
54506 Vandoeuvre-lès-Nancy - France
Contact author: Nicolas Navet (nicolas.navet@uni.lu)

Abstract

The use of networks for communications between the Electronic Control Units (ECU) of a vehicle in production cars dates from the beginning of the 90s. The specific requirements of the different car domains have led to the development of a large number of automotive networks such as LIN, CAN, CAN FD, FlexRay, MOST, automotive Ethernet AVB, etc.. This report first introduces the context of in-vehicle embedded systems and, in particular, the requirements imposed on the communication systems. Then, a review of the most widely used, as well as the emerging automotive networks is given. Next, the current efforts of the automotive industry on middleware technologies which may be of great help in mastering the heterogeneity, are reviewed, with a special focus on the proposals of the AUTOSAR consortium. Finally, we highlight future trends in the development of automotive communication systems.

*This technical report is an updated version of two earlier review papers on automotive networks: N. Navet, Y.-Q. Song, F. Simonot-Lion, C. Wilwert, "Trends in Automotive Communication Systems", Proceedings of the IEEE, special issue on Industrial Communications Systems, vol 96, n^{o}6, pp1204-1223, June 2005 [66]. An updated version of this IEEE Proceedings then appeared as chapter 4 in The Automotive Embedded Systems Handbook in 2008 [62].

Contents

1 Automotive communication systems: characteristics and constraints 3
1.1 From point-to-point to multiplexed communications . . 3
1.2 Car domains and their evolution 4
1.3 Event-triggered versus Time-triggered 6
1.4 Different networks for different requirements 7
1.5 New challenges for in-vehicle communication systems . . 8
1.5.1 Real-time processing of large data quantities . . 8
1.5.2 Reducing electrical energy consumption 9
1.5.3 Security issues 9

2 In-car embedded networks 10
2.1 The CAN network 11
2.1.1 The CAN protocol 11
2.1.2 CAN's use in today's automobiles 15
2.1.3 Other priority buses: VAN and J1850 17
2.1.4 CAN FD: a high speed CAN network 17
2.2 Time-Triggered networks 18
2.2.1 The FlexRay Protocol 19
2.2.2 The TTCAN protocol 21
2.3 Low-cost automotive networks 22
2.3.1 The LIN network 23
2.3.2 The TTP/A network 25
2.3.3 The PSI5 and SENT networks 25
2.4 Multimedia and infotainment networks 25
2.4.1 The MOST network 26
2.4.2 The 1394 Automotive network 26
2.5 Automotive Ethernet 27
2.5.1 Motivation and use-cases for Ethernet 27
2.5.2 BroadR-Reach physical layer 28
2.5.3 Ethernet AVB 28
2.5.4 TTEthernet 29

3 Automotive middleware 29
3.1 Objectives of an embedded middleware 29
3.2 Former propositions of middleware 31
3.3 AUTOSAR - a standard for the automotive industry . . 32
3.3.1 The reference model 32
3.3.2 The communication services 33
3.3.3 The end-to-end communication protection library 37

4 Conclusions and discussion 38

1 Automotive communication systems: characteristics and constraints

1.1 From point-to-point to multiplexed communications

Since the 1970s, one observes an exponential increase in the number of electronic systems that have gradually replaced those that are purely mechanical or hydraulic. The growing performance and reliability of hardware components and the possibilities brought by software technologies enabled implementing complex functions that improve the comfort of the vehicle's occupants as well as their safety. In particular, one of the main purposes of electronic systems is to assist the driver to control the vehicle through functions related to the steering, traction (*i.e.,* control of the driving torque) or braking such as the ABS (Anti-lock Braking System), ESP (Electronic Stability Program), EPS (Electric Power Steering), active suspensions or engine control. Another reason for using electronic systems is to control devices in the body of a vehicle such as lights, wipers, doors, windows as well as entertainment and communication equipments (e.g., radio, DVD, hand-free phones, navigation systems). More recently appeared a large set of Advanced Driver Assistance Systems (ADAS), often camera based, such as brake and park assistance, lane departure detection, night vision assistance with pedestrian recognition or suspension proactively scanning the road surface [55]. In the future, vehicles will become entities of large intelligent transportation systems [51], involving cooperation mechanisms [50] through car-to-car and car-to-infrastructure communications.

In the early days of automotive electronics, each new function was implemented as a stand-alone Electronic Control Unit (ECU), which is a subsystem composed of a micro-controller and a set of sensors and actuators. This approach quickly proved to be insufficient with the need for functions to be distributed over several ECUs and the need for information exchanges among functions. For example, the vehicle speed estimated by the engine controller or by wheel rotation sensors, has to be known in order to adapt the steering effort, to control the suspension or simply to choose the right wiping speed. In today's luxury cars, up to 2500 signals (i.e., elementary information such as the speed of the vehicle) are exchanged by up to 70 ECUs [1]. Until the beginning of the 90s, data was exchanged through point-to-point links between ECUs. However this strategy, which required an amount of communication channels of the order of n^2 where n is the number of ECUs (*i.e.,* if each node is interconnected with all the others, the number of links grows in the square of n), was unable to cope with the increasing use of ECUs due to the problems of weight, cost, complexity and reliability induced by the wires and the connectors. These issues motivated the use of networks where the communications are multiplexed over a shared medium, which consequently required defining rules - protocols - for managing communications and, in particular, for granting bus access. It was mentioned in a 1998 press release (quoted in [45]) that the replacement of a "wiring harness with LANs in the four doors of a BMW reduced the weight by 15 kilograms". In the mid-1980s, the third part supplier Robert Bosch developed Controller Area Network (CAN) which was first integrated in Mercedes production cars in the early 1990s. CAN has become today the most widely used network in automotive systems, and already in 2004 the number of CAN nodes sold per year was estimated [38] to be around 400 millions (all application fields). Other communication networks, providing different services, are now being integrated in automotive applications. A description of the major networks is given in section 2.

1.2 Car domains and their evolution

As all the functions embedded in cars do not have the same performance or safety needs, different Quality of Services (e.g. response time, jitter, bandwidth, redundant communication channels for tolerating transmission errors, efficiency of the error detection mechanisms, etc.) are expected from the communication systems. Typically, an in-car embedded system is divided into several functional domains that correspond to different features and constraints. Two of them are concerned specifically with real-time control and safety of the vehicle's behavior: the "powertrain" (*i.e.* control of engine and transmission) and the "chassis" (*i.e.,* control of suspension, steering and braking) domains. The third, the "body", mostly implements comfort functions. The "telematics" (i.e. integration of wireless communications, vehicle monitoring systems and location devices), "multimedia" and "Human Machine Interface" (HMI) domains take advantage of the continuous progress in the field of multimedia and mobile communications. Finally, an emerging domain is concerned with the safety of the occupant and with functions providing driver assistance.

The main function of the powertrain domain is controlling the engine. This domain is increasingly constrained by stringent regulations, such as EURO 5 in force at the time of writing in Europe, for the protection of environment (e.g., emissions of particulate matter) and energy efficiency. The powertrain function is realized through several complex control laws with sampling periods of a magnitude of some milliseconds (due to the rotation speed of the engine) and implemented in micro-controllers with high computing power. In order to cope with the diversity of critical tasks to be treated, multi-tasking is required and stringent time constraints are imposed on the scheduling of the tasks (see [58] for typical automotive scheduling solutions). Furthermore, frequent data exchanges with other car domains, such as the chassis (e.g. ESP, ABS) and the body (e.g. dashboard, climate control), are required.

The chassis domain gathers functions such as ABS, ESP, ASC (Automatic Stability Control), 4WD (4 Wheel Drive), which control the chassis components according to steering/braking solicitations and driving conditions (ground surface, wind, etc). Communication requirements for this domain are quite similar to those for the powertrain but, because they have a stronger impact on the vehicle's stability, agility and dynamics, the chassis functions are more critical from a safety standpoint. Furthermore, the "X-by-Wire" technology, currently used for avionic systems, is now slowly being introduced to execute steering or braking functions. X-by-Wire is a generic term referring to the replacement of mechanical or hydraulic systems by fully electrical/electronic ones, which led and still leads to new design methods for developing them safely [100] and, in particular, for mastering the interferences between functions [4]. Chassis and powertrain functions operate mainly as closed-loop control systems and their implementation is moving towards a time-triggered approach [82, 44, 76, 72], which facilitates composability (*i.e.* ability to integrate individually developed components) and deterministic real-time behavior of the system.

Dashboard, wipers, lights, doors, windows, seats, mirrors, climate control are increasingly controlled by software-based systems that make up the "body" domain. This domain is characterized by numerous functions that necessitate many exchanges of small pieces of information among themselves. Not all nodes require a large bandwidth, such as the one offered by CAN; this lead to the design of low-cost networks, such as LIN (see section 2), or more recently, PSI5 and SENT. On LIN, only one node, termed the master, possesses an accurate clock and drives the communication by polling the other nodes - the slaves - periodically. The mixture of different communication needs inside the body domain leads to a hierarchical network architecture where integrated mechatronic sub-systems based on low-cost networks are interconnected through a CAN backbone. The activation of body functions is mainly triggered by the driver and passengers' solicitations (e.g. opening a window, locking doors, etc).

Telematics functions are becoming more and more numerous: hand-free phones, car radio, CD, DVD, in-car navigation systems, rear seat entertainment, remote vehicle diagnostic, stolen vehicle tracking, etc. These functions require a lot of data to be exchanged within the vehicle but also with the external world through the use of wireless technology (see, for instance, [81]). Here, the emphasis shifts from messages and tasks subject to stringent deadline constraints to multimedia data streams, bandwidth sharing, multimedia quality of service where preserving the integrity (*i.e.,* ensuring that information will not be accidentally or maliciously altered) and confidentiality of information is crucial. HMI aims to provide Human Machine Interfaces that are easy to use and that limit the risk of driver inattention [19].

Electronic-based systems for ensuring the safety of the occupants are increasingly embedded in vehicles. Examples of such systems are: impact and roll-over sensors, deployment of airbags and belt pretensioners, tyre pressure monitoring, Adaptive Cruise Control (or ACC - the car's speed is adjusted to maintain a safe distance with the car ahead), lane departure warning system, collision avoidance, driver intent and driver drowsiness detection, night vision assistance. These functions form an emerging domain usually referred to as "active and passive safety".

1.3 Event-triggered versus Time-triggered

One of the main objectives of the design step of an in-vehicle embedded system is to ensure a proper execution of the vehicle functions, with a pre-defined level of safety, in the normal functioning mode but also when some components fail (e.g., reboot of an ECU) or when the environment of the vehicle creates perturbations (e.g., EMI causing frames to be corrupted). Networks play a central role in maintaining the embedded systems in a "safe" state since most critical functions are now distributed and need to communicate. Thus, the different communication systems have to be analysed in regard to this objective; in particular, messages transmitted on the bus must meet their real-time constraints, which mainly consist of bounded response times and bounded jitters.

There are two main paradigms for communications in automotive systems: time-triggered and event-triggered. Event-triggered means that messages are transmitted to signal the occurrence of significant events (e.g., a door has been closed). In this case, the system possesses the ability to take into account, as quickly as possible, any asynchronous events such as an alarm. The communication protocol must define a policy to grant access to the bus in order to avoid collisions; for instance, the strategy used in CAN (see §2.1.1) is to assign a priority to each frame and to give the bus access to the highest priority frame. Event-triggered communication is very efficient in terms of bandwidth usage since only necessary messages are transmitted. Furthermore, the evolution of the system without redesigning existing nodes is generally possible which is important in the automotive industry where incremental design is a usual practice. However, verifying that temporal constraints are met is not obvious and the detection of node failures is problematic.

When communications are time-triggered, frames are transmitted at pre-determined points in time, which is well-suited for the periodic transmission of messages as it it required in distributed control loops. Each frame is scheduled for transmission during one pre-defined interval of time, usually termed a slot, and the schedule repeats itself indefinitely. This medium access strategy is referred to as TDMA (Time Division Multiple Access). As the frame scheduling is statically defined, the temporal behavior is fully predictable;

thus, it is easy to check whether the timing constraints expressed on data exchanges are met. Another interesting property of time-triggered protocols is that missing messages are immediately identified; this can serve to detect, in a short and bounded amount of time, nodes that are presumably no longer operational. The first downside is the inefficiency in terms of network utilization and response times by comparison with the transmission of aperiodic messages (i.e. messages that are not transmitted in a periodic manner). A second drawback of time-triggered protocols is the lack of flexibility even if different schedules (corresponding to different functioning modes of the application) can be defined and switching from one mode to another is possible at run-time. Finally, the unplanned addition of a new transmitting node on the network induces changes in the message schedule and, thus, may necessitate the update of all other nodes. TTP/C [95] is a purely time-triggered network but there are networks, such as TTCAN [37], FTT-CAN [18], FlexRay [11] or TTEthernet [94], that can support a combination of both time-triggered and event-triggered transmissions. This capability to convey both types of traffic fits in well with the automotive context since data for control loops as well as alarms and events have to be transmitted.

