UIB technical report: A-01-2010

Evaluation of different approaches for the media management in ReCANcentrate nodes

David Geßner, Manuel Barranco, Julián Proenza
Dpt. Matemàtiques i Informàtica
Universitat de les Illes Balears, Spain
manuel.barranco@uib.es

Luís Almeida
IEETA / DEEC-FEUP
Universidade do Porto, Portugal
lda@fe.up.pt

July 7, 2010

Abstract

CAN, due to its non-redundant bus topology, presents some reliability problems which limit its use for highly reliable systems. In order to increase CAN's reliability we have proposed a CAN-compliant replicated star topology, ReCANcentrate, which incorporates a series of fault-treatment and fault-tolerance mechanisms in its hubs and nodes. This paper introduces different potential approaches for the media management in ReCANcentrate nodes and explains why the approach we are currently implementing is superior in terms of fault-tolerance.

1 Introduction

Controller Area Network (CAN) is a widespread, inexpensive and electrically robust multi-master serial bus suitable for real time communication systems. Unfortunately, because of its bus topology, it presents multiple points where a single fault can prevent the communication between several nodes [1]. These points are called *points of severe failure* [2]. Moreover, within a CAN network, data consistency [3] is not always guaranteed. There are some error scenarios affecting the last-but-one bit of a frame where some nodes accept a frame while others reject it [3, 4]. The existence of points of severe failure and the lack of a guarantee of data consistency are two of the most important shortcomings that make the use of CAN in highly reliable systems, e.g. in safety-critical ones, controversial.

Some solutions have been already proposed to achieve data consistency in CAN, e.g. [3, 4]. However, the elimination of all points of severe failure from CAN was still an open issue. To overcome this limitation, we developed a CAN-compliant replicated star topology, called ReCANcentrate, which can be fully implemented with commercial of the shelf (COTS) components [2].

ReCANcentrate's basic architecture is depicted in Figure 1. It includes two hubs with fault-treatment capabilities which are connected to each other through interlinks,

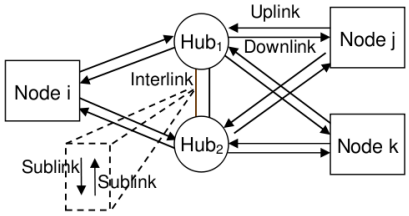

Figure 1: ReCANcentrate architecture.

which are themselves comprised of a sublink for each direction. Moreover, each node is connected to each hub through a link comprised of an uplink and a downlink [2]. Each hub isolates faulty nodes and links and is also able to isolate the other one if it is faulty.

Regarding the wired-and functionality of a CAN bus [5], it is performed by the hubs. Each hub couples the signals received from the nodes through its uplinks in an internal AND-gate. For a particular hub, this coupled signal is called the *contribution* of that hub. Each hub then sends its contribution to the other hub using the interlinks. Once a hub receives the other hub's contribution it couples it with its own contribution. The final result is then broadcast to the nodes through its downlinks. This coupling ensures that each hub sends the same frame, bit by bit, through its non-faulty downlinks; thereby enforcing a single logical broadcast domain. Moreover, since the hubs perform the coupling in a fraction of the bit time, each node will receive each bit value quasi-simultaneously through both its downlinks [2].

Since the traffic is replicated in both hubs, it is necessary to provide nodes with the adequate media management mechanisms, i.e. with mechanisms that allow each node to transmit and receive through the replicated star, while tolerating faults. More specifically, we intend to design media management mechanisms that allow each node to use the replicated media as a single CAN channel. Moreover, our goal is to also take advantage of the replicated traffic in order to tolerate some of the inconsistency error scenarios of CAN.

This paper presents several approaches for the media management in the nodes and explains the advantages in terms of fault-tolerance of the approach currently developed for ReCANcentrate.

2 Fault model

ReCANcentrate's fault model includes stuck-at, medium partition, shorted medium and bit-flipping faults and furthermore assumes that at least one hub is non-faulty and that there is at least one non-faulty sublink in each direction [2]. We also include CAN controller and CAN transceiver stuck-at and bit-flipping faults in ReCANcentrate's fault model.

Notice that from the nodes point of view, faults at the media, hubs, transceivers and CAN controllers manifest themselves as stuck-at or bit-flipping bits received through a downlink [6]. For the sake of simplicity, from now on, we will say that the node detects faults at its downlink, even though faults can occur in other network locations.

