Maintaining data consistency in ReCANcentrate during hub decouplings

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

Luís Almeida
DET/IEETA
Universidade de Aveiro, Portugal
lda@det.ua.pt

Abstract

The use of CAN in safety-critical applications has been controversial due to dependability limitations. To overcome some of them, we proposed a replicated star topology, ReCANcentrate, whose hubs incorporate fault-treatment and fault tolerance mechanisms. The two hubs of ReCANcentrate are coupled with each other, thereby forcing a single communication domain that simplifies the management of the replicated media that each node performs to ensure data consistency. This paper proposes additional mechanisms to enforce data consistency even when hubs become temporarily or permanently decoupled.

1 Introduction

The use of CAN in critical applications has been controversial due to dependability limitations. Some of them arise from its bus topology, which has scarce error-containment and fault-tolerance mechanisms. To overcome these limitations, we developed a replicated star topology, called ReCANcentrate that includes two hubs [1]. In ReCANcentrate each node is connected to each hub by a dedicated link that contains an uplink and a downlink. Both hubs are interconnected by means of at least two *interlinks* each of which contains two independent sublinks, one for each direction. Each hub includes fault-treatment capabilities to contain errors originated at nodes, links, interlinks and at the other hub, and to provide tolerance to hub, link and interlink faults. ReCANcentrate is fully compatible with off-the-shelf (COTS) CAN components and any CAN-based application or protocol.

When using replicated media for fault-tolerant communication, one approach is to transmit the same data over all media (or channels) in parallel. However, a bit error in one channel is enough for its traffic to evolve differently with respect to the other channels, raising some *data consistency* issues [2], namely the detection of replicated frames arriving from different channels at different instants of time (duplicates) and the detection of frames that are omitted in only some channels (omissions). To prevent the traffic from evolving differently in both stars, the hubs of ReCANcentrate exchange their traffic through the interlinks (Figure 1) and couple with each other [1]. In this way, both hubs transmit the same value bit by bit in their downlinks, even when the uplinks from one node differ. Therefore, we say that ReCANcentrate provides a *single communication domain* behaving like a CAN bus to which nodes connect through two communication controllers (Figure 1). Consistency issues, then, are reduced to handling different views of the same channel instead of two different channels, which is substantially simpler.

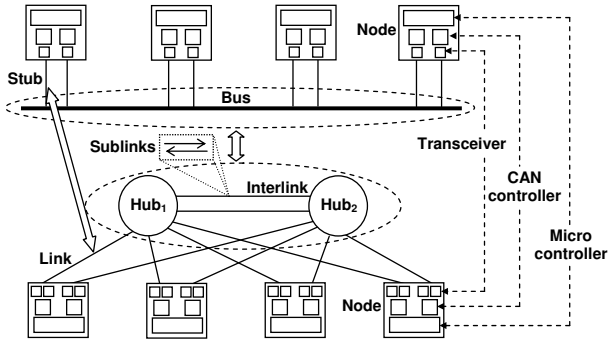

Figure 1. Bus and replicated star analogy

In [1] we outlined a simple replicated media management that takes advantage of this single communication domain. Each node is constituted by COTS components only: two CAN controllers and a micro-controller (Figure 1). Each CAN controller is connected to one hub, only, by means of a dedicated uplink and downlink, using two COTS transceivers [1]. Basically, each node manages the media by transmitting through one hub, only, while receiving from both hubs at the same time. Detection of duplicates and omissions becomes straightforward because each frame is broadcast simultaneously by both hubs.

ReCANcentrate includes replicated interlinks to enable the referred simple media management even in presence of interlink failures. However, hubs may be temporarily or permanently decoupled due to faults affecting all interlinks or any of the hubs. This paper discusses the threats to the hub coupling and, thus, to the single communication domain and briefly explains on-going work towards the design and implementation of a set of mechanisms that will allow nodes to consistently communicate even if hubs are temporarily or permanently decoupled.