Several comparisons have been done between event-triggered and time-triggered approaches, the reader can refer to [42, 1, 18] for good starting points.

1.4 Different networks for different requirements

The steadily increasing need for bandwidth[1] and the diversification of performance, costs and dependability[2] requirements lead to a diversification of the networks used throughout the car. In 1994, the Society for Automotive Engineers (SAE) defined a classification for automotive communication protocols [88, 87, 13] based on data transmission speed and functions that are distributed over the network. *Class A* networks have a data rate lower than 10 Kbit/s and are used to transmit simple control data with low-cost technology. They are mainly integrated in the "body" domain (seat control, door lock, lighting, trunk release, rain sensor, etc.). Examples of class A networks are LIN [79, 48] and TTP/A [27]. *Class B* networks are dedicated to supporting data exchanges between ECUs in order to reduce the number of sensors by sharing information. They operate from 10 Kbit/s to 125 Kbit/s. The J1850 [89] and low-speed CAN [33] are the main representatives of this class. Applications that need high speed real-time communications require *class C*

[1] For instance, in [4], the average bandwidth needed for the engine and the chassis control was estimated to reach 1500kbit/s in 2008 while it was 765kbit/s in 2004 and 122kbit/s in 1994.

[2] Dependability is usually defined as the ability to deliver a service that can justifiably be trusted, see [3] for more details.

networks (speed of 125Kbit/s to 1Mbit/s) or *class D* networks[3] (speed over 1Mb/s). Class C networks, such as high-speed CAN [35], are used for the powertrain and currently for the chassis domains, while class D networks are devoted to multimedia data (e.g., MOST [59]) and safety critical applications that need predictability and fault-tolerance (e.g., TTP/C [95] or FlexRay [11] networks) or serve as gateways between sub-systems (see the use of FlexRay at BMW in [85] and [39]).

It is common, in today's vehicles, that the electronic architecture includes four, five or more different types of networks interconnected by gateways. For example, the first-generation Volvo XC90 [38] embeds up to 40 ECUs interconnected by a LIN bus, a MOST bus, a low-speed CAN and a high-speed CAN. More recent high-end cars have much more complex architectures, such as the BMW 7 series launched in 2008 (see [39]), which implements four CAN buses, a FlexRay bus, a MOST bus, several LIN buses, an Ethernet bus and wireless interfaces. The architecture is organized around a central gateway connecting most of the data buses, and providing external connections for diagnostic and large data transfer (code upload, navigational maps, etc).

1.5 New challenges for in-vehicle communication systems

We discuss here three main evolutions that will probably profoundly shape future automotive communication networks and architectures.

1.5.1 Real-time processing of large data quantities

The first trend is the requirement for an increasing number of functions to *process, at a high rate, large quantities of data,* for instance produced by video and infrared cameras. A good example of such functions is a pedestrian detection system with auto-brake, which relies on image recognition and machine learning algorithms, and has to work at the highest possible vehicle speed. In the case of the pedestrian detection system introduced in 2010 in Volvo's S60, a collision with a pedestrian can be avoided at up to 35km/h [98]. This system has to analyse the flow of images and provide, on time, a deterministic verdict about whether there is a pedestrian in front of the car, and then possibly activate braking if the driver fails to respond in time. Such systems, often belonging to the active and passive safety car domain, require powerful computational resources. This is one of the rationale for the introduction of multicore architectures in automotive embedded systems [58]. Furthermore, these functions are often implemented on a distributed architecture, and usually require sensor fusion, so that the

[3] Class D is not formally defined but it is generally considered that networks over 1Mb/s belong to class D. This SAE classification is outdated today but was often refered to in the past.

underlying communication architecture has to provide a strong and guaranteed temporal quality of service. Another driver for increasing data streams are of course infotainment systems, consider for example a rear-set video system in HD quality.

1.5.2 Reducing electrical energy consumption

Energy efficiency is now a major concern when designing new embedded electronic architectures, while it was for the last 10 years a concern for actuators mainly. A reason is that, by regulation, the average CO2 emission per passenger car sold in the EU must fall below the 95g/km threshold from 2020 onwards. Similar regulations will come into force in many countries outside EU, for instance in China or in the USA [102]. At the same time, it is estimated [8] that, between 2002 and 2012, the average power requirement for the automotive electronics was multiplied by 4 and reaches now 3kW.

This lead to the introduction of two main strategies to achieve better energy efficiency: ECU degradation and partial networking. ECU degradation is the ability to switch off or reduce the execution rate of software modules, shutdown some I/O ports, or even put a complete CPU into idle mode. These low-power features are already supported in AUTOSAR. Partial networking means putting temporarily into sleep mode a group of nodes, or some entire network, when they do not need to be operational. For example, a parking assistance system that cannot be used above 20km/h can be deactivated at higher speeds. Many control modules in the body of the vehicle (seats, mirror, door, etc) can remain in sleep mode most of the time [8]. Then, the nodes are waken up when needed through specific messages. Audi estimates for instance that the potential savings of partial networking would be around 2.6g CO2/km [97, 8].

Partial networking has been an ongoing work in the automotive industry since 2008 [47], already fully supported in AUTOSAR for CAN. A working group of carmakers and semiconductor manufacturers was set up in 2010 to develop a specification for CAN transceivers with wake-up capability [97, 8]. The outcomes of this working group have been submitted for standardisation as ISO 11898-6, an extension to the existing CAN standard (ISO 11898). Several components supporting partial networking are commercially available [102] and a first implementation in a car has been announced for 2013 or 2014 [47].

1.5.3 Security issues

The last issue to be dealt with is the security of communication, and more generally security of the embedded systems. If safety has always been a matter of concern for automotive embedded systems, security issues have been in our view largely overlooked. The need for more security is driven by the ever increasing connectivity between the car and the external world, which includes not only telematics services and internet access, but also upcoming vehicle-to-vehicle or vehicle-to-infrastructure communication.

As demonstrated in [43], some existing security mechanisms, for instance to control ECU reprogramming, are weak and many current automotive systems usually do not resist to attacks, sometimes even relatively unsophisticated ones according to [43]. With a physical access to the vehicle, through a connection to the standard OBD-II port, the authors were for instance able to reflash ECUs, even critical ones controlling the engine and the brakes. The authors were also able to remotely compromise the security of the car, by injecting malicious code, and "ultimately monitor and control the car remotely over the Internet" [43]. Obviously such security breaches could have catastrophic consequences, especially with ADAS systems and future intelligent transportation systems requiring cooperation between vehicles. One may even imagine a scenario where an infected vehicle would contaminate other vehicles.

The technical answer to this security challenge will probably encompass a spectrum of solutions [43], ranging from cryptography (support for embedding a cryptographic module is available in AUTOSAR since release 4.0 [7]), message authentication mechanisms, security breach detection systems, etc. These functionalities will require additional bandwidth, and, for instance, CAN FD (see §2.1.4) would probably be better suited than normal CAN to fulfil these needs. The use of virtualization technologies [63] will also increase security because it enables to isolate functions or sub-systems running on the same ECUs having different security requirements, while enabling them to share information in a controlled manner. Typically, virtualization would help to prevent infotainment applications, that are connected to the external world through wireless connections, from accessing the internal communication buses.

2 In-car embedded networks

The different performance requirements throughout a vehicle, as well as the competition among companies, have led to the design of a large number of communication networks. The aim of this section is to give a description of the most representative networks for each main domain of utilization.

2.1 The CAN network

To ensure at run-time the "freshness"[4] of the exchanged data and the timely delivery of commands to actuators, it is crucial that the Medium Access Control (MAC) protocol is able to ensure bounded response times for frames. An efficient and conceptually simple MAC scheme that possesses this capability is the granting of bus access according to the priority of the messages (the reader can refer to [93, 67, 14, 56] for how to compute bound on response times for priority buses). To this end, each message is assigned an identifier, unique to the whole system. This serves two purposes: giving priority for transmission (the lower the numerical value, the greater the priority) and allowing message filtering upon reception. Beside CAN two other representatives of such "priority buses" are VAN and J1850, they will be briefly presented in §2.1.3.

2.1.1 The CAN protocol

CAN (Controller Area Network) is without a doubt the most widely used in-vehicle network. It was designed by Bosch in the mid 80's for multiplexing communication between ECUs in vehicles and thus for decreasing the overall wire harness: length of wires and number of dedicated wires (e.g. the number of wires has been reduced by 40%, from 635 to 370, in the Peugeot 307 that embeds two CAN buses with regard to the non-multiplexed Peugeot 306 [52]). Furthermore, it allows to share sensors among ECUs.

CAN on a twisted pair of copper wires became an ISO standard in 1994 [33, 35] and is now a de-facto standard in Europe for data transmission in automotive applications, due to its low cost, its robustness and the bounded communication delays (see [38]). In today's car, CAN is used as an SAE class C network for real-time control in the powertrain and chassis domains (at 250 or 500KBit/s), but it also serves as an SAE class B network for the electronics in the body domain, usually at a data rate of 125Kbit/s.

On CAN, data, possibly segmented in several frames, may be transmitted periodically, aperiodically or on-demand (*i.e.* client-server paradigm). A CAN frame is labeled by an identifier, transmitted within the frame (see Figures 1 and 2), whose numerical value determines the frame priority. There are two versions of the CAN protocol differing in the size of the identifier: CAN 2.0A (or "standard CAN") with an 11 bit identifier and CAN 2.0B (or "extended CAN") with a 29 bit identifier. For in-vehicle communications,

[4] The freshness property is verified if data has been produced recently enough to be safely consumed: the difference between the time when the data is used and its production time must be always smaller than a specified value. The delay experienced by a data is made up of the network latency, plus the latencies on the sending and receiving ends which can be large especially if the application tasks are not synchronized with the transmissions on the network (see [25]).

only CAN 2.0A is used since it provides a sufficient number of identifiers (i.e. the number of distinct frames exchanged over one CAN network is lower than 2^{11}).

CAN uses Non-Return-to-Zero (NRZ) bit representation with a bit stuffing of length 5. In order not to lose the bit time (i.e., the time between the emission of two successive bits of the same frame), stations need to resynchronize periodically and this procedure requires edges on the signal. Bit stuffing is an encoding method that enables resynchronization when using Non-Return-to-Zero (NRZ) bit representation where the signal level on the bus can remain constant over a longer period of time (e.g. transmission of '000000..'). Edges are inserted into the outgoing bit stream in such a way to avoid the transmission of more than a maximum number of consecutive equal-level bits (5 for CAN). The receiver will apply the inverse procedure and de-stuff the frame. CAN requires the physical layer to implement the logical "and" operator: if at least one node is transmitting the "0" bit level on the bus, then the bus is in that state regardless if other nodes have transmitted the "1" bit level. For this reason, "0" is termed the dominant bit value while "1" is the recessive bit value.

The standard CAN data frame (CAN 2.0A, see Figure 1) can contain up to 8 bytes of data for an overall size of, at most, 135bits, including all the protocol overheads such as the stuff bits. The sections of the frames are:

- the header field (see Figure 2), which contains the identifier of the frame, the Remote Transmission Request bit that distinguishes between data frame (RTR set to 0) and data request frame (RTR set to 1) and the Data Length Code (DLC) used to inform of the number of bytes of the data field,
- the data field having a maximum length of 8 bytes,
- the 15 bit Cyclic Redundancy Check (CRC) field which ensures the integrity of the data transmitted,
- the Acknowledgment field (Ack). On CAN, the acknowledgment scheme solely enables the sender to know that at least one station, but not necessarily the intended recipient, has received the frame correctly,
- the End-of-Frame (EOF) field and the intermission frame space which is the minimum number of bits separating consecutive messages.

