

A Comparison of TTP/C and FlexRay

Research Report 10/2001

5

H. Kopetz

hk@vmars.tuwien.ac.at
Institut für Technische Informatik
Technische Universität Wien, Austria

10

May 9, 2001

15

Abstract: With the announcement of BMW and DaimlerChrysler to develop the new time-triggered protocol FlexRay for safety critical “X-by-Wire” applications, the time-triggered technology is moving into the mainstream of the automotive electronics market. This paper compares the established protocol TTP/C with the new protocol HexRay from BMW and DaimlerChrysler. This comparison is based on the sparse information about FlexRay that is currently in the public domain and is therefore subject to revisions as a more detailed specification becomes available. This paper identifies five failure scenarios that have the potential for a single node failure to result in a system-wide safety-relevant incident. It discusses how TTP/C controls these scenarios, but cannot answer the question how FlexRay handles these failures, because the information about FlexRay that is in the public domain does not address these safety relevant issues. The comparison comes to the conclusion that FlexRay and TTP/C were designed against the same set of automotive requirements, but that there is a difference in goals: The inherent conflict between flexibility and safety is tilted towards flexibility in FlexRay and safety in TTP/C.

20

25

30

Keywords: TTP/C, FlexRay, real-time systems, safety critical systems, Time-triggered communication.

35 1. INTRODUCTION

It is the objective of this paper to compare the two protocols TTP/C and FlexRay. This comparison is difficult, because the available description of the FlexRay protocol (Berwanger, Ebner et al. 2001) is vague and the precise specification of FlexRay is not publicly available. Therefore a number of guesses and assumptions have to be made that could be wrong. The basis of this comparison is the TTP/C protocol specification that is available on the Internet under the URL tpforum.org and the above cited FlexRay paper. Many design decisions of TTP/C and FlexRay are the same. This comparison paper does not reiterate the common design decisions, but focuses on the differences between the two protocols. Section 2 deals with the requirements for the new class of protocols for automotive “X-by-wire” applications. Section 3 presents the principles of operation of TTP/C and FlexRay. The core Section of this paper, Section 4, introduces five safety-relevant failure scenarios in order to compare the safety strategy of the two protocols. Section 5 discusses issues of system complexity and cost. Section 6 compares the two protocols in tabular form. The paper terminates with a conclusion in Section 7.

50 The reader should note that at present there is a certain competition between FlexRay and TTP/C on the market and that this paper has been written by the author of TTP/C. Although every effort has been made to support all claims by rational arguments, this paper nevertheless represents the view of TTP/C. A more balanced picture would evolve if a similar paper, written by the authors of FlexRay and representing the view of FlexRay, were available. In order to initiate such a discussion, an earlier version of this paper has been sent to some of the authors of FlexRay, asking them to examine the arguments and provide a critical response. No such response has been received.

2. REQUIREMENTS

60 The FlexRay document (Berwanger, Ebner et al. 2001) distinguishes between general requirements and specific automotive requirements.

2.1 General Requirements

According to the FlexRay document the following requirements for new protocols for the next generation of automotive applications have been identified by BMW and DaimlerChrysler:

- 65 (i) Higher Bandwidth: The presently available bandwidth, which is limited to about 500kbit/second for CAN systems, is not adequate for future applications.
- (ii) Determinism: Future control systems need the determinism provided by the time-triggered paradigm.

- 70 (iii) Fault-Tolerance: If mechanical and hydraulic control systems are replaced by computer control, the architecture must support fault-tolerance.
- (iv) Support for distributed control: Distributed control algorithms require a coordinated snapshot of the controlled object. Such a coordination can be achieved by the provision of a global time base.
- 75 (v) Unification of bus systems within vehicles: CAN will remain the selection of choice for event-triggered systems, while a new protocol is needed for applications which require high performance, determinism, fault-tolerance, and flexibility.

Exactly this set of requirements has also driven the development of the TTP/C protocol. However, in TTP/C an additional general requirement, the requirement for composability, has been considered of *utmost* importance. Composability is needed if systems are to be
80 built constructively out of prevalidated components and if the software reuse problem is to be solved. Only if the interfaces of components can be precisely specified in the value domain and in the temporal domain, it is possible to achieve composability of independently developed components. Composability is also a fundamental requirement for the productivity improvement in the area of testing and validation, and for composable safety
85 arguments.

Fundamental Conflict between Safety and Flexibility: If these general requirements are analyzed (independently of a particular architecture) it is evident that the fundamental conflict between the requirements for high safety and composability on one side and the requirement for flexibility on the other side must be addressed. For example, if it is known *a priori*
90 *priori* that a message must arrive from a node at exactly every full second (*restricted flexibility*), it can be decided immediately after the full second tick that something has failed and an appropriate remedial action can be taken instantly. If, on the other hand, a node is free to send a message whenever needed (*more flexibility*), such a prompt error detection and reaction is not possible. The precise interface specification in the temporal domain is thus a prerequisite for prompt error detection and composability, but restricts the flexibility.
95

In its recent recommendations for an industry standard, the aerospace industry, which has a long experience in building safety critical flight-control systems, is taking an even stronger position away from flexibility and towards determinism than in previous documents in order to further improve the system safety. In (WG-48 1999) Section 2.1 it is stated: *The Avionics Computing Resource (ACR) shall include internal hardware and software management methods as necessary to ensure that time, space and I/O allocations are deterministic and static. "Deterministic and static" means in this context, that time, space and I/O allocations are determined at compilation, assembly or link time, remain identical at each and every initialization of a program or process, and are not*
100 *dynamically altered during runtime.* The highly successful previous standard (ARINC 1992) for the development of safe systems, has not been so outspoken with respect to the elimination of flexible dynamic mechanisms in safety relevant systems.
105

2.2 Automotive Requirements

In addition to the general requirements, the FlexRay document (Belschner, Berwanger et al.
110 2000) enumerates also the following list of specific automotive requirements:

- (i) Configurable synchronous and asynchronous transmission

- (ii) High bandwidth of up to 10 Mbit/sec.
- (iii) Deterministic data transmission with guaranteed latency and minimal jitter.
- (iv) Support of scalable redundancy.
- 115 (v) Prompt error detection and error reporting.
- (vi) Global time.
- (vii) Fault-containment at the level of the physical layer.
- (viii) Media-access without arbitration.
- (ix) Support for a fiber-optics physical layer as well as for an electrical physical layer.
- 120 (x) Flexibility, expandability and easy configuration in automotive applications.

