ReCANcentrate: A replicated star topology for CAN networks

Manuel Barranco[1], Luís Almeida[2] and Julián Proenza[1]
[1]Dpt. Matemàtiques i Informàtica Universitat de les Illes Balears, Spain
[2]DET/IEETA Universidade de Aveiro, Portugal

Abstract

Controller Area Network (CAN) is nowadays widespread in distributed embedded systems due to its electrical robustness, low price, and deterministic access delay. However, its use in safety-critical applications has been controversial due to dependability limitations. In particular, in a CAN bus there are multiple components such that a single fault of any of them can prevent the communication capabilities of several nodes and may provoke a general failure of the communication system, i.e. there are multiple severe points of failure. In [1] we proposed a new active star topology, called CANcentrate[1], that solves these limitations by means of an active hub with enhanced fault-treatment capabilities. However, the center of the star still represents a severe point of failure, thus not being suitable for more demanding safety-critical systems. In this paper, we propose a replicated star topology, called ReCANcentrate[2], which has no severe points of failure and is fully compatible with existing CAN controllers. The paper analyzes related work, describes the CANcentrate basics, explains the design and functionalities of ReCANcancentrate, and finally describes the implementation and test of its prototype.

1 Introduction

Controller Area Network (CAN) fulfills the communication requirements of many distributed embedded systems. In particular, CAN includes an event-triggered data link layer that provides high reliability and good real-time performance with very low cost. Due to this, the CAN protocol is nowadays used in a wide range of applications, such as factory automation or in-vehicle communication.

Nevertheless, communication systems based on CAN present several specific dependability problems that are caused by the bus topology of this protocol. The main drawback of using a bus is that the structure of the network presents multiple points such that a fault in any of them can prevent the communication among several nodes. These are called *severe points of failure*, which include the commonly referred to as single point of failure [1].

Our general framework aims at improving the dependability of CAN to make it more suitable for safety critical applications. In particular, the main objective of this work is to provide a CAN network without any severe point of failure.

In [1] we proposed a new star topology for CAN, called CANcentrate, which has enhanced fault treatment mechanisms that reduce the number of severe points of failure inherent to bus topologies to a unique single point of failure, namely the hub.

In this paper we extend the previous work, eliminating the single point of failure of CANcentrate by means of a replicated star topology. We call it *ReCANcentrate*. We will also show that this replicated topology facilitates the management of redundancy in event-triggered communication systems.

In the following Section we describe the existing main approaches for providing CAN with redundancy, paying attention to the problems they pose and identifying their pros and cons. Section 3 outlines the basics of CANcentrate. Section 4 explains the design, the characteristics and the functionalities of ReCANcentrate and addresses issues related to the cabling length and the bit rate. Section 5 describes the implementation of a ReCANcentrate prototype and the tests that were conducted to check its functionalities and performance. Finally, Section 6 considers future work and concludes the paper.

2 Problem statement and related work

Removing all severe points of failure from a communication system can only be achieved with redundancy, either temporal or spatial. However, permanent communication failures, such as medium partition, can only be tolerated with spatial redundancy. This can be found in several existing safety critical protocols, such as TTP [2], FlexRay [3], or FlexCAN [4], which rely on replicated media architectures.

[1]CANcentrate has been the subject of a patent filing that was submitted on 16th of September of 2004 and that is being currently evaluated by the *Oficina Española de Patentes y Marcas*

[2]The contents of this article concerning ReCANcentrate have been the subject of a patent filing and must be considered as confidential until the date of publication of this material in the proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation the 19th of September of 2005.

Regardless of the specific topology, we can differentiate between two main uses of media replication: increased throughput, when the replicated media are used independently to transmit different data; or fault-tolerance, when the replicated media are used to transmit the same data [5]. Each communication medium is commonly referred to as channel and in this work we will focus on the use of replicated channels for fault-tolerance purposes.

The main problem of using replicated channels is that nodes must be able to manage the redundant frames they receive. Particularly, they must determine when two received frames are in fact copies of the same frame (duplicates), or when a frame received from one channel is omitted from the other (omissions). Synchronizing the transmission and the reception of frames across the network is a possible solution. This synchronization is easily achieved in time-triggered protocols due to their inherent transmission schema. In fact, each frame is expected to be transmitted quasi-simultaneously in both channels at predefined time slots. Hence, removal of duplicates and detection of omissions is done on the fly. This is the basic transmission mechanism specified in protocols such as TTP and FlexRay, which provide communication via dual channel either using a bus topology, a star topology or, in the case of FlexRay, also using an hybrid topology.

