Fault-Tolerant and Fail-Safe Control Systems Using Remote Redundancy

Klaus Echtle, Thorsten Kimmeskamp, University of Duisburg-Essen, Institute for Computer Science and Business Information Systems, 45141 Essen, Germany, (echtle | kimmeskamp)@dc.uni-due.de

Abstract

This paper presents a novel redundancy concept for safety-critical control systems. By using signature-protected communication, it allows connecting each redundant peripheral just to the most proximate control computer while forwarding information to or from any other units (sensors, actuators, further control computers) over a bus system. We will show that wiring harness can thus be reduced drastically with regard to both weight and complexity without compromising fault tolerance characteristics. Moreover, since function and location are decoupled, remote redundancy can be shared between different subsystems if more than one control loop (e. g. brakes and steering) exists in the overall system. Finally, our approach is highly flexible and not at all restricted to a certain degree of fault tolerance, as example systems for both a fault-tolerant and a fail-safe application (steer-by-wire/flap control) will demonstrate.

1 Motivation

Highly safety-critical real-time control systems have to satisfy strong fault tolerance requirements to achieve, depending on the application, fail-safe or fault-tolerant behaviour. The necessary degree of redundancy depends on the fault assumption with respect to a formal fault model, the accepted probability of system failure and the number of simultaneous faults to be tolerated [1]. Typical system designs apply duplex or triplex redundancy for the control computers, the connecting communication channels, the sensors and the actuators [2].

For cost reasons, one often tries to benefit from special safety-related properties in order to reduce redundancy. By withdrawing energy from an actuator, a safe state may be guaranteed, for example, without having a redundant actuator. However, in the general case, a TMR control computer, dual communication channels, TMR sensors, and duplex actuators with passivation units are needed to tolerate a single fault in any device.

Many safety-critical systems are composed of relatively independent subsystems each of which is subject to a different control loop with fault-tolerant control computers. Thus, for the overall system, we obtain a network with quite a quantity of TMR and duplex control computers, which play their individual part according to the wired connections to peripherals [3]. In principle, redundancy can be hardly shared among these control loops, because wires cannot be replaced by field buses without violating the independence of redundant units. Considerable total costs for those redundant units and for the wiring and plugs between computers, sensors and actuators hence seem to be unavoidable.

For a substantial reduction of such costs we have, however, developed a new concept called “remote redundancy”, which is subject of this paper. The basic idea can be sketched as follows: Each peripheral, whether sensor or actuator, is connected to just one control computer, which acts as one of the redundant units. Other redundant units may be implemented as software on a remote computer belonging to a different control loop. Consequently, hardware expenses are cut down with respect to both control computers and wiring of peripherals. In contrast to the traditional approach, the computer hardware redundancy does not exceed the unavoidable degree of redundancy required by the peripherals. In case of two redundant motors, there are only two control computers. Together with a third remotely redundant process, they act as a complete TMR system without compromising fault tolerance characteristics.

To make this approach possible, we have developed a new scheme of fault-resilient signatures. It protects data transmission between remote processes and local sensors or actuators, respectively. Faults in the forwarding computer in between therefore cannot corrupt relayed data undetectably. In case of detectable corruption, however, subsequent error processing, e. g. fault masking, can be performed.

A second “enabler” of our concept is the availability of nowadays fast real-time communication systems, such as FlexRay [4]. The transmission delay between a control computer and a redundant process thus does not cause violation of deadlines for most real-time applications [5]. Moreover, software architectures with sufficient modularity have been developed [6].

In the following section 2, we outline the traditional approach using control computers with dedicated redundancy. We further define a fault assumption, which is also taken as a basis for the new approach with remote redundancy. Section 3 depicts the structure of a fully fault-tolerant system using remote redundancy, whereas section 4 shows two variants for a system with fail-safe behaviour. The new fault-resilient signature scheme is explained in section 5. In section 6, a distinction of fault-cases demonstrates that complete fault tolerance (or fail-safe behaviour, respectively) is achieved. Section 7 summarises our results and presents an outlook on future work.

