A Columbus' Egg Idea for CAN Media Redundancy*

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

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

Guilherme Arroz
IST-UTL
pcegsa@alfa.ist.utl.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). In our endeavor to provide CAN with media redundancy we ended-up devising a scheme which is extraordinarily simpler than previous approaches known for CAN or other LANs and field-buses.

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 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 [4, 6], at a 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 [11]. 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. With 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 [16]. Cost-effectiveness and shorter design cycles are, among others, strong arguments in favor of using off-the-shelf LAN and field-bus technologies in the design of fault-tolerant distributed systems.

Field-buses are in essence a technology whose area of application requires continuity of service. 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, that we address in this paper, in the context of the Controller Area Network (CAN) [5]. CAN is a low-cost field-bus that is getting more and more attractive for areas as diverse as shop-floor control, robotics, automotive or avionics. CAN is traditionally viewed as a robust field-bus. Together with protocol level extensive error checking capabilities, the use of differential two-wire communication medium and the use of physical level fault-tolerant mechanisms allows CAN to operate in the presence of one-wire failures in the network cabling [5]. 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.

The work presented here is part of a broader effort aiming at designing an embedded fault-tolerant distributed system around CAN. In previous works, regarding the definition of CAN fault-tolerant communication and time services [13, 12], we have assumed a system model where network components only temporarily refrain from providing service [17]. In this paper we substantiate that assumption by defining a CAN-based network infrastructure resilient to the permanent failure of physical layer components, such as medium partitions.

CAN-based redundant architectures using replicated buses have been identified as being too costly, when compared with alternative designs based on ring topologies [9]. An existing commercial redundant CAN solution implements a self-healing ring/bus architecture [9], but does not solve

* *Columbus' egg*: popular expression, with origin in a story about the navigator Christopher Columbus, that is widely used to refer an extremely simple solution to a difficult problem, hard to find, but that once known, looks trivial and even obvious. The navigator, in front of a meeting of lords, demonstrated how to make an egg stand on end... by cracking its shell in one of the poles!

† Instituto Superior Técnico - Universidade Técnica de Lisboa, Avenida Rovisco Pais, 1049-001 Lisboa, Portugal. Tel: +351-1-8418397/99 - Fax: +351-1-8417499. NavIST Group CAN WWW Page - http://pandora.ist.utl.pt/CAN.

‡ Faculdade de Ciências da Universidade de Lisboa, Campo Grande - Bloco C5, 1700 Lisboa, Portugal. Tel: +351-1-7500087 - Fax: +351-1-7500084. Navigators Home Page: http://www.navigators.di.fc.ul.pt.

the problem of CAN continuity of service efficiently: ring reconfiguration takes time and meanwhile the network is partitioned.

In this paper, we do a systemic analysis of how bus redundancy mechanisms can be implemented in CAN. We started by studying the adaptation of techniques we had developed with success for LANs [15, 8]. 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.

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 [5, 3], for details about the CAN protocol.

2 Related Work

The LAN-based media redundant architectures described in [15, 8] rely on the replication of the physical path – transmission medium and medium interfaces – used by the MAC[1] entities to communicate (channel). It is assumed: channel redundancy is used, through replicated media (physical and medium layers), but only one MAC 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.

In the implementation of LAN media redundancy [15, 8], one medium is selected at a time, for frame reception. A *frame-wise* strategy takes a decision at the start of each frame reception, based on the quality of the signals currently received from each medium. A complementary selection strategy, uses an indication on whether or not network errors occur during the reception of one frame, to choose the medium from which the next frame should be received. A *frame-wise* decision always supersedes an *error-based* selection, unless media switching is locked by fault treatment procedures, i.e measures to ensure the fault is passivated.

A similar approach can be found in the standard specification of some field-buses, such as WorldFIP [2] and PROFIBUS. An alternative dual-ring architecture is foreseen in PROFIBUS, when using fiber optics media [14]. In LONWORKS, a self-healing bus/ring architecture is specified [7]: each end of the bus terminates in an *intelligent switch* which is responsible for network reconfiguration in the event of an open-wiring fault in the network cabling. Since the switch is not replicated, it is a single-point of failure. The MIL-Std-1553 specifies a dual redundant bus option, but it also has a single-point of failure in a centralized bus controller: only one bus is active at a time with the bus controller initiating all message transfers.

