Design of Bus Media Redundancy in CAN

José Rufino[1], Paulo Veríssimo[2], and Guilherme Arroz[1]

[1] Instituto Superior Técnico - Universidade Técnica de Lisboa,
Avenida Rovisco Pais, 1049-001 Lisboa, Portugal.
NavIST Group CAN Page - http://pandora.ist.utl.pt/CAN
[2] Faculdade de Ciências da Universidade de Lisboa,
Campo Grande - Bloco C5, 1700 Lisboa, Portugal.
Navigators Home Page - http://www.navigators.di.fc.ul.pt

Abstract. Network media redundancy is a clean and effective way of achieving high levels of reliability against temporary medium faults and availability in the presence of permanent faults. This is specially true of critical control applications such as those supported by the Controller Area Network (CAN). This paper deals with the complexity of implementing bus media redundancy in CAN, discussing how this can be done through a simple scheme that uses inexpensive off-the-shelf components.

1 Introduction

Continuity of service and determinism in message transmission delays are two fundamental requirements of fault-tolerant real-time applications, that must be fulfilled through the use of some form of network-level redundancy.
Safety-critical applications would resort to full space redundancy, replicating media and network controllers to provide a broad coverage of faults and glitch-free communication [1, 3], usually at high design and implementation costs. Other approach is simple media redundancy, such as exists off-the-shelf in some standard LANs or as developed in Delta-4 [7, 13], where space redundancy is restricted to the physical - electrical signaling in the medium - level. This alternative design may lead to simpler and thus less expensive solutions, yet satisfying a wide spectrum of fault-tolerant real-time applications, with exception of those with very stringent safety and timeliness requirements [14].
Fieldbuses are in essence a technology whose area of application requires continuity of service. Fieldbuses are widely used in systems intended for real-world interfacing (i.e. integrating sensors and/or actuators) which are specially sensitive to the availability of the network infrastructure, a problem that we address in this paper, in the context of CAN, the Controller Area Network [2]. CAN is a low-cost fieldbus, with widespread acceptance in control applications, able to extremely robust operation. However, resilience to medium partitioning has to be provided as an extension to the standard specification.
Arguing with the high costs of other solutions, an existing commercial redundant CAN design uses a self-healing ring/bus architecture [4]. Though such a scheme provides resilience to open/short-circuits in the physical wiring, it does not solve

the problem of CAN continuity of service efficiently: ring reconfiguration takes time (it can last as long as $100ms$) and meanwhile the network is partitioned. Bus media redundancy does not exhibit those shortcomings and do represent a natural design choice, but efficient mechanisms for its implementation were not known, until recently.
In [11], we have presented a *Columbus' egg* idea that opens room for a simple implementation of bus media redundancy in CAN. However, to be of practical use such scheme should be enhanced with mechanisms handling not only medium partitions but also stuck-at and omission failures, as discussed in [11]. The present paper addresses how those mechanisms can be efficiently implemented using off-the-shelf components, discussing how CAN properties may be exploited in the design of the different media redundancy management modules in order to keep complexity low, and explaining how these modules interface with the standard CAN components.
This work is part of a broader effort aiming at designing an embedded fault-tolerant distributed system around CAN [15, 12, 8]. An overview of the results achieved so far can be found in [10].

2 CAN Physical Layer Fault-Tolerance

Some commercial CAN *fault-tolerant transceivers* (e.g. [6]) have been designed to provide resilience to one of the following failures in the CAN communication medium: one-wire interruption; one-wire short-circuit to power or ground; two-wire short-circuit. Continuity of network operation is achieved by switching from the normal two-wire differential operation to a single-wire mode, at the expense of a reduced signal-to-noise ratio. Switching upon the detection of a failure and switching back to the differential mode when recovered, is automatic.
However, no standard mechanism supports CAN non-stop operation if both wires of the network cabling get simultaneously interrupted.