Unfortunately, since CAN is an event-triggered protocol, it does not provide any mechanism for synchronizing the frames in the different channels. Therefore, additional mechanisms have been proposed in the literature to provide some sort of synchronization, as in FlexCAN, SMART-1 [6], or the Columbus Egg Idea [7]. FlexCAN uses a strategy based on triplicated CAN buses and nodes. The nodes are coordinated by means of a software that uses timers to control the transmissions and the receptions on the channels. In [6] the CAN network used in the lunar mission SMART-1 (*small Missions for Advanced Research in technology*) is described. This network includes replicated nodes and two replicated CAN buses, one active and other inactive that is used as a spare. Nodes detect when the active channel is faulty and then switch to the spare one, thus there is no need of synchronization at the frame level. Finally, the solution proposed in [7] uses several CAN buses and eliminates the need of dealing with duplicates and omissions by coupling the streams received from all buses, at the bit level, in each node.

Although these systems provide synchronization between channels, they rely on a bus topology, which still has severe points of failure as referred in [1], e.g. nothing prevents a faulty node from continuously sending erroneous information to all channels. Instead, star topologies may provide improved fault-treatment capabilities with respect to buses [1]. Hence, we decided to use a star topology in order to provide a replicated CAN network. Particularly, in our previous work [1] we proposed a simplex star topology, called CANcentrate, whose hub is provided with enhanced fault treatment mechanisms, beyond the capabilities of any other existing star solution for CAN, and that reduces all severe points of failure to one single point of failure, the hub. The present work addresses the replication of CANcentrate in order to definitively eliminate all severe points of failure.

In what concerns the problems that arise when using replicated channels, notice that CANcentrate is transparent with respect to any application executed at CAN nodes [1]. Thus, in principle, any of the replication strategies referred above could be used, replacing each bus by a hub. However, none of them is suitable for this purpose. On one hand, systems such as FlexCAN or SMART-1 rely on a quite complex solution that increases the requirements of nodes in terms of hardware and software. On the other hand, the approach proposed in [7] would limit the dependability features of the replicated star as described later in Section 4.2.

Therefore, we propose a new replicated star topology, *ReCANcentrate*, which takes advantage of the dependability properties already achieved by CANcentrate and which provides nodes with an easy way to manage the replicated star. Despite replicated stars being available for protocols such as TTP and FlexRay, to the best of our knowledge, ReCANcentrate is the first one for CAN.

3 CANcentrate basics

The main characteristic of CANcentrate [1] is that it relies on a hub that prevents single faults from generating a severe communication failure. Its fault model includes stuck-at faults, medium partition faults, shorted medium faults and bit-flipping faults. Moreover, CANcentrate makes no assumption on the frequency and duration of errors that may occur. The only assumption made, since hub replication is not considered in the original design, is that the hub itself will not fail.

One requirement was imposed from the beginning on the design of CANcentrate, namely to preserve all the characteristics of the CAN protocol that are related to dependability. Concerning this requirement, particular care was taken to maintain the frame format and all mechanisms for channel error detection and signalling exactly as they are defined in CAN. As a consequence of this compatibility with the standard CAN specification, commercial off-the-shelf components can be used for building the CAN nodes of CANcentrate. Similarly, the hub is transparent for any CAN application executed on the nodes.

Each node is connected to the hub by means of two independent and dedicated links. An *uplink* that carries the node contribution to the hub, and a *downlink* that carries the coupled signal from the hub to the node. From the hub point of view, each node and its links constitute an error confinement region that is managed as a port. The hub monitors each port contribution through its uplink in order to detect errors and isolates it when faulty. In this way, the hub prevents propagation of errors from a given faulty port to the others.

The hub is constituted by three modules: the Coupler Module, the Input/Output Module and the Fault-Treatment Module as depicted in Figure 1. The Coupler Module takes into account each port contribution ($B_{1..n}$) and calculates the resultant coupled signal, B_0, that is broadcasted to the nodes. The Input/Output Module translates the physical signals of each uplink into a logical form that can be understood by the other modules of the hub. Additionally, it translates B_0 into a physical signal that is sent to each node through its downlink. Finally, the Fault-Treatment Module monitors each port contribution in order to detect errors. When a given port has accumulated too many errors, this module isolates it by means of the corresponding Enabling/Disabling signal ($ED_{1..n}$ in Figure 1).

Note that in CAN the logical '0' value is normally referred as the dominant value, whereas the logical '1' is referred as the recessive [8]. Taking this into account, the Coupler Module is formed by an AND gate, AND_C, which replaces the wired-AND functionality of the CAN bus, and a set of OR gates used to enable and disable a specific port contribution. The frames that result from coupling all enabled ports contributions, i.e., after the AND_C gate, are called *resultant frames* hereafter.

The Input/Output Module is composed by a pair of transceivers for each node: one for the uplink and another for the downlink.