2 System with Dedicated Redundancy and Fault Model

According to the state of the art, TMR schemes with dedicated redundant units are used to guarantee single fault tolerance. To explain the structure, we depict an example system which could be used for a steer-by-wire application (see figure 1). Two electrical motors M_1 and M_2 operate as redundant actuators, coupled by a non-redundant differential gear D. The triplicated control computers C_1, C_2 and C_3 compute the control function (based on target value transmitted over buses B_1 and B_2 and actual value provided by the redundant position sensors P_1, P_2 and P_3). Error detection using rotation sensors R_1 and R_2 allows for disconnecting a faulty motor by withdrawing electrical energy via the activation units A_{12}, A_{31}, A_{32} and A_{21}.

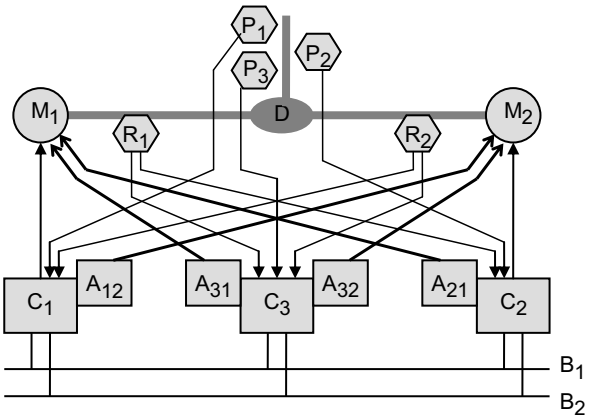

Figure 1 Fault-tolerant system with dedicated redundancy

Component Type	Malfunction in case of a fault
Control computers C	Loss or corruption of information at any time (except undetectable forging of digital signatures, which is very unlikely as a fault)
Communication buses B	Loss or corruption of CRC- or signature-protected messages. Benign Byzantine behaviour [7] in case of multicast messages. Spontaneous generation of a protected message is excluded.
Sensors P and R	Loss or corruption of information at any time. Malicious Byzantine behaviour of sensors connected to more than one control computer.
Activation units A	Spontaneous activation or passivation at any time
Motors M, including power stage and shaft	Blocking or reduced rotation caused by friction. Accelerating torque without supply of electrical energy is excluded.
Differential gear D	Assumed faultless by means of perfection

Table 1 Behaviour of faulty components

As can be seen in figure 1, motor M_1 is controlled by C_1. A fault of C_1 or M_1 is handled by C_2 and C_3. These control computers may jointly switch off M_1's power supply via A_{21} and A_{31}, respectively. The decision on passivation is taken according to the information obtained from rotation sensor R_1, observing the movement of M_1. Unjustified passivation by, say, a faulty C_3 or A_{31} is prevented by C_2, which still supplies M_1 with electrical power via A_{21}. Symmetric cases exist for faults of C_2 or M_2.
The assumed behaviour of faulty components is described in table 1. All single faults are tolerated except for a defective differential gear D. For a complete synopsis of the respective fault regions (i. e. sets of components whose simultaneous faults are tolerated), please refer to section 6.

3 Fault-Tolerant Systems with Remote Redundancy

Having figure 1 in mind, one can easily derive criteria for optimisation. Due to their physical nature, some unavoidable redundancy is needed for sensors, actuators and control computers. In order to cut costs, the complete system should, however, not exceed this minimum redundancy, but use existing resources most efficiently and share them whenever possible. This not only applies to control computers or peripherals themselves, but particularly also to the wiring between control computers and peripherals. Moreover, simple wiring schemes would not only contribute to cost reduction, but also result in an increased reliability (fewer plugs) and probably a better maintainability. The removal of complex wiring structures can finally lead to recombinant components and therefore highly improve the composability of the overall system. In a nutshell, the basic principles behind remote redundancy thus are:

- Decouple function and location of system components. If necessary, forward information through other components.
- Peripherals are only connected to the most proximate control computers, which relay the corresponding information via the redundant communication channels B_1 and B_2 (signature-protection compulsory).
- Software components share control computers whenever safe and possible. Residual computing resources are made available as redundant elements for other control loops.

