Centro de Sistemas Telemáticos e Computacionais

Instituto Superior Técnico

NavIST Group
Fault-Tolerant Real-Time Distributed
Systems and Industrial Automation

CAN Bus Media Redundancy

Published in:1st. CAN in Space Workshop

José Rufino, *et al.*

December 2002

Centro de Sistemas Telemáticos e Computacionais - NavIST Group
Fault-Tolerant Real-Time Distributed Systems and Industrial Automation

CAN Bus Media Redundancy

Invited paper presented at the European Space Agency (ESA)
ESTEC - Spacecraft Control and Data Systems Division
1st. CAN in Space Workshop, 5-6 December 2002
Noordwijk, The Netherlands.

Published in: 1st. CAN in Space Workshop
Authors: J. Rufino, P. Veríssimo, G. Arroz
Date: December 2002

Avenida Rovisco Pais, 1049-001 Lisboa - PORTUGAL, Tel: +351-218418397/99, Fax: +351-218417499.
NavIST WWW Page URL - http://pandora.ist.utl.pt.

CAN Bus Media Redundancy

José Rufino
ruf@digitais.ist.utl.pt
IST-UTL*

Paulo Veríssimo
pjv@di.fc.ul.pt
FC/UL†

Guilherme Arroz
egsa@alfa.ist.utl.pt
IST-UTL

Abstract

The Controller Area Network (CAN) is a fieldbus technology that is increasingly being used in critical control applications in areas as diverse as shop-floor control, robotics, automotive, avionics and aerospace. This kind of applications are specially sensitive to the availability of the network infrastructure. 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. Addressing the problem comprehensively, we ended-up devising a scheme for the implementation of bus media redundancy in CAN which is extraordinarily simpler than previous approaches known for CAN or other LANs and fieldbuses. In addition, we study how media and full-space redundancy can be combined in the same network infrastructure.

1 Introduction

Continuity of service and determinism in message transmission delays are two fundamental requirements of fault-tolerant real-time applications. Though reliable real-time protocols, such as those described in [25], can provide such guarantees in the presence of sporadic transient faults, they are helpless when faced with aggressive omission failure bursts or even permanent failure of the medium. There is no solution but using some form of space redundancy.

Safety-critical applications would resort to full space-redundant network architectures, replicating media and attachment controllers, providing a broad coverage of faults and glitch-free communication [3, 8], at a traditionally high design and implementation cost. An alternative approach is simple media redundancy, such as it exists off-the-shelf in some standard LANs, or as developed in Delta-4 [17]. In these architectures, space redundancy is restricted to the physical – electrical signaling at the medium – level, which may lead to simpler and thus less expensive solutions. Using the appropriate design techniques, the timeliness, reliability and accessibility guarantees achieved, satisfy a wide spectrum of fault-tolerant real-time applications, with exception of those with very stringent safety and timeliness requirements [28].

In any case, cost-effectiveness and shorter design cycles, are strong arguments in favor of using off-the-shelf LAN and fieldbus technologies in the design of fault-tolerant real-time distributed systems. Fieldbuses are in essence a technology whose area of application requires continuity of service, since they are widely used to convey information from and to the boundaries of the system: the sensors and the actuators. Systems intended for real-world interfacing are specially sensitive to the availability of the network infrastructure.

The CAN fieldbus is traditionally viewed as a robust network. Together with protocol level extensive error checking capabilities, the use of differential two-wire communication medium

* Instituto Superior Técnico - Universidade Técnica de Lisboa, Avenida Rovisco Pais, 1049-001 Lisboa Codex - Portugal. Tel: +351-218418397 - Fax: +351-218417499. NavIST Group CAN WWW Page - http://pandora.ist.utl.pt/CAN.

† Faculdade de Ciências da Universidade de Lisboa, Portugal. Navigators Home Page: http://www.navigators.di.fc.ul.pt.

and the use of physical level fault-tolerant mechanisms allows CAN to operate in the presence of one-wire failures in the network cabling [7]. However, these standard fault-tolerant mechanisms are helpless in the provision of CAN non-stop operation in harsher conditions, such as the simultaneous interruption of both wires in the network cabling.

CAN-based redundant architectures using replicated buses have been identified as being too costly, when compared with alternative designs based on ring topologies [14]. An existing commercial redundant CAN solution implements a self-healing ring/bus architecture [14], but does not solve the problem of CAN continuity of service efficiently: ring reconfiguration takes time and meanwhile the network is partitioned.

A systemic analysis of how bus redundancy mechanisms can be implemented in CAN was required. Initially, we tried the adaptation of techniques that have been developed with success for LANs [27, 9]. Unexpectedly, we discovered that these techniques would become extremely complex to apply in the setting of CAN. Moreover, in this process we ended-up with a Columbus' egg idea: an extremely simple mechanism that makes bus-based redundancy easy to implement in CAN using off-the-shelf components. In addition, we study how media and full space-redundancy can be combined in the same network infrastructure.

The following discussion assumes the reader to be fairly familiar with CAN operation. In any case, we forward the reader to the relevant standard documents [18, 7, 1], for details about the CAN protocol.

2 CAN Physical Level Fault-Tolerance

The CAN transmission medium is usually a two-wire differential line. The CAN physical layer specified in [7] allows resilience against some of the transmission medium failures illustrated in Figure 1, by switching from the normal two-wire differential operation to a single-wire mode. After mode switch-over bus operation is allowed to proceed, though with a reduced signal-to-noise ratio, in the presence of one of the following failures:

- one-wire interruption (A or B failures, in Figure 1);
- one-wire short-circuit either to power (C or D) or ground (E or F);
- two-wire short-circuit (G).

CAN medium interfaces that automatically switch to single-wire operation upon the detection of any of these failures and switch back to differential mode when recovered, are commercially available. Usually, such devices are intended for low-speed applications (up to 125 kbps) with no more than 32 nodes [16].

The CAN bus line is usually terminated at both ends by its characteristic impedance [1]. Resilience to the failure of one termination (H failure, in Figure 1) can be achieved simply by taking into account the extra time needed for bus signal stabilization, when dimensioning the CAN *bus timing parameters* [7, 5].

In any case, no standardized mechanisms exist to provide resilience to the simultaneous interruption of both bus line wires (A and B failures, in Figure 1). Upon such a failure, there may be subsets of the nodes which cannot communicate with each other. Because damaging of all wires in a bus line may result from single incidents with the network cabling, the probability of its occurrence is not negligible.

The provision of resilience to CAN physical partitioning is thus a requirement of dependable real-time systems that is not fulfilled by the standard CAN specification. A solution to this

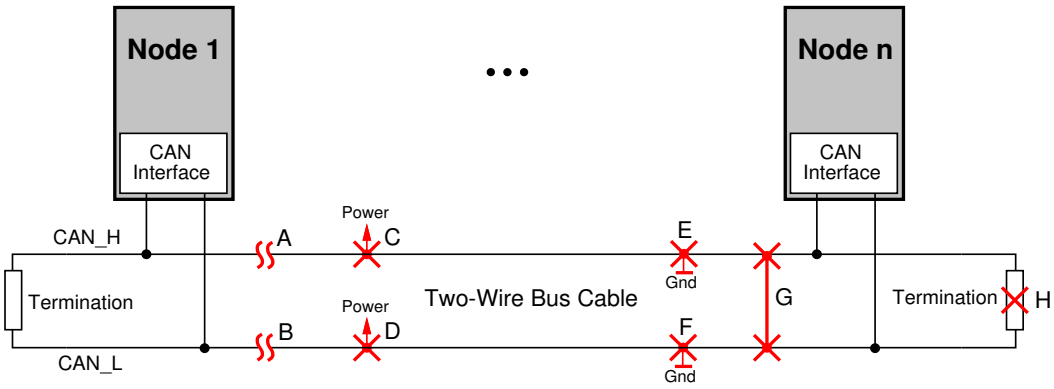

Figure 1: Resilience to medium failures in the ISO 11898 CAN standard

problem must resort to some form of network redundancy: media redundancy allows to achieve high levels of availability in the presence of permanent medium faults; full space-redundancy provides glitch-free operation and tight timeliness guarantees.