[1] Medium Access Control.

3 CAN Physical Level Fault-Tolerance

In CAN, bus signaling takes one out of two possible representations: *recessive*, otherwise the state of an idle bus, occurs when all competing nodes send recessive bits; *dominant*, which always overwrites a recessive value. This behavior, together with the uniqueness of frame identifiers, is exploited for bus arbitration. A *carrier sense multi-access with deterministic collision resolution* policy is used: several nodes may jump on the bus at the same time, but while transmitting the frame identifier each node monitors the bus; for every bit, if the transmitted bit is recessive and a dominant value is monitored, the node gives up transmitting and starts to receive incoming data. Frames that have lost arbitration or have been destroyed by errors are automatically scheduled for retransmission.

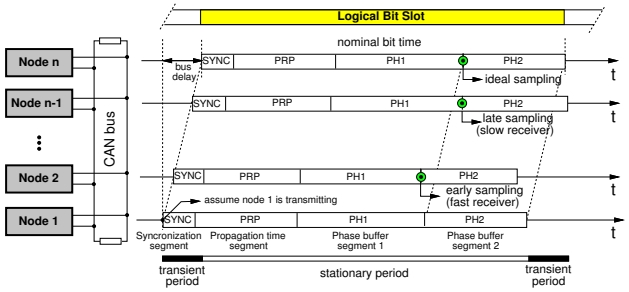

Figure 1. Sketch of CAN bus operation timing

Synchronization of receiver circuitry with the incoming bit stream is performed in CAN through a complex process that we summarize next. The nominal bit time is divided in four segments (Figure 1). The *propagation time segment* (PRP) accounts for physical level delays: the bus propagation time and the transmitter/receiver delays at the medium interface devices. The idea is to give enough time for signal stabilization along the bus before nodes perform sampling, which occurs at the end of the PH1 segment (Figure 1). Bus signal transitions are expected to lie within the *synchronization segment* (SYNC). Deviations from this ideal behavior produce phase errors which are compensated for by using one of the two *phase segments* as elastic buffers: the PH1 segment is lengthened in fast receivers, upon the detection of a phase error; slow receivers compensate phase errors by shortening PH2. Further details on bit synchronization can be found in [5].

With exception of the transient periods at bit boundaries, a single bit is present in the CAN bus line at a time. As a consequence, all nodes get the same bit – with regard the

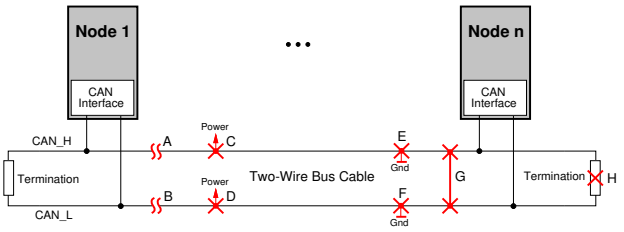

Figure 2. Resilience to medium failures in the ISO 11898 CAN standard

incoming stream – when sampling the bus[2]. Such kind of operation is known as *quasi-stationary*, and it will be of fundamental importance for the definition of CAN media redundancy strategies.

The CAN transmission medium is usually a two-wire differential line. The CAN physical layer specified in [5] allows resilience against some of the transmission medium failures illustrated in Figure 2, 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 2);
- 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 kbaud) with no more than 32 nodes [10]. One exception is the CAN interface described in [1].

The CAN bus line is usually terminated at both ends by its characteristic impedance [3]. Resilience to the failure of one termination (H failure, in Figure 2) can be achieved simply by taking into account the extra time needed for bus signal stabilization, when dimensioning the *propagation time segment* [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 2). 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, through redundancy, is the objective of this paper.