Although the two protocol developments, TTP/C and FlexRay, consider the same set of requirements, they *differ from the point of view of the tradeoff* between conflicting requirements.

125 **TTP/C:** In TTP/C the concern for *safety* is always the primary driver. Second to safety is *composability* and third *flexibility*. Flexibility is only introduced if the goals of safety and composability are not compromised. If the above list would be reordered according to the priorities of TTP/C, the requirements related to safety would be at the top of the list.

130 **FlexRay:** The FlexRay document does not discuss the requirement conflicts. A critical reader of the FlexRay document will however find out that the development of FlexRay is tilted away from safety and composability and more towards flexibility than TTP/C. This is also demonstrated by the order of the requirements on the above list.

135 **Discussion:** What is the utility of flexibility if a system is not safe? A prerequisite for flexibility and a constructive safety case is composability. Composability (that enables the reuse of prevalidated ECU's in differing contexts) requires that the interfaces of an ECU are fully specified in the value domain and in the temporal domain and that the components will operate correctly as long as the interface specifications are maintained. The Byteflight protocol section of FlexRay makes the protocol non-composable in the temporal domain, since the processing of the sporadically arriving event messages can interfere with the established schedules of the time-critical tasks in the host computer (see Fig. 1).
 140 Of what value is flexibility if a system is not composable, i.e., if it cannot be determined *a priori* constructively (i.e., without costly system tests that are hardly ever complete) that an ECU will work in the changed environment?

3. PRINCIPLES OF OPERATION OF FLEXRAY AND TTP/C

145 In this section the principles of operation of TTP/C and FlexRay are elaborated. For a more detailed description of the protocols, the interested reader is advised to consult the respective information on the Internet. While the TTP/C information policy is open—the TTP/C specification is available on the net and has been downloaded by more than 2000 organizations--the FlexRay specification is not openly available.

3.1 History

150 **TTP/C:** The TTP/C protocol was developed at the Technical University of Vienna in the
 course of more than twenty years of research on dependable real-time systems. TTP is
 based on the MARS system that was exposed to extensive fault-injection experiments
 during the European Research Project PDCS. In the European Research Projects X-by-
 Wire and TTA and in a cooperative research between the Technical University of Vienna
 155 and DaimlerChrysler a number of prototype implementations of TTP were developed and
 evaluated. After the completion of these research projects at the end of 1998, TTTech, a
 high-tech spin-off company of the TU Vienna, was founded with the charter to further
 develop and market the TTP technology.

160 **FlexRay:** The FlexRay protocol is a combination of the Byteflight (BMW 1999) protocol
 from BMW and TTP/C from the Technical University of Vienna. Byteflight was originally
 designed for passive safety systems for applications like Airbag release, where short
 response times are required. Since the active mission time of an Airbag system is short (in
 the millisecond range), the probability that a failure will occur during such a short mission is
 low. Therefore Byteflight does not need to support fault-tolerance and as such is not
 165 suitable for active control systems, such as “X-by-wire”, which have a long mission time and
 therefore require fault-tolerance. The FlexRay consortium between BMW and
 DaimlerChrysler was formed with the goal to enhance the Byteflight protocol from BMW to
 make the protocol suitable for “X-by-Wire” systems.

3.2 System Structure

170 The structure of a TTP/C and FlexRay node is similar, as outlined in Fig. 1. Every node
 consists of a host computer and a communication controller. Between the host computer
 and the communication controller is the communication network interface (CNI). A fully
 specified CNI, both in the temporal domain and in the value domain, is an important
 prerequisite for the composability of an architecture.

175

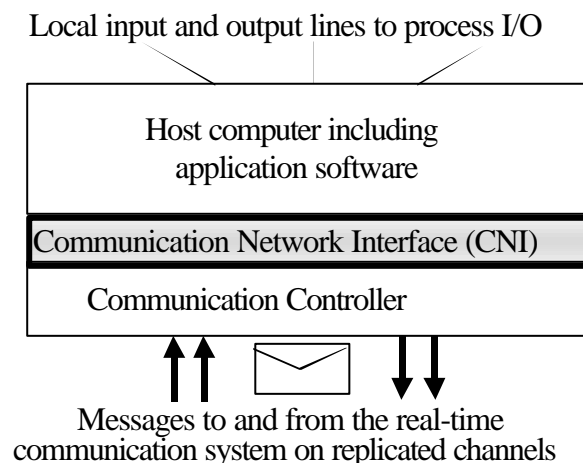


Figure 1: Structure of a node in TTP/C and FlexRay

3.3 Time-Triggered Message Transmission

180 Both protocols, TTP/C and FlexRay, use a TDMA (time-division-multiple-access) strategy for the transmission of time-triggered messages. Both protocols contain a fault-tolerant clock synchronization and protect the communication channel from babbling nodes by a bus guardian (BG). The maximum data field length of a message is 12 bytes in FlexRay and 236 bytes in TTP/C.

185 **TTP/C:** In order to detect any consistency violation within the communication system promptly, TTP/C contains a distributed membership service as part of the communication protocol. This membership service is instrumental for the safety strategy of TTP/C. The clock synchronization and the membership protocol of TTP/C have been formally analyzed (Pfeifer, Schwier et al. 1999; Pfeifer 2000; Rushby 2000).

190 **FlexRay:** FlexRay does not provide a distributed membership service at the protocol level. The correctness arguments for the fault-tolerant clock-synchronization algorithm, as presented in (Belschner, Berwanger et al. 2000) are not convincing and require a further substantiation. According to our understanding, a faulty node can acquire an undue weight in the clock synchronization algorithm.

195 3.4 Event Triggered Message Transmission

The protocols provide *very different* mechanisms for the transmission of asynchronous messages. TTP/C provides the event-channels on top of the time-triggered protocol, while FlexRay provides the event-channels parallel to the time-triggered protocol.

200 **TTP/C:** TTP/C requires the reservation of an *a priori* specified number of bytes in a message for the transmission of event-triggered messages. This implies that the bandwidth assigned for asynchronous message transmission cannot be shared among nodes. It is possible to implement with these reserved bytes an event-triggered protocol, e.g., CAN, in such a way that the communication network interface that is visible to the user presents the event information according to the CAN standard and the available CAN legacy software can be reused with minimal changes. From the point of view of a user, such a CAN implementation on top of TTP/C can provide to the user all CAN facilities, such as priority arbitration at the sender and message filtering at the receiver, that are available in CAN systems. No change whatsoever is required to the underlying TTP/C protocol, implying that the composability of a TTP/C system is not compromised by the support for event-triggered messages.