Fault-treatment is aimed at preventing faults from being activated again and is typically performed in two steps: *fault diagnosis*, to find out the cause of each error including both its location and nature; and *fault passivation*, to prevent faults from being activated again, i.e. making the faults passive. These are the main purposes of the Fault-Treatment Module, which detects permanently faulty ports and isolates them from the system, so they cannot cause severe communication failures. It is basically formed by the Rx_CAN Module and the set of Enabling/Disabling units (Ena/Dis in Figure 1). Rx_CAN monitors the coupled signal B_0 to calculate the *current state* of the resultant frame. This current state identifies which is the meaning of the bit of the resultant frame that is currently being broadcast to all ports and forecasts which should be the proper contribution of each node to the following bit. Rx_CAN outputs to each of the Enabling/Disabling units a set of signals, C, which together with the coupled signal, B_0, describe the current state of the *resultant frame*, e.g., whether the bit is a stuff bit, the expected correct bit value according to the stuff rule, the type of frame and the specific frame field the bit belongs to, whether the bit is the last bit of the End-Of-Frame field, whether the frame has passed the CRC checking, etc.

The fault diagnosis and fault passivation are carried out by the Enabling/Disabling units. Each one of these units uses C, B_0, the contribution from its port (B_i), and a set of error counters to diagnose whether its port is permanently faulty or not. If found permanently faulty, the respective Ena/Dis unit removes the port contribution by setting ED_i to '1'.

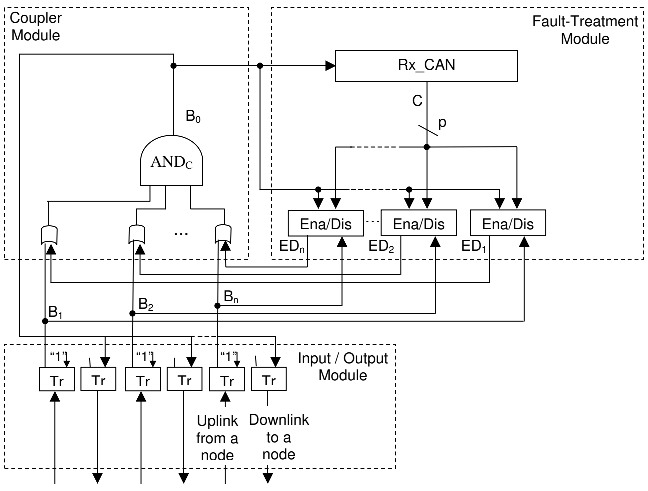

Figure 1. Internal structure of the hub

4 Replication of CANcentrate

In order to eliminate all severe points of failure from the communication system, we designed an infrastructure, called ReCANcentrate, which relies on a replicated star topology using two interconnected CANcentrate hubs. This infrastructure maintains all CAN properties related to dependability, and is compatible with commercial off-the-self CAN components.

Furthermore, ReCANcentrate overcomes the problems, regarding dependability and management of redundancy, that arise when using a replicated event-triggered communication system, namely the management of duplicates and omissions.

4.1 Consideration on the fault model

The fault model considered by ReCANcentrate gathers the same kind of faults included in the fault model of CANcentrate (see Section 3). However, the fault assumptions are now slightly different. While in CANcentrate the unique fault assumption made was that the hub would not fail, in the present work we relax this fault assumption considering that at least one of the hubs will be non-faulty. As a consequence, faults may occur not only at nodes and links, but also at one of the hubs.

Furthermore, the fault model of ReCANcentrate includes a new type of fault that may occur in a replicated star topology, which arises from the fact that nodes can be isolated on one hub but still be able to communicate using the other one. Therefore, particular combinations of faults occurring in different ports of both hubs may lead nodes to have an inconsistent view of the nodes that are available for communicating, i.e. an *inconsistent membership* fault occurs. Figure 2 depicts such a situation showing the failure of two different links connecting two different nodes to different hubs. Node A can communicate with nodes B and C. However, nodes B and C cannot communicate with each other.