Notice that it is practically impossible to know the real probabilities of fundamental dependability parameters such as component failure rates or the proportion with which

a component fails exhibiting the different failure modes included in our fault model. Fortunately, it is still possible to estimate what are widely assumed as realistic values for these probabilities and, then, to model CAN and ReCANcentrate to compare their dependability properties. Particularly, it is possible to assume that stuck-at dominant, stuck-at recessive and bit-flipping faults are equiprobable and, afterwards, to perform sensitivity analyses with respect to them. Since, in principle, none of the failure modes can be considered as negligible, the approaches for the media management in the nodes should deal with all of them.

A comparison of the dependability of CAN and CANcentrate, the predecessor of ReCANcentrate, has already been done [7]. But it is still ongoing work for ReCANcentrate itself.

3 Approaches for the Media management in ReCANcentrate nodes

We will consider single CAN controller approaches and two CAN controller approaches for the media management in the nodes.

3.1 Single CAN controller approaches

The first approach we will consider uses one CAN controller per node with a switching mechanism which allows each node to connect its controller to either one or the other hub. The switching mechanism could be implemented with a multiplexer that is controlled by the node's microcontroller. When the microcontroller detects that it cannot communicate through the CAN controller it currently uses, e.g. through the CAN controller's error warning notification[1], it could connect it to the other hub using the multiplexer.

The biggest caveat to this approach is that a node cannot receive frames between the instant where the reception from one hub fails and the instant its controller switches to the other hub. This loss of frames may lead to data inconsistencies and therefore we consider this solution unacceptable.

The second approach we will consider, which has been previously implemented for an experimental assessment of ReCANcentrate [8], is depicted in Figure 2 and is similar to an approach proposed by Rufino *et al.* [4]. It incorporates for each node one CAN controller with four CAN transceivers grouped in two pairs. Each pair connects to a different hub using one transceiver for the downlink and the other one for the uplink. Each of the node's downlinks bifurcates into two branches after having entered the downlink's transceiver. One branch enters an OR-gate while the other enters a *stuck-at dominant detector* (StDD1 and StDD2 in Figure 2), whose output then enters the same OR-gate.

This stuck-at-dominant detector is a simple circuit that outputs a logical '0' as long as the number of consecutive dominant bits received through the corresponding downlink does not exceed a specific threshold. Otherwise, it permanently outputs a logical '1'.

In the absence of faults, each of the OR-gates will receive a stream of zeros from its attached stuck-at dominant detector and, thus, will output the stream received through the downlink. As ReCANcentrate enforces a single broadcast domain, each bit value

[1] CAN controllers typically include a threshold for their error counters called *error warning limit*.

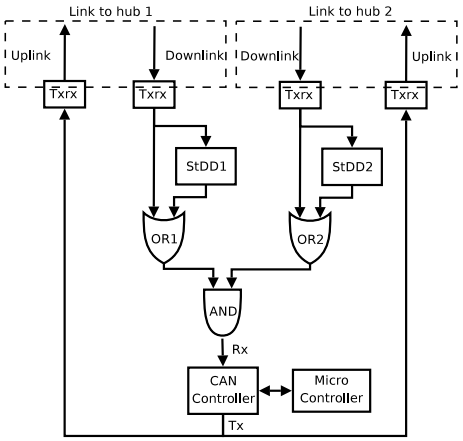

Figure 2: Rufino et al. inspired approach.

is quasi-simultaneously received on both of the node's downlinks. Therefore, as can be deduced from Figure 2, the node's CAN controller will receive the bit stream being broadcast by both hubs.

If a fault manifests as a stuck-at *recessive* downlink, the sequence of ones coming from that downlink will enter the AND-gate and therefore the stream of bits that gets to the CAN controller will be the one from the non-faulty downlink.

If a fault manifests as a stuck-at *dominant* downlink, the corresponding stuck-at dominant detector will be the one producing the sequence of ones that passivates the faulty contribution received through that downlink.

Unfortunately this approach is not recommendable for highly fault tolerant systems. First, it does not tolerate faults that manifest as a bit-flipping downlink. Second, there is another big disadvantage of this approach, and in general of any approach where a node uses a CAN controller that sends the same signal through both uplinks. Notice that since the CAN controller will send error frames through both uplinks in response to errors it detects in any of the downlinks, both hubs will observe these error frames at the respective uplink ports. This will likely cause both hubs to isolate the node even when the fault only affects the connection of the node to one of the hubs. Finally, and this also applies to the other one CAN controller approaches, it would require higher level protocol layers to tolerate Rufino *et al.*'s inconsistency scenario.

