Harmonizing Dependability and Real Time in CAN Networks *

Guillermo Rodríguez-Navas, Manuel Barranco, Julián Proenza
Unversity of the Balearic Islands
Palma de Mallorca, SPAIN
vdmigrg0@uib.es

Abstract

The particular mechanisms which the Controller Area Network (CAN) protocol defines to guarantee dependable communication with real-time constrains makes this fieldbus very suitable for many small-size distributed embedded systems. Nevertheless, it has been also reported that CAN protocol exhibits some liabilities concerning both dependability and real-time; though solutions to these problems have been already suggested. In this paper, an analysis of the atomic broadcast property in the Time-Triggered CAN (TTCAN) protocol is used to reason about the compatibility between techniques intended to improve CAN dependability and techniques intended to improve CAN real-time behavior. Authors claim that, although a number of problems have been solved individually, further research has to be carried out in order to properly integrate such solutions.

1. Introduction

Nowadays, the *Controller Area Network* (CAN) fieldbus [5] is one of the preferred solutions to implement the communication subsystem of small-size distributed embedded systems. Some of the reasons for the success experienced by CAN are the particular mechanisms it implements in order to achieve dependable communication with real-time constrains. Nevertheless, CAN protocol still exhibits some problems concerning dependability and real time performance, which have been thoroughly reported in the literature (e.g. [13,15]). Although solutions to such problems have been already suggested, it has not been studied whether these solutions can be combined in order to improve the properties of a CAN network in a balanced manner.

1.1. Dependability properties of CAN

CAN presents five error detection mechanisms, which lead to five different kinds of errors: bit error, stuff error, CRC error, ACK error and format error [5]. Whenever a node detects one of these errors, it signals this situation to the rest of nodes, by transmitting a specific bit pattern (the so-called *error flag*) on the bus which overwrites the transmitted frame. As a consequence of this error flag, all the nodes detect an error condition and reject the frame. This mechanism allows *globalization* of local errors and is assumed to guarantee *data consistency* within the nodes. Once a frame is rejected, the transmitter CAN controller immediately reschedules the frame for retransmission. In this way, any message issued to the network will eventually be received by all the correct nodes, provided the transmitter does not fail. This property is known as *atomic broadcast*.

However, the error-detection and error-recovery mechanisms of CAN may fail in some situations. It has been reported that, due to the bit stuffing, CAN controllers are not able to detect some very specific combinations of multi-bit errors[16]. Moreover, it has been reported that in the presence of inconsistent errors in the last bits of the end of frame field [13,10], the atomic broadcast property is not guaranteed. In some cases, some nodes may receive the same message twice (a so-called *Inconsistent Message Duplicate* failure) whereas in other cases, some nodes may not receive a message which the others do receive (a so-called *Inconsistent Message Omission* failure).

Due to the important benefits that a distributed system may obtain from having atomic broadcast (e.g. in order to maintain replica determinism [7]) some solutions have been proposed to achieve this property. The solutions suggested in [13,9] are higher-layer protocols which are based on message exchanges. The solution proposed in [6] is based on a hardware circuit which substitutes the transmitter whenever it fails. In [10], a modification to the CAN protocol, the *MajorCAN* protocol, is suggested. The main difference between the latter solution and the three former ones is that MajorCAN does not rely on message retransmissions in order to achieve data consistency.

The basic philosophy of MajorCAN is that frames with errors in the last bits can be accepted, though in such a case, the node accepting the frame must signal this to the rest of nodes by means of a specific bit

* This work has been partially supported by the Spanish MCYT grant DPI2001-2311-C03-02, which is partially funded by the European Union FEDER programme.

pattern, which is called *extended error flag*. Therefore, MajorCAN ensures data consistency in a frame by frame basis. Nevertheless, the main drawback of MajorCAN is that it is not compatible with the standard CAN protocol.

1.2. Real-time properties of CAN

CAN protocol implements a prioritized access to the medium. Each message has a unique identifier, which indicates its priority with respect to the other messages: the lower the identifier, the higher the priority. When the bus is *idle*, any CAN controller is allowed to transmit. If more than one controller attempt to transmit simultaneously then a non-destructive bit-by-bit comparison of the message identifiers is performed. In this way, only the highest priority message is transmitted, whereas the nodes willing to send a message with lower priority back off and schedule their messages for retransmission. This mechanism ensures that messages are serialized on the bus conforming to their priorities. Thus, the *worst-case response time* of any message can be calculated by applying results from processor scheduling [14]. The effect of the channel errors in the response time of a CAN message has been also evaluated by means of bounded fault models [11] as well as non-bounded (probabilistic) fault models [8,3].