3 Bus Media Redundancy in CAN

In [11], we have thoroughly addressed CAN resilience to medium partitioning and, dismissing existing cost and complexity arguments, we have came up with an extremely simple scheme for the implementation of bus-based media redundancy in CAN. Next, we shortly discuss that *Columbus' egg* idea.
Let us assume a simplified fault model, considering only medium partitioning. Let us also assume every bit issued from a MAC[1] sub-layer is simultaneously transmitted on all redundant media interfaces. If a medium is partitioned: nodes located at the *in-partition*[2] receive a correct signal on all redundant media interfaces; nodes at the *out-partition* receive a recessive[3] signal from the (idle)

[1] Medium Access Control.
[2] I.e., the partition that includes the transmitter.
[3] A recessive symbol is represented by a logical one level.

failed medium. Given the *quasi-stationary* operation of CAN and its wired-AND nature, a correct *channel*[4] receive signal can be produced simply combining the redundant media receive signals in a logical AND function, as exemplified in Fig. 1 for a dual-media architecture.

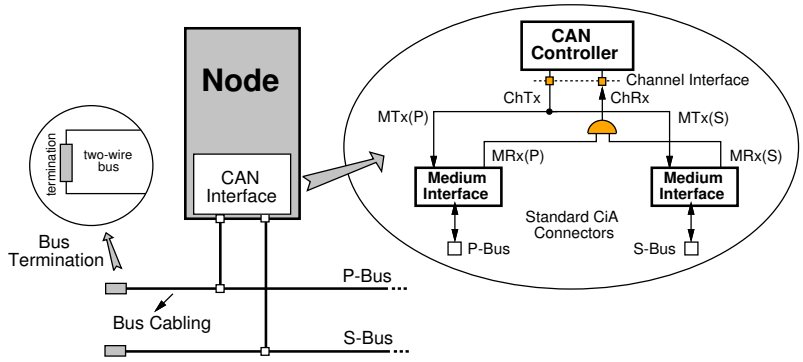

Fig. 1. A *Columbus' egg* idea for bus media redundancy in CAN

The only disadvantage of this approach is too restrictive a fault model. Only medium partitions and stuck-at-recessive faults are tolerated. However, this basic architecture can be enhanced to support a less restrictive and thus more realistic fault model [11]: stuck-at-dominant faults can be handled through a special-purpose *watchdog* timer; omission faults can be monitored and a medium exhibiting an excessive number of omission errors can be put in *quarantine* until (and if) it "behaves well" again.

4 Design of Bus Media Redundancy in CAN

This paper addresses how to support non-stop network operation, through bus-based media redundancy, in CAN. A *media selection and management unity*, to be inserted between the redundant media interfaces and the CAN controller (Fig. 2), is intended to provide resilience to crash (stuck-at/broken) and omission failures, through the mechanisms specified in [11].

Our design does not assume the use of any particular CAN controller, and multiple solutions are allowed for the physical layer: inexpensive two-wire differential cabling used together with classical (non fault-tolerant) transceivers [5] or fiber optics technology [9]. That means, apart from replication, commercial CAN components are used. All provisions for media replication management are made as extensions to the standard.

[4] We call *channel* to the physical path - cable medium and transceivers - used by MAC entities to communicate.