Any CAN node may start a transmission when the bus is idle. Possible conflicts are resolved by a priority-based arbitration process, which is said non-destructive in the sense that, in case of simultaneous transmissions, the highest priority frame will be sent despite the contention with lower priority frames. The arbitration is determined by the arbitration fields (identifier plus

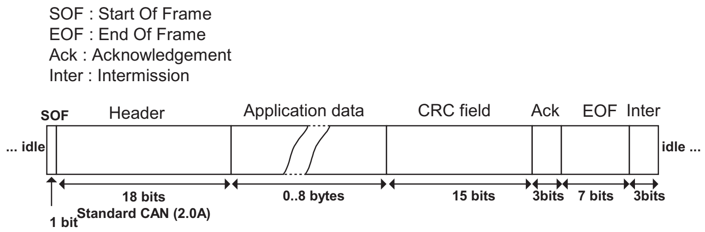

Figure 1: Format of the CAN 2.0A data frame

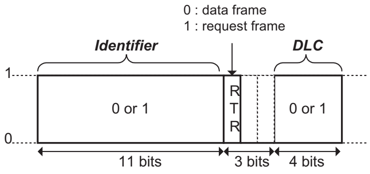

Figure 2: Format of the header field of the CAN 2.0A data frame

RTR bit) of the contending nodes. An example illustrating CAN arbitration is shown on Figure 3. If one node transmits a recessive bit on the bus while another transmits a dominant bit, the resulting bus level is dominant due to the “and” operator realized by the physical layer. Therefore, the node transmitting a recessive bit will observe a dominant bit on the bus and then will immediately stop transmitting. Since the identifier is transmitted “most significant bit first”, the node with the numerically lowest identifier field will gain access to the bus. A node that has lost the arbitration will wait until the bus becomes free again before trying to retransmit its frame.

CAN arbitration procedure relies on the fact that a sending node monitors the bus while transmitting. The signal must be able to propagate to the most remote node and return back before the bit value is decided. This requires the bit time to be at least twice as long as the propagation delay which limits the data rate: for instance, 1Mbit/s is feasible on a 40 meter bus at maximum while 250Kbit/s can be achieved over 250 meters. To alleviate the data rate limit, and extend the lifespan of CAN further, car manufacturers are starting to optimize the bandwidth usage by implementing “traffic shaping” strategies that are very beneficial in terms of response times (see, for instance, [24]).

CAN has several mechanisms for error detection. For instance, it is checked that the CRC transmitted in the frame is identical to the CRC computed at the receiver end, that the structure of the frame is valid and that no bit-stuffing error occurred. Each station which detects an error sends an "error flag" which is a particular type of frame composed of 6 consecutive dominant bits that allows all the stations on the bus to be aware of the transmission error. The corrupted frame automatically re-enters into the next arbitration

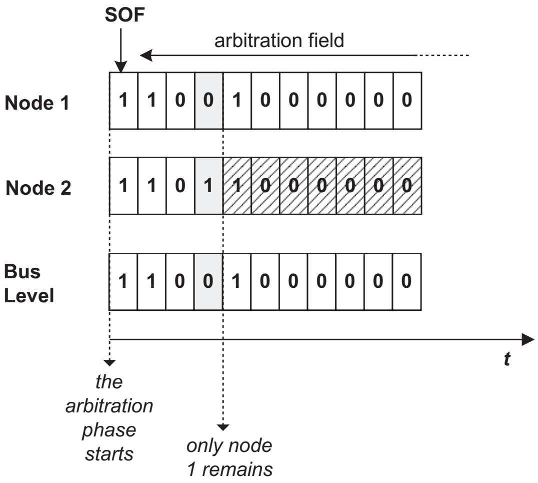

Figure 3: CAN arbitration phase with two nodes starting transmitting simultaneously. Node 2 detects that a frame with a higher priority than its own is being transmitted when it monitors a level 0 (i.e. dominant level) on the bus while it has sent a bit with a level 1 (i.e. recessive level). Afterwards, Node 2 immediately stops transmitting.

phase, which might lead it to miss its deadline due to the additional delay. The error recovery time, defined as the time from detecting an error until the possible start of a new frame, is 17 to 31 bit times. CAN possesses some fault-confinement mechanisms aimed at identifying permanent failures due to hardware dysfunctioning at the level of the micro-controller, communication controller or physical layer. 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, etc.). The relevance of the algorithms involved is questionable (see [22]) but the main limitation is that a node has to diagnose itself, which can lead to the non-detection of some critical errors. For instance, a faulty oscillator can cause a node to transmit continuously a dominant bit, which is one manifestation of the "babbling idiot" fault, see [74]. 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. Without additional fault-tolerance facilities, CAN is not well suited for safety-critical applications. For instance, a single node can perturb the functioning of the whole network by sending messages outside their specification (*i.e.* length and period of the frames). Many mechanisms were proposed for increasing the dependability of CAN-based networks (see [74] for an excellent survey), but if each proposal solves a particular problem, they have not necessarily been conceived to be combined. Furthermore, the fault-hypotheses used in the design of theses mechanisms are not necessarily the same and the interactions between them remain to be studied in a formal way.

The CAN standard only defines the physical layer and Data Link layer (DLL). Several higher level protocols have been proposed, for instance, for standardizing startup procedures, implementing data segmentation or sending periodic messages (see OSEK/VDX and AUTOSAR in §3). Other higher-level protocols standardize the content of messages in order to ease the interoperability between ECUs. This is the case for J1939 which is widely used, for instance, in Scania's trucks and buses [99].

2.1.2 CAN's use in today's automobiles

There is now more than 20 years of experience in automotive CAN applications, and CAN has certainly proven very successful as a robust, cost effective and all-around network technology. But the use of CAN in vehicles is evolving, in particular because of more complex and heterogeneous architectures with FlexRay or Ethernet networks, and because of recent needs like hybrid, electric propulsion or driver assistance that involves more stringent real-time constraints. Besides, there are other new requirements on CAN: more fine-grained ECU mode management for energy savings [8, 102], multi-ECU splitted functions and huge software downloads. In parallel, safety issues request more and more mechanisms to protect against potential failures and provide end-to-end integrity (see §3.3 and [2]).

Increased bandwidth requirements. The robustness and performance of the CAN technology, as well as the new possibilities brought by distributed software functions, have lead engineers to use more and more bandwidth in order to improve existing Electrical and Electronic (EE) functions and introduce new ones. This trend has never ceased, and along with topology and functional domain constraints, has led to the use of several CAN clusters within a car, sometimes more than 4 or 5 [64, 39]. Also, the data rates of the CAN buses are now higher (e.g., 250kbit/s for a body network when it used to be 125kbits/) and the bus load level has increased (e.g., greater than 50%).

More complex architectures. At the beginning of CAN, just a few ECUs were connected while today there are thousands of signals exchanged by several tens of ECUs, with some signals having timing constraints below 5ms. Besides, the architectures are becoming complex because of gateways between the CAN buses or between a CAN bus and another networking technology (typically FlexRay). The use of several CAN clusters raises also technical issues regarding for instance fault-handling, diagnosis timing response, wake-up and sleep synchronization. And, whatever is done, there is an overlap between the data sent on the buses connected to a gateway, which induces a significant waste a bandwidth. To face the EE architecture complexity, and be able to push the limits of CAN, car makers have established rigorous development processes. Besides, there are now several well-suited COTS toolsets available on the market to help them with the optimization and verification using simulation (possibly with fault-injection), schedulability analysis and trace analysis (see [64] for the use of RealTime-at-Work's tools).

Optimizing CAN networks. When CAN was introduced, the bus load levels were limited, typically much less than 30% (see [65] for a typical set of messages of the years 1995-2000). Optimizing CAN networks, which includes reaching higher load levels, has now become an industrial requirement for several reasons:

- It helps to master the complexity of the architectures,
- It reduces the hardware costs, weight, space, consumption, etc,
- It facilitates an incremental design process,
- It may avoid the industrial risk, the costs and the time to master new technologies such as FlexRay,
- It leads to better communication performances and helps to match the bandwidth needs. Sometimes, a 60%-loaded CAN network can be more efficient that two 40% CAN networks interconnected by a gateway causing delays and high jitters.

The first obvious way to optimize a CAN network is to keep the amount of data transmitted to a minimum, specifically limit the transmission frequency of the frames. This requires a rigorous identification and traceability of the temporal constraints. Given a set of signals or frames, and their associated temporal constraints (freshness, jitters, etc), they are in addition a few configuration strategies than can be used:

1. Desynchronize the stream of frames by using offsets. The reader may refer to [26] for comprehensive experiments on the large gains achieved using offsets,
2. Reassign the priorities of the frames, so that the priority order better reflects the timing constraints,
3. Re-consider the frame-packing, that is the allocation of the signals to the frames and choice of the frame periods, so as to minimize the bandwidth usage while meeting timing constraints (see [83]),

4. Optimize the ECU communication stacks so as to remove all implementation choices that cause a departure from the ideal CAN behavior (see [64]).

However, because there is less margin for error, using complex CAN-based architectures at high load levels involves more detailed supplier specifications on the one hand, and, on the other hand, to spend more time and effort in the integration/validation phase.

2.1.3 Other priority buses: VAN and J1850

The J1850 [89] is an SAE class B priority bus that was adopted in the USA for communications with non-stringent real-time requirements, such as the control of body electronics or diagnostics. Two variants of the J1850 are defined: a 10.4Kbit/s single-wire version and 41.6Kbit/s two-wire version. For quite a long time, the trend in new designs seems to be the replacement of J1850 by CAN or a low-cost network such as LIN (see §2.3.1).

In the 90s, another competing technology was the French Vehicle Area Network (VAN, see [34]) which is very similar to CAN (e.g., frame format, data rate) but possesses some additional or different features that are advantageous from a technical point of view (e.g., no need for bit-stuffing, in-frame response: a node being asked for data answers in the same frame that contained the request). VAN was used for years in production cars by the French carmaker PSA Peugeot-Citroën in the body domain (e.g, for the 206 model) but, as it was not adopted by the market, it was abandoned in favor of CAN.

2.1.4 CAN FD: a high speed CAN network

CAN FD is a new protocol that was presented for the first time by Robert Bosch GmbH at the International CAN Conference in 2012 [30] that combines CAN's core features with a higher data rate and larger data payloads. The data field length can be up to 64 bytes long: from 0 to 8 bytes, then above 8 bytes, the possible values are 12, 16, 20, 24, 32, 48 and 64 bytes. Technically, two bit rates are used successively in the transmission of a CAN FD frame: one lower for the arbitration phase, as required by the CAN bitwise arbitration (see §2.2.2), and a higher one used immediately after the arbitration for the transfer of the data payload and related fields (e.g., DLC, CRC). The bit rate for this data part of the frame transmission can be freely chosen (e.g., 10MBit/s are mentioned in [30]) but in practice it will depend much on the topology of the network and the efficiency of the transceivers. It is not clear at the time of writing what is the kind of data rate, and thus the speedup with regard to standard CAN, that can be achieved in typical automotive applications and this will largely determine the acceptance of

CAN FD. The target mentioned in [30] is an average data rate of 2.5MBit/s with existing CAN transceivers.

Four main use-cases are identified for CAN FD in [49]: faster software download (end-of-production line or maintenance), avoiding segmentation of long messages (and possibly securing normal CAN messages with message authentication information), offering higher-bandwidth to car domains requiring it (e.g., powertrain [49]), and enabling faster communication on long CAN buses (e.g., in trucks and buses). An important advantage of CAN FD is that there is an easy migration path from CAN systems to CAN FD systems since existing CAN application software can basically remains unchanged, the changes taking place in the communication layers and their configuration. Following FPGA's implementations, microcontrollers with CAN FD are already available (e.g., [20]) and the protocol has been submitted as ISO 11898-7 for international standardization.

2.2 Time-Triggered networks

Among communication networks, as discussed before, one distinguishes time-triggered networks where activities are driven by the progress of time and event-triggered once where activities are driven by the occurrence of events. Both types of communication have advantages but one considers that, in general, dependability is much easier to ensure using a time-triggered bus (refer, for instance, to [82] for a discussion on this topic). This explains that mainly time-triggered communication systems are being considered for use in the most critical applications such as X-by-Wire systems. In this category, multi-access protocols based on TDMA (Time Division Multiple Access) are particularly well suited; they provide deterministic access to the medium (the order of the transmissions is defined statically at the design time), and thus bounded response times. Moreover, their regular message transmissions can be used as "heartbeats" for detecting station failures. The three TDMA based networks that could serve as gateways or for supporting safety critical applications are TTP/C (see [95]), FlexRay (see §2.2.1) and TTEthernet (see §2.5.4). FlexRay, which was backed by the world's automotive industry, is becoming a standard technology in the industry, though without gaining broad acceptance, and is already in use in production cars since 2006 (see [85, 28]). TTCAN (see §2.2.2) was, before the advent of FlexRay, considered for use and is briefly discussed in this section. In the following, we choose not to discuss further TTP/C which, to the best of our knowledge, is no more considered for vehicles but is used in aircrafts. However, the important experience and know-how gained over the years with TTP/C, in particular regarding its fault-tolerance features (see [23]) and their formal validation (see [73]), is certainly beneficial to FlexRay and any Ethernet-based protocols such as TTEthernet (see §2.5.4) or even AVB (see §2.5.3).