210 **FlexRay:** FlexRay partitions the time line into two recurring intervals, one synchronous channel for time-triggered messages and one asynchronous channel for event-triggered messages. In the asynchronous part of the protocol the bandwidth is shared among the nodes in order to achieve a better bandwidth utilization. Media access for the asynchronous messages is controlled by the Byte-flight mini-slotting protocol. The messages in both parts have the same structure and length of 16 bytes.

215 Let us assume, that the arrival of a new asynchronous message is reported to the host computer by raising an interrupt (the FlexRay document does suggest this interpretation). Given the slot length for an asynchronous message of about 200 bits, in a 10 Mbit system, a message arrival interrupt to the host computer will arrive about every 20 μ sec.

220

Discussion: The tradeoffs between the two different methods for handling event traffic are shown Table 1:

Characteristic	TTP/C	FlexRay
Number of media access protocols required in the controller	one	two
Dynamic Bandwidth Sharing among nodes	no	yes
Precise temporal specification of interfaces	yes	no
Composability of the architecture	yes	no
Bus guardian protection of event messages	yes	no

Table 1: Tradeoffs between the different methods of handling event traffic

225

The TTP/C decision to provide static channels for the dynamic asynchronous traffic was driven by the following arguments:

- 230 (i) Despite the support for dynamic asynchronous traffic, the communication network interface (CNI) of every node remains fully defined in the temporal domain and in the value domain, *a prerequisite for temporal composability*.
- (ii) With static channel assignment, the protocol can perform a complete error detection on the arrival of all messages. This improves the responsiveness of the membership service and increases the performance of higher-level protocols for event-message channels.
- 235 (iii) In our opinion, the advantages of a single intelligible media access mechanism in the protocol (TDMA) and the associated temporal composability far outweigh the performance improvements gained by dynamic channel sharing for non time-critical asynchronous traffic. If, for example, in TTP/C a static bandwidth of 150kbits/second is assigned to each node for the asynchronous message transmissions, the total bandwidth requirement in a 10 node 10 Mbits/s system is about 15% (This is much less than the data-efficiency advantage of TTP/C compared to FlexRay, as depicted in
- 240 Table 5). Such a system can transport about 10 000 CAN messages every second with a guaranteed response time of about 1 msec for the highest priority CAN message at each node.
- (iv) The asynchronous traffic is also protected by the Bus Guardian.
- 245 (v) In case of an orthogonal operating mode (e.g., a maintenance mode to download new software versions) a new set of consistent time-triggered schedules (message descriptor lists MEDLs) can be loaded into the TTP controllers within a fraction of a second in order to meet the specific bandwidth requirements of this operating mode.

250 Since in TTP/C safety and composability have priority over flexibility, it was decided that the fault-isolation of event messages and the composability are more important than the flexibility of dynamic bandwidth assignment.

3.5 Physical Interconnection Structure

Both protocols, TTP/C and FlexRay support different physical interconnection structures, a bus configuration and a star or multi-star configuration, which is recommended for safety critical fail-operational configurations. There is a safety relevant difference among the protocols in the placement of the BG in the star configuration.

TTP/C: In the star configuration of TTP/C, the BG is integrated into the replicated central star couplers (Bauer, Frenning et al. 2000). The TTP bus guardian is placed in physical distance from the node its protects, contains its own independent power supply, its own clock with an autonomous clock synchronization, and reshapes the transmission signals using its local clock and local power supply. This independent reshaping of the transmission signals is necessary in order to mask slightly-off-specification (SOS) faults (see Section 4.2).

FlexRay: The FlexRay publication states (Berwanger, Ebner et al. 2001), p. 11: *Each communication node has a host with a subordinate communication controller. One or two bus drivers and bus guardians can be connected to this. All components are coupled through a power supply, which is connected directly with the vehicle battery. . . . The bus guardian is controlled by the communication controller, while the bus driver controls the power supply (p.12).*

Discussion: In TTP/C the bus guardian is logically and physically independent of the communication controller, while in FlexRay the bus guardian shares the power supply with the communication controller, is controlled by the communication controller, depends on clock synchronization of the communication controller, and is in physical proximity of the communication controller. It is difficult to argue that in FlexRay the bus guardian is independent of the communication controller. Such an independence is however a requirement from the safety point of view.

3.6 Fault Hypothesis

The design of any fault-tolerant system starts with the specification of the fault-hypothesis. The fault hypothesis states the assumptions made (Powell 1995) about the types and number of faults that the system should tolerate and determines the design of the fault-tolerance mechanisms.

During fault analysis, the following two cases must be clearly distinguished:

- (i) A fault occurs that is covered by the fault hypothesis. We call such a fault a *level-1* fault. Level-1 faults must be tolerated by a fault-tolerant system.
- (ii) A fault occurs that is outside the fault hypothesis. We call such a fault a *level-2* fault. It cannot be expected that the fault-tolerance mechanisms mask level-2 faults, since the fault-tolerance mechanisms have not been designed to handle these faults. A well designed architecture for high-dependability applications will provide additional mechanisms that bring a system back into a safe state (if any), even after the occurrence of a level-2 fault. These mechanisms are sometimes called NGU (never-give-up) mechanisms.

During the design of a safety case, estimates for the occurrence of unit failures must be provided and it must be shown that the probability of a level-2 failure is sufficiently small,

295 i.e., *that level-2 failures are rare events*. A precise fault-hypothesis is an important
 criterion for allocating resources during system design. The focus during the design must
 concentrate on the elimination of those faults that are not covered by the fault hypothesis. It
 is not possible to validate the fault-tolerance mechanisms without a clearly stated fault
 hypothesis.

300 In Table 2 we state the failure rate assumption (order of magnitude estimate) concerning the
 occurrence of independent failures:

Message transmission time	0.36 msec	10^{-7} hours
2 TDMA rounds	3.6 msec	10^{-6} hours
Cluster recovery	36 msec	10^{-5} hours
Independent message loss	every six minutes	10^{-1} hours
ECU failure transient		10^2 hours
ECU failure permanent		10^7 hours

Table 2: Failure rate assumptions for independent failures

305 We assume that the failure rate of a correlated failure that affects the message transmissions
 on both replicated channels, i.e., both messages of a node, is subsumed in the ECU
 transient failure rate. This failure rate is orders of magnitude higher than the failure rate
 calculated under the assumption of independent message loss

310 **TTP/C:** The fault-hypothesis of TTP/C states that any properly configured TTP/C cluster
 will tolerate any single failure and many multiple failures of its constituent parts. If we assume
 the independent failure rates outlined in Table 2, then the probability that a level-2 fault
 occurs once during the lifetime of car is extremely low. In a properly designed application,
 the membership service is used to detect a violation of the fault hypothesis (a level-2 fault) in
 the communication system and to activate the NGU strategy.