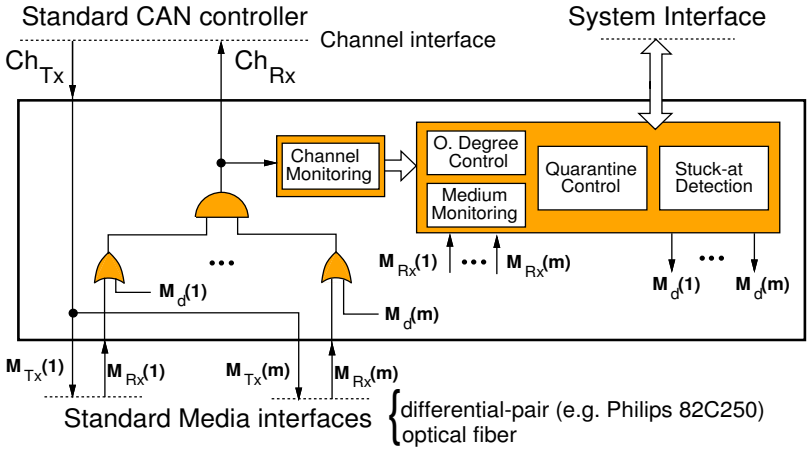

Fig. 2. CAN media selection and management unity

The architecture of the *CAN media selection and management unity* is depicted in Fig. 2. The central component of this unity is a conventional AND gate that, as defined by our *Columbus' egg* strategy, combines the values of the different media receive signals[5] to generate the channel receive signal (Ch_{Rx}), to be delivered at the CAN controller interface.

The functionality provided by this simple mechanism is the required to ensure resilience to *medium partition* and *stuck-at-recessive* medium failures, because the recessive signal of a failed medium does not disturb the result of the AND function. Nevertheless, it is relevant to detect any bus idle (recessive state) period with an abnormal duration. Suck kind of indication is of fundamental importance to the design of high-level diagnose applications aiming: to distinguish a stuck-at-recessive from a medium partition failure; to pinpoint the location of the medium partition failure in the network cable.

The *stuck-at detection* module of Fig. 2 has been designed in order to provide a low-level support for those functions, in addition to the detection of *stuck-at-dominant* medium failures. The occurrence of *stuck-at-dominant* medium failures must have a special treatment in our architecture, because an erroneous dominant value would otherwise propagate from a failed medium receiver (one of the M_{Rx} signals in Fig. 2) to the channel interface (Ch_{Rx} signal).

The contribution of a medium receive signal to the AND function is allowed to be disabled, whenever such signal assumes a dominant value that lasts beyond what it may be expected, in the worst-case, from the normal operation of the CAN protocol [2]. The disabling of a given $M_{Rx}(m)$ signal can be achieved by combining that signal with a medium disable indication[6] in a simple OR gate,

[5] We use $M_{Tx}(m)$ and $M_{Rx}(m)$ to denote the transmitter and receiver interfaces of medium $m \in \mathcal{M}$, the set of all media interfaces.

[6] In Fig. 2, we use $M_d(m)$ to denote the medium m disable indication.

as shown in Fig. 2. This simple method secures resilience to *stuck-at-dominant* medium failures.
The design of the *stuck-at detection* module (Fig. 2) can be optimized in several ways: the properties of the CAN protocol allow the error detection thresholds to be defined independently of network configuration parameters; stuck-at recessive and dominant failures are mutually exclusive events, and this can be exploited to reduce the complexity of the module circuitry.
The media redundancy strategies in [11] define specific mechanisms able to detect and account for frame omission errors. Short-term frame omissions usually have its origin in noise sources, that corrupt frames by electromagnetic interference. Other omission error patterns, with origin in more subtle causes such as a defective connector mount or a faulty medium interface circuit, may affect network operation during longer periods.
The *omission degree control*[7] module of Fig. 2 accounts for the frame consecutive omission errors in data/remote frame transmissions, at each medium. Network errors occurring in the interframe space are not accounted as frame omissions. The *omission degree control* module uses the indications provided by the *channel monitoring* and the *media monitoring* modules (Fig. 2), to take a decision on whether the *omission degree* of a given medium (m) should:

- be incremented by one unit, due to the detection of a frame omission with origin in that particular medium;
- maintain the same value, because the origin of the error cannot be assigned to any particular medium;
- set to zero, upon a frame has been successfully transfered in the channel, without any error being detected in medium m.

A medium exceeding a given *omission degree* bound should be considered failed. In addition, special precautions should be taken upon the occurrence of a frame omission error, whose origin cannot be localized. Usually, such kind of errors are due to single medium faults that nevertheless generate omission errors in all media. These scenarios call for fault treatment procedures where the set of "incorrect" media are temporarily disabled (*quarantined*), to allow operation to proceed with the "correct" set. A detailed discussion of CAN-oriented *quarantine* techniques, similar to those introduced in [13] for LANs, to be supported by the *quarantine control* module of Fig. 2, is out of the scope of this paper and it will be reported in a future work.