2.2.1 The FlexRay Protocol

A consortium of major companies from the automotive field developed the FlexRay protocol. The core members of the consortium, now disbanded, were BMW, Bosch, Daimler, General Motors, NXP Semiconductors, Freescale Semiconductor and Volkswagen. The first publicly available specification of the FlexRay Protocol was released in 2004, the current version of the specification is now available as a set of ISO standards issued in 2013 (ISO 17458 part 1 to 5), based on the 3.0.1 release from the FlexRay consortium [11]. FlexRay did not gain very wide acceptance yet, since most car manufacturers still rely on CAN-based architectures, but it has been used since 2006 in some production cars by BMW and Audi. For instance, the latest series 5 from BMW implements up to 17 FlexRay nodes [28].

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. It is not mandatory that each station possesses replicated channels nor a bus guardian. 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, size of which is set statically at design time, two distinct protocols are 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 in FlexRay might possess several slots in the time-triggered window, but the size of all the slots is identical (see Figure 4). 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[5] 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. A mini-slot remains idle if the station has nothing to transmit which actually induces a loss of bandwidth (see [10] for a discussion on that topic). An example of a dynamic window is shown in Figure 5: on channel B, frames have been transmitted in mini-slots n and $n+2$ while mini-slot $n+1$ has not been used. It is 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 2047 overall) and since the frames are only transmitted if necessary in the dynamic part of the communication cycle. In a similar way as with TTP/C, the structure of the communication cycle is statically stored in the nodes, however, unlike TTP/C, mode changes with a different communication schedule for

[5] Different nodes can send frames in the same slot but in different cycles, this is called “slot multiplexing” and it is only possible only in the dynamic segment.

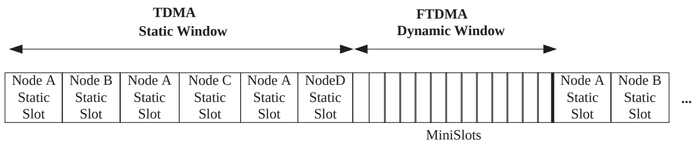

Figure 4: Example of a FlexRay communication cycle with 4 nodes A, B, C and D

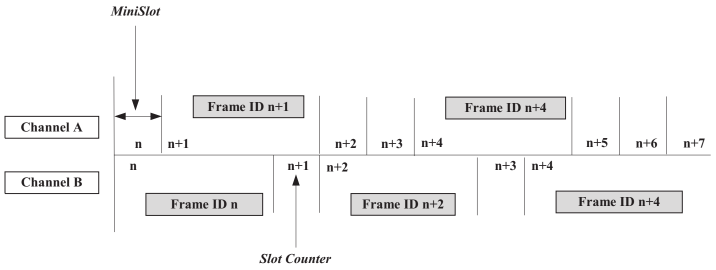

Figure 5: Example of message scheduling in the dynamic segment of the FlexRay communication cycle

each mode are not possible.

The FlexRay frame consists of 3 parts : the header, the payload segment containing up to 254 bytes of data and the CRC of 24 bits. The header of 5 bytes includes the identifier of the frame and the length of the data payload. The use of identifiers allows to move a software component, which sends a frame X, from one ECU to another ECU without changing anything in the nodes that consume frame X. It has to be noted that this is no more possible when signals produced by distinct components are packed into the same frame for the purpose of saving bandwidth (i.e., which is refer to as frame-packing or PDU-multiplexing - see [83] for this problem addressed on CAN).

From the dependability point of view, the FlexRay standard solely specifies a bus guardian (optional), passive and active star coupler (optional) and the clock synchronization algorithms. Other features, such as mode management facilities or a membership service, will have to be implemented in software or hardware layers on top of FlexRay (see, for instance, [5] for a membership service protocol that could be used along with FlexRay). This will allow to conceive and implement exactly the services that are needed with the drawback that correct and efficient implementations might be more difficult to achieve in a layer above the communication controller.

In the FlexRay specification, 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 links with single and dual transmission supports on the same network, subnetworks of nodes without bus-guardians or with different fault-tolerance capability with regards to clock synchronization, etc. In the automotive context where critical and non-critical functions co-exist and interoperate, this flexibility can prove to be efficient in terms of cost and re-use of existing components. The reader interested in more information about FlexRay can refer to [86, 11], and to [77, 25, 103] for how to configure the communication cycle.

2.2.2 The TTCAN protocol

TTCAN (Time Triggered Controller Area Network - see [37]) is a communication protocol developed by Robert Bosch GmbH on top of the CAN physical and data-link layers. TTCAN uses the CAN standard but, in addition, requires that the controllers have the possibility to disable automatic retransmission of frames upon transmission errors and to provide the upper layers with the point in time at which the first bit of a frame was sent or received. 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 the CAN protocol. Channel redundancy is possible (see [57] for a proposal), but not standardized and no bus guardian is implemented in the node. The key idea was to propose, as with FlexRay, a flexible time-triggered/event-triggered protocol. As illustrated in Figure 6, 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 by their organization in exclusive and arbitrating windows and by the messages sent inside exclusive windows, can be defined. The list of successive basic cycles is called the system matrix, which is executed in loops. Interestingly, the protocol enables the master node (*i.e.* 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 mode. Later, the master node can switch back to TTCAN mode by sending a reference message.

TTCAN is built on a well-mastered and low-cost technology, CAN, but, as defined by the standard, does not provide important dependability services such as the bus guardian, membership service and reliable acknowledgment (see [74]). It is, of course, possible to implement some of these mechanisms at the application or middleware level but with reduced efficiency. About 10 years ago, it was thought that carmakers could be interested in

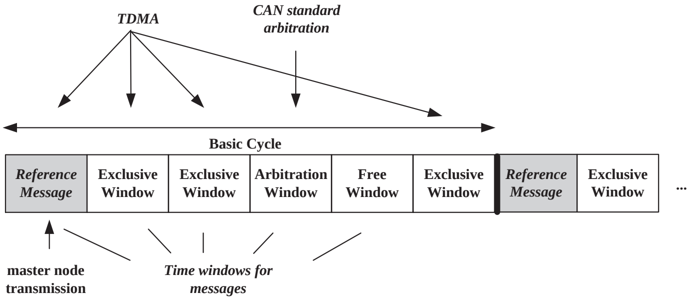

Figure 6: Example of a TTCAN Basic Cycle

using TTCAN during a transition period until FlexRay technology is fully mature but this was not really the case and TTCAN has not been used in production cars to the best of our knowledge.

2.3 Low-cost automotive networks

Several fieldbus networks have been developed to fulfill the need for low-speed / low-cost communication inside mechatronic based sub-systems generally made of an ECU and its set of sensors and actuators. Two representatives of such networks are LIN and TTP/A. The low-cost objective is achieved not only because of the simplicity of the communication controllers but also because the requirements set on the micro-controllers driving the communication are reduced (i.e., low computational power, small amount of memory, low-cost oscillator). Typical applications involving these networks include controlling doors (e.g., door locks, opening/closing windows) or controlling seats (e.g., seat position motors, occupancy control). Besides cost considerations, a hierarchical communication architecture, including a backbone such as CAN and several sub-networks such as LIN, enables reducing the total traffic load on the backbone.

Both LIN and TTP/A are master/slave networks where a single master node, the only node that has to possess a precise and stable time base, coordinates the communication on the bus: a slave is only allowed to send a message when it is polled. More precisely, the dialogue begins with the transmission by the master of a “command frame” that contains the identifier of the message whose transmission is requested. The command frame is then followed by a “data frame” that contains the requested message sent by one of the slaves or by the master itself (i.e., the message can be produced by the master).

This paragraph also presents the SENT and PSI5 networks which are other more recent low-cost alternatives to CAN.

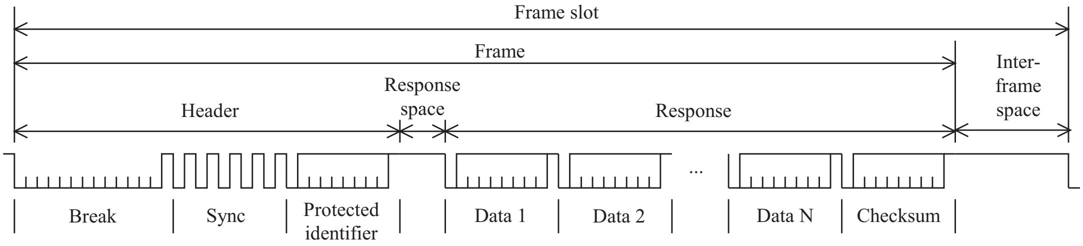

Figure 7: Format of the LIN frame. A frame is transmitted during its "frame slot" which corresponds to an entry of the schedule table

2.3.1 The LIN network

LIN (Local Interconnect Network, see [48, 79]) is a low cost serial communication system used as SAE class A network, where the needs in terms of communication do not require the implementation of higher-bandwidth multiplexing networks such as CAN. LIN is developed by a set of major companies from the automotive industry (e.g., Daimler, Volkswagen, BMW and Volvo) and is widely used in production cars.

The LIN specification package (LIN version 2.2A [48]) includes not only the specification of the transmission protocol (physical and data link layers) for master-slave communications but also the specification of a diagnostic protocol on top of the data link layer. A language for describing the capability of a node (e.g., bit-rates that can be used, characteristics of the frames published and subscribed by the node, etc.) and for describing the whole network is provided (e.g., nodes on the network, table of the transmissions' schedule, etc.). These description language facilitates the automatic generation of the network configuration by software tools.

A LIN cluster consists of one "master" node and several "slave" nodes connected to a common bus. For achieving a low-cost implementation, the physical layer is defined as a single wire with a data rate limited to 20Kbit/s due to EMI limitations. The master node decides when and which frame shall be transmitted according to the schedule table. The schedule table is a key element in LIN; it contains the list of frames that are to be sent and their associated frame-slots thus ensuring determinism in the transmission order. At the moment a frame is scheduled for transmission, the master sends a header (a kind of transmission request or command frame) inviting a slave node to send its data in response. Any node interested can read a data frame transmitted on the bus. As in CAN, each message has to be identified: 64 distinct message identifiers are available. Figure 7 depicts the LIN frame format and the time period, termed a "frame slot", during which a frame is transmitted.

The header of the frame that contains an identifier is broadcast by the master node and the slave node that possesses this identifier inserts the data in the response field. The “break” symbol is used to signal the beginning of a frame. It contains at least 13 dominant bits (logical value 0) followed by one recessive bit (logical value 1) as a break delimiter. The rest of the frame is made of byte fields delimited by one start bit (value 0) and one stop bit (value 1), thus resulting in a 10-bit stream per byte. The “sync” byte has a fixed value (which corresponds to a bit stream of alternatively 0 and 1), it allows slave nodes to detect the beginning of a new frame and be synchronized at the start of the identifier field. The so-called “protected identifier” is composed of two sub-fields: the first 6 bits are used to encode the identifier and the last two bits, the identifier parity. The data field can contain up to 8 bytes of data. A checksum is calculated over the protected identifier and the data field. Parity bits and checksum enable the receiver of a frame to detect bits that have been inverted during transmission.

LIN defines five different frame types: unconditional, event-triggered, sporadic, diagnostic and user-defined. Frames of the latter type are assigned a specific identifier value and are intended to be used in an application-specific way that is not described in the specification. The first three types of frames are used to convey signals. Unconditional frames are the usual type of frames used in the master-slave dialog and are always sent in their frame-slots. Sporadic frames are frames sent by the master, only if at least one signal composing the frame has been updated. Usually, multiple sporadic frames are assigned to the same frame-slot and the higher priority frame that has an updated signal is transmitted. An event-triggered frame is used by the master willing to obtain a list of several signals from different nodes. A slave will only answer the master if the signals it produces have been updated, thus resulting in bandwidth savings if updates do not take place very often. If more than one slave answers, a collision will occur. The master resolves the collision by requesting all signals in the list one by one. A typical example of the use of the event-triggered transfer given in [48] is the doors’ knob monitoring in a central locking system. As it is rare that multiple passengers simultaneously press a knob, instead of polling each of the four doors, a single event-triggered frame can be used. Of course, in the rare event when more than one slave responds, a collision will occur. The master will then resolve the collision by sending one by one the individual identifiers of the list during the successive frame slots reserved for polling the list. Finally, diagnostic frames have a fixed size of 8 bytes, fixed value identifiers for both the master’s request and the slave answers and always contain diagnostic or configuration data whose interpretation is defined in the specification.

It is also worth noting that LIN offers services to send nodes into a sleep mode (through a special diagnostic frame termed “go-to-sleep-command”) and to wake them up, which is convenient since optimizing energy consumption, especially when the engine is not running, is a real matter of concern in the automotive context.