FlexRay: There is no fault-hypothesis specified in the available FlexRay documents. It is
 therefore not possible to validate the fault-tolerance mechanisms of FlexRay.

315 **3.7 Never-Give-Up (NGU) Strategy**

320 Any safety critical system must contain multiple defenses against catastrophic failures. One
 such defense is the NGU strategy. The failure to provide an NGU mechanism was the cause
 of the well-publicized ARIANE accident on June 4, 1996 (Ladkin 1998). The designers of
 the ARIANE 5 rocket assumed that the software is free of design faults and therefore no
 mechanisms for handling software failures were provided (no level-2 fault analysis). The
 occurrence of the same software failure on replicated hardware components caused the loss
 of the rocket.

325 In the automotive environment the occurrence of correlated multiple transient failures must
 be addressed. Replication is not very effective at handling correlated failures, because a
 single external fault may cause all replicated units to fail. Correlated failures must be handled
 in the design phase, e.g., by designing a proper physical layer (e.g., fiber optics). Correlated
 failures should be rare events if a system is properly designed and installed.

TTP/C: The membership protocol of TTP/C detects promptly a level-2 fault, i.e., a violation of the fault-hypothesis caused by multiple correlated failures. Based on his extensive experience in the design of safety-critical systems in the aerospace industry, Rechlin has formulated the following design heuristics p. 153: *Chances for recovery from a single failure, even with complex consequences, are fairly good. Recovery from two or more independent failures is unlikely in real time and uncertain in any case.* The NGU strategy of TTP/C is therefore as follows: after the detection of a multiple failure (which is outside the fault hypothesis) by the membership algorithm, the application can decide to put the actuators under local control (normally frozen in their current state) and the system is restarted with a clean state within the order of 10 milliseconds. This recovery time is below the requirements stated by Thurner and Heiner (Thurner and Heiner 1998): *Taking the vehicle dynamics into account, the transient outage-time can be estimated to be less than 50ms. This time is crucial for the design of recovery mechanisms.*

If level 2 failures, i.e., the occurrence of faults outside the specified fault hypothesis, are not rare events during the test and operation of a safety critical system, *then they must be caused by a fundament design problem* (e.g., inadequate shielding against EMI or an inadequate fault hypothesis). *Such systems should not be deployed in safety critical applications.*

FlexRay: No NGU strategy is discussed in the available FlexRay documents. There is no mechanism in FlexRay that triggers an NGU mechanism.

3.8 Communication Network Interface (CNI)

The communication network interface (CNI) is the interface between the communication controller and the host computer. The amount of memory (either in the form of registers or dual ported RAM) provided by the controller at the communication network interface is an implementation characteristic that determines the temporal coupling between communication controller and host. In one extreme case, a single send and receive register is provided in the CNI, which then must be serviced very frequently by the host (after the arrival of each message an interrupt is generated—*information push interface*). In the other extreme case, the communication controller will provide a large dual-ported memory in the CNI to store all state data and a large circular buffer to store the event data in order to avoid spontaneous interruptions of the host computer (*information pull interface*). Considering the temporal complexities introduced by *information push* interfaces versus the simplicity of *information pull* interfaces (Deline 1999), and the decreasing cost of memory, the tradeoff between these two extreme cases is shifting to the provision of an *information pull interface* in the CNI.

Any time-triggered protocol requires dynamic configuration data that must be loaded into a communication controller before the operation of the time-triggered protocol can start. The size of this dynamic configuration data depends on the flexibility supported by a particular implementation. If all parameters are fixed (e.g., message length, where a message will be stored in the communication network interface, when a message stored at a particular location will be sent, etc.) at the time of protocol design, most of the configuration data is static and can be placed into a read-only memory of the communication controller (or can become part of the wired logic), while the dynamic configuration data can be minimized to a few bytes. The amount of dynamic configuration data required is thus not a protocol characteristic, but a characteristic of a particular implementation.

375 **TTP/C:** In TTP/C the configuration data is stored in a controller internal data structure MEDL (Message Descriptor List) that must be loaded into the controller before the start of the protocol. In the first controller implementation it has been tried to keep all options open for changing dynamically as many parameters as possible. This resulted in a rather large MEDL. If, in a particular mass market application, the communication requirements are established, then many of the MEDL data elements can be placed into ROM and the dynamic data structure can be minimized.

380 **FlexRay:** FlexRay minimizes the configuration data by fixing many parameters in the protocol definition. The remaining dynamic configuration data must be stored into the controller at startup: *The protocol must be parameterized before startup.* (Berwanger, Ebner et al. 2001) p.5.

3.9 Data Efficiency

385 The data efficiency of a protocol can be expressed by the percentage of time that application data is transported over a channel.

Flexray supports a data field of up to 12 bytes and a frame length of up to 16 bytes, including a 2-byte CRC field.

390 TTP supports a data field of up to 236 bytes and a frame length of up to 240 bytes, including a 3-byte CRC field.

The minimum length of the inter-frame-gap of time-triggered messages that are protected by a bus guardian has been thoroughly studied in the PhD thesis by Temple (Temple 1998). This minimum inter-frame-gap depends on the propagation delay, the precision of the distributed clock synchronization, the microtick duration, the sampling frequency of the bus-guardian and other terms. Under realistic assumptions the minimum inter-frame gap is in the range between 5 μ sec and 20 μ sec.

400 In a 10 Mbit/second system with 5 μ sec inter-frame-gap the maximum achievable data efficiency for time-triggered messages is around 95.8 % in a TTP/C system (1888 bits of user data in a slot that corresponds to a bit length of 1970 bits), whereas the maximum data efficiency of a corresponding FlexRay system is around 45.7 % (96 bits of user data in a slot that corresponds to a bit length of 210 bits). If the systems are scaled to higher bandwidths (e.g., 100 Mbits/second, 5 μ sec inter-frame-gap) the data efficiency difference between TTP/C (78%) and FlexRay (14.5%) is even more pronounced.

405 **Discussion:** The longer message size in TTP/C has been introduced in order to increase the data efficiency in higher-speed systems and to provide sufficient bandwidth for distributed state recovery and the inclusion of the event-traffic in the time-triggered messages. It can be expected that higher speed-systems (100 Mbits/second and beyond) will become possible in the foreseeable future.