5 Conclusions

There is a demand for fault-tolerant and real-time distributed systems based on fieldbuses. Many of these systems are intended for critical control applications, where continuity of service is a strict requirement. Network media redundancy is

[7] Informally, the *omission degree* of a component is the number of consecutive omission errors in an interval of reference.

a clean and effective way of achieving high levels of reliability against temporary medium faults and availability in the presence of permanent faults.
In a previous work [11], we have discussed a strategy for the implementation of bus media redundancy in CAN, based on an extremely simple idea. This paper is concerned with the design and implementation of a *media selection and management unity* based on that strategy, discussing how the media redundancy mechanisms of [11] can be structured in a modular and simple architecture.
These implementation issues are extremely important with regard the integration of the *media selection and management unity* in a single, inexpensive, medium capacity Programmable Logic Device.

References

1. F. Cristian, R. Dancey, and J. Dehn. High availability in the Advanced Automation System. In *Digest of Papers, The 20th International Symposium on Fault-Tolerant Computing*, Newcastle-UK, June 1990. IEEE.
2. ISO. *International Std. 11898 - Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication*, November 1993.
3. H. Kopetz and G. Grunsteidl. TTP - a protocol for fault-tolerant real-time systems. *IEEE Computer*, 27(1):14–23, January 1994.
4. RED-CAN a fully redundant CAN-system. NOB Elektronik Product Note, Sweden. http://www.nob.se.
5. Philips Semiconductors. *PCA82C250 - CAN Controller Interface*, April 1994.
6. Philips Semiconductors. *TJA1053 - Fault-tolerant CAN transceiver*, October 1997.
7. D. Powell, editor. *Delta-4 - A Generic Architecture for Dependable Distributed Computing*. ESPRIT Research Reports. Springer Verlag, November 1991.
8. L. Rodrigues, M. Guimarães, and J. Rufino. Fault-tolerant clock syncronization in CAN. In *Proceedings of the 19th Real-Time Systems Symposium*, pages 420–429, Madrid, Spain, December 1998. IEEE.
9. M. Rucks. Optical layer for CAN. In *Proceedings of the 1st International CAN Conference*, pages 2.11–2.18, Mainz, Germany, September 1994. CiA.
10. J. Rufino, P. Veríssimo, and G. Arroz. Defining a CAN-based infrastructure for faul-tolerant real-time distributed computing. In *Proceedings of the 19th Real-Time Systems Symposium*, Work In Progress, pages 27–30, Madrid, Spain, December 1998. IEEE. http://www.cse.unl.edu/rtss98wip/proceedings.
11. J. Rufino, P. Veríssimo, and G. Arroz. A Columbus' egg idea for CAN media redundancy. In *Digest of Papers, The 29th International Symposium on Fault-Tolerant Computing Systems*, pages 286–293, Madison - USA, June 1999. IEEE.
12. J. Rufino, P. Veríssimo, G. Arroz, C. Almeida, and L. Rodrigues. Fault-tolerant broadcasts in CAN. In *Digest of Papers, The 28th Int. Symposium on Fault-Tolerant Computing Systems*, pages 150–159, Munich, Germany, June 1998. IEEE.
13. P. Veríssimo. Redundant media mechanisms for dependable communication in Token-Bus LANs. In *Proceedings of the 13th Local Computer Network Conference*, Minneapolis-USA, October 1988. IEEE.
14. P. Veríssimo. Real-time Communication. In S.J. Mullender, editor, *Distributed Systems*, ACM-Press, chapter 17. Addison-Wesley, 2nd edition, 1993.
15. P. Veríssimo, J. Rufino, and L. Ming. How hard is hard real-time communication on field-buses? In *Digest of Papers, The 27th International Symposium on Fault-Tolerant Computing Systems*, pages 112–121, Seattle - USA, June 1997. IEEE.