A first design for CANsistant: a mechanism to prevent inconsistent omissions in CAN in the presence of multiple errors

Julián Proenza and Ernesto Sigg
Dept. de Matemàtiques i Informàtica, Universitat de les Illes Balears, Palma de Mallorca, SPAIN
julian.proenza@uib.es and ernesto.sigg@gmail.com

Abstract

Despite the significant advantages of the Controller Area Network (CAN) there is an extended belief that CAN is not suitable for critical applications, mainly because of several dependability limitations. One of them is its limited data consistency. Several solutions to this problem have been previously proposed but they are not able to efficiently ensure consistent broadcasts in the presence of multiple channel errors. This paper introduces a circuit called CANsistant, that detects all scenarios potentially leading to the inconsistent omission of a frame in the presence of up to 4 channel errors and, if necessary, retransmits the affected frame.

1. Introduction

The Controller Area Network (CAN) [3] protocol is a fieldbus communication protocol that was first devised for in-vehicle control applications and that has been widely adopted in many other areas within the distributed embedded control systems field. CAN is nowadays a mature technology whose tremendous success has been mainly caused by its error control features, low latency, network wide bus access priority and real-time response. In addition, CAN's widespread use has caused the price of its components to drop to some levels where other protocols cannot compete.

Despite these significant advantages, there is an extended belief that CAN is not suitable for critical applications, mainly because of the following dependability limitations [6]: (1) Limited data consistency; (2) Limited error containment; (3) Limited support for fault tolerance and (4) Lack of clock synchronization. Nevertheless, several researchers state that CAN will be able to support safety-critical applications if these limitations are overcome with the proper enhancements [6]. This possibility is very appealing for many application domains, since CAN components are much cheaper than those of the natural competitors of CAN in highly dependable systems, e.g., FlexRay or TTA, and because the use of CAN permits to take advantage of the know-how and expertise that engineering teams have gained in using this technology during the last decades.

The recently started CANbids (*CAN-Based Infrastructure for Dependable Systems*) project purports to design, implement and validate a CAN-based infrastructure for supporting the execution of highly-dependable distributed control applications. CANbids will use as building blocks various mechanisms and enhancements intended to overcome the aforementioned CAN dependability limitations.

This paper focuses on how CANbids overcomes CAN's limited data consistency. In fact, the CAN specification [3] claims that it presents *data consistency*. This means that in a CAN network it is guaranteed that a frame is either simultaneously accepted by all nodes or by none. Besides the existence of the *error passive state* [9] in which this property does not hold, Rufino *et al.* identified [9] some specific scenarios, with all nodes in the error active state, in which some nodes receive a frame and some others do not, which is called an *inconsistent message omission*. The same authors proposed a set of protocols to be executed on top of CAN to solve the problem [9]. It is important to say that all these protocols require the transmission of at least an additional frame for each frame that would have been transmitted in the network, even if this frame was consistently transmitted.

Afterwards, Livani [5] presented his SHAdow REtransmitter (SHARE): a mechanism to be included in the network as a regular node and that is able to detect the bit pattern that according to Rufino *et al.* indicates that a frame-level inconsistency is possible, and retransmit the potentially inconsistently received frame. This had the significant advantage of only requiring the transmission of extra frames in case an inconsistency was possible.

In a later analysis, new scenarios of inconsistent communication were identified in which both CAN, the proposed higher-layer protocols and SHARE fail [7]. Those scenarios are characterized by the presence of multiple bits affected by errors in the channel.

In order to achieve the advantages of the SHARE approach this paper presents a mechanism that is capable of performing the function of SHARE even in the presence of multiple errors in the channel. Said mechanism is called CANsistant (*CAN Assistant for Consistency*).

2. Related Work

In order to illustrate the scenarios of Rufino *et al.* let us consider the case in Fig. 1. A disturbance corrupts the last

 doi:10.1109/ETFA.2009.5347236

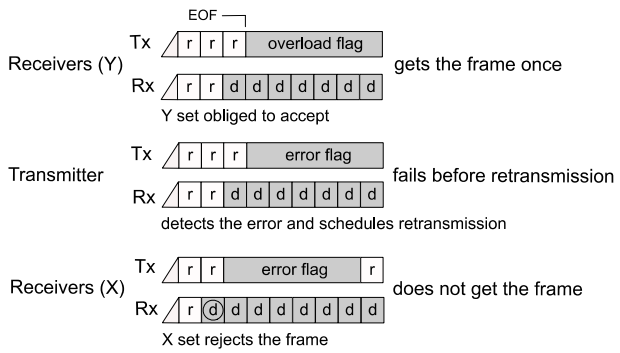

Figure 1. Inconsistency scenarios in CAN [9]