Note that, although the response time of a CAN message can be bounded, it presents a high variability, which mainly depends on the error conditions of the channel as well as on the current workload. Since this variability may have a negative impact in the system, a number of extensions to the CAN protocol have been proposed, which aim at solving this issue. Some of them are introduced next.

The *Latest Sent Time CAN* (LST-CAN) protocol [2] is an extension to CAN which is intended to guarantee *timeliness* regardless of environmental error conditions. In LST-CAN, every message is given a time such that if the message has not been sent by this time then it is not transmitted at all. Simulation shows that this mechanism significantly reduces the effect which channel errors have on the response time.

The Time-Triggered CAN (TTCAN) [4] is an extension to CAN which relies on a static TDMA schedule in order to guarantee deterministic response times. A relevant characteristic of this protocol is that automatic frame retransmission upon error is disable, so that an slot assigned for a message cannot be interfered by the retransmission of a previous message. This characteristic has a strong impact on the data consistency of the TTCAN communication, as we will show in Section 2.

The Flexible Time-Triggered CAN (FTTCAN) [1] protocol is another extension to CAN which implements a dynamic TDMA schedule, though with a centralized online admission control. This protocol is intended to guarantee deterministic response time while allowing certain level of flexibility.

2. Atomic broadcast in TTCAN

At this point, some of the liabilities, regarding both dependability and real time, which the CAN protocol presents have been highlighted. It has been also remarked that solutions to these problems are currently available. Therefore, one may think that, for instance, combining a solution to achieve good real-time performance and a solution to achieve atomic broadcast will improve both properties. However, in this Section we show that this is not the case.

First, we show that the problem of data consistency becomes worse when TTCAN is used instead of CAN. After that, we show that most of the traditional solutions for achieving atomic broadcast in CAN are not suitable for the TTCAN protocol. A more complete discussion on this topic can be found in [12]

2.1. Problem statement

The problem of the inconsistency failures in CAN was firstly described in [13]. This work shows that in the presence of *inconsistent* local errors, data consistency is not guaranteed in all cases. In particular, inconsistencies appear when an error occur in the last but one bit of the end of frame field.

As remarked in [13], inconsistencies can drive to two different failures. The first one is called an *Inconsistent Message Duplicate* (IMD) failure, whereas the second one is called *Inconsistent Message Omission* (IMO) failure. Both kind of failures are complementary, in the sense that any inconsistency due to the last but one bit will drive to an IMD, unless the transmitter crashes before being able to retransmit the frame, which therefore would drive to an IMO.

The probability of having some of these failures has been also evaluated in [13]. Analytical results were obtained for a network made up of 32 nodes, with a bit rate of 1Mbps, a overall load of 90% and an average frame length of 110 bits. Table 1 shows the results which were obtained assuming different bit error rates and different probabilities of node failures. Note that IMDs are more likely to occur (in a range between 2.87 $\times$ 10 and 2.84 $\times$ 10^3 IMD/hour) than IMOs (in a range between 3.94 $\times$ 10^{-7} and 3.98 $\times$ 10^{-9}) since an IMO requires a *crash* of the transmitter to happen, and this is a very unlikely failure.

Bit Error Rate (*ber*)	Node failures per hour	IMD/hour	IMO/hour
10^{-4}	10^{-4}	2.84×10^3	3.94×10^{-7}
10^{-5}	10^{-4}	2.86×10^2	3.98×10^{-8}
10^{-6}	10^{-4}	2.87×10	3.98×10^{-9}

Table 1. Frequency of the inconsistency failures in CAN, as calculated in [13].

Nevertheless, in TTCAN this relation between IMDs and IMOs does not hold. Since the automatic

retransmission of frames is not allowed, IMDs cannot happen. Therefore, any inconsistency due to the last but one bit will result in an IMO. The severity of the IMOs in TTCAN has been evaluated in [12]. Table 2 shows the results that were obtained by assuming the same conditions described in [13].

Bit Error Rate (*ber*)	IMO/hour
10^{-4}	2.84×10^{3}
10^{-5}	2.86×10^{2}
10^{-6}	2.87×10

Table 2. Frequency of the inconsistency failures in TTCAN, as calculated in [12].