2 Threats to the hub coupling

The physical layer of CAN implements a wired-AND function of every node contribution, thereby providing a dominant/recessive transmission [2]. Additionally, CAN communication relies on a complex bit synchronization mechanism that guarantees that nodes have a simultane-

 doi:10.1109/WFCS.2008.4638750

ous view of every single bit on the medium, i.e. CAN presents *in-bit response*.

To cope with our fault model [1], each hub is able to detect faults at nodes, links, interlinks or at the other hub that manifest as stuck-at-recessive, stuck-at-dominant or bit-flipping streams. Notice that a hub cannot build or buffer CAN frames, so that its failure can only manifest, in principle, as the generation or propagation of stuck-at or bit-flipping bits.

The hub isolates faults by disabling the port of origin [1]. Additionally, the hub uses a specific reintegration policy to re-enable a port contribution whose failure is only temporary. Since stuck-at-dominant and bit-flipping faults corrupt the data conveyed through the media, a hub detects and isolates them with a short delay [1]. In contrast, a stuck-at-recessive port does not issue errors that necessarily corrupt data, hence its detection implies a longer delay being based on the detection of consecutive omissions of the *ACK bit*. Note that in CAN every operational controller contributes to all frames in transmission with a dominant bit, the ACK bit, within the *ACK slot* [2].

As explained before, nodes can easily manage the replicated media as long as both stars constitute a single communication domain. The unique situation in which this single domain is not enforced is when any hub continues coupling the contributions of its own nodes, but does not couple with the other hub. This can only happen in the following cases. Firstly, considering our fault model, when all interlinks are faulty or one hub sends a stuck-at or a bit-flipping contribution to the other hub through all interlinks. Secondly, if a new type of fault, which we add to our fault model, leads a hub to erroneously decide not to couple with the other hub. This may occur if the faulty hub does not send its traffic to the other hub, or if it stops considering the traffic received from that hub.

In order to allow nodes to consistently communicate even in presence of faults that lead to a hub decoupling, we propose to enhance the fault-treatment capabilities of ReCANcentrate. Firstly, we prevent nodes from communicating while a decoupling fault is not treated, otherwise data consistency could be violated. Secondly, we force both hubs and all nodes to agree on whether it is possible to recover from a hub decoupling provoked by a temporary fault, or if it is only possible to use both stars independently. We focus on enforcing such agreement; the strategy nodes use to communicate using two decoupled hubs is beyond the scope of this paper.

3 ACK interdependence

As explained before, it is necessary to prevent nodes from exchanging any frame as long as the decoupling is not treated. Unfortunately, a decoupling cannot be always instantaneously detected and, meanwhile, nodes could exchange frames. For example, if all interlinks are stuck-at-recessive, then a hub will only detect the decoupling after observing that the *ACK bit* is omitted from the other hub during several frames.

Therefore, we propose a new mechanism, based on the ACK bit, that instantaneously blocks the communication in both stars when hubs become decoupled. Basically, during the ACK slot, each hub transmits the resulting ACK bit received from the other hub, only, without regard of the ACK bits received from its own ports. Thus, no frame will be exchanged in any star if both hubs are not correctly coupled and synchronized bit by bit with each other. We call this mechanism *ACK interdependence*.

After detecting and treating a decoupling, as explained later on in Section 4, the hubs may decide that it is not possible to couple with each other again. In such case, each hub changes its mode of operation to broadcast the *ACK bits* received from its own nodes. This allows nodes to use both stars independently when necessary.

Note that the *ACK interdependence* mechanism relies on a new hypothesis: no hub propagates the *ACK bits* of its own nodes unless it has correctly treated its decoupling with the other hub and nodes use both stars independently. Thus, it is necessary to restrict the failure semantics of the hub to ensure that it cannot maliciously fail by not fulfilling this hypothesis. To achieve this, the hub could be internally duplicated and compared. However, to avoid excessively increasing the hub complexity, we are devising a guardian for each hub, called *Hub Guardian*, which will prevent its corresponding hub from incorrectly broadcasting the *ACK bits* received from its own nodes. An important requirement has been imposed to the design of this guardian: to guarantee the failure independence between itself and its corresponding hub.