but one bit of the *End Of Frame* field (EOF) received by the set of nodes called X through their Rx input. Therefore instead of a recessive value, they see a dominant one in this bit. In the next bit, these receivers start the transmission of an error flag (i.e. six consecutive dominant bits) through their Tx output. The *dominant* first bit of this error flag is seen by the nodes belonging to set Y and by the transmitter (through their Rx input) as an error in the last bit of their EOF, thus they will start the transmission of their own error flags in the next bit. The nodes belonging to X will reject the frame, the nodes belonging to Y will accept the frame following the CAN's last bit rule and the transmitter will schedule the frame retransmission. If the transmitter suffers a hardware failure that prevents it from completing the retransmission, the nodes belonging to Y receive the frame whereas those of X will never do it, i.e. an *inconsistent message omission* (IMO) occurs.

As indicated above, SHARE [5] is an effective mechanism designed to detect this scenario and autonomously retransmit the affected frame, in order to provide a backup for the original transmitter.

2.1. SHARE description

A SHARE is a dedicated node that has to be added to the CAN network. This node preventively stores each frame that is transmitted through the CAN bus. If a specific pattern is detected at the end of a frame, SHARE immediately tries to retransmit it. This retransmission is a regular one, in the sense that it abides by the CAN's arbitration mechanism. In contrast, if the pattern is not detected, SHARE simply discards the stored frame and restarts for the next frame.

The specific pattern that SHARE detects is a sequence of 6 consecutive dominant bits starting at the last bit of the EOF [5], which is characteristic of the scenario described above (Fig. 1). In case the pattern is detected, the retransmission performed by SHARE covers the potential crash of the original transmitter. In case this transmitter does not crash, its own retransmission would take place bit-by-bit synchronized with the retransmission by SHARE. According to SHARE's description, this simultaneous transmission should not cause any error since both nodes would send for each bit exactly the same value. However this is a controversial statement. Given that in CAN only receivers adapt the duration of the bit time to the edges caused by the transmitter, having two transmitters that will naturally drift apart could make receivers to detect errors in the bit stream by the end of the frame.

2.2. Scenarios with multiple channel errors

Another scenario that was later described in [7] happens if the transmitter can not see the error flag in the last bit of EOF due to an additional disturbance in that bit. In this case, the transmitter does not even try to retransmit the frame and the same IMO takes place. Note that this scenario has a higher probability of occurrence than the one described in Fig. 1, as estimated in [7]. Indeed an additional channel error is much more likely to occur than the crash of the transmitter.

Unfortunately SHARE's design did not take into account the possibility of having additional bits affected by errors. Note that if SHARE had the same view of the channel as the transmitter in the scenario that has been just discussed, then it will neither have considered that a retransmission was required. This is the origin of our goal of designing a mechanism that is able to detect IMOs even in the presence of multiple channel errors, but there are more reasons for it.

Although SHARE was designed for systems using CAN in its original way of operation, there is an increasing interest in disabling the automatic retransmission of frames upon error occurrence. This *single-shot transmission* mode is natural in TDMA and FTDMA-like systems [1], or in real-time ones that purport to prevent the jitter caused by retransmissions, such as CANbids. In these kinds of systems, since they are usually intended for critical applications, it is also important to be able to detect inconsistencies. Thus, they could make good use of a mechanism that provided the IMO-detection functionality offered by SHARE. Moreover, disabling the automatic retransmission dramatically increases the probability of IMOs as calculated in [8], since it is not necessary that the transmitter crashes to suffer an IMO. Furthermore, it is also important to be able to detect IMOs in the presence of multiple channel errors, since this situation (i.e. IMO together with multiple errors) is also more likely to happen than in a normal CAN network.

3. CANsistant operation description

Among the two basic functions of CANsistant, which are potential IMO detection and frame retransmission, we are going to focus on the first one, since it is the one that has to take into account the possibility of multiple channel errors.

The first aspect to be determined is in the presence of how many channel errors is CANsistant going to be able to properly detect all possible scenarios leading to an IMO, without

raising too frequent and avoidable false alarms. This fundamental parameter will be denoted as m. In order to determine its value it is necessary to take into account that in the absence of errors, after the EOF field, three recessive bits are transmitted which are called *intermission*. After these bits, the bus is idle and a new frame may start in any bit with a first dominant value called *Start Of Frame* (SOF), followed by a series of bits that are the new frame's *identifier* [3].