[2] Although the sampled value may not be the same at all nodes, due to errors. Examples of causes for erroneous bit sampling are: electromagnetic interference, loss of synchronism or defective receiver circuits.

4 Redundancy Mechanisms for CAN

This section starts with a description of existing approaches to physical layer redundancy in CAN. Next, we analyze how LAN-based techniques could be adapted to CAN and finally present our Columbus' egg idea for CAN media redundancy.

Existing solutions

In [9] 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 ($2.5ms@1Mbps$ - transmitter failure) [17].

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 [15, 8], that proved quite effective. We maintain the assumptions stated in Section 2 for LAN-based approaches, but take into account the CAN own properties 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.

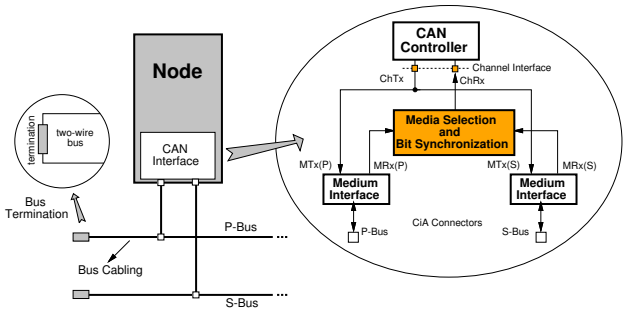

Figure 3. A complex approach to CAN media redundancy using off-the-shelf components

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.

Though a solution integrating media redundancy mechanisms and MAC layer functionalities may exhibit a moderate level of complexity, such a specialized design will be too costly. On the other hand, the architecture of current CAN controllers does not favor the implementation of media switching strategies with off-the-shelf components (Figure 3). CAN controllers include a *bit synchronization* module, that internally recovers the receiving clock. To maintain synchronism on media switching, a smooth data signal transition would be required. Bus data would have to be delayed by one bit time, until a decision is available, but that prevents a transmitting node from correctly performing bus state monitoring. One possible solution would be to allow abrupt transitions without being concerned with a possible loss of synchronism, and rely on MAC level mechanisms to recover from the error. Thus, the fundamental obstacles to the implementation of CAN bus redundancy using media switching and off-the-shelf components do concern both complexity and effectiveness.

However, some questions remain: how much of the complexity associated with the architecture of Figure 3 would really be needed to ensure CAN non-stop operation? Would it be possible to provide an equivalent functionality with a simpler architecture?

The 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. 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, nodes at the *out-partition* receive a recessive signal from the (idle) failed media.

As a consequence, we came up with this Columbus' egg idea of extending the bare properties of CAN bus operation to the media interface level. 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.

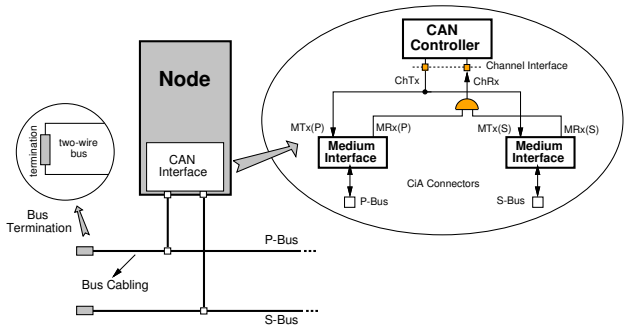

Figure 4. The Columbus' egg idea for bus media redundancy in CAN

This solution can be implemented by a conventional AND gate, to be inserted between the medium interfaces and the CAN controller, as exemplified in Figure 4 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.

5 System Model

In this section we define a model for a redundant media network infrastructure, explain our fault assumptions and discuss a relevant set of CAN physical-level properties.

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). We denote the channel transmitter and the channel receiver of node n as Ch^n_{Tx} and Ch^n_{Rx}, respectively. If the Channel is composed of several media $m \in \mathcal{M}$, we use $M^n_{Tx}(m)$ and $M^n_{Rx}(m)$ 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 module (as depicted

[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* [5].

[4] Informally the *omission degree* is the number of consecutive omission errors of a component in an interval of reference [16].