4. FAILURE SCENARIOS

410 In this Section the handling of the following failure modes by TTP/C and FlexRay is analyzed:

- (i) Outgoing link failure
- (ii) Slightly-off-specification (SOS) failure
- (iii) Spatial proximity failure
- 415 (iv) Masquerading failure
- (v) Babbling idiot failure

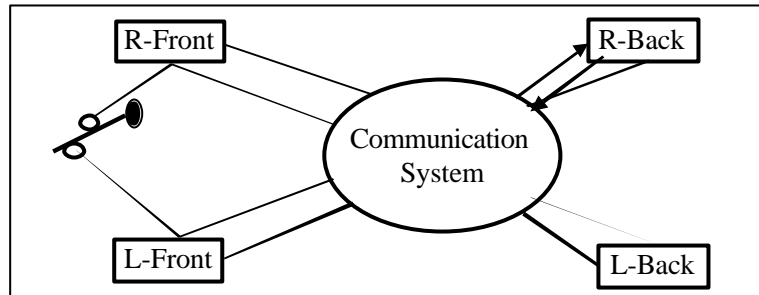


Figure 2: Simple “brake-by-wire” application

420 The simple “brake-by-wire” application of Fig. 2 is introduced in order to demonstrate the effects of these failures. In this application four Electronic Control Units (ECUs), one at each wheel, are connected to a fault-tolerant TTP/C or FlexRay communication system. The *R-Front* and the *L-Back* ECU accept the brake pedal pressure from one fail-silent brake pedal sensor, the *L-Front* and *R-Back* ECU accept the brake pedal pressure from the other fail-silent brake pedal sensor. Every wheel ECU informs all other ECUs about its view of the brake pedal sensors, performs a distributed algorithm to allocate the brake force to each wheel and controls its local wheel. The brake is assumed to be designed in such a way that the brake autonomously visits a defined state, e.g., *wheel free running—no brake force applied* in case the wheel ECU crashes or the electric or mechanic mechanism in the local brake fails. As soon as the other three wheels learn about the failure at one wheel, they redistribute the brake force to the other wheels in order that the car is stopped safely with three braking wheels. The time interval between the instant of brake failure and the instant of redistribution of the brake-force, the error detection interval, is a *safety critical parameter of this application*. During this error detection interval the braking system is in an inconsistent state. We conjecture that there is a potential for a fatal accident if this inconsistent state is not detected and corrected within at most a few sampling intervals

In the aerospace community the safety analysis of such a system must assume that any single unit can fail with a probability of 10^{-6} /hour in an *arbitrary* failure mode. In order to achieve a system failure rate of better than 10^{-9} /hour it must be shown that the architecture provides mechanisms to tolerate any arbitrary unit failure without causing a system wide safety-relevant incident. If two units depend on each other and there is no effective error detection and error propagation boundary between these two units, or the units are placed in the same fault-containment region, than these two units are considered a single unit from the point of view of safety analysis. Such an approach seems to be justified in the mass market of electronic safety systems in cars, because if tens of millions of cars are on the road for many hundred hours each year, it can be expected that any failure-mode of dependant units is possible. Since it is very difficult to establish a failure rate 10^{-9} /hour by experimental means, logical analysis and analytical arguments must be presented to demonstrate that the safety goals are achieved.

450 In the following analysis we focus on single unit failures during a protocol execution that can cause a system wide safety relevant incident.

4.1 Outgoing Link Failure

455 An outgoing link failure occurs if a node cannot send a message to the other nodes in a cluster, e.g., because of a failure in the send logic of the communication controller caused by physical aging of a component. According to the end-to-end argument (Saltzer, Reed et al. 1984) only the receiving nodes can decide if a sender had an outgoing link failure. Any local mechanism for detecting this failure that is part of the sending unit, e.g., loop-back of the send signal, reduces the probability of this failure mode but is not sufficient for the safety argument as introduced above.

460 Assume that in the example of Fig. 2 the *R-Back node* has an outgoing link failure. If the *R-back node* does not know about its outgoing link failure, the *R-back node* will continue braking. The other nodes, not hearing any response from the *R-Back node*, assume that the *R-Back node* has failed and its brake is in the position “*wheel free running*”. As a consequence, there is a safety relevant inconsistent state in the braking system.

465 **TTP/C:** The TTP/C membership service informs every node about the “health -state” of every other node within the theoretical minimal time of one TDMA rounds. If *the R-back node* learns from this membership service that all other nodes assume that it has failed, it will visit the defined state (*wheel free running*) autonomously and thus reestablish consistency within at most two TDMA rounds.

470 **FlexRay:** FlexRay does not provide a membership service. It is not clear how the FlexRay protocol handles this safety critical scenario.

Discussion: It is sometimes proposed to solve this problem at the application level, by implementing some form of a membership protocol at the application level. Establishing a consistent membership at the application level takes significantly longer than establishing membership at the protocol level in silicon. This implies that the error detection interval and therefore ~~the~~ *safety critical state of inconsistency* lasts longer than in the case where membership is established at the protocol level. Furthermore, the load imposed on the host CPU for implementing a responsive membership protocol is hefty, making the host-software more complex.

480 4.2 Slightly-off-Specification (SOS) Failure

Slightly-off-specification failures can occur at the interface between the analog and the digital world. Assume the situation as depicted in Fig. 3. The specification for a type of nodes requires that every correct node must accept input signals if they are within a specified receive window of a parameter (e.g., frequency or voltage). Every individual node will have a wider actual receive window than the one specified in order to ensure that even if there are slight variations in manufacturing it can accept all input signals as required by the specification. These actual receive windows will be slightly different for the individual nodes, as shown in Fig. 3. If an erroneous node produces an output signal (in time or value) slightly outside the specified window, some nodes will correctly receive this signal, while others might fail to receive this signal. Such a scenario will result in an inconsistent state of the distributed system.