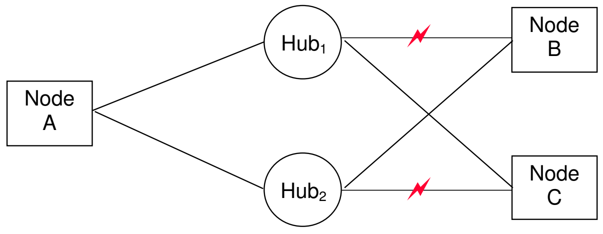

Figure 2. Example of inconsistent membership fault

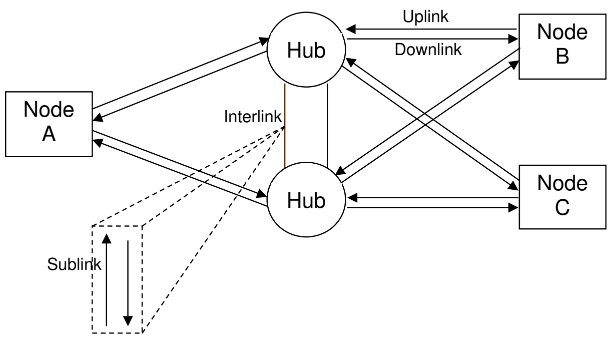

Figure 3. Architecture of ReCANcentrate

4.2 Star replication rationale

The main architectural characteristic of ReCANcentrate is that it is constituted by two CANcentrate hubs interconnected by means of two dedicated links called *interlinks* (Figure 3). Nodes are connected to each hub via an uplink and a downlink, as in CANcentrate.

A given interlink is also formed by two *sublinks* used by each hub to send the coupling of the contributions of its own nodes to the other hub. For the sake of simplicity, we will refer to this signal as the *contribution* of that hub. Then, the resulting signal that each hub broadcasts to its own nodes is the one that results from coupling its own contribution with the contribution received from the other hub. This coupling creates a single logical broadcast domain since both hubs behave like one, transmitting the same value bit by bit in their downlinks, i.e., in-bit response [8] is enforced in the whole replicated domain.

This tight coupling has deep consequences on the whole communication system. Firstly, nodes can be either connected to both hubs, for improved fault-tolerance, or they can be connected to one hub, only. In any case, the node transmissions will be broadcasted to all nodes. Therefore, regardless the hub or hubs a node is connected to, all nodes will have a coherent view of which nodes are available for communicating thus preventing inconsistent membership faults to occur.

Secondly, a node connected to both hubs can still communicate even if it transmits through one hub and receives from the other, i.e., as long as one of its uplinks and one of its donwlinks remain non-faulty, regardless the hub they are connected to. Thereby, the system tolerates simultaneously one fault affecting an uplink and another fault affecting one downlink.

Thirdly, nodes connected to both hubs receive the same frames within the same bit time thus easily detecting duplicates and omissions, which is one of the major problems when designing an event-triggered system that relies on a replicated communication system. Specifically, duplicated frames are always expected in each reception, whereas an omission can be easily detected by checking, at the reception of each frame, that two copies of the same frame are effectively received from both stars.

In what concerns the fault-treatment capabilities of ReCANcentrate, note that each node can be considered as two CAN nodes, each one connected to a different hub. In such a way, a hub monitors the contribution of a CAN node, regardless the contribution the node sends to the other hub, and isolates the corresponding port when faulty. Therefore, the fault-treatment capabilities of CANcentrate with respect to faults occurring at nodes or links apply to the whole network by means of the actions performed by both hubs at their respective ports.

Additionally, each hub includes mechanisms for detecting and isolating faults occurring at the interlinks. Specifically, it monitors the two sublinks, within both interlinks, that carry the contribution from the other hub. When any of these sublinks fails, the hub isolates it, but continues using the other one. Therefore, hubs will communicate with each other as far as there are two non-faulty sublinks (regardless the interlinks they are located in) that make possible them to interchange bits in both directions.

Moreover, the hub also uses these mechanisms to detect and isolate a faulty hub. Specifically, this occurs when both contributions received from the other hub are diagnosed as being faulty. In such a way, the errors generated by the faulty hub cannot propagate through the non-faulty hub to the nodes.

Note that despite ReCANcentrate removes all severe points of failure, the properties derived from having both hubs coupled only apply when hubs can still correctly receive the contribution from each other. Otherwise, the communication system would be equivalent to have two independent CANcentrate stars. This still provides a valid replicated communication system but losing most of the features referred above. Namely, the in-bit response may be lost, the domains of each hub may be strictly isolated from each other and then duplicates could arrive at very different instants in time. However, communication is still possible and thus graceful degradation is provided.

Finally, different schemas can be used in order to connect a node to both hubs. For instance, the node can use two CAN controllers, each one connected to one hub as depicted in Figure 4. As in CANcentrate, two transceivers ($Txrx$) are needed to connect a CAN controller to a hub [1], one for the uplink and other for the downlink. However, a node will only transmit using one of the CAN controllers at a given time, while receiving from both.