Our starting point for the determination of m is the scenario described in Fig. 1. Let us assume that while the receivers belonging to X detect the first dominant value in the last but one bit of EOF, CANsistant does not see that value because it is not affected by this first error. Moreover let us assume that CANsistant suffers a series of consecutive $m-1$ errors that prevent it from seing the first $m-1$ dominant bits of the error flag transmitted by the nodes belonging to X. The question is then for what value of $m-1$ it would be impossible for CANsistant to tell whether the first dominant bit it detects could correspond to one of the bits of an error flag or to the normal operation of the protocol in the absence of errors. This second possibility is exactly what happens when $m-1=4$. In that case the first dominant bit of the error flag that would be detected by CANsistant would appear in the position that in complete absence of errors would have been occupied by the first bit of the next frame, i.e. SOF, which has a dominant value as indicated above. Thus, for $m=5$, each time a new frame with an identifier with several consecutive dominant values in its most significant bits starts right after the three bits of intermission, a false IMO alarm will be raised. Since this happens in the absence of errors, we conclude that it would significantly and unnecessarily increase the number of false alarms and we establish as our goal to detect all IMOs for $m=4$.

3.1. Other CANsistant parameters

As illustrated by the analysis above, designing CANsistant to be able to detect inconsistencies even in the presence of multiple channel errors opens room for deciding that an IMO could have happen even if the first dominant value is detected some bits after the last bit of EOF. Indeed the view that CANsistant has of the channel has the same possibilities of being affected by errors as the view of any other node of the network. But not only the detection of the first dominant bit can be affected, multiple errors can also alter the detection of the dominant bits that are supposed to come after the first one. We have reduced the problem of dealing with the effects of multiple errors to determining the following parameters:

(1) *First Dominant Detection Window* (FDDW): This window corresponds to the group of bits of any frame where CANsistant has to find the first dominant bit in order to consider that an inconsistency may have happened. If CANsistant does not detect any dominant bit inside the FDDW, it will deem that an IMO is not possible for this frame.

(2) *Number of Dominant bits* (ND): This is the number of dominant bits that CANsistant has to find after having detected a first dominant bit in the FDDW, in order to signal a potential IMO and schedule the retransmission.

(3) *Additional Dominant Detection Window* (ADDW): This window corresponds to the group of bits starting with the next bit after the first dominant value detected within the FDDW. If and only if CANsistant finds ND additional dominant bits in the ADDW, it signals a potential IMO and schedules the retransmission of the affected frame.

In order to determine the position of both windows and the value of ND we propose to start by establishing the position of the FDDW. This can be done using the same kind of reasoning that was used to determine the maximum number of errors in whose presence CANsistant is able to detect all possible IMOs. Obviously the first bit of the FDDW is the last bit of EOF since, in the presence of only one channel error, the first dominant value must be detected in this place by any node (including CANsistant), in case the only scenario that may cause an IMO with a single channel error (depicted in Fig. 1) had happened. On the other hand, the last bit of the FDDW is the last bit of intermission since, in the presence of the additional $m-1=3$ errors that are possible, the actual reception of the first dominant by the nodes (or at least by CANsistant) could be delayed at most until this bit.

Considering again the scenario represented in Fig. 1 it is also possible to determine the value of ND. In case CANsistant was the only one suffering additional errors during the transmission of these two overlapped error flags, and in case it suffered the maximum number of remaining errors ($m-1=3$) in the very first dominant bits of said flags, then it would have detected the first dominant value in the last bit of intermission and, due to the fact that it has already suffered the maximum number of errors, it would see the next 3 bits as dominant. We conclude that the value of ND would be 3. On the one hand, considering ND $=4$ would cause the scenario just described to be not recognized as an IMO. And on the other hand, considering ND $=2$ would cause many scenarios to be mistaken as IMOs since for an IMO to take place at least 7 dominant values have to be transmitted and, according to our fault assumptions, at most 3 of them can be affected by additional errors, therefore any node must be able to see at least 4 of them.

Finally, the ADDW by definition starts in the next bit after the first dominant that is detected. The determination of the last bit of this window requires additional considerations. Although the basic scenario causing an IMO is the one shown in Fig. 1 and in this scenario the last bit of the ADDW should be the third one after intermission (the last one represented in Fig. 1), it is true that additional errors could prevent all nodes from seeing the first bits of the error flag transmitted by the receivers belonging to the X set. This would cause a delay in the transmission of the error flag by the receivers of the Y

set and by the transmitter. However it is clear that as soon as no error is detected, the next bit will have a dominant value. Thus, we can see the sequence to be detected as a minimum of 7 dominant bits that at most can be interrupted by 3 erroneous recessive bits. This implies that actually placing the last bit of the ADDW in the bit proposed (the third one after intermission) guarantees that CANsistant would be able to detect ND $= 3$ dominant bits within the ADDW whenever an IMO had happened. Extending this window to include later bits would simply increase the number of false alarms.