4 Re-Start Procedure

The second enhancement specified in Section 2 demands for a treatment mechanism that ensures that once a hub decoupling is detected, the hubs and all nodes agree on whether or not the single communication domain is reestablished. To achieve this, we are designing a mechanism called *Re-Start Procedure*, which is based on a set of frames the hubs and nodes interchange to consistently treat the decoupling. During this procedure, each hub sends what we call *Identification Frames* (IF) and *Filling Frames* (FF), whereas each controller of each node sends a different *Identification Request Frame* (IRF). Note that in CAN, each frame begins with an identifier that indicates its priority, so that any collision is solved by means of a bit-wise arbitration of the identifiers [2]. All the IFs of a given hub always have the same identifier, which is reserved for that hub and whose priority is one of the two highest ones. Each hub also owns one of the two lowest prioritized identifiers and uses it to build its FFs. Besides, each controller builds its IRF using an identifier reserved for that controller. The IRFs have the lower priorities after the two lowest prioritized identifiers reserved for the FFs.

Due to space limitations, next we only describe a brief sketch of the *Re-Start Procedure*. Figure 2 summarizes the steps a hub executes during this procedure.

4.1 Sublink Restoration

The first phase of the *Re-Start Procedure* is the *Sublink Restoration Phase* (SRP), during which both hubs consistently decide whether or not they become coupled again. Each hub executes it immediately after detecting that all the interlinks are faulty, i.e. when detecting a hub decoupling. First of all, each hub sends a constant dominant logical value during a given period of time through all its outgoing sublinks and its downlinks. This compels the other hub to isolate all its incoming sublinks and to also execute this phase if it had not detected the decoupling yet. Additionally, all CAN controllers connected to the hub will receive the stuck-at-dominant value through their downlinks and will eventually enter into the *bus-off state* [2], in which they will not try to communicate.

When both hubs are executing this phase, they test the interlinks to decide whether or not it is possible to communicate with each other. In order to do that, the hubs try to alternatively exchange, through all the interlinks in parallel, their IFs and rule out a given interlink whenever its port has accumulated too many errors. On the one hand, the phase finishes whenever both hubs successfully interchange a given number, M, of IFs. On the other hand, a given hub also stops executing the phase if it rules out all interlinks. When this happens, the hub decides that it is impossible to re-couple and, then, permanently sends a stuck-at-dominant value through all its outgoing sublinks. This forces the other hub to finish the phase too.

Note that both hubs will consistently decide whether or not to re-couple. This is because they use the CAN protocol to communicate with each other and, thus, they consistently interchange each IF. Moreover, due to the *in-bit response* property of CAN, in case hubs re-couple, both of them will simultaneously finish the phase when interchanging the last bit of the last IF.

4.2 Re-coupling & Decoupling Node Agreement

After the SRP, the hub executes the *Re-coupling Node Agreement Phase* (RNAP) or the *Decoupling Node Agreement Phase* (DNAP), depending on whether it is re-coupled with the other hub or not, respectively (Figure 2). By means of these phases all nodes consistently diagnose whether or not the single communication domain is re-established. In each one of these phases each node CAN controller requests the presence of the *other* hub, i.e. the one it is not directly connected to. A controller performs such a request by sending its IRF, and expects that the required hub replies by broadcasting its IF. The targeted hub will only be able to receive the request and reply to it if both hubs are re-coupled, i.e. they are executing the RNAP. Thus, in principle, if a controller receives the IF it requests, its node could assume that the single broadcast domain is re-established. Conversely, if hubs are not re-coupled, i.e. they are executing the DNAP, no hub will receive an IRF. Thus, if a controller does not receive the IF it requests, its node could decide that both hubs are decoupled. However, the node only comes to a final conclusion