3 Media Redundancy Mechanisms for CAN

This section is devoted to the analysis of media redundancy mechanisms for CAN. Based on the results presented in [23], it starts with a description of existing approaches to physical layer redundancy in CAN. Next, it is analyzed how LAN-based techniques could be adapted to CAN and finally our Columbus' egg idea for CAN media redundancy is presented.

Existing solutions

In [14] it is described a commercial solution (RED-CAN) that uses a self-healing ring/bus architecture to ensure resilience against open and short-circuits in the network physical wiring. Each RED-CAN node has its own reconfiguration switch. In case of failure, nodes perform a sequence of steps to find out the failure location and heal the physical network by isolating the failed segment. However, this reconfiguration process takes time and meanwhile the network is partitioned: communication blackouts can last as long as $100ms$. This is an extremely high figure when compared, for instance, with the worst-case time required by the standard CAN protocol to recover from severe network errors ($2.9ms@1Mbps$ - CAN 2.0B transmitter failure) [29].

An adaptation of the CAN protocol to an actively-coupled optical fiber ring which appear to the CAN controller as a conventional bus, is described in [12]. Fault-tolerance is attained through operation of the self-healing mechanism, which is used when single or multiple faults occur. The drawbacks of this proposal are: the need to modify (even if slightly) the CAN protocol; the extra delays introduced by the active coupling, which significantly reduces the achievable bit rate.

Are redundant media bus architectures feasible?

Our initial approach to the design of an infrastructure supporting CAN non-stop operation tries to exploit the techniques used in former works on LANs [27, 9], that proved quite effective. We maintain the assumptions defined in those works for LAN-based approaches, namely:

- channel[1] redundancy is used, through replicated media (physical and medium layers), but only one MAC[2] layer;
- each transmission medium replica is routed differently, being reasonable to consider failures in different media as independent;
- all media are active, i.e. every bit issued from the MAC layer is transmitted simultaneously on all media.

However, the CAN own properties are taken into account in the definition of media switching rules. For example, the *quasi-stationary* operation of CAN, guarantees the simultaneous reception at all redundant media interfaces of the same bit, in a given stream ordering [23, 22].

That property is exploited in the definition of a *frame-wise* strategy for CAN. The frame bit values are continuously compared and switching to a medium receiving a *dominant* value is required, whenever the current medium is receiving a *recessive* value: at the *start-of-frame* delimiter; within the frame arbitration field; at the *acknowledgment slot*[3]. The reasons that justify this strategy are: in a correct medium, a *dominant* bit transmission always overwrites a *recessive* value; physical disconnection from the network partition that includes a transmitter leads to a recessive idle bus; the frame bits where switching is allowed are the intervals, in the normal transmission of a frame, where several nodes may be transmitting simultaneously.

For the definition of an *error-based* media selection strategy, the CAN media redundancy entities must be able to identify the medium originating the error before the MAC layer performs error signaling to all media. Fault treatment procedures should: avoid switching to a medium exhibiting omission errors; declare failure when the allowed *omission degree*[4] is exceeded.

However, one key point is that the architecture of current CAN controllers does not favor the implementation of media switching strategies with off-the-shelf components. For example, to maintain synchronism on media switching, a smooth data signal transition would be required but that prevents a transmitting node from correctly performing bus state monitoring. The fundamental obstacles to the implementation of CAN bus redundancy using media switching and off-the-shelf components do concern both complexity and effectiveness.

Nevertheless, some questions remain: how much of the complexity associated with a media switching strategy would really be needed to ensure CAN non-stop operation? Would it be possible to provide an equivalent functionality with a simpler architecture?

A Columbus' egg idea

To answer those questions, let us analyze the problem under a slightly different perspective. Let us assume a simplified fault model, considering only the abrupt interruption of a transmission medium, as shown in Figure 2. In addition, let us assume a single node (Node 1 in Figure 2) is transmitting, for example a data or a remote frame.

When a given node transmits a frame, all the nodes located at the *in-partition*[5] receive a correct signal on all redundant media interfaces. On the other hand, the nodes at the *out-partition* only receive a correct signal on all media interfaces when the transmitter is sending recessive symbols.

[1] The *channel* is the physical path – transmission medium and medium interfaces – used by the MAC entities to communicate.

[2] Medium Access Control.

[3] The CAN protocol obliges a correct node to acknowledge the reception without errors of a frame, by asserting a *dominant* value at the *acknowledgment slot* [18, 7].

[4] Informally, the *omission degree* is the number of consecutive omission errors of a component in a time interval of reference [28].

[5] I.e., the partition that includes the transmitter.

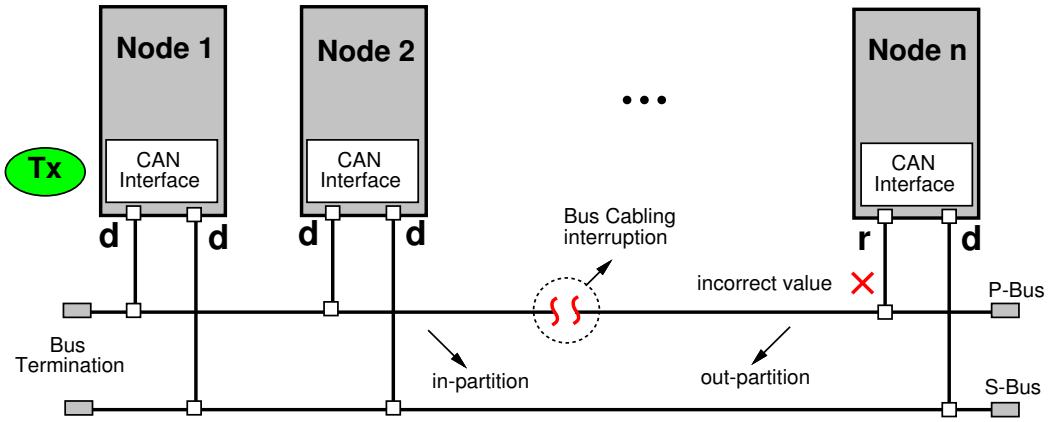

Figure 2: Abrupt partition of a media redundant CAN fieldbus

Otherwise, the recessive signal from the (idle) failed media does not match the dominant symbol transmitted in the correct media.

A fault treatment component, to be inserted in the incoming path, between the medium interfaces and the CAN controller, will allow the delivery of correct results. Hence, we came up with this Columbus' egg idea of extending the bare properties of CAN bus operation to the media interface level [20, 23, 22].