The result of applying these principles to the aforementioned steer-by-wire application can be seen in figure 2. Most strikingly, C_3 may be any control computer possibly already existing for a different control loop somewhere in the network, only some free computing capacity and sufficient responsiveness, to be guaranteed by an appropriate real-time scheduling algorithm, are required. Physical wiring of C_3 to the peripherals is superfluous. C_3 does of course still need to be logically connected to the sensors P_1, P_2, R_1 and R_2, as well as to the activation units A_1 and

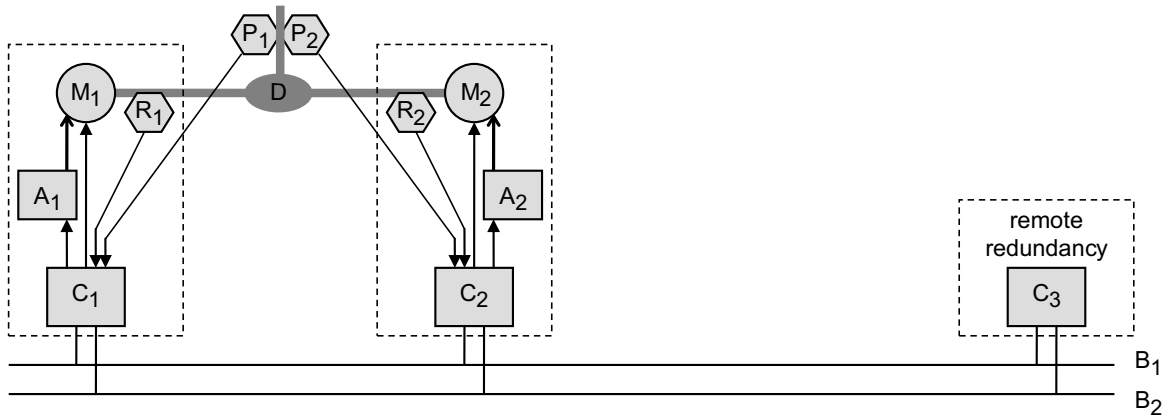

Figure 2 Fault-tolerant system with remote redundancy

A_2 (sensor P_3 can be replaced by a consistency check between all other sensors). However, these links are not implemented by direct wires. Instead, all relevant data is forwarded on the double bus system via control computers C_1 or C_2, respectively. Any fault-induced modification is revealed by the signature scheme presented in section 5. Forwarded sensor values or activation signals thus cannot undetectably be forged or generated. Activation units A_1 and A_2 (remark: A_{31} and A_{21} are combined into a single unit A_1, A_{32} and A_{12} are replaced symmetrically by A_2) interpret the loss of any activation signals due to an error of C_1 or C_2 as an implicit passivation command, which always preserves the functionality of the system as a whole, as will be pointed out in section 6 in more detail.

A short example shall be given in advance: Assume C_1 to be faulty. It may direct M_1 in wrong rotation sense and, moreover, cut C_2 and C_3 from sensors P_1 and R_1. However, since the information flow from P_1 and R_1 to C_2 and C_3 is signature-protected, C_1 cannot generate wrong sensor data. Both C_2 and C_3 thus decide for passivation via A_1 (by omitting periodic activation signals which cause A_1 to provide M_1 with electrical energy). Motor M_2 is unaffected and continues its operation, possibly controlled with double speed by C_2 to compensate the loss of M_1.

Please note that omitting one position sensor and two activation units as mentioned above is also possible with dedicated redundancy. Only remote redundancy, however, reduces both the complexity and the amount of wiring and (if shared between different control loops) the number of necessary control computers in the overall system.

4 Fail-Safe Systems with Remote Redundancy

The principle of remote redundancy is not at all restricted to a certain degree of fault tolerance. In fact, decoupling location and function makes it possible to flexibly mix or amend different degrees of redundancy.

As an example, we have developed the structure for a fail-safe system which could for instance be used for a flap-control application in aircrafts. On error detection, the flaps are fixed in the current position. The resulting system-state is safe, provided that a sufficiently long runway for landing is available.