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

in Figure 5), which internally connects to each Medium, accordingly to a given strategy.

For simplicity of exposition, we will omit the superscript identifying the node, 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.

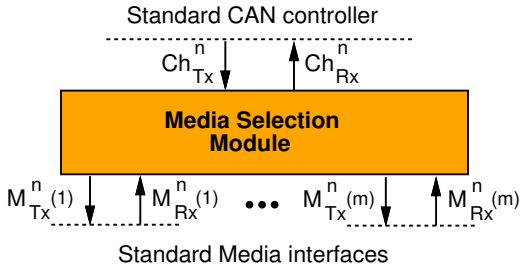

Figure 5. Channel and Media interfaces

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 an interval of reference.

In the context of CAN network components, an omission is an error that destroys a data or remote frame. It does not matter how many individual bits get corrupted: a single omission is accounted for each destroyed data/remote frame.

The fault assumptions are drawn from our previous works on CAN (e.g. [13]). 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 an 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.

[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 6. CAN physical level properties

The omission degree is also a general measure of the reliability of the CAN components to transient errors: failure bursts affect at most f_o transmissions in an interval of reference. However, 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.

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 $t_B^s(p)$ 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 $v_B^s(p)$. Given the current CAN implementations, the logical value one represents the *recessive* state and the logical value zero represents the *dominant* state.

Figure 6 presents a relevant set of CAN physical layer properties. The PCAN1 property formalizes the *quasi-stationary* propagation of signals in CAN where, unlike longer and faster 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 and are exploited in our method to implement bus-based media redundancy in CAN.

6 Media Redundancy Strategies

We now use the model defined in Section 5 to discuss our implementation of bus-based media redundancy in CAN.

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 [10, 1]. 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 or fiber optics technology.

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 to the Media interface level, taking into account the PCAN1 property: the receive signals of each Medium interface are combined in an AND function (equation 2), before interfacing the MAC layer.

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

where, $\mathcal{M}$ is the set of Medium interfaces. For example, in the dual-media architecture of Figure 4, $\mathcal{M} = \{P, S\}$.

This technique provides resilience to Medium partitions (e.g. A and B failures in Figure 2) and to stuck-at-recessive failures (e.g. failure D in Figure 2), 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 may be 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 $\mathcal{T}_{skd}$. This parameter is important because it provides an upper bound for the delay in the detection of a stuck-at-dominant Medium failure. We account for two different contributions:

$$\mathcal{T}_{skd} = (\mathcal{T}_{stuff} + 1) + 2\,.\mathcal{T}_{eflag} \tag{3}$$

The first term of equation (3) represents the minimum number of consecutive dominant bits violating the bit-stuffing coding rule, being $\mathcal{T}_{stuff}$ the bit-stuffing width. The second term of equation (3) represents the maximum duration of an error signaling action, being $\mathcal{T}_{eflag}$ the normalized duration of an error flag [5].

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:

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

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 $\mathcal{T}_{skd}$, the Medium has failed. The values *true* and *false* are represented by a logical one and a logical zero, respectively.

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

$$M_d(m) = M_{skd}(m) \tag{5}$$

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.

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

With this technique we have made our architecture resilient to Medium stuck-at-dominant failures, such as failures C or G in Figure 2.

7 Omission Error Detection

In this section we extend the functionality of our architecture by introducing mechanisms able to detect and to account for omission errors. These errors may 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. Another cause may be the incorrect dimensioning of CAN physical layer parameters.

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 [5]:

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 N4, a Ch_{EFS} 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 Frame Sequence* ($\mathcal{T}_{EFS}$) is equal for data/remote and for error/overload frames and includes the three bit intermission [5]. Equation (7) takes into account that a transmission may start at the last bit of the intermission, being $\mathcal{T}_L = \mathcal{T}_{EFS} - 1$.

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

If a data or remote frame transmission ends without errors, a *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 [5, 13].

$$Ch_{Fok} \mapsto \begin{cases} true & \textbf{if } Ch_{Rx} = rdrrrrrr \\ false & \textbf{when } Ch_{EFS} \end{cases} \quad (8)$$

Conversely, 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 through a detectable sequence of dominant bits (assumption N6), violating the bit-stuffing coding rule.