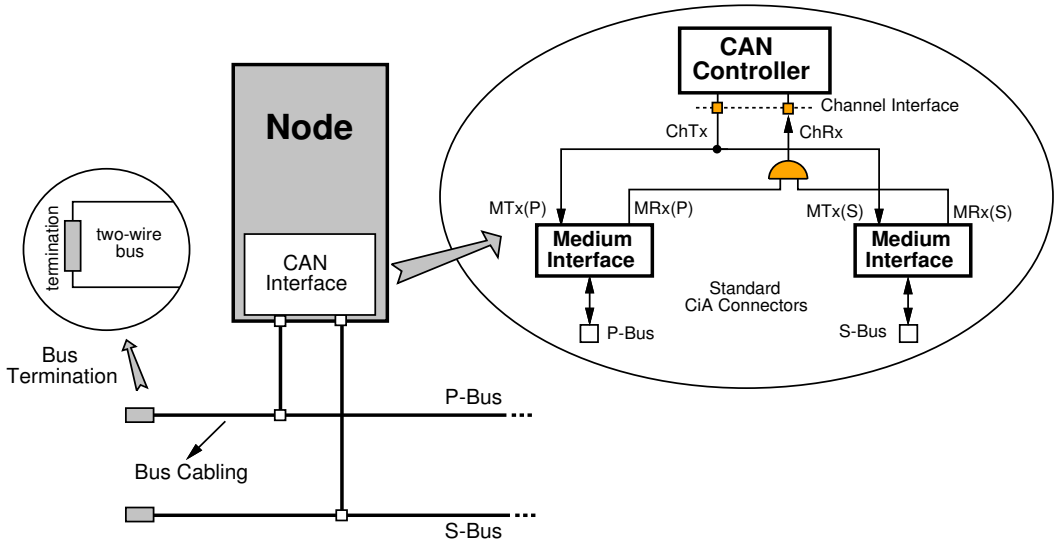

Figure 3: The Columbus' egg idea for bus media redundancy in CAN

Assuming a common CAN implementation, where a *dominant* value is represented by a logical zero and a *recessive* value is represented by a logical one, all the media will operate in parallel, being seen at the channel interface, as an unique bus implementing a logical AND function.

This solution can be implemented by a conventional AND gate, as exemplified in Figure 3 for a dual-media architecture. The complexity associated with media switching is avoided. The only disadvantage of this approach is that it is based on too restrictive a fault model. However, this basic architecture can be enhanced in order to support a less restrictive and thus more realistic fault model.

4 System Model

This section summarizes the model of CAN defined in [23], for a redundant media network infrastructure.

Let us assume a network composed of $\mathcal{N}$ nodes interconnected by a Channel. Each node $n \in \mathcal{N}$ connects to the Channel by a channel transmitter (outgoing bit stream) and a channel receiver (incoming bit stream). The channel transmitter and the channel receiver of node n are denoted as Ch^n_{Tx} and Ch^n_{Rx}, respectively. If the Channel is composed of several media $m \in \mathcal{M}$, $M^n_{Tx}(m)$ and $M^n_{Rx}(m)$ are used to represent the Medium m transmitter and receiver interfaces, at node n. The node is connected by the Channel transmitter and receiver to a media selection unit, which internally connects to each Medium, accordingly to a given strategy.

For simplicity of exposition, the superscript identifying the node will be omitted, whenever Channel and Medium refer to the same node. Medium is used to refer an instantiation of the Channel, comprising the network physical layer and the communication medium itself.

4.1 Fault model

We introduce the following definition: a component is **weak-fail-silent** if it behaves correctly or crashes if it does more than a given number of omissions – called the component's *omission degree* – in a time interval of reference.

The **CAN bus** is viewed as a single-channel broadcast local network with the following failure semantics for the network components:

- individual components are **weak-fail-silent** with *omission degree* f_o;
- failure bursts never affect more than f_o transmissions in a time interval of reference[6];
- omission failures may be inconsistent (i.e., not observed by all recipients);
- there is no permanent failure of the Channel (e.g. the simultaneous partition of all media).

Establishing a bound for the omission degree ($Od = f_o$) of individual components provides a general method for the detection of failed components. If each omission is detected and accounted for, the component fails once it exceeds the omission degree bound. In particular, a Medium fails if it crashes (stuck-at or broken failures) or if it exceeds Od. For the particular set $\mathcal{M}$ of media, we make the additional fault assumptions:

- failures in different media are independent;
- permanent omission failures never affect more than $\#\mathcal{M} - 1$ media.

4.2 CAN physical-level properties

We define *logical bit slot*, that we denote Bit from now on, as the logical entity corresponding to a nominal bit time interval. A Bit occupies an interval of constant length T_B and $t^s_B(p)$ is the (unobservable) real time instant when Bit p starts at s (s is a transmitter, a receiver or the channel).

In absence of faults, a Bit p at s assumes one and only one logical value $v^s_B(p)$. Given the current CAN implementations, the logical value one represents the *recessive* state and the logical value zero represents the *dominant* state.

A relevant set of CAN physical layer properties is presented in Figure 4. The PCAN1 property formalizes the *quasi-stationary* propagation of signals in CAN where, unlike longer and faster

[6] For instance, the duration of a broadcast round. Note that this assumption is concerned with the total number of failures of possibly different components.

PCAN1 - Bit **Simultaneity**: for any $Bit\ p$ of any transmitter s starting at $t_B^s(p)$, if $t_B^r(p)$ is the start of $Bit\ p$ as seen by receiver r, for any r, then in absence of faults, $t_B^s(p) = t_B^r(p)$.

PCAN2 - Wired-AND Multiple Access: for all transmitters s in $\mathcal{N}$, the value of any $Bit\ p$ seen by the channel c is, in absence of faults, $v_B^c(p) = \prod_{s \in \mathcal{N}} v_B^s(p)$.

PCAN3 - Bit **Broadcast**: in absence of faults, for any $Bit\ p$ on the channel c, and for any receiver r, $v_B^r(p) = v_B^c(p)$.

Figure 4: CAN physical level properties

networks, a transmitted bit stream has the same phase along the bus. A single Bit is transmitted on the channel at a time. Property PCAN2 specifies the function that combines the signals from multiple simultaneous transmitters on the bus, into a single Bit value. A dominant value always overwrites a recessive state. The symbol $\prod$ is used to represent a logical AND function. Property PCAN3 is required by the CAN protocol for arbitrating accesses to the shared medium, bus state monitoring and data transfer. Properties PCAN1 and PCAN2 are the foundation of CAN operation, being exploited in our method to implement bus-based media redundancy in CAN.

5 CAN Media Redundancy Strategies

This section discusses our implementation of bus-based media redundancy in CAN, which is based in the model defined in Section 4. In essence, we follow the results of [23].

Operational assumptions

Let us start with a description of some additional assumptions about the network infrastructure:

N1 - *channel redundancy is used, through replicated media (physical and medium layers), but only one MAC layer.*

However, apart from replication, standard CAN components are used. In particular, we do not exploit any of the fault-tolerant mechanisms of [16]. Furthermore, we do not assume the use of any specific transmission medium. Hence, different solutions are allowed for the physical layer: inexpensive differential pair wiring and non fault-tolerant medium interfaces [15] or fiber optic technology [19].