4. On-going work on CANsistant

Even though it may seem that the CANsistant's IMO detection, in the way it has been presented, requires no further proof of correctness, in fact it does need it, due to the many different scenarios that are possible when we consider any distribution of the m erroneous bits and that each node can see or not see each one of these bits and react accordingly. In order to ensure that CANsistant is able to detect IMOs caused by any possible scenario, we are currently performing a formal verification using model-checking techniques. More specifically we have modelled a system with different CAN nodes and a CANsistant device with the UPPAAL [4] model-checker and we are using its verification engine for checking the appropriate properties. Due to space limitations we cannot provide further details on this verification, however we can say that the models follow the modelling patterns that were already used in [2] for the MajorCAN protocol. The results obtained so far show the correctness of the design and that reducing by a single bit the value of ND or the end bit of the ADDW or reducing the length of the FDDW leaves room for IMOs that are not detected by CANsistant. Obviously with the actual design CANsistant will generate some false alarms but this is inherent to the inaccurate information that can be obtained by simply sensing the value of the bus in the presence of multiple errors. In any case the priority is to be able to detect all possible IMOs rather than not generating false alarms. Additional future work is to be carried out in the quantification of the probability of false alarms.

Besides the formal verification of this mechanism, we are also implementing a prototype that will be added to the hub of a network that presents a star topology and is called CANcentrate [6]. The CANsistant functionality is designed to be applicable both in a regular CAN bus and in a CAN-based star. The inclusion of the mechanism in the CANcentrate hub is a first step towards the construction of a complete CAN-based dependable architecture, which is the goal of the CANbids project. In this architecture the automatic retransmission of frames that characterizes the CAN protocol is to be restricted in order to ensure higher levels of predictability. Since these kinds of restrictions significatively increase the chances of IMOs in a CAN-based system, it is necessary to include mechanisms to at least detect the possibility of IMOs, for the system to be able to later take the appropriate actions that, in any case, will prevent the potential conflicts that a simultaneous transmission of the same frame by several nodes could have caused.

5. Summary

We have presented a first design of a mechanism that can be added to any CAN network to detect all possible inconsistent message omissions in the presence of up to m erroneous bits in the channel. We have determined that the maximum value of m is 4 and we are currently model-checking our design and developing a prototype to be included in a CAN-based network that presents a star topology.

6. Acknowledgement

This work was supported in part by the Spanish Science and Innovation Ministry with grant DPI2008-02195, and in part by FEDER funding.

References

[1] L. Almeida, P. Pedreiras, and J. A. Fonseca. The FTT-CAN protocol: Why and how. *IEEE Transactions on Industrial Electronics*, 49(2), December 2002.

[2] M. Bonet, G. Donaire, and J. Proenza. Modelling MajorCAN with UPPAAL. In *Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2007), Patras, Greece*, September 2007.

[3] ISO. *International Standard 11898 – Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High-Speed Communication*. 1993.

[4] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, Oct. 1997.

[5] M. Livani. SHARE: A transparent approach to fault-tolerant broadcast in CAN. In *Proceedings of the 6th International CAN Conference*, 1999.

[6] J. Pimentel, J. Proenza, L. Almeida, G. Rodríguez-Navas, M. Barranco, and J. Ferreira. Dependable automotive CAN networks. In N. Navet and F. Simonot-Lion, editors, *Automotive Embedded Systems Handbook*. CRC Press.

[7] J. Proenza and J. Miro-Julia. MajorCAN: A modification to the Controller Area Network protocol to achieve Atomic Broadcast. In *IEEE Int. Workshop on Group Communications and Computations. IWGCC. Taipei, Taiwan*, 2000.

[8] G. Rodríguez-Navas and J. Proenza. Analyzing atomic broadcast in TTCAN networks. In *Proceedings of the 5th IFAC International Conference on Fieldbus Systems and their Applications(FeT03). Aveiro. Portugal*, pages 153–156, 2003.

[9] J. Rufino, P. Veríssimo, G. Arroz, C. Almeida, and L. Rodrigues. Fault-tolerant broadcast in CAN. In *Proceedings of the IEEE 28th Int. Symp. Fault-Tolerant Computing. FTCS-28. Munich (Germany)*, June 1998.