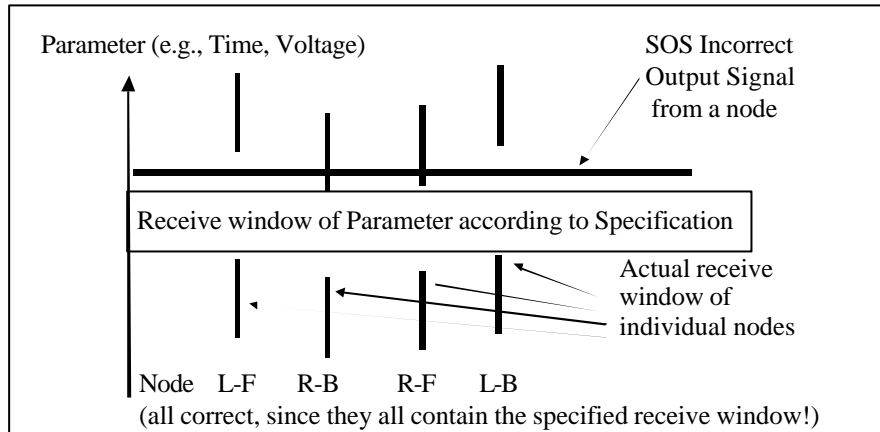


Figure 3: Slightly off Specification (SOS) failure

495 In the example of Fig. 2, an SOS output failure of the *R-Back* node will cause confusion in the distributed system. According to Fig. 3, the *L-Front* and the *L-Back* node will assume that *R-Back* node is operating correctly, while the *R-Front* node will assume that *R-Back* node has failed. This inconsistency is safety relevant.

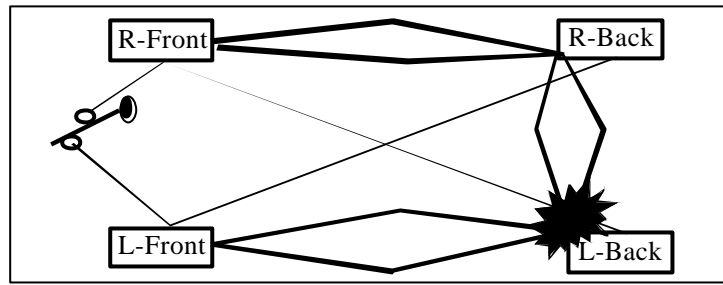
500 **TTP/C:** The membership algorithm of TTP/C will detect this inconsistency within at most two TDMA rounds, the theoretical limit for distributed agreement. The bus guardians in the star coupler of TTP will transform this SOS signal into either a correct or an incorrect signal, thus eliminating this inconsistency. This is the reason why the bus guardian of TTP/C reshapes all analog signals with its own independent clock and its own independent power supply. Since there are two stars with two bus guardians, a single SOS fault will be
505 tolerated.

FlexRay: It is not known, how FlexRay handles this failure scenario.

510 **Discussion:** The replication of channels will not solve this problem, if the signal on both channels are generated by the same timing source and/or the same power supply. It is known that crystals slightly change their physical characteristics as they age, increasing the probability that SOS signals in the frequency domain will occur during the lifetime of a car. The distributed clock synchronization, neither in TTP/C nor in FlexRay, will be effected by such an SOS failure, since both systems deploy synchronization algorithms that can handle this failure class.

4.3 Spatial Proximity Failure

515 A spatial proximity fault occurs, if the replicated units introduced for the purpose of fault-tolerance are in close physical proximity and are thus in a single fault-containment region. A single external event, e.g., a fire in a cabinet or the local impact in an accident, can thus disable all replicated units. For example, a replicated bus must be brought into close physical proximity at each node. An external event that annihilates all matter in a defined
520 volume of space (Fig. 4) will cause the disruption of both replicated busses. This is one of the reasons, why both FlexRay and TTP/C propose a replicated star as the physical interconnection structure.



525 **Figure 4:** Single point of physical failure in replicated busses

TTP/C: The central guardian in the star coupler of TTP/C, which enforces a fail-silent behavior of a node in case of an arbitrary node failure, is in a *different* physical fault-containment region than the node it protects.

530 **FlexRay:** The guardian which enforces a fail-silent node failure is at the node and thus in the *same* physical fault containment region as the node. It is not discussed how FlexRay handles physical proximity faults.

535 **Discussion:** The FlexRay paper (Berwanger, Ebner et al. 2001) states on p.11: *The active stars are fitted with any number of bus drivers and the accompanying shared power supply, which is again connected directly to the vehicle battery. Bus Guardians are not included. The validity of a branch claiming the broadcast channel through the star is derived from the validity of signal entry into the star, which is derived from the validity of the transmitting node. No further intelligence is necessary in the star.* If this is the case, a babbling idiot on both channels caused by a single physical fault in one fault-containment region (node plus bus guardian) has the potential to propagate through the central star and disable the communication in the total system.

4.4 Masquerading Failure

545 A masquerading failure occurs if an erroneous node assumes the identity of another node and causes harm to the system. Systems that rely only on names stored in a message to identify the transported message and the information contained therein are vulnerable to masquerading faults. It opens the possibility that a single faulty node can masquerade other nodes, without the receiver having a chance to detect the fault. For example, if a bit in the name of a message to-be-sent that is stored in the sending node is incorrect, this message could, after arrival at its destination, overwrite correct messages at correct receivers. This problem is discussed at some length in the safety critical SafeBus protocol (Driscoll and Hoyme 1993) p.36: *Any protocol that includes a destination memory address in a message is a space-partitioning problem.*

555 **TTP/C:** In order to avoid masquerading failures, TTP/C identifies the information by *static a priori* knowledge stored in both components, the sender and receiver, and dynamically establishes the state consistency between sender and receiver to detect a node that is considered faulty by a majority of receivers. This eliminates the possibility that a node can masquerade as another node.

560 **FlexRay:** The FlexRay paper (Berwanger, Ebner et al. 2001) states on p.7: *The receive units decode the encoded bit stream and fill the corresponding message buffer with the received data.* It follows that if an incoming bit stream contains a wrong message name

(ID), generated by a faulty sender, the message will be placed into the wrong buffer at all receivers, possibly overwriting correct message from other nodes and giving rise to masquerading faults.

4.5 Babbling Idiot Failure

565 A babbling idiot failure occurs, if in a broadcast communication system an erroneous node does not observe the rules of the media access protocol and sends messages at arbitrary instants. The probability for the occurrence of babbling idiot failures in a broadcast communication system has been experimentally investigated by different fault injection methods (Karlsson, Folkesson et al. 1995). Since this probability is too high for safety critical systems, independent bus guardians are installed in safety critical architectures to prevent this failure mode.

570 **TTP/C:** In TTP/C the replicated guardians are integrated into the replicated star couplers. The bus guardians are thus at a physical distance from the node they monitor, and are completely independent with their own power supply, their own clock and their own distributed clock synchronization algorithm.