A first variant (see figure 3) could feature a single bus B_1 (which is sufficient, as fail-operational behaviour is not required), a single motor M_1 with an activation unit A_1, two redundant position sensors P_1 and P_2 and a control computer C_1. Using remote redundancy, C_1 forms a duplex unit together with C_2, which may already exist for a completely different control loop. The system is passivated via A_1 in case of any deviation between C_1 and C_2 or the two sensors P_1 and P_2. Using signature protection, C_2 is connected only to the bus system, thus radically reducing the wiring overhead in comparison to a solution with dedicated redundancy (not depicted here).

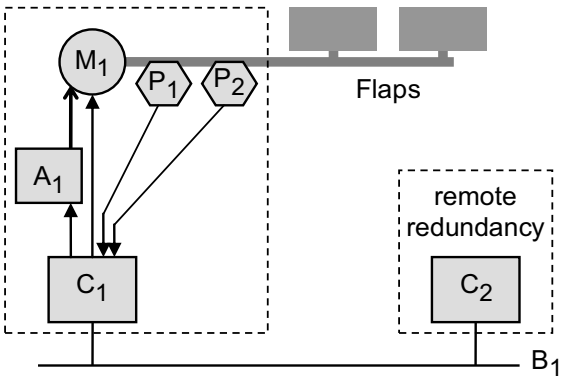

Figure 3 Fail-safe system with remote redundancy, first variant

In a second variant (cf. figure 4), two nodes control individual motors each for the flap of one wing (without a mechanical connection between them). The two control computers C_1 and C_2 need to keep the flaps in identical positions to preserve the symmetry of the aircraft. Therefore, on error detection, both flaps are fixed in the current position. In this variant, the second characteristic principle of remote redundancy becomes evident: although the two control computers C_1 and C_2 serve as mutual redundancy for each other, no crosswise wiring is needed (only the single bus B_1 is required). Thus, wiring harness und weight (important in avionics) can be reduced substantially.

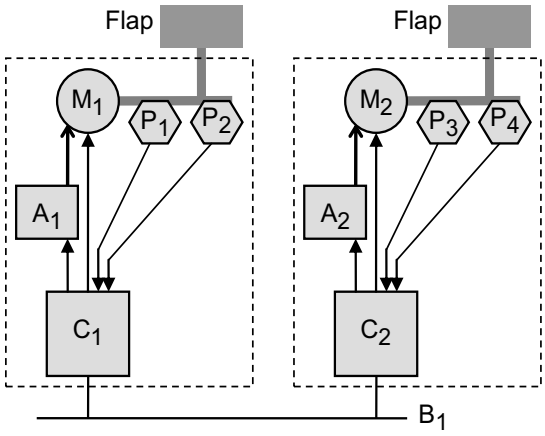

Figure 4 Fail-safe system with remote redundancy, second variant

5 Protection by Fault-Resilient Signatures

The concept of remote redundancy highly depends on reliably detecting modifications of any information forwarded on the bus system, either sent from a sensor via one control computer to another control computer, or in the opposite direction towards an actuator. Protecting the information flow just by CRCs might be sufficient for more benign assumptions on malfunctions. However, since a forwarding node might generate a CRC which is consistent to wrong information, additional countermeasures are required, which exhibit exactly the properties of digital signatures. Sensors thus sign their values (see figure 5) together with a sequence number by using a private key. The destination node checks the received information by using the corresponding public key. In the reverse direction, an actuator has to do the signature check (cf. figure 6).

A potential problem of signatures lies in the high computation overhead associated with them. These computations must be performed in the semiconductor device of the sensor or actuator, respectively. Yet, in contrast to an intentional attack, faults are considerably less malicious. For this reason, even cryptographically weak signatures are sufficient [8, 9].