N2 - *each medium replica is routed differently.*

N3 - *all media are active, meaning every bit issued from the MAC layer is transmitted simultaneously on all media.*

Assumption N3 is simply enforced by logically connecting the Channel and all the Medium outgoing links together, thus implementing the function:

$$M_{Tx}(m) = \ Ch_{Tx} \qquad \forall\ m \in \mathcal{M} \tag{1}$$

The Columbus' egg strategy

The Columbus' egg strategy extends the PCAN2 wired-AND multiple access property (*vide* Figure 4) to the Media interface level, taking into account the PCAN1 property. The receive signals of each Medium interface are combined in an AND function before interfacing the MAC layer, as defined by equation:

$$Ch_{Rx} = \prod_{m \in \mathcal{M}} M_{Rx}(m) \tag{2}$$

where: the symbol $\prod$ is used to denote the AND function; $\mathcal{M}$ is the set of Medium interfaces. For example, in the dual-media architecture of Figure 3, $\mathcal{M} = \{P, S\}$.

This technique provides resilience to Medium partitions (e.g. A and B failures in Figure 1) and to stuck-at-recessive failures (e.g. failure D in Figure 1), without violating property PCAN3.

Handling stuck-at-dominant failures

A Medium stuck-at-dominant failure prevents equation (2) from delivering correct results. To detect these failures a special-purpose watchdog timer is used. In CAN, a correct Medium is not allowed to be at a dominant state for more than a given number of bit times, that we denote $\mathcal{T}_{stuck \leftarrow dm}$. This parameter is important because it provides an upper bound for the delay in the detection of a stuck-at-dominant Medium failure. The method that we specify for the treatment of stuck-at-dominant Medium failures is inspired by the rules defined in [18, 7] for the treatment of stuck-at-dominant Channel failures. The value of $\mathcal{T}_{stuck \leftarrow dm}$, defined accordingly with those rules, is given by equation:

$$\mathcal{T}_{stuck \leftarrow dm} = [\, 2 \,.\, l_{stk\_d} + (l_{stk\_d} + 1) \,.\, err_{stuck \leftarrow rx(bus)} \,] \,.\, \mathcal{T}_{bit} \tag{3}$$

where, l_{stk_d} represents the length of the sequence of consecutive dominant bits, tolerated by the CAN fault confinement mechanisms upon the transmission of an active error flag [18, 7]. We denote: $err_{stuck \leftarrow rx(bus)}$, as the allowed Medium *receive-error-tolerance margin* in the violation of the active error flag tolerance [22]; $\mathcal{T}_{bit}$, represents one bit-time.

The state of each Medium is permanently monitored. Upon the detection of a stuck-at-dominant condition, an indication of Medium m failure is provided:

$$M_{stk-d}(m) = \begin{cases} true & \textbf{if } \mathcal{T}_D(m) > \mathcal{T}_{stuck \leftarrow dm} \\ false & \textbf{if } \mathcal{T}_D(m) \leq \mathcal{T}_{stuck \leftarrow dm} \vee M_{Rx}(m) = r \end{cases} \tag{4}$$

where, $\mathcal{T}_D(m) = \mathcal{T}(M_{Rx}(m) = d)$ represents the normalized time elapsed since Medium m is at a dominant state. If it exceeds $\mathcal{T}_{stuck \leftarrow dm}$, the Medium has failed. The values *true* and *false* are represented by a logical one and a logical zero, respectively.

The M_{stk-d} failure indication can be used to directly request the disabling of the failed Medium, as follows:

$$M_{dis}(m) = M_{stk-d}(m) \tag{5}$$

The receive signal – Ch_{Rx} – delivered at the channel interface is established by the receive signals of the non-failed media interfaces, as specified in equation (6), where the symbols $\prod$ and $+$ are used to denote the AND and the OR functions, respectively.

$$Ch_{Rx} = \prod_{m \in \mathcal{M}} (M_{Rx}(m) + M_{dis}(m)) \tag{6}$$

This simple technique secures resilience to Medium stuck-at-dominant failures, such as failures C or G in Figure 1.

6 Error Detection Mechanisms

This section extends the functionality of our architecture by introducing mechanisms able to detect and to account for omission errors[7]. Provisions for the detection and monitoring of Medium and node permanent failures are also included.

Operational assumptions

We begin by making the following operational assumptions concerning the observable behavior of CAN at the PHY-MAC interface, as *per* the standard [18, 7]:

N4 - *there is always a detectable minimum idle period preceding the start of every CAN data or remote frame transmission.*

N5 - *there is a detectable and unique fixed form sequence that identifies the correct reception of a CAN data or remote frame.*

N6 - *there is a detectable bit sequence that identifies the signaling of errors in the CAN bus.*

Let us shortly justify these assumptions. With regard to N4, the Ch_{EOT} signal is asserted at the end of each frame transmission, when the minimum bus idle period that precedes the start of every data or remote frame transmission has elapsed. It is negated at the start of a frame transmission. The normalized duration of the *End-Of-Transmission* sequence ($\mathcal{T}_{EOT}$) includes the nominal three bit *intermission* ($\mathcal{T}_{ifs}$) and is equal for data/remote and for error/overload frames[8]. Equation (7) takes into account that a transmission may start at the last bit of the *intermission*, being $\mathcal{T}_L = \mathcal{T}_{EOT} - \mathcal{T}_{bit}$.

$$Ch_{EOT} = \begin{cases} true & \textbf{if } \mathcal{T}(Ch_{Rx} = r) \geq \mathcal{T}_L \\ false & \textbf{if } \mathcal{T}(Ch_{Rx} = r) < \mathcal{T}_L \vee Ch_{Rx} = d \end{cases} \tag{7}$$

With regard to assumption N5, if a data or remote frame transmission ends without errors, the *Frame correct* signal (Ch_{Fok}) is asserted, changing of state accordingly to expression (8). The fixed form sequence of assumption N5 includes the recessive (r) CRC delimiter, the dominant (d) *acknowledgment slot* and the recessive *acknowledgment delimiter* plus the first six recessive bits of the seven bit *end-of-frame* delimiter. The frame's last bit was not included because it is never considered by the recipients in the evaluation of frame correctness [18, 7, 25].

$$Ch_{Fok} \mapsto \begin{cases} true & \textbf{if } Ch_{Rx} = rdrrrrrrr \\ false & \textbf{when } Ch_{EOT} \end{cases} \tag{8}$$

Conversely, as defined by assumption N6, when a frame transmission is aborted due to errors, the Ch_{Err} signal changes of state accordingly to expression (9). Errors are signaled on the bus

[7] The occurrence of omission errors is usually due to electromagnetic interference. However, they may also have their origin in subtle causes, such as a defective connector mount or a smashed cable, causing impedance mismatches that may introduce a reflection pattern which sporadically prevents communication. Other cause may be the incorrect dimensioning of CAN physical layer parameters.

[8] Being: $\mathcal{T}_{EOT} = \mathcal{T}_{bit} + \mathcal{T}_{EOF} + \mathcal{T}_{ifs} = \mathcal{T}_{del} + \mathcal{T}_{ifs}$, where $\mathcal{T}_{EOF}$ and $\mathcal{T}_{del}$ are the normalized durations of the *end-of-frame* and of the error/overload delimiters [18, 7]. The normalized duration associated to the ending sequence of data/remote frames includes the one-bit *acknowledgment delimiter* [18, 7].

through a detectable sequence of dominant bits (assumption N6), violating the bit-stuffing coding rule. Defining l_{stuff}, as the bit-stuffing width:

$$Ch_{Err} \mapsto \begin{cases} true & \textbf{if } \mathcal{T}(Ch_{Rx} = d) \geq (l_{stuff} + 1).\mathcal{T}_{bit} \\ false & \textbf{when } Ch_{EOT} \end{cases} \tag{9}$$

Frame monitoring

In order to evaluate whether or not a given Medium is exhibiting omission errors, the reception of data and remote frames is continuously monitored. For every bit, the signal received from Medium m – $M_{Rx}(m)$ – is compared with the channel receive signal (Ch_{Rx}), until the frame transfer is successfully completed or aborted by errors. A *frame mismatch* signal – $M_{Fm}(m)$ – is asserted for Medium m, if the two signals do not exhibit the same value:

$$M_{Fm}(m) \mapsto \begin{cases} true & \textbf{if } M_{Rx}(m) \neq Ch_{Rx} \wedge Ch_{TiP} \\ false & \textbf{when } Ch_{EOT} \end{cases} \tag{10}$$

where, $Ch_{TiP} = \neg Ch_{Fok} \wedge \neg Ch_{Err}$ signals that a frame transfer is in progress. Once asserted, the $M_{Fm}(m)$ signal is kept in that state even if the two signals become equal again. It is negated only when Ch_{EOT} becomes true.