575 **FlexRay:** The bus guardians are controlled by the node they supervise and are at the same place as the node they supervise, sharing some of the resources with the node they supervise. It is difficult to argue that the bus guardians and the node they supervise are independent units.

580 5. SYSTEM COMPLEXITY AND COST

The most important issue in the design of future automotive computer systems is the reduction of the system complexity. Unmanaged system complexity is at the root of the software problems that plagues our modern society (Littlewood 1997). In safety critical applications, system complexity has been diagnosed as the cause of many fatal accidents (Perrow 1999). Leveson (*Leveson 1995*) p.8. states: “*Many of the new hazards are related to increased complexity (both product and process) in the systems we are building.*” System complexity can be reduced by the following techniques:

- (i) The specification of a coherent architecture that guides the designer and avoids property mismatches at the interfaces.
- 590 (ii) The partitioning of a system into nearly autonomous components connected by fully specified deterministic interfaces, both in the value domain and in the temporal domain in order to support composability. In the cited SafeBus paper (Driscoll and Hoyme 1993) it is stated on p. 35: *If the system design does not build in time determinism, a function can be certified only after all possible combinations of events, including all possible combinations of failures of all functions, have been considered. Clearly, this would drastically increase the cost of certification, as well as software maintenance.*
- 595 (iii) The once-for-all implementation, preferably in silicon, of generic functions that are needed in many different applications.

600 5.1 Complexity in TTP/C

TTP/C is the most important building block of the time-triggered architecture (TTA) (Kopetz 1997) that provides a well-researched framework for the implementation of composable dependable real-time applications according to the time-triggered paradigm. By solving the difficult membership problem generically at the protocol level in silicon,
 605 TTP/C simplifies the application software complexity and reduces the effort and time-to-market for new control systems. There are already additional generic solutions in the research pipeline that will simplify the application software even further:

- (i) At last years Distributed Systems and Network Conference (Bauer and Kopetz 2000) a generic hardware supported fault-tolerance layer for the time-triggered architecture has
 610 been presented. This fault-tolerance layer hides all fault-tolerance mechanisms from the operating system and the application software, in order that the complexity of the software is not increased by the provision of fault tolerance.
- (ii) At the Technical University of Vienna, a research project is under way to implement high-level event-triggered protocols (e.g., TCP/IP or CAN) on top of the time
 615 triggered TTP/C protocol in order that existing legacy software can be reused with minimal effort.
- (iii) The time-triggered smart-transducer bus protocol TTP/A (Kopetz, Holzmann et al. 2000) which is in the process of standardization by the OMG, provides the same
 620 interfaces to the host computer CNI as the TTP/C protocols. This simplifies the operating system of the host computer.

5.2 Complexity in FlexRay

The complexity of FlexRay-based systems is influenced by the following design decisions:

- (i) FlexRay is a combination of mechanisms from two different protocol worlds, synchronous and asynchronous, and does not support a single coherent architectural
 625 view. The complexity caused by the feature interactions of the two protocols incorporated in FlexRay are an issue of concern.
- (ii) The event-triggered Byteflight part of the FlexRay protocol makes it impossible to specify the temporal properties of component interfaces independently of the global context of an application. Composability is thus not supported in FlexRay.
- (iii) The decision to push the already generically solved hard membership problem back
 630 into the software realm makes the implementation of a generic fault-tolerance layer more complex. It furthermore increases the load on the application processor, the software size and complexity and thus the probability of software errors.

5.3 Cost Comparison

635 From a cost perspective, the most distinguishing difference between the TTP/C and the FlexRay architecture is the functionality and the placement of the bus guardians. In TTP/C two bus guardians are integrated into the two star couplers, whereas FlexRay requires two bus guardians at each node. This implies that a TTP/C system with n nodes needs $n+2$ oscillators, whereas a comparable FlexRay system requires $3n$ oscillators. If the bus
 640 guardian of FlexRay are designed to be *fully independent* of the nodes they protect (i.e.,

they have their own power supply, their own distributed fault-tolerant clock synchronization algorithm, contain their own configuration data, and are placed in a separate package at a physical distance from the node they are to protect), then the cost of each of the $2n$ bus guardians in FlexRay is in the same cost category as the cost of the two central bus guardians in TTP/C. This means that in a ten node system, TTP/C will require twelve packages (10 nodes, 2 star couplers) and 12 oscillators, whereas FlexRay will require 32 packages (10 nodes, 20 bus guardian, 2 star couplers) and 30 oscillators. Since the number of packages and the number of oscillators are significant drivers for system costs, a FlexRay system is expected to be considerably more expensive to implement than a comparable TTP/C system.

6. TABULAR COMPARISON OF TTP/C AND FLEXRAY

TTP/C and FlexRay have been developed with consideration of the same set of automotive requirements. There are, however, three fundamental differences in the design of TTP/C and FlexRay. These differences relate to the safety, composability and the consistency of the respective systems:

- (i) **Safety:** Whenever there is an inherent requirements conflict between safety and flexibility, TTP/C considers safety as more important than flexibility. FlexRay, as the name implies, puts more emphasis on flexibility than TTP/C.
- (ii) **Composability:** In order to enable a precise interface specification of a node in the temporal domain and in the value domain (*a necessary prerequisite for composability*), TTP/C does not support dynamic bandwidth sharing for asynchronous traffic. Dynamic bandwidth sharing, as used in FlexRay, makes it impossible to obtain a precise interface specification in the temporal domain.
- (iii) **Consistency:** TTP/C contains, at the protocol level, services (e.g., a membership service) that continuously monitor the consistency of the distributed system and detect a violation of the fault-hypothesis within a short delay. These services are necessary for the prompt activation of an NGU strategy in case a rare level-2 fault scenario occurs. In FlexRay these services are not part of the protocol and must be implemented at the application level.