We have developed a signature technique which is appropriate for systems with remote redundancy, based on arithmetics modulo m. Typically, m is a power of two, like 2^{16} for 16-bit arithmetics for example. For generating a key, two integer numbers $a \in [m/8, m-1]$ and $b \in [m/8, m-1]$ are chosen randomly and the product $c = a \cdot b$ is calculated (modulo m). Factor a is kept secret for signing, b and c are public for signature checking. Knowing b and c, the secret factor a cannot be derived easily, because $a = c / b$ does not work modulo m. Euklid's algorithm doesn't help either if m is not prime. Instead, a somewhat extended version of Euklid's algorithm would be needed (It is extremely unlikely that a fault "generates" such an algorithm by chance.). Data d and sequence number n are signed by a signature function σ as follows ("multiplication with the secret factor"):

$$\sigma(n, d) := \mathrm{CRC}(\mathrm{concat}(n, d)) \cdot a \pmod m,$$

where CRC is an ordinary ld(m)-bits-CRC-function applied to the concatenation of n and d. Signature checking is done by the boolean signature test function τ for sequence number n, received data d and received signature s:

$$\tau(n, d, s) := (s \cdot b = \mathrm{CRC}(\mathrm{concat}(n, d)) \cdot c) \pmod m.$$

Test function τ is correct, because substitutions can be made according to key generation $c = a \cdot b$ and signature generation $s = \sigma(n, d) = \mathrm{CRC}(\mathrm{concat}(n, d)) \cdot a$. Consequently, we obtain the following equality for $\tau(n, d, s)$: $\mathrm{CRC}(\mathrm{concat}(n, d)) \cdot a \cdot b = \mathrm{CRC}(\mathrm{concat}(n, d)) \cdot a \cdot b$.

The computation overhead of this scheme is rather low. Special hardware associated with a sensor only needs to calculate an ordinary CRC and perform one multiplication, where the overflow is cut. An actuator's hardware needs just one more multiplication.

The usage of a sequence number n is essential in the signature scheme, because otherwise out-dated (instead of current) data could be forwarded by a faulty node. Through using n, the receiver only accepts recent data. In order to avoid the danger of wrongly accepting data from a previous round, we use a 36 bit counter, delivering a different number every millisecond over a duration of more than two years.

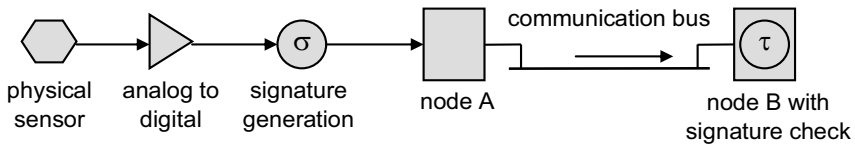

Figure 5 Signature protecting the data of a sensor

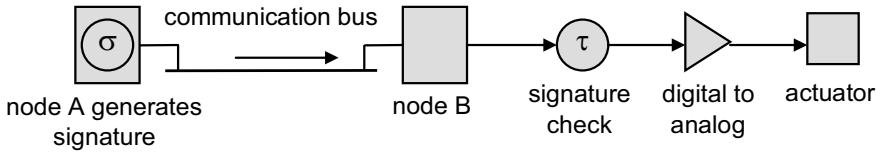

Figure 6 Signature protecting the data for an actuator

In principle, the sequence number should be transmitted with each message in clear text. Otherwise it could become difficult to reconstruct the context, once it is lost. On the other hand, the overhead of transmitting additional 36 bits would be considerable for payload values consisting of much less than 36 bits. We reduce such overhead by subdividing the 36 bit sequence number n into 9 blocks n_8, n_7, n_6, n_5, n_4, n_3, n_2, n_1, n_0 of 4 bits each. The least significant block n_0 is always transmitted with data d. In addition we transmit block n_i where $i = n_0 \bmod 8$ (see figure 7).

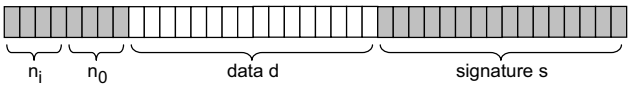

Figure 7 Example of a message format with partial sequence number and signature

When using this approach, n_0 provides a fine-grained number that enables the receiver to detect message losses, duplications etc. Moreover, after having received 8 subsequent messages (initially or after a blackout phase), a node possesses the full context information n. This number is stored locally and is incremented after every 16 receive operations. Thus, all received partial sequence numbers can be completed correctly.