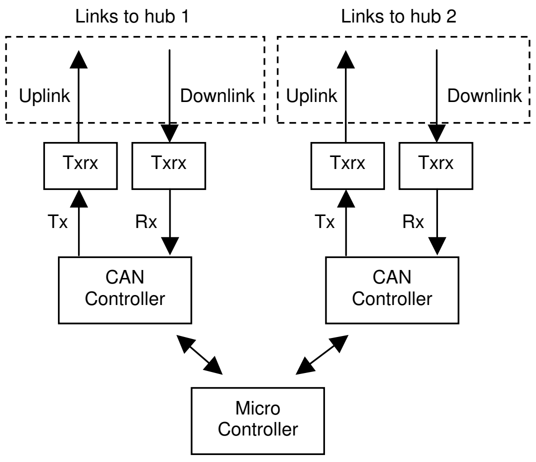

Figure 4. Example of how connect a node to both hubs

This selective behavior in the node transmissions can be enforced by several ways. For instance, each node could monitor the state of its CAN controllers. When a given controller is not able to communicate through the hub it is connected to, the node switches to the other controller thereby communicating through the other hub.

For simplicity of implementation, it is also possible to use the mechanism proposed in [7] to allow nodes to connect to both channels using a single CAN controller. However, this mechanism couples both downlinks and uplinks. Hence, upon detection of errors the node will signal them to both hubs, regardless the star in which the error was detected. This causes propagation of errors from one star to the other, unless the node isolates a faulty downlink. Therefore, in the general case, one fault occurring at any of the node's links may be enough to lead both hubs to isolate the node. Therefore, despite practical and simple, this solution should not be used when high fault-tolerance is desired.

4.3 Internal structure of the hub

In order to replicate CANcentrate, the internal structure of the hub has been slightly modified. As depicted in Figure 5, the ReCANcentrate hub continues being constituted by the same three main modules as in CANcentrate: the Coupler Module, the Input-Output and the Fault-Treatment Module.

The Coupler Module includes an additional AND gate and two additional OR gates. The first AND gate, AND_C, plays the same role as in CANcentrate. It couples the contributions from every node, $B_{1..n}$, that is connected to the hub. However, in this case, the output of this AND gate, B_0, is not broadcasted to these nodes. Instead, it is sent to the other hub by means of two identical contributions, B_{00} and B_{01}, that are routed into different sublinks within distinct interlinks. By sending these two copies of B_0, the failure of the sublink that carries one of them is tolerated.

The second AND gate, AND_T, is aimed at coupling B_0 with the contribution of the other hub. Two copies of this contribution are received by means of two identical signals, B'_{00} and B'_{01}, routed within different sublinks. Finally, the resultant signal from coupling $B_{1..n}$, B'_{00} and B'_{01}, is then broadcast to the nodes, B_T.

As in CANcentrate, each OR gate connected to AND_C is used to enable or disable the contribution of a specific node connected to the hub. In contrast, each one of the additional OR gates, which are connected to AND_T, is used to enable or disable one of the signals that carry the contribution from the other hub (either B'_{00} or B'_{01}).

Note that the output of the gate AND_T within both hubs would be the same as a CAN bus interconnecting all non-faulty nodes connected to any of the hubs. As in CANcentrate (see Section 3), the frame that results from coupling the frames from all these nodes is called *resultant frame*. In such a way, nodes can consider both hubs as one unique CANcentrate hub. This allows ReCANcentrate to keep all the CAN properties related to dependability, as well as to enforce a synchronization between both stars at bit level.

Furthermore, the usage of two AND gates within the Coupler Module makes it possible to separate the contributions of both hubs, thus allowing each hub to detect errors in the contribution of the other hub and isolate it when faulty.

In what concerns the modifications in the Input/Output Module, simply note that four transceivers have been included for transmitting and receiving the contributions of both hubs. Two of these transceivers are used to send the copies of the hub contribution (B_{00} and B_{01}), whereas the other two are aimed at receiving the copies of the contribution of the other hub (B'_{00} and B'_{01}).

Finally, regarding the changes within the Fault-Treatment Module, the internal structure and functionalities of the modules it already included in CANcentrate [1] have not been modified. The only change performed on these modules is that now they monitor B_T, instead of B_0 (see Section 3), to know the value of each bit of the *resultant frame* that is broadcasted to the nodes.

For instance, each Enabling/Disabling Unit (Ena/Dis in the Figure 5) continues using the coupled signal, now B_T, and the set of signals C together with the contribution from its corresponding port, B_i, in order to diagnose whether its port is permanently faulty or not. For removing the contribution of its port, the Enabling/Disabling Unit also uses the appropriate Enabling/Disabling signal, $ED_{1..n}$.

Additionally, two new units have been added, namely Hub Enabling/Disabling units ($HubEna/Dis_0$ and $HubEna/Dis_1$). Each one of them is responsible for detecting errors in the contribution received from the other hub at a specific sublink (either B'_{00} or B'_{01}). When one of these signals is permanently faulty, the corresponding $HubEna/Dis$ disables it by means of the appropriate Enabling/Disabling signal.