Frame monitoring is suspended: after the detection of an error and assertion of the corresponding Ch_{Err} signal; after the successful reception of a data/remote frame, recognized through the assertion of the Ch_{Fok} signal. Network errors occuring at the last bit of the *end-of-frame* delimiter or during the *minimum intermission*[9] period are not accounted for as frame omissions[10].

Detecting Medium omissions

The $M_{Fm}(m)$ signals are used, together with the Ch_{Fok} and Ch_{Err} signals, in the update of the Medium omission degree. The following set of auxiliary functions is defined:

$$M_{Fm-s} = \sum_{m \in \mathcal{M}} M_{Fm}(m) \tag{11}$$

$$M_{Oer}(m) = Ch_{Fok} \wedge M_{Fm}(m) \tag{12}$$

$$M_{Och}(m) = Ch_{Err} \wedge M_{Fm-s} \wedge \neg M_{Fm}(m) \tag{13}$$

$$M_{Uer}(m) = Ch_{Err} \wedge (\neg M_{Fm-s} \vee M_{Fm}(m)) \tag{14}$$

The symbol $\sum$ is used in equation (11) to denote the logical sum of the $M_{Fm}(m)$ signal associated to each Medium. These auxiliary functions are used in the accounting of Medium m omission degree, $M_{Od}(m)$, updated upon the assertion of the Ch_{EOT} signal[11]:

$$M_{Od}(m) \overset{=}{\uparrow_{Ch_{EOT}}} \begin{cases} M_{Od}(m) + 1 & \textbf{if } M_{Oer}(m) \vee (M_{Och}(m) \wedge \neg Ch_{Fok}) \\ M_{Od}(m) & \textbf{if } M_{Uer}(m) \wedge \neg Ch_{Fok} \\ 0 & \textbf{if } Ch_{Fok} \wedge \neg M_{Fm}(m) \end{cases} \tag{15}$$

If the $M_{Fm}(m)$ and the Ch_{Fok} signals are simultaneously asserted at the end of a frame transfer, that means: a correct data or remote frame has been successfully received; at least one bit in the

[9] The mandatory two-bit bus idle period between every correct frame transmission [18, 7].

[10] In CAN operation, these are considered *overload errors* [18, 7].

[11] This condition is signaled in equation (15) through the symbol $\uparrow_{Ch_{EOT}}$.

stream received from Medium m did not have a correct value. Thus, the omission degree of Medium m should be incremented.

On the other hand, if the frame transfer is aborted due to errors, a Medium having its $M_{Fm}(m)$ signal negated can be made responsible for the errors and its omission degree count should be incremented. However, we define one exception to this rule: the omission degree count should not be modified, despite the assertion of Ch_{Err}, when no Medium has the $M_{Fm}(m)$ signal asserted. This situation have its origin in **common-mode errors**, caused, for example, by a node with a failed transmitter [29]. The omission degree count of Medium m should also remain unchanged, when the Ch_{Err} and $M_{Fm}(m)$ signals are both asserted, because one cannot be sure Medium m has not exhibit omission errors[12]. One particular case concerns **single-medium errors**, that nevertheless generate *frame mismatches* in all media. The "incorrect" media need to be temporarily disabled (*quarantined*), to allow operation to proceed with a "correct" set.

A Medium that exceeds its omission degree bound should be declared failed and its contribution to equation (6) disabled, by layer management entities [22]. For a Medium exhibiting a correct behavior, i.e. when a frame is correctly received and no *frame mismatches* have been reported for that Medium, the corresponding omission degree counter is set to zero.

Media monitoring

The signaling of an incorrect recessive value at a given Medium interface does not disturb the results of the AND function that constitutes the core of our media redundancy mechanisms. Nonetheless, such incidents should be detected and monitored.

A special-purpose watchdog timer, together with the monitoring of abnormally long Medium idle periods, may be used to detect those failures, signaling them to high layer management entities, e.g. for diagnose purposes [22].

Channel monitoring

The error detection mechanisms introduced earlier in the architecture of the CAN media selection unit do not take into account that a given node may fail and start to transmit only dominant symbols at the Channel interface. The operation of the entire network may be disturbed by this single failure.

Again a dedicated watchdog timer may be used to detect such failures [22]. Such kind of indication can be used to disable the medium interface transceiver circuitry, whenever those components support that functionality (e.g. [6]). Otherwise, the architecture of the media selection unit should include the circuitry required for that purpose (*vide* Figure 5).

7 CAN Media Selection Unit Design

The architecture of the *media selection unit*, to be inserted between the redundant media interfaces and a standard CAN controller, is depicted in Figure 5. Apart from replication, commercial CAN components are used. No assumption is made concerning the use of a 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 [15] or fiber optic technology [19].

[12] One exception can be found in a dual-media architecture, when only one $M_{Fm}(m)$ signal is negated. Under a single-failure assumption the Medium not exhibiting omission errors will have its $M_{Fm}(m)$ signal asserted.

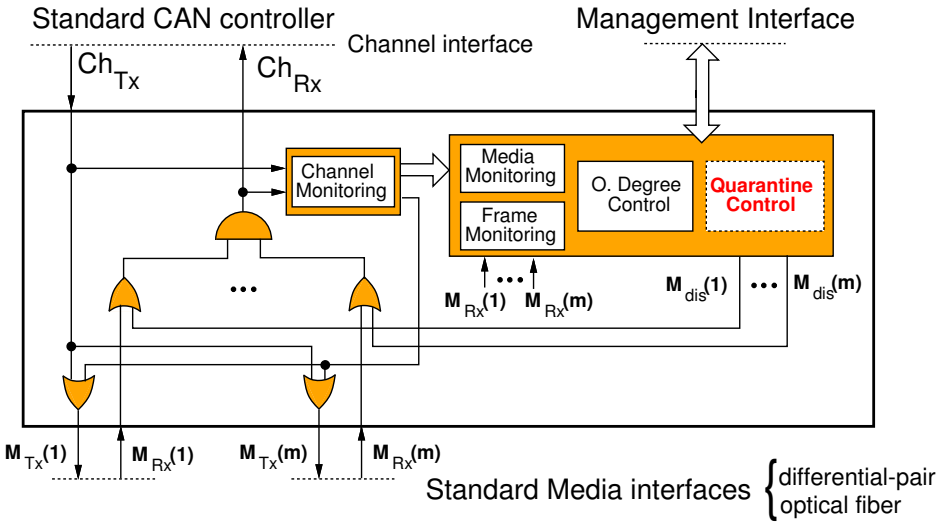

Figure 5: Architecture of the CAN media selection unit

All provisions for media replication management are made as extensions to the standard. Resilience to Medium crash (stuck-at/broken) and omission failures is achieved through the mechanisms described earlier, in Sections 5 and 6.