The results obtained are so high that a solution to guarantee atomic broadcast in TTCAN seems to be mandatory. However, an analysis of the applicability of the solutions for atomic broadcast in CAN to TTCAN shows that most of these solutions are not suitable for TTCAN.

2.2. Applying previous solutions to TTCAN

Two approaches are followed in order to solve the problem of atomic broadcast in CAN networks. The first approach is the one followed in [13,9,6] and relies on message exchanges in order to achieve an agreement between the nodes. This approach, though being suitable for event-triggered systems, exhibits significant drawbacks when it is used in time-triggered systems such as TTCAN. First, these protocols would require reservation of a given bandwidth for retransmission of messages, thus driving to a reduction of the throughput of the network. Moreover, these protocols go against the basics of the TTCAN communication, as they introduce a certain level of unpredictability.

In contrast, the solution proposed in [10] does not rely on message retransmissions. As we briefly explained in the Introduction, the MajorCAN protocol solves the inconsistency problems in a *frame by frame* basis. Therefore the fact of not having automatic retransmission of erroneous frames does not affect the MajorCAN functionality.

3. Conclusions and open issues

Data consistency and predictability are relevant attributes of any critical distributed embedded system. Since there is a growing interest in using CAN for these applications, especially because of the low cost of the components, both properties must be improved in a balanced manner. Nevertheless, current solutions to these problems, though being suitable to solve each problem separately, present important drawbacks when they are combined. Due to this, further research is required in order to properly integrate such solutions.

Our current research interest is focused on the provision of atomic broadcast in those extensions to CAN which improve the predictability of the response time. In the remaining of this Section, some issues which still remain open are introduced.

3.1. Evaluate the severity of the inconsistency scenarios

The first topic we are addressing is the evaluation of the severity of the inconsistency scenarios in LST-CAN, TTCAN and FTT-CAN. Since all these protocols limit the automatic retransmission of errors to a certain extent, they are expected to have a higher probability of inconsistency failures. The way to calculate such a probability is conceptually simple, though in some cases the required mathematics are complex. Figure 1 will help to understand this calculation.

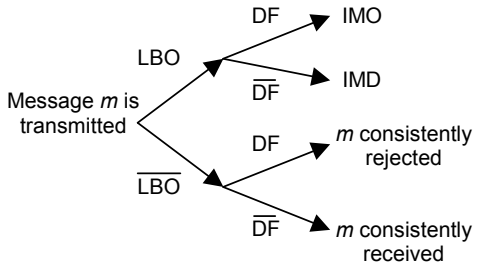

Figure 1. Consequences of inconsistent errors and deadlines failures in CAN

Any message which is transmitted in a CAN network may suffer an error in the last but one bit (LBO error, for short). Moreover, depending on the current load and error conditions, every message may miss its deadline. We call this failure a deadline failure (or DF). Figure 1 shows all the possible combination of these two events, and their consequences. Note that the probability of a node failure is neglected. Therefore, the probability of having a inconsistent message omission turns out to be

$$prob(IMO) = prob(LBO) \times prob(DF \mid LBO)$$

The solution to this equation is trivial for TTCAN. Since any time a message is not transmitted within its corresponding slot can be considered as constituting a deadline failure, the equation can be written as

$$prob(IMO) = prob(LBO)$$

Nevertheless, in the case of LST-CAN and FTT-CAN, this calculation is far from being trivial. First, it requires a better error model of the CAN bus. However, the lack of actual data about error conditions in real CAN systems difficults this work, so calculations usually have to rely on assumptions not very well substantiated (e.g. a Poisson distribution for the channel errors). And

second, this calculation requires a better modeling of the effect of channel errors in the response time of the CAN messages. Though some research has been carried out in this direction [3], only little advances have been achieved, mainly because of the complex mathematics that are required for integrating statistics and scheduling theory.

3.2. Fault injection

Besides the analytical results, there is an interest in testing the behavior of commercial CAN controllers in response to the inconsistency scenarios. However, this test cannot be done because there is no fault injector able to inject the specific error scenarios which drive to an inconsistent message omission. In particular, current fault injectors do not achieve the high spatial and temporal *controllability* which this test would require.