3.2 Two CAN controller approaches

In this section we examine the use of two CAN controllers per node. These approaches will all use the same hardware architecture, which is shown in Figure 3. A given CAN controller is connected to only one hub and it uses two transceivers for the connection, one for the uplink and one for the downlink. The difference between the approaches considered in this section is how the node's microcontroller uses its two CAN controllers.

In the first considered approach, one CAN controller is used for both transmission and reception while the other acts as a spare. When the node's microcontroller is notified by the active CAN controller of a communication failure it switches over to the spare controller. This approach is similar to the first one CAN controller per node

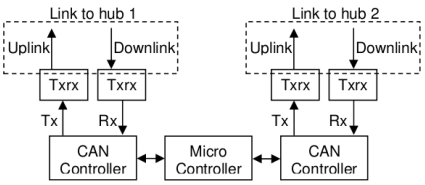

Figure 3: Architecture for two CAN controller approaches.

approach we introduced earlier. It also has the same major problem, there is a delay, during which frames could be missed, between a communication failure on the active controller and the switch to the spare. Therefore, this approach may also lead to a data inconsistency.

The next approach uses the two controllers simultaneously, instead of keeping one as a spare. When the node acts as a receiver, it merely uses each CAN controller to receive a copy of the frame being broadcast. Similarly, when it acts as the transmitter, it simultaneously transmits through its two CAN controllers.

More specifically, when the node acts as as a receiver, its microcontroller will expect to be notified quasi-simultaneously by both its controllers of the reception of a frame. In response, the node's microcontroller can then load the frame from either controller as it will be the same due to ReCANcentrate's single broadcast domain.

If a fault occurs, it will generate errors that block the communication in all the ReCANcentrate domain. This is because since ReCANcentrate enforces a single broadcast domain, all CAN controllers will be signalling error frames as long as the fault continues generating errors. If the fault is temporary, the communication will be reestablished as soon as the fault becomes inactive. Otherwise, the communication will only resume after the fault is isolated at the corresponding hub ports.

Notice that after a fault is isolated at a given hub port, the CAN controller directly connected to that port will not be able to communicate any more. Therefore, the corresponding microcontroller will observe that only one of its CAN controllers notifies about transmissions and/or receptions. The node can handle this situation just accepting as valid the notifications received from that surviving CAN controller. Additionally, it can discard the CAN controller that cannot communicate, after observing that this controller omits a specific number of notifications. Moreover, if the CAN controller isolated at the hub port continues detecting errors, sooner or later, its error counters will reach a specific threshold, e.g. the error warning limit, and will prompt the microcontroller. In this case, the node can directly rule out that controller for communicating.

Accepting the notifications from the non-omitting controller will permit the node to tolerate stuck-at and bit-flipping faults at one of its connections. Also, this approach prevents nodes from missing any frame while a fault is being treated. Thus, so far, this approach is an improvement in terms of fault-tolerance with respect to the approaches we outlined before.

Lamentably, to try to transmit frames through two CAN controllers simultaneously poses some problems. When the node wants to transmit a frame, it instructs both its controllers to send the same frame. Therefore both transmissions will have the same priority and consequently both controllers will win the arbitration [5], which inevitably will lead to a collision. This approach could only work if both controllers initiated their transmissions simultaneously and sent their bits in exact lockstep with each other,

which is very hard to guarantee. Otherwise they might get so out-of-sync that the controllers would no longer see on the media the bit value they transmitted, and as a result, both controllers would signal an error [5] and abort the transmission.

The approach that we have adopted for ReCANcentrate uses both controllers simultaneously for reception but only one of them to transmit frames, therefore it solves the previous approach's transmission problem. The controller that is used to transmit and receive is called the transmission controller, whereas the other one is called the reception controller [6].

When the node acts as a receiving node, the reception mechanism is very similar to the one used in the previous approach, where both controllers are used simultaneously for both transmission and reception. The difference is basically that when the microcontroller discards a CAN controller for communicating, it also has to check if that controller is the one which is currently marked as the transmission controller. If affirmative, it will have to assign the transmission controller role to the surviving controller.

This approach has an additional advantage. Notice that when the transmission controller sends a frame to its hub, that frame will be received not only by itself, but also by the reception controller, due to ReCANcentrate's single broadcast domain. A transmitting node therefore expects to always receive at the reception controller a frame that it has sent through the transmission controller. This fact can be used to implement further fault-tolerance mechanisms. Particularly, it can be used to tolerate the CAN inconsistency scenarios reported by Rufino *et al.* [4]. We hope to accomplish exactly that through a driver we are currently implementing for the nodes.