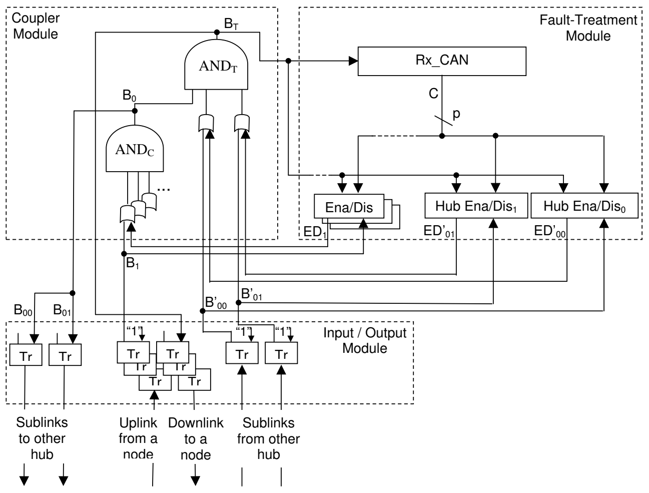

Figure 5. New internal structure of the hub

4.4 Fault diagnosis mechanisms

As explained above, diagnosis and passivation of faults occurring at nodes and links are performed by the Enabling/Disabling Units, whereas for faults occurring at the sublinks the diagnosis and passivation are carried out by the Hub Enabling/Disabling Units.

The fault-diagnosis and fault passivation functionalities performed by each Enabling/Disabling Unit are kept as they were in CANcentrate. Specifically, these functionalities consist of detecting and counting errors in the contribution of the corresponding port, produced by any of the types of faults gathered in the fault model. For each kind of fault there are specific rules for detecting the errors that may appear due to the fault and an associated threshold [1]. When the number of errors related to a given fault exceeds the corresponding threshold, the port is diagnosed as being permanently affected by that fault.

The Hub Enabling/Disabling Unit performs essentially the same functionalities as the Enabling/Disabling Unit, but introducing some small changes: the former uses slightly different rules for detecting errors due to bit-flipping faults, i.e. bit-flipping errors; and it also uses higher thresholds for diagnosing each kind of fault.

The Enabling/Disabling Unit detects bit-flipping errors by checking each one of the bits issued from its corresponding port. Particularly, the Enabling/Disabling Unit considers that the value of a bit is correct if it matches with the set of allowed values that are expected for that bit. This set of values are calculated, bit by bit, taking into account which is the current state of the *resultant frame*, as well as which is the role currently played by the node that sends the bit, i.e. whether it is a transmitter or a receiver.

In contrast, the Hub Enabling/Disabling Unit must take into account that the contribution received from the other hub can be already the coupling of the contribution sent from a transmitting node with the contributions sent by several receiving nodes. This implies only few changes in the rules followed for calculating the set of correct values for each bit.

In what concerns the differences about the thresholds, notice that as long as a given hub does not isolate a faulty port, it sends the errors issued through this port to the other hub. In such situations, the other hub will detect errors in the contribution received from this hub. Hence, if the thresholds of the Hub Enabling/Disabling Unit are not higher enough than the thresholds of the Enabling/Disabling Unit, then faults at some ports of a hub occurring near in the time may lead the other hub to unfairly diagnose that the hub is faulty.

In the worst case, all ports of a hub may consecutively fail. Thus, it may be required to consider thresholds N times greater in the Hub Enabling/Disabling Unit, where N is the number of ports. Nevertheless, this would imply a big latency for detecting a real failure of a hub contribution. Fortunately, since port failures occurring very near in the time are very unlikely, we can consider that a value of $N = 3$ is wise enough, and does not increase significatively the latency for diagnosing a real failure of a hub contribution.

4.5 Considerations on the cabling and bit rate

The length of the cabling is an important factor in a distributed embedded system, mainly due to its cost in terms of wire and the limitations it imposes on the bit rate. When compared with a bus, CANcentrate demands a higher amount of cabling, since every node is connected to the hub by means of two dedicated links. However, note that signals travel in parallel to both hubs and then in parallel in all links back to the nodes. Hence, the maximum length applies only two every pair of links. This feature may represent a substantial increase in the capacity to interconnect nodes when compared with a bus topology [1].

The increment of cabling is even bigger in ReCANcentrate because nodes are normally connected to two hubs. Nevertheless, since in ReCANcentrate the hubs are coupled, nodes are not required to be connected to both hubs for communicating. This allows to achieve higher dependability than CANcentrate without needing to duplicate the cost of the cabling.

In what concerns the limitations on the bit rate, CAN imposes an inverse relationship between the length of the cable an the maximum bit rate, as a consequence of the synchronization at bit level among all nodes [8]. In both CANcentrate and ReCANcentrate this synchronization is preserved, thus the same kind of relationship applies.