The central component of the CAN media selection unit (Figure 5) is the AND function specified by our strategy. The remaining functions are structured in a modular way [24], as follows:

- **Channel monitoring** – analyzes Channel activity with regard to the occurrence of permanent node failures and checks whether or not a frame transfer succeeds;
- **media monitoring** – analyzes the state of each Medium interface with regard to the occurrence of permanent failures;
- **frame monitoring** – compares Channel activity with the behavior of each Medium interface, thus allowing the detection of omission errors;
- **omission degree control** – this module uses the indications provided by Channel and frame monitoring modules to keep track of the omission degree of each Medium;
- **quarantine control** – aims to control the effects of single-medium errors that nevertheless affect the behavior of media monitoring functions at all media interfaces.

The attributes of the CAN protocol can and should be exploited in the design of media redundancy management modules, in order to keep complexity low [24, 22]. For example: stuck-at dominant and recessive failures are mutually exclusive events, and this may be used to reduce the complexity of the *media monitoring* circuitry; the detection of the bit sequences that identify error signaling and frame termination can be combined in a common pattern recognition machine; the assignment of pre-defined constants to a relevant set of protocol-related parameters, such as the *error tolerance margins* [22], simplifies the implementation of monitoring mechanisms, at the cost of a lower flexibility in the configuration of those parameters.

Complexity issues are of major relevance to the integration of the media selection unit in a single, inexpensive, medium capacity programmable logic device. A prototype of the media selection unit, aiming its integration in a *field programmable gate array*, is being specified in VHDL[13]. A preliminary specification of such a design is described in [10].

[13] Very High-Speed Integrated Circuits (VHSIC) Hardware Description Language.

Timing issues

The inaccessibility[14] period associated with bus reconfiguration in the presence of *stuck-at-dominant* medium failures is analyzed next.

Let us denote by $\mathcal{T}_{stuck \leftarrow dm}$ the detection latency of a stuck-at-dominant medium failure, as specified by equation (3). The duration of corresponding inaccessibility period accounts for two different contributions: the time required to detect the error and the time needed for recovery. Under a slightly pessimistic approach, we consider the recovery process includes the transmission of a complete error frame, despite the resume of normal bus operation requires only the signaling of the error frame ending delimiter. The best-case inaccessibility time due to bus reconfiguration, $\mathcal{T}_{ina \leftarrow bus}$, that we signal through the superscript ^{bc}, is then given by equation:

$$\mathcal{T}^{bc}_{ina \leftarrow bus} = \mathcal{T}_{stuck \leftarrow dm} + \mathcal{T}^{bc}_{error} + \mathcal{T}_{ifs} \tag{16}$$

where: $\mathcal{T}^{bc}_{error}$, is the normalized minimum duration of an error frame; $\mathcal{T}_{ifs}$, is the normalized duration of the *nominal intermission* period.

To obtain the corresponding worst-case inaccessibility time, that we signal through the superscript ^{wc}, we consider the medium failure occurs while transmitting the last bit of a maximum size data frame:

$$\mathcal{T}^{wc}_{ina \leftarrow bus} = \mathcal{T}^{wc}_{data} - \mathcal{T}_{bit} + \mathcal{T}_{stuck \leftarrow dm} + \mathcal{T}^{wc}_{error} + \mathcal{T}_{ifs} \tag{17}$$

where, $\mathcal{T}^{wc}_{data}$ and $\mathcal{T}^{wc}_{error}$ are, respectively, the normalized maximum durations of data and error frames.

Considering a value of $err_{stuck \leftarrow rx(bus)} = 2$ for the *receive-error-tolerance margin* associated to a stuck-at-dominant Medium failure, we obtain the values inscribed in Table 1 for the normalized inaccessibility times of bus reconfiguration [22]. Notice the extremely low worst-case figure of bus reconfiguration time (209 μs @ 1 Mbps), compared with the 100 ms of the RED-CAN system [14].

$\mathcal{T}_{ina \leftarrow bus}$ (*bit-times*)			
CAN 2.0A		**CAN 2.0B**	
min. (^{bc})	*max.* (^{wc})	*min.* (^{bc})	*max.* (^{wc})
47	184	47	209

Table 1: Normalized CAN inaccessibility time due to bus reconfiguration

8 CAN Full Space-Redundancy

In order to complete our analysis of highly available CAN fieldbus architectures, we discuss next how complex it will be to design and implement full space-redundancy in CAN.

Full space-redundancy, replicating media and network controllers, traditionally exhibits a cost higher than simple media redundancy. However, CAN is a low-cost technology: stand-alone CAN controllers are widely available at rather low prices; a few microcontroller families already provide devices with dual CAN interfaces [13, 4].

[14] Informally, *inaccessibility* is a state where a component temporarily refrains from providing service without that having to be necessarily considered a failure [28, 29].

This opens room for the design and implementation of CAN-based full space-redundant network infrastructures, at acceptably low costs. The architecture of a CAN space-redundant platform is not complex to design, as illustrated in Figure 6, for a dual-redundant solution. For each CAN controller, the channel interface transmit and receive signals are connected directly to the corresponding medium interface signals. Redundancy management is performed by a software layer (not represented in Figure 6) built directly on the top of the standard MAC sub-layers.

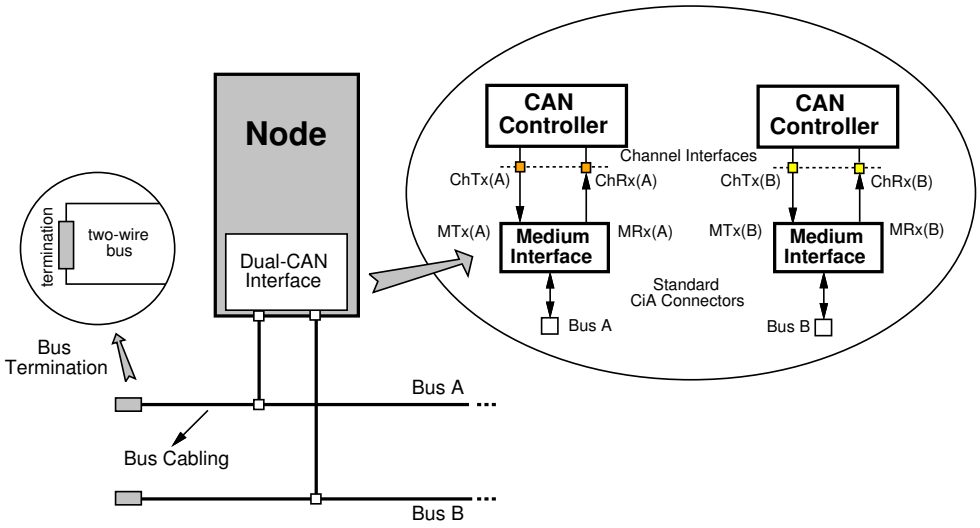

Figure 6: Full space-redundancy in CAN

Next, we discuss the pros and cons of using either simple media-redundancy or full space-redundancy techniques in the design of a CAN-based fault-tolerant real-time communication system.

9 CAN Media and Full Space-Redundancy

This section discusses the main attributes of CAN media and full space-redundancy schemes, performs their comparison and shows how the two redundancy approaches can be combined in the same architecture [21].

Attributes of CAN media redundancy

The main attributes of CAN media redundant architectures are characterized next. Both positive and negative aspects are analyzed.