6 Verification of Complete Fault Tolerance

In recent studies, the systems presented in sections 3 and 4 have been thoroughly analysed with respect to their fault tolerance characteristics: For each system, the set of components used has been subdivided into fault regions. After this, it was examined whether or not a fault of any single fault region is tolerated (this means, that all components within the faulty region simultaneously exhibit an arbitrary behaviour out of the set of malfunctions specified in section 2. The named systems are composed of the following fault regions:

- Fault-tolerant system with dedicated redundancy (for comparison only, see figure 1): $\{C_1, P_1, A_{12}, M_1\}$, $\{C_2, P_2, A_{21}, M_2\}$, $\{C_3, P_3, A_{31}, A_{32}\}$, $\{R_1\}$, $\{R_2\}$, $\{B_1\}$, $\{B_2\}$
- Fault-tolerant system with remote redundancy (see figure 2): $\{C_1, P_1, M_1\}$, $\{C_2, P_2, M_2\}$, $\{C_3\}$, $\{A_1, R_1\}$, $\{A_2, R_2\}$, $\{B_1\}$, $\{B_2\}$
- Fail-safe system with remote redundancy, first variant (see figure 3): $\{C_1, P_1, M_1\}$, $\{P_2, A_1\}$, $\{C_2\}$, $\{B_1\}$
- Fail-safe system with remote redundancy, second variant (see figure 4): $\{C_1, P_1, P_3, A_2, M_1\}$, $\{C_2, P_2, P_4, A_1, M_2\}$, $\{B_1\}$

For the fault-tolerant system with remote redundancy, three examples of fault tolerance analysis are conducted in short words below:

- Region $\{C_1, P_1, M_1\}$ assumed to be faulty: C_1 could direct M_1 to rotate in wrong sense of direction. Fault-tolerant reaction of C_2 and C_3: Plausibility-check of actual rotation (R_1) against difference of target position value (obtained via B_1 and B_2) and actual position value (P_2). Conjoint abortion of activation messages from C_2 and C_3 to A_1, which leads to passivation of M_1. Motor M_2 continues operation.
- Region $\{C_1, P_1, M_1\}$ assumed to be faulty: Value provided by P_1 could be wrong: Fault-tolerant reaction of C_2 and C_3: Comparison of P_1 and P_2 against the time integral of rotation sensors R_1 and R_2. In case of a deviation, C_2 and C_3 will detect consistency for P_2 only, which means they ignore P_1.
- Region $\{C_1, P_1, M_1\}$ assumed to be faulty: Periodic activation signals from C_1 to A_2 could be dropped. Fault-tolerant reaction of C_3 and A_2: Since C_3 still provides A_2 with activation signals, M_2 continues operation.
- Region $\{A_1\}$ assumed to be faulty: M_1 could be passivated unjustifiedly. Fault-tolerant reaction of C_1, C_2 and C_3: Motor M_2 is controlled to continue its operation with double speed to compensate the loss of M_1.

Please note that a simultaneous occurrence of all three faults mentioned for region $\{C_1, P_1, M_1\}$ is also tolerated because C_2 and C_3 will jointly passivate M_1 via A_1 but C_1 cannot passivate M_2 by himself without C_3. We have proven complete single fault tolerance for all considered systems by similar reasoning. The concept of remote redundancy thus evidentially permits a substantial reduction of the required structural hardware redundancy without a loss in fault tolerance.

By assuming a stochastic model, one can furthermore calculate the availability of a system or the probability of system-survival in a given time interval. A comprehensive analysis of such figures has been carried out for systems using dedicated redundancy and for systems using remote redundancy in cooperation with Max Walter from Technical University of Munich. The quantitative results obtained show that, though dedicated redundancy performs slightly better as the number of redundant components is higher, the reliability of both systems using dedicated redundancy and systems using remote redundancy is at large in the same order of magnitude. Detailed results are subject of a separate paper [10].