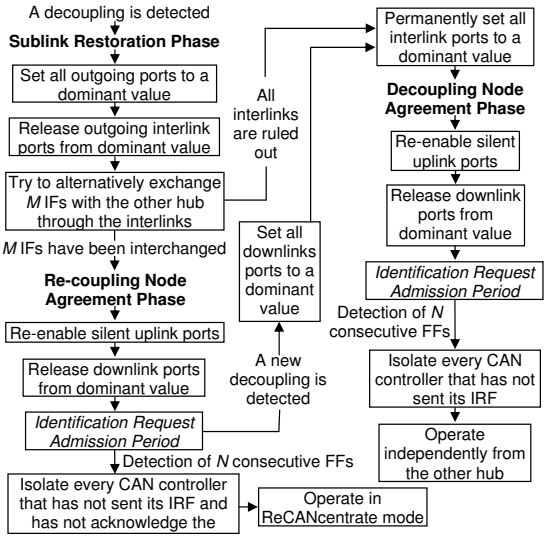

Figure 2. Re-Start Procedure by one hub

depending on the view its two controllers have about the single communication domain, as explained later.

The behavior of a hub during the RNAP is as follows. First of all, the hub stops sending the stuck-at-dominant value through the downlinks and re-enables the hub ports corresponding to controllers that have been silent during the *Sublink Restoration Phase*. This is because, during the SRP, the hub was sending a suck-at-dominant value trough all its downlinks thereby forcing all its correct controllers to reach the *bus-off* state. Thus, a silent port indicates that the corresponding controller is possibly correct. Then, the hub begins an *Identification Request Admission Period* during which it performs three different actions. As first action, it expects to observe that each one of the controllers connected to it sends an IRF in order to request the presence of the other hub. The hub knows what is the identifier of the IRF of each one of these controllers. If the hub detects that any of them sends a frame other than the IRF with the appropriate identifier, or does not acknowledge the IF sent by the other hub, it isolates that port. Moreover, when the phase finishes, the hub disconnects each controller that has not sent its IRF. The other action the hub performs during this phase is to reply to any IRF sent by a controller connected to the other hub. Each hub knows what are the IRFs it can receive from these controllers and will only reply to those IRFs. To detect possible forged IFs, the hub includes a *password* within the data field of the IF that only itself and the requesting controller know. The third action the hub performs is to constantly try to broadcast its FF when it is not replying to a given IRF. As explained later, the FFs allow hubs and nodes to quasi-simultaneously detect the end of the RNAP.

Concerning the role of a node during this phase, when it detects that any of its controllers exits the *bus-off state*, it requests the transmission of the corresponding IRF through that controller. If the next frame the controller monitors in the channel after sending the IRF is the requested IF and its password is correct, the node diagnoses that, in principle, the hubs are re-coupled. Notice that if

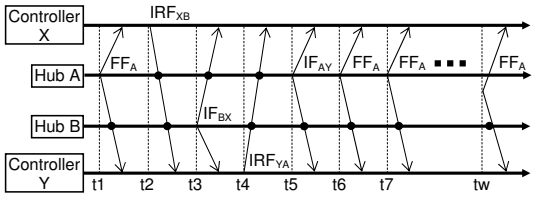

Figure 3. RNAP case example

the hubs are actually coupled, the next frame that will be broadcast through the network after an IRF is the corresponding IF. This is because the hub whose presence has not been requested by the IRF will not try to transmit its IF and, thus, it is guaranteed that the requested IF is the frame with the highest priority. This simple rule consisting in expecting the IF just after the IRF allows the node to avoid the use of a timeout to detect that the controller cannot communicate with the requested hub.