In CANcentrate it is needed to take into account the extra delay introduced by the hub (additional transceivers and internal gates) when dimensioning the bit time. In particular, from the point of view of signal propagation, the hub is equivalent to have extra cable length. In [1] it is explained which is the maximum bit rate, B', that could be achieved in a CAN bus with a bus length equal to the diameter of a CANcentrate star operating at a maximum bit rate of B.

$$B' = \frac{1}{t_{ps}} = \frac{1}{1/B - t_h} = \frac{B}{1 - B * t_h} > B$$

Where t_{ps} is the bit time of the star equivalent bus, and t_h is the delay introduced by the hub (around 300ns [1]). Note that since a signal must go through the hub two times (from the transmitting node to the receiving node and viceversa), t_h includes twice the time a signal is delayed when crossing the hub.

The comparison between ReCANcentrate an a CAN bus is slightly different. Note that in ReCANcentrate two nodes communicate simultaneously through different paths, which can include one or both hubs. Hence, in order to allow the bit value to settle before sampling, the bit time must take into account the slower communication path in the network. In such a way, we define a ReCANcentrate equivalent bus as a CAN bus whose length is equal to the slower communication path between two any nodes. The previous equation still holds, but now the factor concerning the delay introduced by the hub, t_h, must take into account if the slower path includes both hubs or not. For instance, if both hubs are included, it is necessary to double t_h.

The above discussion shows that both CANcentrate and ReCANcentrate are, from a electrical signal transmission point of view, equivalent to a bus operating at a higher bit rate. This actually means that the length of the slower communication path in ReCANcentrate, operating at bit rate B, is the maximum length of standard CAN operating at bit rate B'. Moreover, the higher the bit rate, the larger the difference. For instance, if we consider that both hubs are included in the slower communication path and that $t_h = 300ns$, the maximum length of a communication path in ReCANcentrate operating at $B = 1Mbit/s$ is equal to the length of a CAN bus operating at $B' = 2.5Mbit/s$. In contrast, when $B = 125Kbit/s$, the maximum length of a communication path in ReCANcentrate is equal to the maximum length of a CAN bus operating at $B' = 135Kbit/s$, which implies a negligible reduction in length.

5 Prototype implementation

In order to verify experimentally the proposed replicated star topology, a ReCANcentrate prototype was built including two hubs and three CAN nodes. The internal part of the hubs (the Coupler Module and the Fault-Treatment Module) was implemented using the VHSIC Hardware Description Language (VHDL) and a Field Programmable Gate Array (FPGA). A dedicated board was built for implementing the Input/Ouput Module with capacity to connect three nodes and one interlink. UTP (Unshielded Twisted Pair) Category 5/5e/6 cables were used for the links and the interlink. Each uplink / downlink pair used two wire differential lines within the same cable. Similarly, one cable was used for the interlink, using a two wire differential line for each sublink.

Each CAN node was totally implemented using commercial off-the-self components, and basically includes a PIC micro-controller [9], which incorporates one CAN controller, and four CAN transceivers. Each pair of transceivers was aimed at connecting the node with one of the hubs, following the schema specified in [1] for interfacing a CAN node with a CANcentrate hub. For managing the replicated channels we used the simplified approach referred in 4.2, which is similar to the mechanism proposed in [7]. Despite limiting the fault-tolerance properties of ReCANcentrate, this approach is very simple to deploy and still allows verifying the main features of this architecture, namely the error detection, isolation and masking capabilities applied to both nodes and hubs.

The tests to check the functionalities of the ReCANcentrate hubs were carried out at two different levels: at the level of the VHDL design of the hubs and at the level of the physical network. Concerning the first level, the state machines that constitute the hubs were checked under error-free conditions. Additionally, many different scenarios concerning the faults included in the scope of the present work were tested. Specifically, we checked that the hub correctly counts errors and isolates the corresponding ports whenever the pertinent thresholds are exceeded. In every case, the observed behavior of the hub was correct during both error and error-free conditions.

For thoroughly testing CANcentrate at the level of the physical network, a test software in each node was continuously trying to transmit CAN frames without any additional delay. This ensures that the network load is close to the maximum and that there is an arbitration every frame.

In addition, we injected faults at different hubs' ports, either corresponding to nodes or to interlinks, as well as we provoked the failure of one of the hubs. For injecting stuck-at-dominant and bit flipping faults we basically used a signal generator device and a CAN transceiver to send periodic square signals with different frequencies to several ports. Stuck-at-recessive faults were injected by disconnecting specific links or interlinks. Regarding the failure of one of the hubs, the VHDL implementation of the hubs includes a mechanism for sending, when pressing a button of the FPGA board, a constant dominant bit value or a bit-flipping stream through all the interlinks, and a constant dominant bit through all downlinks. By means of this mechanism, we conducted many tests and we satisfactorily checked the fault-treatment capabilities of ReCANcentrate.