2.3.2 The TTP/A network

As TTP/C, TTP/A [27] was initially invented at the Vienna University of Technology. TTP/A pursues the same aims and shares the main design principles as LIN and it offers, at the communication controller level, some similar functionalities, in particular, in the areas of plug-and-play capabilities and on-line diagnostics services. TTP/A implements the classic master-slave dialogue, termed "master-slave round", where the slave answers the master's request with a data frame having a fixed length data payload of 4 bytes. The "Multi-partner" rounds enable several slaves to send up to an overall amount of 62 bytes of data after a single command frame. A "broadcast round" is a special master-slave round in which the slaves do not send data; it is, for instance, used to implement sleep / wake-up services. The data rate on a single wire transmission support is, as for LIN, equal to 20Kbit/s, but other transmission supports enabling higher data rates are possible. To our best knowledge, TTP/A is not currently in use in production cars.

2.3.3 The PSI5 and SENT networks

PSI5 (Peripheral Sensor Interface, see `http://www.psi5.org`) is another low-cost bus that was originally developed for communication between ECUs and airbags, but is now considered for use with any kinds of sensors. It provides 125kbit/s communication and needs only two wires for both data communication and sensor power supply. When used in bus mode with several sensors on the bus, as opposed to the point-to-point mode, the communication is organized in a time-triggered manner according to a TDMA-based MAC protocol, each sensor sending in a predefined time slot.

Another competing technology is SENT (Single-Edge Nibble Transmission), normalized as SAE J2716, which provides one-way communication, but it can be complemented by the SPC protocol (Short PWM code, see [6]) which adds functionalities such as bi-directional communication.

2.4 Multimedia and infotainment networks

Several protocols have been adapted or specifically conceived for transmitting the large amount of data needed by multimedia and infotainment applications increasingly available in cars. Two prominent protocols in this category are MOST and 1394 Automotive (formerly known as IDB-1394) that are discussed below. Another technology, not specifically developed for the automotive domain, though that is currently broadly used for infotainment and automotive cameras, is LVDS (Low-Voltage Differential Signaling) which enables communication at 655Mbit/s and above [54] over twisted pair copper cables. In the near future, it is very likely that Ethernet-based networks, probably compliant with the IEEE 802.1 AVB QoS standard (see §2.5.3), will be used to transport the high data volumes needed by multimedia and infotainment applications [28, 91].

2.4.1 The MOST network

MOST (Media Oriented System Transport, see [59]) is a multimedia network, development of which was initiated in 1998 by the MOST Cooperation (a consortium of carmakers and component suppliers). MOST provides point-to-point audio and video data transfer with different possible data rates. MOST supports end-user applications like radios, GPS navigation, video displays and entertainment systems. MOST's physical layer has been so far a Polymer Optical Fiber (POF) transmission support which provides a better resilience to EMI and higher transmission rates than classical cooper wires, but a new coaxial standard has been introduced recently [41]. It was estimated in 2008 [53] that around 50 model series, for example from BMW and Daimler, implement a MOST network (e.g., MOST25 at 25Mbit/s at Daimler, see [46]). MOST has now become a de-facto standard for transporting audio and video streams within vehicles (see [60, 61]).

The third revision of MOST [59] has introduced the support of a channel that can transport standard Ethernet frames and is thus well suited to transmit IP traffic. These features, along with the higher bandwidth provided by MOST150 introduced in 2007, are already taken advantage of in some new vehicle projects [46] to provide access to internet services and web browsing, and independent access to audio and video sources from all seats with a single bus. In [41], the MOST consortium announces its intention to develop a next generation of the protocol, with a target bandwidth of 5Gbit/s, that would be suited as well for ADAS requiring the transmission of uncompressed video streams, competing thus with automotive Ethernet solutions.

2.4.2 The 1394 Automotive network

1394 Automotive is an automotive version of IEEE-1394 (FireWire) for in-vehicle multimedia and telematics applications that is developed by the 1394 Trade Association (see `http://www.1394ta.org`). The system architecture of 1394 Automotive permits existing IEEE-1394 consumer electronics devices to interoperate with embedded automotive grade devices. 1394 Automotive is advertised to support a data rate of 800Mbps [92, 84] over several physical layers including twisted pair or POF. Thanks to its large bandwidth and the interoperability with existing IEEE-1394 consumer electronic devices, 1394

Automotive was at some time considered a serious competitor for MOST technology but, despite a few early implementations at Renault and Nissan, as far as we know the protocol did not reach wide acceptance in the automotive market.

2.5 Automotive Ethernet

At the time of writing, the question is not if, but when Ethernet will become a standard technology in cars [101, 28, 96]. For instance, Continental expects to start series production of Ethernet-capable control units as soon as in 2015 [12]. The introduction of Ethernet will however be gradual, and Ethernet is probably going to fulfil different use-cases over time. This paragraph discusses these use-cases and the main Ethernet technologies that are considered for use.

2.5.1 Motivation and use-cases for Ethernet

The first motivation for Ethernet is that it is a low-cost and mature technology that offers much more bandwidth that what is available today, which is of interest for infotainment and active safety especially [29]. From the user perspective, as well as from an economic point of view, the re-use of non automotive-specific networking technologies such as Ethernet can be beneficial. However, automotive specific requirements must be taken into account: e.g., the need for partial networking and power over Ethernet to reduce the wiring, provision of specific QoS, robustness to severe environmental conditions (e.g., EMI, heat, etc).

In [29], the authors describe a plausible roadmap for the use of Ethernet. First generation Ethernet network, based on 100BASE-TX physical layer, will be for diagnostics (using ISO 13400 standard) and code upload (as already done since 2008, see [39]). The second generation Ethernet network, from 2015 onwards, will support infotainment and camera-based ADAS using Ethernet AVB (see §2.5.3) and BroadR-Reach physical layer (see §2.5.2). At the third stage, from 2020 onwards, gigabit Ethernet should become the backbone interconnecting most other networks, and replacing thus today's gateways (see section 4). This communication architecture will benefit from the ability of Ethernet switches to handle ports having different speeds. The backbone will have to support various kinds of traffic with different QoS requirements, and thus require QoS policies that may be offered, according to the authors of [29], by a second generation of AVB protocols. A new physical layer derived from today's 100Mbit/s BroadR-Reach will be needed, and Time-Triggered transmissions would be available.

2.5.2 BroadR-Reach physical layer

BroadR-Reach is a 100MBit/s physical layer over low-cost unshielded copper wires that is being developed by the OPEN Alliance SIG consortium (see `http://www.opensig.org/`). This consortium, that already counts 140 member companies with many of the carmakers along with Robert Bosch and Continental, aims at establishing BroadR-Reach as the standard for the automotive Ethernet physical layer, with Gb/s speed in its roadmap [32]. For that purpose, BroadR-Reach was designed to meet the automotive requirements in terms of cost, power saving modes, robustness to environmental conditions, etc. One clear advantage of BroadR-Reach over standard 100BASE-TX Fast Ethernet is that it only requires a single pair of copper cables. At the time of writing, the Broad-R-Reach specification is close to being final [101].

2.5.3 Ethernet AVB

IEEE 802.1 Audio/Video Bridging (AVB) is a set of standards aimed at providing low-latency streaming services on 802.3 Ethernet. The development and adoption of AVB is promoted by the AVnu Alliance consortium (see `http://www.avnu.org/`) which counts several car manufacturers and automotive suppliers among its members.

IEEE 802.1AS is a synchronization protocol that builds on IEEE 1588 Precision Time Protocol to enable sub-microsecond time synchronization between nodes. IEEE 802.1Qav specifies a leaky-bucket Credit Based traffic Shaper (CBS). 802.1Qat is an end-to-end bandwidth reservation protocol. Finally, 802.1BA specifies default parameters and profiles that manufacturers of AVB equipments can use. AVB defines three classes of traffic: Stream Reservation (SR) Class-A with a guaranteed latency of 2ms maximum, SR Class-B with a guaranteed 50ms maximum latency, and best effort for the rest of the traffic. At most, 75% of the bandwidth can be reserved for SR Class-A and SR Class-B streams.

Experiments in [91] suggest that AVB would be an excellent option for in-car multimedia streams, with the advantage that it enables to dynamically register data flows and can be used with higher level protocols for handling audio/video streams such as IEEE 1733. To the best of our knowledge, there is not clear cut answer yet about the suitability of the current AVB standard for the transmission of critical control data. In October 2012, AVnu Alliance has announced the formation of the AVB Gen2 Council with the aims to improve AVB functionality in areas such as time-sensitive transmission and fault tolerance. This new specification is currently in development within the IEEE (Audio/Video Bridging Task Group).

2.5.4 TTEthernet

TTEthernet [94] is a time-triggered switched Ethernet protocol, promoted by the TTA-Group (see `http://www.ttagroup.org/`), that was standardized as SAE AS6802 in 2011. TTEthernet natively supports mixed-criticality temporal requirements as it defines 3 types of traffic streams: time-triggered (with offline configurated transmission schedule), rate constrained (i.e., the maximum workload is bounded) and best effort. Upper bounds on the jitters and latencies can be derived for both the time-triggered and rate constrained traffic. The temporal QoS in terms of jitters is better for the time-triggered traffic but sending and receiving must be done at pre-defined points in time. Conceptually, the Rate-constrained traffic model is similar to Virtual links in AFDX, and SR-Class A and B in AVB.

TTEthernet is designed to be operational in cross-domain critical applications, up to highest criticality level such as DO178 DAL-A in aeronautics, and thus possesses a wide range of fault-tolerant mechanisms. Besides, the time-triggered traffic offers a temporal QoS that cannot be matched by the current version of AVB. These features could be needed for some future automotive applications. Another idea currently investigated [101, 91] is the integration of AVB and TTEthernet, which the aim to come up with an efficient solution for all common automotive Ethernet use-cases.

3 Automotive middleware

3.1 Objectives of an embedded middleware

The design of automotive electronic systems has to take into account several constraints. First, nowadays, the performance, quality and safety of a vehicle depend on functions that are mainly implemented in software (for example, the engine control, body electronics, ADAS, etc.) and moreover depend on a tight cooperation between these functions. For example, the control of the engine is done according to requests from the driver (speeding up, slowing down as transmitted by the throttle position sensor or the brake pedal) and requirements from other embedded functions such as climate control, ESP or brake assist. Second, in-vehicle embedded systems are produced through a complex cooperative multi-partner development process shared between OEMs and suppliers. Therefore, in order to increase the efficiency of the production of components and their integration, two important problems have to be solved: 1) the portability of components from one Electronic Control Unit to another one enabling some flexibility in the architecture design, and 2) the reuse of components between platforms which is a keypoint for car manufacturers. Thus the cooperative development process raises the problem of interoperability of components. A classic approach for easing the integration of software components is to implement a *middleware layer* that provides application programs with common services and a common interface. In particular, the common interface allows the design of an application disregarding the hardware platform and the distribution, and therefore enables the designer focusing on the development and the validation of the software components and the software architecture that realize a function.

Among the set of common services usually provided by a middleware, those that related to the communication between several application components are crucial. They have to meet several objectives:

- *Hide the distribution* through the availability of services and interfaces that are the same for intra-ECU, inter-ECU, inter-domain communications whatever the underlying protocols,
- *Hide the heterogeneity* of the platform by providing an interface independent of the underlying protocols, of the CPU architecture (e.g., 8/16/32 bits, endianness), of the Operating Systems, etc.
- *Provide high-level services* in order to shorten the development time and increase quality through the re-use of validated services (*e.g.,* periodic transmission, working mode management such as partial networking, etc.). A good example of such a function is the "frame-packing" (sometimes also called "signal multiplexing") that enables application components to exchange *signals* (e.g. the number of revolutions per minute, the speed of the vehicle, the state of a light, etc.) while, at runtime, *frames* are transmitted over the network; so, the "frame-packing" service of a middleware consists in packing the signals into frames and sending the frames at the right points in time for ensuring the deadline constraint on each signal it contains,
- *Ensure QoS properties required by the application*, in particular, it can be necessary to improve the QoS provided by the lower-level protocols as, for example, by furnishing an additional CRC (Cyclic Redundancy Code), transparent to the application, if the Hamming distance of the CRC specified by the network protocol is not sufficient in regard to the dependability objectives (cf. the end-to-end communication protection library of AUTOSAR in §3.3.3). Other examples are the correction of "bugs" in lower level protocols such as the "inconsistent message duplicate" of CAN (see [75]), the provision of a reliable acknowledgment service or message authentication mechanism on CAN, the status information on the data consumed by the application components (e.g., data was refreshed since last reading, its freshness constraint was not respected, etc.) or filtering mechanisms (e.g., notify the application for each k reception or when the data value has changed in a significant way).