In order to make this test possible, we are working on the design of a physical fault injector which exhibits the desired properties. Our fault injector is made up of a set of specifically designed circuit, which we call *Individual Fault Injectors* (IFI). An IFI is attached to every node of the network. The function of each IFI is to inject faults in the link between the CAN controller and its corresponding transceiver. In this way, the response of the CAN controller (as well as the response of the higher-level fault tolerance mechanisms) in the presence of very specific scenarios can be tested. IFIs are able to work cooperatively in order to build very complex error scenarios. Nevertheless, in order to simplify the complexity of the IFI's circuitry, a initial step is required. Every error scenario which is going to be tested is pre-processed by a software tool, which converts this scenarios in a set of individual instructions for the IFIs. The name of this tool is CANfidant, which states for *CAN Fault Injection Design Assistant*. Through the use of this tool, a high temporal controllability can be achieved with a reduced hardware complexity.

3.3. The MajorCAN protocol

In our opinion, the MajorCAN protocol merits further research as it seems to be the most suitable solution to provide atomic broadcast in LST-CAN, TTCAN and FTT-CAN protocols. A first prototype of a MajorCAN controller has been already developed in programmable logic, but a dependability evaluation of this protocol has not been performed yet.

In a first step, we wish to perform a formal validation of MajorCAN. Particularly, we are working on the validation by means of *model checking*. We also wish to use our fault injector in order to thoroughly test the behavior of the MajorCAN controller. Finally, we are studying the implementation of a MajorCAN controller with a standard CAN controller. Although we are not certain about the achievement of this goal, this would eliminate one of the main drawbacks of MajorCAN; its lack of compatibility with CAN controllers .

References

[1] L. Almeida, P. Pedreiras and J. Fonseca, "The FTT-CAN protocol: why and how", *IEEE Transactions on Industrial Electronics,* Vol. 49(6), 2002.

[2] I. Broster and A. Burns, "Timely Use of the CAN Protocol in Critical Hard Real-Time Systems with Faults", *Proc. of the 13th Euromicro Conference on Real-time Systems,* 2001.

[3] I. Broster, A. Burns and G. Rodríguez-Navas, "Probabilistic Analysis of CAN with Faults", *Proc. of the 23rd Real Time Systems Symposium,* 2002.

[4] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, M. Walther and Robert Bosch GmbH, "Time Triggered Communication on CAN", *Proc. of the 7th Int. CAN Conference,* 2000.

[5] ISO, "ISO11898. Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication", 1993.

[6] M. A. Livani, "SHARE: A Transparent Approach to Fault-tolerant Broadcast in CAN", *Proc. of the 6th International CAN Conference,* 1999.

[7] S.J. Mullender (Ed.), "Distributed Systems, 2nd edition", ACM Press, Addison Wesley, 1987.

[8] N. Navet, Y. Q. Song and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over Controller Area Network", *Journal of Systems Architecture,* Vol.46(1), 2000.

[9] L. Pinho and F. Vasques, "Improved Fault-Tolerant Broadcasts in CAN", *Proc. of the 8th IEEE International Conference on Emerging Technologies and Factory Automation,* 2001.

[10] J. Proenza and J. Miro-Julia, "MajorCAN: A modification to the Controller Area Network to achieve Atomic Broadcast", *IEEE International Workshop on Group Communication and Computations,* 2000.

[11] S. Punnekkat, H. Hansson and C. Norström, "Response time analysis under errors for CAN", *Proc. of RTAS,* 2000

[12] G. Rodríguez-Navas and J. Proenza, "Analyzing atomic broadcast in TTCAN networks", P*roc. of the 5th IFAC International Conference on Fieldbus Systems and their Applications,* to be published*,* 2003.

[13] J. Rufino, P. Veríssimo, G. Arroz, C. Almeida and L. Rodrigues, "Fault-tolerant broadcasts in CAN", *Digest of papers, The 28th IEEE International Symposium on Fault-Tolerant Computing,* 1998.

[14] K. Tindell, A. Burns and A. J. Wellings, "Calculating controller area network (CAN) message response time", *Control Engineering Practice,* Vol. 3(8), 1995.

[15] M. Törngren, "A perspective to the Design of Distributed Real-time Control Applications based on CAN", *Proc. of the 2nd International CAN Conference,* 1995.

[16] E. Tran, *Multi-bit Error Vulnerabilities in the Controller Area Network Protocol*, Carnegie Mellon University, thesis, 1999.