Other experiments have been done in order to measure the performance of ReCANcentrate. The most interesting concerns the use of several cables of different lengths as well as different bit rates, in order to measure the performance of the network with respect to the length of the slower communication path. Due to implementation limitations, the maximum used bit rate was 625kbit/sec. At this bit rate, normal communication was achieved with a slower communication path of 25 meters. With 30 meters, although the communication capabilities were not disrupted, an error frame was observed each 5 ms. Notice that the maximum length of a bus operating at the same bit rate would be around 75 meters [10].

6 Conclusions and future work

Despite CAN has good dependability properties, its network structure presents several single points of failure due to the bus topology it relies on. Since the communication medium can be considered a single point of failure itself, several solutions based on replicated buses architectures have been proposed for CAN. However, they still suffer from several drawbacks concerning dependability and difficulty managing redundancy in event-triggered communications.

In [1] we explain how star topologies can represent a positive step towards improving dependability in CAN networks. We proposed a new active star topology, namely CANcentrate, that improves dependability, beyond the capabilities of any other existing star topology for CAN. In particular, it reduces the multiple severe points of failure present in a CAN bus into one single point of failure, i.e. the hub.

However, more demanding safety critical systems require to eliminate any single point of failure. In order to take profit from the advantages already achieved by CANcentrate, we propose a new communication infrastructure, called ReCANcentrate, that relies on a replicated star topology, and which overcomes the drawbacks of any other replicated communication system already proposed for CAN.

This infrastructure basically includes two interconnected CANcentrate hubs. In this way, it eliminates any single point of failure and extends the fault-treatment capabilities of CANcentrate to the overall of the communication system. Furthermore, ReCANcentrate is still compatible with commercial off-the-shelf CAN components and can be used with any CAN-based protocol.

Beyond the good properties of CANcentrate, ReCANcentrate exhibits three additional advantages. First, it provides a synchronization mechanism for transmitting and receiving frames in both stars, which is very helpful in event-triggered communications. Second, nodes may be able to communicate as long as one of its uplinks and one of its donwlinks remain non-faulty. Finally, it prevents inconsistent membership fault to occur.

In this paper, we describe the internal structure of the hub, its new functionalities, and the implementation and test of a prototype of ReCANcentrate we have developed.

Finally, it is worth noting that ReCANcentrate is suitable for both event-triggered and time-triggered communications. However, the synchronization mechanism of ReCANcentrate may be not needed for time-triggered and, in addition, it imposes extra limitations on the maximum bit rate. Hence, in the short term, we are going to address the design of a replicated star topology based on CANcentrate hubs, but specifically designed for time-triggered communications.

References

[1] M. Barranco, G. Rodríguez-Navas, J. Proenza, and L. Almeida, "CANcentrate: An active star topology for CAN networks," *WFCS'04. IEEE Workshop on Factory Communication Systems, Vienna, Austria*, 2004.

[2] H. Kopetz and G. Grunsteidl, "TTP - A Protocol for Fault-Tolerant and Real-Time Systems," *IEEE COMPUTER*, January 1994.

[3] FlexRay^{TM}, "FlexRay Communications System - Protocol Specification, Version 2.0," FlexRay^{TM}, 2003.

[4] J. R. Pimentel and J. A. Fonseca, "FlexCAN: A Flexible Architecture for Highly Dependable Embedded Applications," *The 3rd International Workshop on Real-Time Networks, Catania, Italy*, July 2004.

[5] R. Belschner, "FlexRay - Requirements Specification," BMW AG, DaimlerChrysler AG, Robert Bosch GmbH, General Motors/Opel AG, 2002.

[6] M. T. K. H. Johansson and L. Nielsen, "Vehicle Applications of Controller Area Network," Department of Signals, Sensors and Systems, Royal Institute of Technology, Stockholm, Sweden; Department of Electrical Engineering, Linkoping University, Sweden," Technical Report, 2003.

[7] J. Rufino, P. Veríssimo, and G. Arroz, "A Columbus' Egg Idea for CAN Media Redundancy," *FTCS-29. The 29th International Symposium on Fault-Tolerant Computing, Winconsin, USA*, June 1999.

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

[9] "PIC18FXX8 Data Sheet - 28/40-Pin High-Performance, Enhanced Flash Microcontrollers with CAN Module," Microchip Technology Inc., 2004.

[10] CiA, "CAN data link layer," CAN in Automation (CiA), Am Weichselgarten 26, Tech. Rep. [Online]. Available: headquarters@can-cia.de