Note that a more advanced features would be to come up with adaptive communication services, thanks to algorithms that would modify at run-time the parameters of the communication protocols (e.g., priorities, transmission frequencies, etc.) according to the current requirements of the application (e.g., inner-city driving or highway driving) or changing environmental conditions (e.g., EMI level). For the time being, to the best of our knowledge, such features exist in series automotive embedded systems only for low-power modes (see §1.5, see also [104] for a research work in the direction of more adaptive communication systems). In fact, this point requires a coordinated approach for the design of function (as the definition of control law parameters, the identification of the parameters acting on the robustness of the function, etc.) and the deployment of the software architecture that implements the function (specifically the communication parameters). By increasing the efficiency and the robustness of the application, such an adaptive strategy would certainly ease the re-usability.

3.2 Former propositions of middleware

Some proprietary middleware (MW) were developed by several carmakers in order to support the integration of ECUs and software modules provided by their third-party suppliers. For instance, the TITUS/DBKOM communication stack, was a proprietary middleware (MW) of Daimler that standardizes the cooperation between components according to a client/server model. *Volcano* [9, 80] is a commercial product of Mentor Graphics, initially developed in tight cooperation with Volvo. The Volcano Target Package (VTP) consists of a communication layer and a set of off-line configuration tools for application distributed on CAN and LIN. It is aimed at providing the mapping of signals into frames under network bandwidth optimization and ensure a predictable and deterministic real time communication system thanks to schedulability analysis techniques (see [93, 9]). To the best of our knowledge, there are no publicly available technically precise descriptions of both TITUS and Volcano.

A first step to define a standard for in-car embedded middleware was started by the OSEK/VDX consortium (`http://www.osek-vdx.org`). In particular, two specifications are of particular interest in the context of this chapter: the *OSEK/VDX Communication* layer [70] and the *Fault-Tolerant Communication* layer [68]. The first one specifies a communication layer [70] that defines common software interfaces and common behavior for internal and external communications between application components. How signals are packed into a frame is statically defined off-line and the *OSEK/VDX Communication* layer automatically realizes the packing / unpacking at run-time as well as the handling of queued or unqueued messages at the receiver side. *OSEK/VDX Communication* runs on top of a transport layer (e.g., [36]) that takes care mainly of possible segmentation of frames and it can operate on any OS compliant with *OSEK/VDX OS* services for tasks, events and interrupt management (see [71]).

OSEK/VDX Communication is not intended to be used on top of a Time-Triggered (TT) network, as for example TTP/C or FlexRay. For this purpose, there is *OSEK/VDX FTCom* specification (Fault-Tolerant Communication, see [68]) whose main role is to manage the redundancy of data needed to achieve fault-tolerance (i.e., the same information can be produced by a set of replicated nodes) by presenting only one copy of data to the receiving application according to the strategy specified by the designer. Two other important services of *OSEK/VDX FTCom*, already offered by *OSEK/VDX Communication*, are to manage the packing/unpacking of messages [83], and to provide message filtering mechanisms for passing only "significant" data to the application. *OSEK/VDX FTCom* was developed to run on top of a time-triggered operating system (OS) such as OSEK Time [69]. OSEK/VDX is no more under active development and has been superseded by AUTOSAR.

3.3 AUTOSAR - a standard for the automotive industry

From 2001 to 2004, the ITEA European project EAST-EEA aimed at the specification of an automotive MW for the automotive industry. To the best of our knowledge, this was the first important initiative targeting the specification of both the services to be ensured by the middleware and the architecture of the middleware itself in terms of components and architecture of components. Similar objectives are guiding the work done in the AUTOSAR consortium, see `http://www.autosar.org` and [17, 21, 40], that gathers most the key players in the automotive industry. The specifications produced by the consortium become quickly de-facto standards for the cooperative development of in-vehicle embedded systems (see, for instance, the migration to AUTOSAR at PSA Peugeot-Citröen [16]).

3.3.1 The reference model

AUTOSAR specifies the software architecture embedded in an ECU. More precisely, it provides a reference model which is comprised of three main parts:

- the application layer,
- the basic software (middleware software components),
- and the Run Time Environment (RTE) that provides standardized software interfaces to the application software.

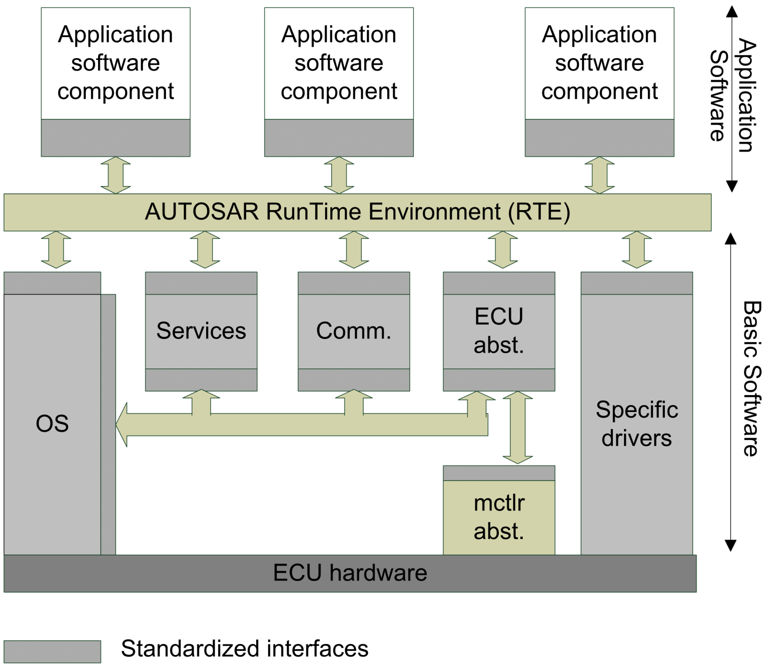
Application software component
Application software component
Application software component
Application Software
AUTOSAR RunTime Environment (RTE)
OS
Services
Comm.
ECU abst.
Specific drivers
mctlr abst.
Basic Software
ECU hardware
Standardized interfaces

Figure 8: AUTOSAR reference architecture

One of AUTOSAR's main objective is to improve the quality and the reliability of embedded systems. By using a well suited abstraction, the reference model supports the separation between software and hardware, it eases the mastering of the complexity, allows the portability of application software components and therefore the flexibility for product modification, upgrade and update, as well as the scalability of solutions within and across product lines. Besides, AUTOSAR ensures a smooth integration process of components provided by different companies while protecting the industrial properties of each actor involved. The AUTOSAR reference architecture is schematically illustrated in figure 8. An important issue is the automatic generation of an AUTOSAR middleware that has to be done from the basic software components, generally provided by suppliers, and the specification of the application itself (description of applicative-level tasks, signals sent or received, events, alarms, etc.). The challenge is to realize such a generation so that the deployment of the middleware layer can be optimized for each ECU.

3.3.2 The communication services

One of the main objectives of the AUTOSAR middleware is to hide the characteristic of the hardware platform as well as the distribution of the application software components. Thus the inter- or intra-ECU communication

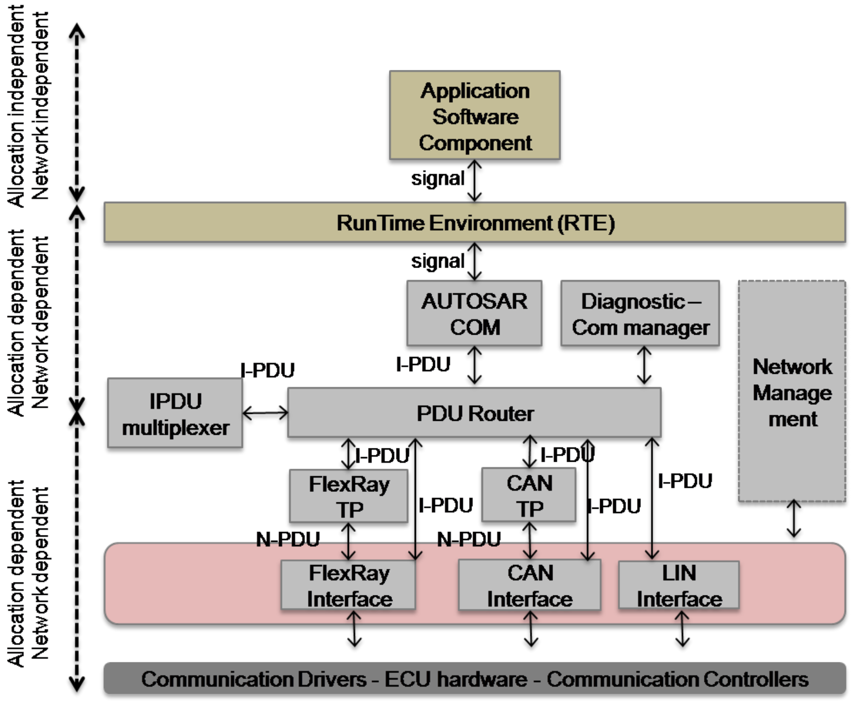

Figure 9: Communication software components and architecture

services are of major importance and are thoroughly described in the documents provided by the AUTOSAR consortium (see figure 9 for an overview of the different modules). The role of these services is crucial for the behavioral and temporal properties of an embedded and distributed application. So, their design, their generation and configuration have to be precisely mastered and the verification of timing properties becomes an important activity. The problem is complex because, as for the formerly mentioned middleware, the objects (e.g., signals, frames, I-PDU, etc.) that are handled by services at one level are not the same objects that are handled by services at another level. Nevertheless each object is strongly dependent of one or several objects handled by services belonging to neighboring levels. The AUTOSAR standard proposes two communication models:

- "sender-receiver" used for passing information between two application software components (belonging to the same task, to two distinct tasks on the same ECU or to two remote tasks),
- "client-server" that supports function invocation.

Two communication modes are supported for the "sender-receiver" communication model:

- the "explicit" mode is specified by a component that makes explicit calls to the AUTOSAR middleware for sending or receiving data,
- the "implicit" mode means that the reading (resp. writing) of data is automatically done by the middleware before the invocation (resp. after the end of execution) of a component consuming (resp. producing) the data without any explicit call to AUTOSAR services; this is away to protect effectively the data between application software components and middleware services.

AUTOSAR identifies three main objects regarding the communication: signal exchanged between software components at application level, I-PDU (Interaction Layer Protocol Data Unit) that consists of a group of one or several signals, and the N-PDU (Data Link Layer Protocol Data Unit) that will actually be transmitted on the network. Precisely AUTOSAR defines:

- *signals* at application level that are specified by a length and a type. Conceptually a signal is exchanged between application software components through ports disregarding the distribution of this component. The application needs to precise a *Transfer Property* parameter that will impact the behavior of the transmission and whose value can be "*triggered*" (each time the signal is provided to the middleware by the application, it has to be transmitted on the network) or "*pending*" (the actual transmission of a signal on the network depends only on the emission rule of the frame that contains the signal). Furthermore, when specifying a signal, the designer has to indicate if it is a *data*, an *event* or a *mode*. For *data* transmission, incoming data are not queued on the receiver side while for *event* exchanges, signals are queued on the receiver side and therefore, for each transmission of the signal, a new value will be made available to the application. The handling of buffers or queues is done by the RTE.

- I-PDU are built by the AUTOSAR COM component. Each I-PDU is made of one or several signals and is passed via the PDU Router to the communication interfaces. The maximum length of an I-PDU depends on the maximum length of the L-PDU (i.e., Data Link Layer PDU) of the underlying communication interface: for CAN and LIN the maximum L-PDU length is 8 bytes while for FlexRay the maximum L-PDU length is 254 bytes. AUTOSAR COM ensures a local transmission when both components are located on the same ECU, or by building suited objects and triggering the appropriate services of the lower layers when the components are remote. This scheme enables the portability of components and hide their distribution. The transformation from signals to I-PDU and from I-PDU to signals is done according to an off-line generated configuration. Each I-PDU is characterized by a behavioral parameter, termed *Transmission Mode* whose possible value is "direct" indicates (the sending of the I-PDU is done as soon as a "triggered" signal contained in this I-PDU is sent at application layer), "periodic" means (the sending of the I-PDU is done only periodically), "mixed" (the rules imposed by the "triggered" signals contained in the I-PDU are taken into account, and additionally the I-PDU is sent periodically if it contains at least one "pending" signal) or "none" (for I-PDUs whose emission rules depend on the underlying network protocol, as e.g., FlexRay; no transmission is initiated by AUTOSAR COM in this mode).

- an N-PDU is built by the basic components CAN TP (Transport Protocol) or FlexRay TP. It consists of the data payload of the frame that will be transmitted on the network and protocol control information. Note that the use of a transport layer is not mandatory and I-PDUs can be transmitted directly to the lower layers (see figure 9).