7 Conclusion and Future Work

The principle of remote redundancy locates redundant software processes to already existing control computers with free capacity. Additionally, communication between control computers and peripherals typically realised by wires is in part replaced by bus communication with digital signatures. As shown in our paper, remote redundancy thus allows for a significant cost reduction for redundant hardware without compromising fault tolerance properties. Particularly with regard to the wiring outlay, the system structures emerging when applying remote redundancy are considerably less complex than commonly usual and highly support the composability of the overall system. The additional delays for communication to remote processes can be neglected, if a sufficiently fast communication system is used (nowadays low-cost bus systems allow for periodic communication with cycle durations below 1 ms). A signature scheme sufficiently powerful to reveal fault-induced modifications of data, yet simple enough to be implemented with very low overhead has been developed. It has been furthermore shown that remote redundancy is not restricted to a certain degree of fault tolerance, but can be used in a variety of applications in the domain of engineering. Systems for both fault-tolerant and fail-safe applications have been presented as examples.

Further work will concentrate on the generalisation of the principle of remote redundancy, including formal models. Simulation models and prototypical implementations with fault injection capabilities shall supplementary serve as a proof of concept. We will also refine and optimise the concept by dealing with specific properties of particular peripherals. Moreover, real-time software architectures will be evaluated concerning their suitability for applying remote redundancy with respect to both encapsulation of processes and scheduling mechanisms. As far as the feasibility of remote redundancy is concerned, surveys of potential users are planned in order to gain insight into non-technical aspects of particular importance. In specific, economic and legal conditions seem to be an interesting object of investigation.

8 References

[1] International Electrotechnical Commission (IEC): Functional safety and IEC 61508, see also http://www.iec.ch/zone/fsafety/pdf_safe/hld.pdf, date of retrieval: 2008-11-19

[2] Baleani, M.; Ferrari, A., Mangeruca, L.; Sangiovanni-Vincentelli, L.; Peri M. and Pezzini, S.: Fault-Tolerant Platforms for Automotive Safety-Critical Applications. International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), Conf. Proc., San Jose, 2003, pp. 170 – 177

[3] Electronic Architecture and System Engineering for Integrated Safety Systems (EASIS): Discussions and findings on fault tolerance, http://www.easis-online.org/wEnglish/download/Deliverables/EASIS_Deliverable_D1.2-5_V1.0.pdf, date of retrieval: 2008-11-19

[4] Rausch, M.: FlexRay. Hanser Publishing House, Munich, 2007

[5] Echtle, K.; Jochim, M.; Tappe, D.: Sicherheit und Fehlertoleranz – Zusammenspiel sicherheitsrelevanter Software und fehlertoleranter Datenbusse. Automotive Elektronik, 3 / 2004, pp. 44 – 48

[6] Automotive Open System Architecture (AUTOSAR) http://www.autosar.org, date of retrieval: 2008-11-19

[7] Echtle K.; Masum A.: A Fundamental Failure Model for Fault-Tolerant Protocols. IEEE International Computer Performance and Dependability Symposium (IPDS2K), Conf. Proc., Chicago, 2000, pp. 69 – 78

[8] Echtle, K.: Avoiding Malicious Byzantine Faults by a New Signature Generation Technique. European Dependable Computing Conference (EDCC), Conf. Proc., Lecture Notes in Computer Science 1667, Springer-Verlag, Heidelberg, 1999, pp. 106 – 123

[9] Leu M.: Relative Signatures for Fault Tolerance and their Implementation. European Dependable Computing Conference (EDCC), Conf. Proc., Lecture Notes in Computer Science 852, Springer-Verlag, Heidelberg, 1994, pp. 563 – 580

[10] Echtle, K.; Kimmeskamp, T.; Jacquet, S.; Malassé, O.; Pock, M.; Walter, M.: Reliability Analysis of a Control System Built Using Remote Redundancy. Advances in Risk and Reliability Technology Symposium (AR2TS), Conf. Proc., Loughborough, 2009