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

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:

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

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_{EFS} becomes true.

The Medium receive signals are not monitored in the interframe space, i.e. in the period between two consecutive data or remote frames. Network errors occuring in this period are not accounted as frame omissions.

Detecting Medium omissions

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

$$\begin{aligned} M_{Fm-s} &= \sum_{m \in \mathcal{M}} M_{Fm}(m) \\ M_{Oer}(m) &= Ch_{Fok} \wedge M_{Fm}(m) \\ M_{Och}(m) &= Ch_{Err} \wedge \neg M_{Fm}(m) \wedge M_{Fm-s} \\ M_{Uer}(m) &= (Ch_{Err} \wedge \neg M_{Fm-s}) \vee (Ch_{Err} \wedge M_{Fm}(m)) \end{aligned}$$

that we use in the accounting of Medium m omission degree:

$$M_{Od}(m) = \begin{cases} M_{Od}(m) + 1 & \textbf{if } M_{Oer}(m) \vee M_{Och}(m) \\ M_{Od}(m) & \textbf{if } M_{Uer}(m) \\ 0 & \textbf{if } Ch_{Fok} \wedge \neg M_{Fm}(m) \end{cases} \quad (11)$$

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 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. 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[7].

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.

Additional remarks on omission errors

As a general rule, a Medium that exceeds its omission degree bound should be declared failed and its contribution to equation (6) disabled. However, we have identified two situations where despite the occurrence of omission errors, those should not be accounted for in the omission degree:

- **common-mode errors**, with origin, for example, in a node with a failed transmitter or a failed receiver [17]. This scenario is easily detectable, because no media will report *frame mismatches*;
- **single Medium errors**, that nevertheless generate *frame mismatches* in all media. This scenario calls for fault treatment procedures where some "incorrect" media are temporarily disabled (*quarantined*), to allow operation to proceed with a "correct" set.

A detailed discussion of CAN-oriented *quarantine* techniques, similar to those introduced in [15] for LANs, is out of the scope of this paper and it will be reported in a future work.

8 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.

All the required functionality can be easily integrated in a single, medium capacity, Programmable Logic Device.

References

[1] Alcatel. *MTC-3054 CAN Interface Data Sheet*, Dec. 1995.

[2] J. Azevedo. *The WorldFIP protocol*. WorldFIP International Technical Center, Antony, France, Nov. 1996.

[3] CiA - CAN in Automation. *CAN Physical Layer for Industrial Applications - CiA/DS102-1*, Apr. 1994.

[4] 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.

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

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

[7] LONWORKS 78kbps self-healing ring architecture. Echelon Marketing Bulletin, Aug. 1993.

[8] C. Mateus. Design and implementation of a non-stop Ethernet with a redundant media interface. Graduation Project Final Report, IST, Lisboa, Portugal, Sept. 1993. (in portuguese).

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

[10] Philips Semiconductors. *TJA1053 - Fault-tolerant CAN transceiver*, Oct. 1997.

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

[12] L. Rodrigues, M. Guimarães, and J. Rufino. Fault-tolerant clock syncronization in CAN. In *Proc. of the 19th Real-Time Systems Symposium*, Madrid, Spain, Dec. 1998. IEEE.

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

[14] EN50170 - PROFIBUS. Presentation Slides - Siemens PROFIBUS Interface Center, Mar. 1998.

[15] P. Veríssimo. Redundant media mechanisms for dependable communication in Token-Bus LANs. In *Proceedings of the 13th Local Computer Network Conference*, Minneapolis-USA, Oct. 1988. IEEE.

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

[17] P. Veríssimo, J. Rufino, and L. Ming. How hard is hard real-time communication on field-buses? In *Digest of Papers, The 27th Int. Symposium on Fault-Tolerant Computing Systems*, pages 112–121, Washington - USA, June 1997. IEEE.

[7] 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.