The RTE (Run Time Environment) implements the AUTOSAR middleware interface and the corresponding services. In particular, when using the *sender/receiver* model, the RTE handles the *implicit/explicit* communication modes and the fact that the communication involves *events* (queued) or *data* (unqueued). The *AUTOSAR COM* component is responsible for several functions: on the sender side, it ensures the transmission and notifies the application about its outcome (success or error). In particular, AUTOSAR COM can inform the application if the transmission of an I-PDU did not take place before a specified deadline (i.e., deadline monitoring). On the receiver side, it also notifies the application (success or error of a reception) and supports the filtering mechanism for signals (dispatching each signal of a received I-PDU to the application or to a gateway). Both at the sending and receiving end, the endianness conversion is taken in charge. An important role of the COM component is to pack/unpack *signals* into/from *I-PDU*s. Note that, as the maximal length of an I-PDU depends on the underlying networks, the design of a COM component has to take into account the networks and therefore it is not fully independent of the hardware architecture. The COM component has also to determine the points in time where to send the I-PDUs. This is based on the attributes *Transmission Mode* of an I-PDU and on the attribute *Transfer Property* of each signal that it contains.

The COM component is generated off-line on the basis of the knowledge of the signals, the I-PDUs and the allocation of application software components on the ECUs. The *AUTOSAR PDU Router* (see figure 9), according to the configuration, dispatches each I-PDU to the right network communication stack. This basic component is statically generated off-line as soon as the allocation of software components and the operational architecture is known. Other basic software components of the communication stack are responsible for the segmenting/reassembling of I-PDU(s) when needed (FlexRay TP, CAN TP) or for providing an interface to the communication drivers (FlexRay Interface, CAN Interface, LIN Interface).

3.3.3 The end-to-end communication protection library

In order to support safer communications between application software components, the AUTOSAR standard specifies the *End-to-End Communication Protection Library* [2]. The specified End-to-End mechanisms aim to ensure that safety-critical data exchanges (up to ASIL D safety level) can be protected against faults that may occur at runtime during the transmission of the data: faults affecting the hardware (e.g., bit-flipping due to EMI in the sending communication stack) or software faults due to the incorrect development of a component of the communication stack. In particular, the E2E protection mechanisms allow:

- to attach, on the sender side, specific control data to data that are to be sent to the RTE,
- to verify, on the receiver side, through the control data, the correction of the data received from the RTE,
- to report, when it occurs, that received data are faulty. In this case, the fault has to be handled by the receiver software components.

The E2E library is implemented as a set of functions that can be invoked at two levels: by application software components (thanks to a wrapping technique) or from the COM component. These functions are stateless and have to report synchronously a verdict (*i.e.*, the success of the exchange or the detected errors) to their calling component. The end-to-end communication protection is based on a standardization of mechanisms under the concept of *E2E profile*. Each profile can have several variants. So the instantiation of one profile for a specific information exchange consists in the choice of a variant and therefore the setting of the corresponding configuration options. Each profile offers all or a subset of the following data protection mechanisms:

- checking the integrity of an exchanged data. This is done by an additional CRC computed by a function belonging to the E2E library,
- checking the systematic transmission of a data thanks to a *Sequence Counter*. The *Sequence Counter* is incremented upon each transmission of the data, allowing thus the receiver to verify that no instance of the data has been lost,
- checking, from the receiver side, if a sender is transmitting a data as expected. For this purpose, an *Alive Counter* is used, it is incremented upon each transmission of the data and the receiver can verify that the counter has been incremented,
- to check if the received data is the one that is expected. This mechanism is based on the definition of a data ID that is not transmitted with the data but that is used in the computation of the CRC,
- finally, to verify that the data is sent and/or received on time. Two mechanisms are provided for this purpose: receiver communication timeout and sender acknowledgment timeout.

The CRC and the counters are integrated into a E2E header, which is an additional control field transmitted along with the data payload. When a faulty transmission is detected, the error is then reported to the calling component.

4 Conclusions and discussion

In our view, the development of in-vehicle embedded communications can be schematically subdivided into 4 main successive periods:

- communication through point-to-point links - until the beginning of the 90s,
- CAN-based architectures as de-facto standard - until 2006,

- the advent of AUTOSAR and FlexRay - until 2010,
- the emergence of automotive Ethernet and security concerns since then.

One may imagine that the next stage will be when the "car will become an IP node in the world wide web" ([28], see also [31]), integrated into global data infrastructures enabling a more efficient type of "collaborative mobility" [78] relying on interactions between users and infrastructure operators.

Over the years, the technologies needed for the interoperability between applications located on different ECUs and sub-networks have been improved. With AUTOSAR, we are now close to the desirable characteristics listed in section 3. However, if the traditional partitioning of automotive applications into several distinct functional domains with their own characteristics and requirements has proven useful in mastering the complexity, it lead to the development of several independent sub-systems with their specific architectures, networks and software technologies. Some difficulties arise from this partitioning since more and more cross-domain data exchanges are needed. The current practice is to transfer data between different domains through a central gateway (see, for example, [39]). This subsystem is recognized as being critical in the vehicle: it constitutes a single point of failure, its design is overly complex and ensuring a guaranteed QoS through gateways is difficult and require a careful design (see, for instance, [90] and [15]). In the future, the interconnection between vehicle subdomains could be more efficiently ensured by an Ethernet backbone [29], instead of a gateway. This backbone will have to support various traffic with different QoS requirements on the same network, which will require to implement QoS policies such as bandwidth reservation.

A perhaps more difficult challenge ahead of us is security: automotive systems must be designed to be more resilient to security attacks. This entails finding the technical solutions, but also the new cooperation schemes between all the automotive stakeholders, that will permit to achieve the right trade-off for the automotive domain, between costs/overhead and security level.

These issues will certainly require more research and development work but the current state of technological maturity in security, network hardware components, communication protocols and middleware engineering is such that in our view everything is at hand to succeed in building safe, secure and cost-optimized embedded communication architectures for the next car generations.

References

[1] A. Albert. Comparison of event-triggered and time-triggered concepts with regards to distributed control systems. In *Proceedings of Embed-*

ded World 2004, Nürnberg, February 2004.

[2] AUTOSAR. Specification of SW-C end-to-end communication protection library, v3.0.0. Available at `http://www.autosar.org`, 2013.

[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. Ayoubi, T. Demmeler, H. Leffler, and P. Köhn. X-by-Wire functionality, performance and infrastructure. In *Proceedings of Convergence 2004*, Detroit, Michigan, 2004.

[5] R. Barbosa and J. Karlsson. Formal specification and verification of a protocol for consistent diagnosis in real-time embedded systems. In *Third IEEE International Symposium on Industrial Embedded Systems (SIES'2008)*, June 2008.

[6] L. Beaurenaut. Short PWM Code: A step towards smarter automotive sensors. In *Advanced Microsystems for Automotive Applications 2009*, pages 383–395. Springer, 2009.

[7] S. Bunzel. Autosar architecture expands safety and security applications. EETimes, available at `http://www.eetimes.com/`, February 2011.

[8] C. Butzkamm and D. Bollati. Partial Networking for CAN bus systems: Any saved gram CO_2/km is essential to meet stricter EU regulations. In *13th International CAN Conference (iCC2012)*, Hambach, Germany, March 5-6 2012.

[9] L. Casparsson, A. Rajnak, K. Tindell, and P. Malmberg. Volcano - a revolution in on-board communications. Technical Report 98-12-10, Volvo, 1999.

[10] G. Cena and A. Valenzano. Performance analysis of Byteflight networks. In *Proceedings of the 2004 IEEE Workshop of Factory Communication Systems (WFCS 2004)*, pages 157–166, September 2004.

[11] FlexRay Consortium. FlexRay communications system - protocol specification - version 3.0.1, December 2010.

[12] Continental. Continental shows Ethernet test setup. Press Release, January 2012.

[13] Intel Corporation. Introduction to in-vehicle networking. Available at `http://support.intel.com/design/auto/autolxbk.htm`, 2008.

[14] R. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, April 2007.

[15] R. Davis and N. Navet. Traffic shaping to reduce jitter in Controller Area Network (CAN). *SIGBED Rev.*, 9(4):37–40, November 2012.

[16] P.H. Dezaux. Migration strategy of in-house automotive real-time applicative software in AUTOSAR standard. In *Proceedings of the 4th European Congress Embedded Real Time Software (ERTS 2008)*, Toulouse, France, 2008.

[17] H. Fennel, S. Bunzel, H.Heinecke, J. Bielefeld, S. Fürstand K.P. Schnelle, W. Grote, N. Maldenerand, T. Weber, F. Wohlgemuth, J. Ruh, L. Lundh, T. Sandén, P. Heitkämper, R. Rimkus, J. Leflour, A. Gilberg, U. Virnich, S. Voget, K. Nishikawa, K. Kajio, K. Lange, T. Scharnhorst, and B. Kunkel. Achievements and exploitation of the AUTOSAR development partnership. In *Convergence 2006*, Detroit, USA, October 2006.

[18] 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.

[19] Ford Motor Company. Ford to study in-vehicle electronic devices with advanced simulators. Available at url `http://media.ford.com/article_display.cfm?article_id=7010`, 2001.

[20] Freescale Semiconductor. New Freescale microcontrollers help streamline automotive body electronics networks and reduce vehicle weight. Press Release available on Reuters.com, March 18 2013.

[21] S. Fürst. AUTOSAR for Safety-Related Systems: Objectives, Approach and Status. In *2nd IEE Conference on Automotive Electronics*, London, UK, March 2006. IEE.

[22] B. Gaujal and N. Navet. Fault confinement mechanisms on CAN: Analysis and improvements. *IEEE Transactions on Vehicular Technology*, 54(3):1103–1113, May 2005.

[23] B. Gaujal and N. Navet. Maximizing the robustness of TDMA networks with applications to TTP/C. *Real-Time Systems*, 31(1-3):5–31, December 2005.

[24] M. Grenier, L. Havet, and N. Navet. *The Automotive Embedded Systems Handbook*, chapter Scheduling frames with offsets in automo-

tive systems: a major performance boost, pages 14.1–14.15. CRC Press/Taylor and Francis, December 2008.

[25] M. Grenier, L. Havet, and N. Navet. Configuring the communication on FlexRay: the case of the static segment. In *ERTS Embedded Real Time Software 2008*, 2008.

[26] M. Grenier, L. Havet, and N. Navet. Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost. In *Proc. of the 4th European Congress Embedded Real Time Software (ERTS 2008)*, Toulouse, France, January 29 - February 1 2008.

[27] H. Kopetz et al. *Specification of the TTP/A Protocol.* University of Technology Vienna, September 2002.

[28] C. Hammerschmidt. Beyond FlexRay: BMW airs Ethernet plan. EETimes, available at `http://www.eetimes.com/document.asp?doc_id=1256700`, May 2013.

[29] P. Hank, T. Suermann, and S. Müller. Automotive Ethernet, a holistic approach for a next generation in-vehicle networking standard. In *Advanced Microsystems for Automotive Applications 2012*, pages 79–89. Springer, 2012.

[30] F. Hartwich. CAN with Flexible Data-Rate. In *13th International CAN Conference (iCC2012)*, Hambach, Germany, March 5-6 2012.

[31] P. Hoschka. W3C launched work on Web and automotive. In *ERCIM News, ISSN 0926-4981, number 94, Special theme on Intelligent Cars*, page 5. ERCIM-EEIG, July 2013.

[32] C. Humig. Next generation automotive network architecture based on Ethernet. Slides presented at the 3rd International Conference Chassis Electrification, May 2012.

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

[34] International Standard Organization. *ISO 11519-3, Road Vehicles - Low Speed serial data communication - Part 3: Vehicle area network (VAN).* ISO, 1994.

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

[36] International Standard Organization. *15765-2, Road Vehicles - Diagnostics on CAN - Part 2: Network Layer Services.* ISO, 1999.

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

[38] K.H. Johansson, M. Törngren, and L. Nielsen. *Handbook of Networked and Embedded Control Systems*, chapter Vehicle Applications of Controller Area Network, pages 741–765. Birkhäuser Boston, 2005.

[39] H. Kellermann, G. Németh, J. Kostelezky, K. Barbehön, F. El-Dwaik, and L. Hochmuth. Electrical and electronic system architecture. *ATZextra worldwide*, 13(8):30–37, 2008.

[40] F. Kirschke-Biller, S. Fürst, S. Lupp, S. Bunzel, S. Schmerler, R. Rimkus, A. Gilberg, K. Nishikawa, and A. Titze. A worldwide standard current developments roll out and outlook. In *15th International VDI Congress Electronic Systems for Vehicles 2011*, 2011.

[41] R. Klos. MOST in future automotive connectivity. EETimes, available at `http://www.automotive-eetimes.com/`, May 2013.

[42] P. Koopman. Critical embedded automotive networks. *IEEE Micro, Special Issue on Critical Embedded Automotive Networks*, 22(4):14–18, July-August 2002.

[43] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy (SP2010)*, pages 447–462, 2010.