- *network availability* – replication of the CAN physical and medium layers provides resilience to network stuck-at, partition and omission failures, on the assumption that permanent failures do not affect simultaneously all media replicas (cf. Section 4).
- *narrowed coverage of faults* – simple media redundancy cannot provide resilience to common mode errors that temporarily may affect all the media simultaneously. These errors may have its origin, for example, in a node with a failed transmitter or a failed receiver. Though CAN has means to recover from such errors, that recovery takes time and meanwhile the network is completely down, because each node can communicate with no one.

- *redundancy transparency* – with the possible exception of network configuration actions by layer management entities, all provisions for redundancy are introduced at the lowest layers of communication. The operation of protocols above the PHY-MAC interface is transparent with regard to the use of media redundancy.

- *hardware-based redundancy management* – the need for a specialized hardware infrastructure is not necessarily a disadvantage. All actions required for the management of redundancy are performed by dedicated machinery, without introducing any protocol processing overheads. This can be of major importance in simple CAN infrastructures, with modest or even inexistent processing resources (e.g. [11]), where media redundancy may be the only solution for achieving high network availability.

Attributes of CAN full space-redundancy

The fundamental attributes of CAN full space-redundant architectures are characterized next, once again analyzing both positive and negative aspects.

- *network availability* – a full space-redundant CAN communication infrastructure should provide availability figures similar to those furnished by simple media redundancy, for the same fault assumptions and level of replication.

- *broad coverage of faults* – assuming independence of channel failures, the use of full space-redundancy provides tolerance to temporary and permanent channel partitions.

- *tight timeliness* – the use of full space-redundancy allows to narrow the variability of CAN timing properties, in the presence of network errors. The timeliness of CAN error-free operation may be preserved by transmitting the same message on all channels. Assuming channel failures are independent, a copy of that message will be delivered to all recipients, at least by one channel.

  The provision of strict real-time guarantees at the MAC sub-layer interface is obtained by securing CAN glitch-free operation in the presence of inaccessibility faults [29].

- *software-based redundancy management* – to assure transparency with regard to the use of network redundancy, a given number of actions have to be performed at a software layer, to be inserted between the redundant MAC sub-layers and the higher layer protocols. For example: the same message may need to be queued for transmission on all the communication channels; each channel receive buffer has to be read upon the reception of a message.

  This software layer introduces processing overheads which may have a non-negligible impact on CAN platforms with modest processing resources. It is common knowledge among CAN practitioners that in some simple CAN infrastructures, built around small microcontrollers, it may be difficult to handle a high data throughput. For these architectures, full space-redundancy may not be recommended.

- *extra network bandwidth* – the glitch-free operation and the tight timeliness characteristics of full space-redundant architectures can be traded with a lower utilization of the available network bandwidth. Under error-free operation, each message is queued for transmission at a single channel interface. Given sufficiently low network error rate figures, that may lead to significant savings in the utilization of the network bandwidth.

The extra network bandwidth can be used to: increase the probability of having soft real-time traffic meeting the corresponding timeliness requirements; increase the network bandwidth available for non real-time traffic.

Upon the detection of a network error, the message queuing policy changes and each message will be submitted for transmission at multiple channel interfaces. The timing guarantees achieved with this adaptive scheme are naturally less tight than in a traditional space-redundant configuration. The worst-case time required to detect the network error and perform mode switch, T_{Ch-ms}, must be added to worst-case message transmission delay, T_{td}, in the absence of faults. The method is effective because $T_{Ch-ms} \ll T_{ina}$, the worst-case duration of an inaccessibility period [29].

Comparing CAN redundant architectures

The main attributes of the different CAN redundancy schemes are compared in Figure 7. If no redundancy is used, the system will exhibit the availability and the fault coverage secured by the native CAN protocol.

MAC architecture	Redundant media	Network availability	Fault coverage	Tight timeliness	Processing overheads
single	no	low	low	no	no
	yes	high	high		
dual	no	high	very high	yes	yes
	yes	very high	very high		

Figure 7: Main attributes of CAN redundant architectures

The use of simple media redundancy allows the provision of resilience to Medium crash (stuck-at/broken) and omission failures in a highly available network architecture. Glitch-free operation and tight timeliness characteristics can only be achieved with full space-redundant architectures, at the cost of additional message processing overheads.

Combining CAN media and full space-redundancy

The combination, in a single network architecture, of full space-redundancy and of simple media redundancy, allows to achieve all the features of network redundancy together with very high levels of network availability.

In CAN, the combination of media and full space-redundancy is extremely simple to achieve, as illustrated in Figure 8. Each network cable has two differential pairs:

- one pair is used as a primary bus[15] (P-Bus) for a given channel interface, as illustrated in Figure 8;
- the other pair, is used as a secondary bus (S-Bus) for a different channel interface.

[15] This designation is completely arbitrary because, in the absence of permanent medium errors, all media replicas are active.

The architecture of Figure 8 illustrates the design of a Dual-MAC/Quad-Bus CAN architecture. Media redundancy is supported by the CAN media selection unit described in Section 7. The medium interface receive signals, for both primary and secondary buses, are combined in the AND-based media selection unit, to obtain the channel receive signal.

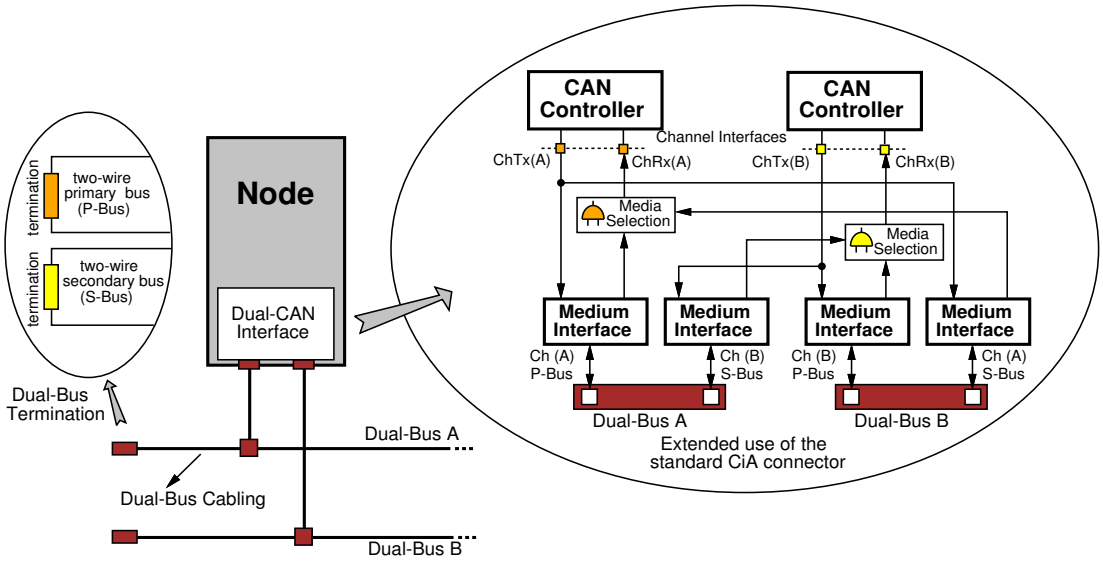

Figure 8: Combining media and full space-redundancy in CAN

The architecture of Figure 8 includes two different CAN controllers. A software/firmware protocol layer (not represented in Figure 8), to be inserted between the CAN standard layer and higher level protocols, is responsible for managing communication redundancy. If required, this layer can ensure a transparent operation of higher level protocols, with regard to network redundancy.