4 Conclusions and future work

In order to improve the reliability of CAN bus networks, we have developed a replicated star topology called ReCANcentrate, which includes two active hubs. The nodes of ReCANcentrate are connected to both of them by means of dedicated links, and the hubs are connected to each other through several interlinks. Each hub of ReCANcentrate includes mechanisms to detect and isolate stuck-at and bit-flipping faults occurring at nodes, the media and the other hub.

The hubs exchange their traffic through the interlinks and couple with each other, within a fraction of the bit time, to create a single logical broadcast domain. As a consequence, each node receives the same bit values from both hubs quasi-simultaneously. This synchronization between both stars at bit level simplifies the way in which each node manages the replicated traffic to transmit, receive and tolerate faults.

This paper explores different approaches for this media management at nodes, to take full advantage of ReCANcentrate's redundancy. We analyze different management strategies to allow nodes to continue communicating tolerating different kinds of faults, while keeping data consistency even in the presence of some CAN inconsistency error scenarios identified in the literature.

We overview the approaches for the media management we have so far considered. First we examined approaches using only one CAN controller per node, coming to the conclusion that they are not adequate when a high level of fault-tolerance is required. Then we took a look at different approaches where two CAN controllers are used per node. Although this adds more complexity to the system and therefore might seem to reduce ReCANcentrate's reliability, it is in fact possible to use a strategy which actually increases ReCANcentrate's reliability by providing better fault-tolerance.

After describing some unsatisfactory two CAN controller approaches we introduced our chosen approach. This approach has been formally verified by means of a model checker and is currently being implemented in a ReCANcentrate prototype to show its viability.

There are also other, more complex approaches, we have not studied yet. For instance it might be possible to use an approach very similar to the one we are implementing currently. The main difference would be that instead of having a dedicated transmission controller we would alternate through which controller to transmit (as long as both are functional and their links non-faulty). A possible advantage of this approach could be that the blocking time of high priority messages due to lower priority messages could be reduced as the transmit queue on each controller would be shorter. On the other hand, the firmware for the microcontroller would be more complex.

Other possible approaches might take even further advantage of ReCANcentrate's redundant architecture by allowing a node to communicate as long as it had one non-faulty uplink and one non-faulty downlink, independent of the hub they are connected to. This could increase ReCANcentrate's fault-tolerance, but we cannot say yet if it would result in an overall increase in reliability as it increases the complexity and would require more hardware.

References

[1] M. Barranco, J. Proenza, G. Rodríguez-Navas, and L. Almeida, "An Active Star Topology for Improving Fault Confinement in CAN Networks", *IEEE transactions on industrial informatics*, vol. 2, no. 2, pp. 78–85, May 2006.

[2] M. Barranco, L. Almeida, and J. Proenza, "ReCANcentrate: a replicated star topology for CAN networks", in *10th IEEE International Conference on Emerging Technologies and Factory Automation, 2005. ETFA 2005.*, volume 2, Sept. 2005, Catania, Italy.

[3] J. Proenza and J. Miro-Julia, "MajorCAN: A Modification to the Controller Area Network Protocol to Achieve Atomic Broadcast", *IEEE International Workshop on Group Communication and Computations, Taipei, Taiwan*, 2000.

[4] J. Rufino, P. Veríssimo, G. Arroz, C. Almeida, and L. Rodrigues, "Fault-Tolerant Broadcasts in CAN", in *FTCS '98: Proceedings of the The Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*, 1998, p. 150, Washington, DC, USA. IEEE Computer Society.

[5] R. Bosch GmbH, "CAN Specification Version 2.0", Technical report, Robert Bosch GmbH, 1991.

[6] M. Barranco, J. Proenza, and L. Almeida, "Designing and Verifying Media Management in ReCANcentrate", in *Proceedings of the 13rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2008), Hamburg, Germany*, Sept. 2008.

[7] M. Barranco, "Improving Error Containment of Controller Area Network (CAN) by means of Adequate Star Topologies", Technical report, Universitat de les illes balears, Nov. 2008.

[8] M. Barranco, J. Proenza, and L. Almeida, "Experimental Assessment of ReCANcentrate, a Replicated Star Topology for CAN", *SAE 2006 Transactions Journal of Passenger Cars: Electronic and Electrical Systems*, 2006.