[44] M. Krug and A. V. Schedl. New demands for in-vehicle networks. In *Proceedings of the 23rd EUROMICRO Conference'97*, pages 601–605, Budapest, Hungary, July 1997.

[45] G. Leen and D. Heffernan. Expanding automotive electronic systems. *IEEE Computer*, 35(1), January 2002.

[46] A Leonhardi, S. Wachter, M. Bösinger, and T. Pech. MOST150's new features in a series project. EETimes, available at `http://www.automotive-eetimes.com/`, May 2011.

[47] T. Liebetrau, U. Kelling, T. Otter, and M. Hel. Energy saving in automotive E/E architectures. Infineon White Paper, December 2012.

[48] LIN Consortium. *LIN Specification Package, revision 2.2A*, December 2010. Available at `http://www.lin-subbus.org/`.

[49] T. Lindenkreuz. CAN FD - CAN with flexible data rate. Slides presented at Vector Congress 2012, May 2012.

[50] Z. Lokaj, T. Zelinka, and M. Srotyr. Cooperative systems for car safety improvement. In *ERCIM News, ISSN 0926-4981, number 94, Special theme on Intelligent Cars*, pages 9–10. ERCIM-EEIG, July 2013.

[51] J. Machan and C. Laugier. Intelligent vehicles as an integral part of intelligent transport systems. In *ERCIM News, ISSN 0926-4981, number 94, Special theme on Intelligent Cars*, pages 6–7. ERCIM-EEIG, July 2013.

[52] Y. Martin. *L'argus de l'automobile*, 3969:22–23, March 2005.

[53] E. Mayer. Serial bus systems in the automobile - part 5: MOST for transmission of multimedia data. Summary of Networking Competence, February 2008. Published by Vector Informatik GmbH.

[54] K. McCrory. Tech tutorial: LVDS offers efficient data transmission for automotive applications. EETimes, available at `http://www.eetimes.com/`, April 2006.

[55] Mercedes-Benz. The new Mercedes-Benz S-Class. Press Release, May 2013.

[56] P. Meumeu-Yomsi, D. Bertrand, N. Navet, and R. Davis. Controller Area Network (CAN): Response time analysis with offsets. In *9th IEEE International Workshop on Factory Communication System (WFCS 2012)*, Lemgo/Detmold, Germany, May 21-24 2012.

[57] B. Müller, T. Führer, F. Hartwich, R. Hugel, and H. Weiler. Fault tolerant TTCAN networks. In *Proceedings of the 8th International CAN Conference (iCC)*, Las Vegas, Nevada, 2002.

[58] A. Monot, N. Navet, B. Bavoux, and F. Simonot-Lion. Multisource software on multicore automotive ECUs - combining runnable sequencing with task scheduling. *IEEE Transactions on Industrial Electronics*, 59(10):3934–3942, 2012.

[59] MOST Cooperation. *MOST Specification Revision 3.0 E2*, July 2010. Available at `http://www.mostcooperation.com`.

[60] H. Muyshondt. Consumer and automotive electronics converge: Part 1 - Ethernet, USB, and MOST. Available at `http://www.automotivedesignline.com/`, February 2007.

[61] H. Muyshondt. Consumer and automotive electronics converge: Part 2 - a MOST implementation. Available at `http://www.automotivedesignline.com/`, March 2007.

[62] F. Simonot-Lion N. Navet. *The Automotive Embedded Systems Handbook*, chapter A Review of Embedded Automotive Protocols, pages 4.1–4.31. Industrial Information Technology series. CRC Press/Taylor and Francis, December 2008.

[63] N. Navet, B. Delord, and M. Baumeister. Virtualization in automotive embedded systems: an outlook. Slides presented at RTS Embedded Systems 2010 (RTS'2010), March 2010.

[64] N. Navet and H. Perrault. CAN in automotive applications: a look forward. In *13th International CAN Conference (iCC2012)*, Hambach, Germany, March 5-6 2012.

[65] N. Navet, Y. Song, and F. Simonot. Worst-case deadline failure probability in real-time applications distributed over CAN (Controller Area Network). *Journal of Systems Architecture*, 46(7):607–617, 2000.

[66] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert. Trends in automotive communication systems. *Proceedings of the IEEE*, 93(6):1204–1223, 2005.

[67] N. Navet and Y.-Q. Song. Validation of real-time in-vehicle applications. *Computers in Industry*, 46(2):107–122, November 2001.

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

[69] OSEK Consortium. *OSEK/VDX time triggered operating system, Version 1.0*, July 2001. Available at `http://www.osek-vdx.org/`.

[70] OSEK Consortium. *OSEK/VDX Communication, Version 3.0.3*, July 2004. Available at `http://www.osek-vdx.org/`.

[71] OSEK Consortium. *OSEK/VDX Operating System, Version 2.2.3*, February 2005. Available at `http://www.osek-vdx.org/`.

[72] M. Peteratzinger, F. Steiner, and R. Schuermans. Use of XCP on FlexRay at BMW. Translated reprint from HANSER Automotive 9/2006, available at url `http://www.vector.com`, 2006.

[73] H. Pfeifer. *The Automotive Embedded Systems Handbook*, chapter Formal Methods in the Automotive Domain: The Case of TTA, pages 15.1–15.27. CRC Press/Taylor and Francis, December 2008.

[74] J. Pimentel, J. Proenza, L. Almeida, G. Rodriguez-Navas, M. Barranco, and J. Ferreira. *The Automotive Embedded Systems Handbook*, chapter Dependable Automotive CAN Networks, pages 6.1–6.41. CRC Press/Taylor and Francis, 2008.

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

[76] S. Poledna, W. Ettlmayr, and M. Novak. Communication bus for automotive applications. In *Proceedings of the 27th European Solid-State Circuits Conference*, Villach, Austria, September 2001.

[77] T. Pop, P. Pop, P. Eles, and Z. Peng. Bus access optimisation for FlexRay-based distributed embedded systems. In *Proceedings of the conference on Design, Automation and Test in Europe (DATE '07)*, pages 51–56, San Jose, CA, USA, 2007.

[78] I. Radusch. Collaborative mobility - beyond communicating vehicles. In *ERCIM News, ISSN 0926-4981, number 94, Special theme on Intelligent Cars*, page 4. ERCIM-EEIG, July 2013.

[79] A. Rajnák. *The Industrial Communication Technology Handbook*, chapter The LIN Standard, pages 31.1–31.13. CRC Press, January 2005. R. Zurawski editor, ISBN 0-8493-3077-7.

[80] A. Rajnák and M. Ramnefors. The Volcano communication concept. In *Proceedings of Convergence 2002*, Detroit, Michigan, 2002.

[81] K. Ramaswamy and J. Cooper. Delivering multimedia content to automobiles using wireless networks. In *Proceedings of Convergence 2004*, Detroit, Michigan, 2004.

[82] J. Rushby. A comparison of bus architectures for safety-critical embedded systems. Technical Report NASA/CR-2003-212161, NASA, March 2003.

[83] R. Saket and N. Navet. Frame packing algorithms for automotive applications. *Journal of Embedded Computing*, 2:93–102, 2006.

[84] W. Saleem. 1394 Automotive network enables powerful, cost-efficient in-vehicle networks for infotainment, navigation, cameras. EETimes, available at `http://www.eetimes.com/`, May 2009.

[85] A. Schedl. Goals and architecture of FlexRay at BMW. Slides presented at the Vector FlexRay Symposium, March 2007.

[86] B. Schätz, C. Kühnel, and M. Gonschorek. *The Automotive Embedded Systems Handbook*, chapter The FlexRay Protocol, pages 5.1–5.22. Industrial Information Technology series. CRC Press/Taylor and Francis, December 2008.

[87] Society of Automotive Engineers. J2056/1 class C application requirements classifications. In *SAE Handbook*. SAE, 1994.

[88] Society of Automotive Engineers. J2056/2 survey of known protocols. In *SAE Handbook*, volume 2. SAE, 1994.

[89] Society of Automotive Engineers. Class B data communications network interface - SAE J1850 standard - rev. nov96, 1996.

[90] J. Sommer and R. Blind. Optimized resource dimensioning in an embedded CAN-CAN gateway. In *IEEE Second International Symposium on Industrial Embedded Systems (SIES'2007)*, pages 55–62, July 2007.

[91] T. Steinbach, H-T. Lim, F. Korf, T.C. Schmidt, D. Herrscher, and A. Wolisz. Tomorrow's in-car interconnect? A competitive evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802). In *IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–5, 2012.

[92] R. Tewell, Z. Freeman, and M. Bassler. The 1394 auto network bus for all high-bandwidth, high-speed communications and infotainment. Slides presented at SAE 2010 World Congress (session AE305), April 2010.

[93] K. Tindell, A. Burns, and A.J. Wellings. Calculating Controller Area Network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, 1995.

[94] TTTech Computertechnik AG. TTEthernet - a powerful network solution for multiple purpose. Marketing whitepaper, 2013.

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

[96] S. Tuohy, M. Glavin, E. Jones, M.M. Trivedi, and L. Kilmartin. Next generation wired intra-vehicle networks, a review. In *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, June 2013.

[97] A. Vollmer. Deutsche OEMs setzen standards. all-electronics.de, available at `http://www.all-electronics.de/texte/anzeigen/42481/Deutsche-OEMs-setzen-Standards`, June 2011.

[98] Volvo. Volvo unveils innovative safety technology - pedestrian detection with full auto brake debuts on the all-new Volvo S60. Press Release available at https://www.media.volvocars.com/, March 2 2010.

[99] M. Waern. Evaluation of protocols for automotive systems. Master's thesis, KTH Machine Design, Stockholm, 2003.

[100] C. Wilwert, N. Navet, Y.-Q. Song, and F. Simonot-Lion. *The Industrial Communication Technology Handbook*, chapter Design of Automotive

X-by-Wire Systems. CRC Press, January 2005. R. Zurawski editor, ISBN 0-8493-3077-7.

[101] J. Yoshida. NXP: 3 phases of Automotive Ethernet. EETimes, available at `http://www.eetimes.com/document.asp?doc_id=1319225`, August 2013.

[102] H. Zeltwanger. Partial networking reduces CO2 emissions. CAN Newsletter 4/2011, December 2011.

[103] H. Zeng, M. Di Natale, A. Ghosal, and A. L. Sangiovanni-Vincentelli. Schedule optimization of time-triggered systems communicating over the FlexRay static segment. *IEEE Trans. Industrial Informatics*, 7(1):1–17, 2011.

[104] T. Ziermann, J. Teich, and Z. Salcic. DynOAA - dynamic offset adaptation algorithm for improving response times of CAN systems. In *DATE*, pages 269–272, 2011.

Index

100BASE-TX, 28
1394 Automotive, 25, 26

Advanced Driver Assistance Systems (ADAS), 3, 10, 26, 27
AFDX, 29
AUTOSAR, 9, 10, 15, 32, 33, 35, 37, 38
AVB, 18, 25, 27–29
AVB Gen2 Council, 28
AVnu Alliance, 28

BMW, 4, 8, 19, 26
BroadR-Reach, 27

CAN, 4–9, 11, 15, 36–38
CAN 2.0A, 11
CAN 2.0B, 11
CAN FD, 10, 17
chassis domain, 5
collaborative mobility, 39
COTS toolsets, 15

Daimler, 26, 31

ECU degradation, 9
end-to-end communication protection library, 30, 37
Ethernet, 8, 26–28, 38, 39

FlexRay, 8, 15, 18, 31, 36, 38
frame-packing, 16, 20
FTT-CAN, 7
functional domains, 4

gateway, 8, 15, 16, 18, 39

ISO 11898-6, 9

J1850, 7, 11, 17
J1939, 15

LIN, 5, 7, 8, 17, 22, 36, 37

middleware, 29, 31–33
MOST, 8, 25, 26

Nissan, 26
Non-Return-to-Zero (NRZ), 11

OPEN Alliance SIG, 27
OSEK/VDX, 15, 31

partial networking, 9, 27, 30
point-to-point communication, 3, 38
powertrain domain, 4
PSA, 17, 32
PSI5, 5, 25

QoS, 25, 27, 29, 39

Renault, 26
Robert Bosch, 4, 17, 21, 27

SAE AS6802, 28
SAE classification, 7
SAE J2716, 25
schedulability analysis, 16, 31
scheduling frames with offsets, 16
security, 9, 10, 38
SENT, 5, 25
simulation, 16
SPC protocol, 25

TDMA, 6, 18, 19
time-triggered, 6, 18, 19, 21, 25, 28, 32
trace analysis, 16
TTCAN, 7, 18, 21, 22
TTEthernet, 7, 18, 28
TTP/A, 7, 22, 25
TTP/C, 7, 18, 19, 31

VAN, 11, 17
virtualization, 10
Volvo, 8, 31

X-by-Wire, 5, 18