The following Table 5 compares the properties and services of the TTP/C and FlexRay protocol in tabular form:

675

Characteristic	TTP/C	FlexRay
Designed to meet automotive requirements	yes	yes
Priority in the “safety versus flexibility” conflict	safety	flexibility
Specification in the public domain	yes	no
Composability (precise interface specification in the value domain and in the temporal domain)	yes	no
Fault-tolerant clock synchronization	yes	yes
Replicated communication channels	yes	yes
Time-triggered message channels	yes	yes
Bus guardians to avoid babbling idiots	yes	yes
Bus guardian and protected node in different fault-containment regions	yes	no
Dynamic asynchronous message channels	yes, local	yes, global
Membership service	yes	no
Fault-hypothesis specified	yes	no
Never-give-up (NGU) strategy specified	yes	no
Critical algorithms formally analyzed	yes	no
Handling of outgoing link failures	yes	?
Handling of SOS failures	yes	?
Handling of Spatial Proximity failures	yes	?
Handling of Masquerading failures	yes	?
Handling of babbling idiot failures	yes	?
Transmission speed planned up to	25 Mbits/sec	10 Mbits/sec
Message data field length up to	236 bytes	12 bytes
Physical layer	copper/fiber	copper/fiber
CRC field length	3 bytes	2 bytes
Maximum achievable data efficiency for time-triggered messages in a 10Mbit/second system, interframe gap 5 microseconds.	95.8 %	45.7 %
Scalability: Maximum achievable data efficiency for time-triggered messages in a 100Mbit/second system, interframe gap 5 microseconds.	78 %	14.5%
Number of oscillators in a system with 10 ECUs	12	30
First system available on the market	1998	planned 2002
Architecture validated by fault injection	yes	no

Architecture viable for aerospace applications	yes	?
--	-----	---

Table 5: Tabular comparison of TTP/C and FlexRay

7. CONCLUSION

685 TTP/C and FlexRay are both intended for the “X-by-Wire” applications without mechanical backup. In these applications, a total failure of the electronic system has safety implications for the vehicle as a whole. The preceding comparison of FlexRay and TTP/C has revealed that the behavior of FlexRay in the safety critical scenarios of Section 4 is not documented in the public domain, and questions about the safety of FlexRay based systems remain unanswered. In this paper it has also been shown that cost-effective solutions to the failure
690 scenarios introduced in Section 4 do exist for the TTP architecture.

The TTP/C specification has been in the public domain for more than 3 years and has been investigated by many independent scientists. In contrast, documentation about the safety mechanisms of FlexRay is not openly available. We feel that the scientific community working in the area of safety critical computer systems has not only the right, but the *duty* to
695 scrutinize the safety mechanism of protocols that intend to become an international standard for safety-critical mass market applications. It is hoped that the FlexRay consortium will open their safety strategy to international scrutiny and thus contribute to the efforts by the international research community to build safe mass-market electronic systems.

ACKNOWLEDGMENT

700 This paper is a substantial revision of a previous draft paper that has been communicated to a selected audience. I would like to thank for the many constructive comments received, particularly by K.Kim from the University of California and from G. Bauer from the TU Vienna.

705 REFERENCES

- ARINC (1992). Software Considerations in Airborne Systems and Equipment Certification, ARINC, Annapolis, Maryland.
- 710 Bauer, G., T. Frenning, et al. (2000). A centralized approach for avoiding the babbling idiot failure in the time-triggered architecture. Int. Conf. on Dependable Systems and Networks 2000, New York, IEEE Press.
- Bauer, G. and H. Kopetz (2000). Transparent Redundancy in the Time-Triggered Architecture. Dependable Systems and Networks (DSN 2000), New York, IEEE Press.
- 715 Belschner, R., J. Berwanger, et al. (2000). Anforderungen an ein zukünftiges Bussystem for fehlertolerante Anwendungen aus Sicht Kfz-Hersteller. VDI Congress, Baden-Baden Germany.
- Berwanger, J., C. Ebner, et al. (2001). FlexRay--The Communication System for Advanced Automotive Control Systems. SAE World Congress, Detroit, SAE Press paper 2001001-0676.
- BMW (1999). Byteflight homepage www.byteflight.com.
- 720 Deline, R. (1999). Resolving Packaging Mismatch. Computer Science Department. Pittsburgh, Carnegie Mellon University: 178.

- Driscoll, K. and K. Hoyme (1993). "SafeBus for avionics." IEEE Aerospace and Electronics Systems Magazine 8(3): 34-39.
- 725 Karlsson, J., P. Folkesson, et al. (1995). Integration and Comparison of Three Physical Fault Injection Techniques. Predictably Dependable Computing Systems. B. Randell, J. L. Laprie, H. Kopetz and B. Littlewood. Heidelberg, Springer Verlag: 309-327.
- Kopetz, H. (1997). Real-Time Systems, Design Principles for Distributed Embedded Applications; ISBN: 0-7923-9894-7, Third printing 1999. Boston, Kluwer Academic Publishers.
- Kopetz, H., M. Holzmann, et al. (2000). A Universal Smart Transducer Interface: TTP/A. Object-Oriented Real-Time Distributed Computing, Newport Beach, Cal., IEEE Press.
- 730 Ladkin, P., B. (1998). The Ariane 5 Accident: A Programming Problem? www.rvs.uni-bielefeld.de/publications/Incidents, University of Bielefeld.
- Leveson, N. G. (1995). Safeware: System Safety and Computers. Reading, Mass., Addison Wesley Company.
- 735 Littlewood, B. (1997). The use of computers in safety-critical applications. London, UK health and safety commission.
- Perrow, C., (1999). Normal Accidents--Living with high-risk technologies, Princeton University Press.
- Pfeifer, H. (2000). Formal Verification of the TTP Group Membership Algorithm. FORTE/PSTV 2000, Pisa, Italy, Kluwer Academic Publishers.
- 740 Pfeifer, H., D. Schwier, et al. (1999). Formal Verification for Time-Triggered Clock Synchronization. Dependable Computing for Critical Applications 7, IEEE Press.
- Powell, D. (1995). Failure Mode Assumptions and Assumption Coverage. Predictably Dependable Computing Systems. B. Randell, J. C. Laprie, H. Kopetz and B. Littlewood. Berlin, Springer Verlag: 123-140.
- 745 Rushby, J. (2000). Formal Verification of Group Membership for the Time-Triggered Architecture. Menlo Park, Cal, SRI International: 41.
- Saltzer, J., D. P. Reed, et al. (1984). "End-to-End Arguments in System Design." ACM Transactions on Computer Systems 2(4): 277-288.
- Temple, C. (1998). Avoiding the Babbling-Idiot Failure in a Time-Triggered Communication System. Fault Tolerant Comp. Symp. FTCS 28, Munich, Germany, IEEE Press.
- 750 Thurner, T. and Heiner (1998). Time-Triggered Architecture for Safety-Related Distributed Real-Time Systems in Transportation Systems. FTCS 28, IEEE Press.
- WG-48, R.-S. -. E. (1999). Minimal Operational Performance Standards for Avionics Computer Resource. Washington D.C, ARINC RTCSA-SC-182/Eurocae WG-48.