Regarding the DNAP, the hub executes it if, after the SRP, it is not re-coupled with the other hub, as said before. However, the hub also executes it whenever it detects a new decoupling during the RNAP (Figure 2). More specifically, a hub diagnoses a new decoupling during the RNAP if it isolates all ports corresponding to the interlinks; or if it diagnoses the other hub as faulty because that hub has not immediately replied to an IRF, it has broadcast an IF that was not requested, or it has broadcast a frame other than an IF, an IRF or a FF. When this happens, the hub definitively abandons the possibility of re-coupling; aborts the RNAP; forces again its controllers to enter into *bus-off* and, then, it executes the DNAP. The actions performed by the hubs and the controllers in the DNAP are equal to those they carry out in the RNAP, except that during the *Identification Request Admission Period* of the DNAP, the hub does not expect that the other hub replies the IRFs sent by its controllers. Notice that when the hub aborts the RNAP in order to execute the DNAP, it firstly forces all controllers connected to it to enter the *bus-off* state. This ensures that each one of these controllers that already sent its IRF during the RNAP sends its IRF again and detects that the hub constitutes an independent star.

The RNAP and DNAP finish when the hub monitors a given number, N, of consecutive FFs. At the end of the RNAP, since the hubs are re-coupled, both of them and all controllers will simultaneously observe the last FF of the consecutive FFs and, thus, the end of this phase. In contrast, although each hub and its corresponding controllers will detect the end of a DNAP simultaneously, nothing ensures that a DNAP finishes at the same time in both stars. This has no negative consequences, since nodes will use the two hubs as two independent stars and they do not need bit-level synchronization between them.

At the end of each one of these phases each node makes a decision about the reestablishment of the single communication domain as follows. If one of its two controllers has accumulated too many errors during the phase, the node declares that controller as faulty, and accepts as valid the view that the non-faulty one has about the communication domain. Similarly, if both controllers are non-faulty and they agree, the node also uses their view of the communication domain. In contrast, if they disagree, the node permanently disconnects itself.

Figure 3 shows an example of the frames interchanged among the hubs, A and B, and the two controllers of the same node: X connected to A, and Y connected to B. Each bullet indicates that a frame goes through a given hub. A has a higher prioritized FF than B, and the IRF of X has a higher priority than the IRF of Y. Before time $t1$ both controllers are in *bus-off* and A constantly broadcasts its FF. At $t2$ each controller exits that state and tries to send its IRF; controller X wins and requests the presence of B with $\mathrm{IRF_{XB}}$. At $t3$, B broadcasts its IF to reply to X, so that X assumes that there is a single communication domain. The same pattern is observed afterwards when Y requests the presence of A. Finally, from $t6$ on, A constantly broadcast its FF. When at tw it has broadcast N FFs, the RNAP finishes and the node concludes that the single communication domain is reestablished.

5 Conclusions and future work

In ReCANcentrate data are transmitted in parallel through both stars to provide fault tolerance. The hubs exchange their traffic through several interlinks and couple with each other. This creates a single communication domain that allows each node to easily manage the media to guarantee data consistency. The paper proposes new mechanisms to enforce data consistency when hubs become temporarily or permanently decoupled due to faults that affect all interlinks or any of the hubs. The first mechanism blocks the communication in each star when hubs become decoupled, to prevent inconsistent communication as long as the decoupling is not treated. Additionally, a treatment procedure allows hubs and nodes to agree on whether hubs can couple again. This ensures data consistency since it allows all nodes to agree on whether or not they can use both stars as a single one.

As future work, we plan to formally verify the correctness of our approach by modelling it using the model checker UPPAAL [3]. Results obtained so far, analyzing media and controller faults, support its correctness.

Acknowledgment

This work is partially supported by DPI 2005-09001-C03-02 and FEDER funding.

References

[1] M. Barranco, L. Almeida, and J. Proenza, "Experimental assessment of ReCANcentrate, a replicated star topology for CAN", in *Safety-Critical Automotive Systems*, 2006. Society of Automotive Engineers, USA.

[2] ISO, "ISO11898. Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication", 1993.

[3] K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a Nutshell", *Int. Journal on Software Tools for Technology Transfer, 1(12):134152*, 1997.