The physical attachment of the network controller to each dual-bus infrastructure has to support a dual-media connection. Fortunately, this attachment can be made compliant with the definition of the CiA standard connector [1]. In our proposal, the standard connector definition is extended to allow the use of two reserved pins, shown in Figure 9 through a gray shaded label, to support the connection of the secondary bus. Other types of connectors, such as those specified in [2], can be easily extended to support a combination of network/media redundancy.

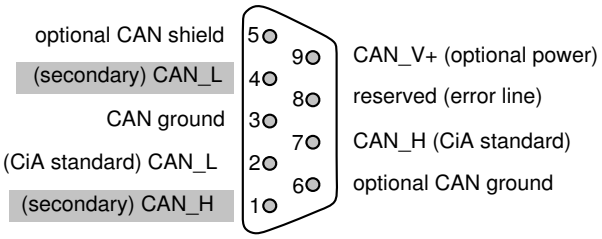

Figure 9: Extending the use of the standard CiA connector

The high levels of dependability achieved by this unified architecture, namely the very high network availability in the presence of permanent failures, glitch-free communication and tight timeliness characteristics do represent a step toward a possible utilization of CAN in safety-critical applications. The same architecture supports an alternative design solution, where the tight

timeliness guarantees are traded by an improvement in the available network bandwidth and overall system throughput.

10 Future Developments

Future developments may address the design of: extremely effective Medium quarantine techniques, aiming to guarantee, with a very high probability, an extremely low Channel omission degree bound at the MAC-level ($Ch_{Od-MAC} = 1$); hardware assistance to the total ordering of messages, in dual-redundant Channel architectures.

A slightly different line of research is inspired by the recent public domain availability of CAN controller VHDL cores [26]. In this context, CAN dependability enforcement mechanisms, such as the support for media redundancy, can be integrated directly into the CAN controller architecture in a cost-effective way. As a matter of fact, some mechanisms may even be optimized.

11 Conclusions

There is an increasing demand for fault-tolerant and real-time distributed systems based on field-buses. Many of these systems are intended for critical control applications, where continuity of service is a strict requirement.

Network media redundancy is an effective solution for resilience against temporary medium faults and availability in the presence of permanent faults.

In this paper, we do a systemic analysis of how bus redundancy mechanisms can be implemented in CAN, the Controller Area Network, and we end-up with a Columbus' egg idea: an extremely simple mechanism that makes bus-based redundancy easy to implement in CAN using off-the-shelf components.

The simplest architecture just uses a conventional AND gate, together with the standard CAN components, to provide resilience to medium partitions and stuck-at-recessive failures in the network cabling. With some extra circuitry, of small complexity, we are able to ensure: resilience to stuck-at-dominant failures; omission failure detection and fault treatment; support to high-layer diagnose and distributed failure detection applications.

The combination of media and full-space redundancy, that we have also analyzed, provides glitch-free communication and tight timeliness characteristics and do represent a step toward a possible utilization of CAN in safety-critical applications.

References

[1] CiA - CAN in Automation. *CAN Physical Layer for Industrial Applications - CiA Draft Standard 102 Version 2.0*, April 1994.

[2] CiA - CAN in Automation. *CANopen Cabling and Connector Pin Assignment - CiA Draft Recommendation DR-303-1*, version 1.0 edition, October 1999.

[3] 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*, pages 6–19, Newcastle, United Kingdom, June 1990. IEEE.

[4] Dallas Semiconductors. *DS80C390 Dual-CAN High-Speed Microprocessor*, preliminary edition, September 1999.

[5] F. Hartwich and A. Bassemir. The configuration of the CAN bit timing. In *Proceedings of the 6th International CAN Conference*, pages xx–xx, Turin, Italy, November 1999. CiA.

[6] Infineon Technologies, Munich, Germany. *CAN Transceiver TLE 6250 Data Sheet*, May 2000.

[7] ISO. *International Standard 11898 - Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication*, November 1993.

[8] H. Kopetz and G. Grunsteidl. TTP - a protocol for fault-tolerant real-time systems. *IEEE Computer*, 27(1):14–23, January 1994.

[9] C. Mateus. Design and implementation of a non-stop Ethernet with a redundant media interface. IST Graduation Project Report, Advisors: J. Rufino and P. Veríssimo, Instituto Superior Técnico, Lisboa, Portugal, September 1993. (in portuguese).

[10] A. Matias. Design and implementation of CAN media redundancy mechanisms. IST Graduation Project Report, Advisors: J. Rufino and G.Arroz, Instituto Superior Técnico, Lisboa, Portugal, February 2000. (in portuguese).

[11] Microchip Technology Inc. *MCP2502X/5X CAN I/O Expander Family*, preliminary edition, 2001.

[12] R. Mores, M. Morse, and W. Kuntz. CAN operated on an optical double ring at improved fault-tolerance. *ETT Journal*, 4(4):465–470, July 1993.

[13] Motorola, Inc., USA. *MPC555 User's Manual*, May 1998.

[14] RED-CAN a fully redundant CAN-system. NOB Elektronik AB Product Note - Sweden, 1998. http://www.nob.se.

[15] Philips Semiconductors. *PCA82C250 - CAN Controller Interface*, April 1994.

[16] Philips Semiconductors. *TJA1053 - Fault-tolerant CAN transceiver*, October 1997.

[17] D. Powell, editor. *Delta-4 - A Generic Architecture for Dependable Distributed Computing*. ESPRIT Research Reports. Springer Verlag, November 1991.

[18] Robert Bosch GmbH. *CAN Specification Version 2.0*, September 1991.

[19] M. Rucks. Optical layer for CAN. In *Proceedings of the 1st International CAN Conference*, pages 2.11–2.18, Mainz, Germany, September 1994. CiA.

[20] J. Rufino. Dual-media redundancy mechanisms for CAN. Technical Report CSTC RT-97-01, Centro de Sistemas Telemáticos e Computacionais do Instituto Superior Técnico, Lisboa, Portugal, January 1997.

[21] J. Rufino. Redundant CAN architectures for dependable communication. Technical Report CSTC RT-97-07, Centro de Sistemas Telemáticos e Computacionais do Instituto Superior Técnico, Lisboa, Portugal, December 1997.

[22] J. Rufino. *Computational System for Real-Time Distributed Control*. PhD thesis, Technical University of Lisbon - Instituto Superior Técnico, Lisboa, Portugal, July 2002.

[23] 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, Wisconsin - USA, June 1999. IEEE.

[24] J. Rufino, P. Veríssimo, and G. Arroz. Design of bus media redundancy in CAN. In D. Dietrich, P. Neumann, and H. Schweinzer, editors, *Fieldbus Technology - System Integration, Networking and Engineering*, pages 375–380. Springer, September 1999. (Proceedings of the Fieldbus Conference FeT'99, Magdeburg, Germany).

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

[26] L. Stagnaro. *HurriCANe - Free VHDL CAN Controller core*. Spacecraft Control and Data Systems Division - Eupean Space Agency, Noordwidjk, The Netherlands, March 2000. Version: Alpha 4.

[27] 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.

[28] P. Veríssimo. Real-time Communication. In S.J. Mullender, editor, *Distributed Systems*, ACM-Press, chapter 17, pages 447–490. Addison-Wesley, 2nd edition, 1993.

[29] 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, Washington - USA, June 1997. IEEE.