CANcentrate: An active star topology for CAN networks

Manuel Barranco, Guillermo Rodríguez-Navas
and Julián Proenza
Dpt. Matemàtiques i Informàtica
Universitat de les Illes Balears, Spain

Luís Almeida
DET/IEETA
Universidade de Aveiro, Portugal

Abstract

Distributed embedded systems that require real-time performance need a network capable of deterministic access delay. CAN is one such network that became widespread in recent years due to its electrical robustness, low price, and priority-based access control. However, its use in safety-critical applications has been controversial due to dependability limitations that arise from its bus topology and non-guaranteed atomic broadcast. In this paper, we propose an active star topology that allows solving many of the limitations related to the first aspect by means of strong error confinement. Nodes are interconnected through an active hub that is fully compatible with existing CAN controllers. The paper compares bus and star topologies, analyzes related work and discusses the hub implementation and dependability properties. [1]

1 Introduction

The Controller Area Network (CAN) protocol is a fieldbus which fulfills the communication requirements of many distributed embedded systems. In particular, CAN 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, in a distributed system, a fault on the communication system may cause a degradation of the communication services that can lead to a malfunction of the entire system. Furthermore, communication systems based on CAN present several specific dependability problems, some of which are caused by the bus topology of this protocol.

The main drawback of any protocol using a bus topology is that the structure of the network presents multiple components, i.e. cables, connectors and circuits in nodes, which have direct electrical connections to each other without any kind of filtering. As a consequence, a fault in the bus interface of one node may generate errors that propagate to the remaining nodes and effectively prevent further communication to take place, leading to a global failure of the communication system. This situation is depicted in

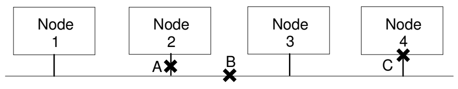

Figure 1. Examples of failures of the communication subsystem

case C of Figure 1 in which a fault in the medium access circuitry of node 4, e.g. with the transmitted bits stuck at a fixed value (a dominant value in case of CAN), blocks the communication channel and none of the nodes can communicate with each other. Similar situations can happen with short circuits in the bus media or connectors.

Moreover, a bus is shared by all communication paths between every subset of nodes. Consequently, a partition in just one point necessarily leads to a disruption of many communication paths. Even if both partitions can continue operating independently, i.e. the respective nodes can still communicate with each other, the global communication capabilities may have been substantially reduced. This is depicted in case B of Figure 1 in which a partition in the bus mid point blocks any further communication between nodes 1 and 2 with nodes 3 and 4.

Finally, case A shows the situation in which there is a partition in the node 2 local connection to the bus that does not affect the bus integrity and which leaves the inputs of the node's reception port floating. Consequently, node 2 becomes isolated but the communication among the remaining nodes is unaffected. From the communication system point of view, this is the desired behavior when a fault occurs in one node or node bus interface, because it exhibits the least impact on the communication system itself.

The general framework within which this work has been developed addresses two main objectives. The first objective is to prevent situations in which one single fault in the communication system affects the communication capabilities of more than one node, e.g. cases C and B in Figure 1. In practice, we achieve this objective by using an appropriate topology, namely a star, whose hub enforces the necessary error confinement. As it will be seen later, the hub in a star is a natural location to perform error detection and filtering, blocking their propagation from faulty nodes at the respective hub port. On the other hand, the links that connect each node to the hub are dedicated and thus, faults

[1] The contents of this article 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 5th IEEE International Workshop on Factory Communication Systems the 22th of September of 2004.

occurring on them can be isolated together with only the respective node. Nevertheless, the star topology still contains one single point of failure, i.e. the hub, which if faulty may lead to a global communication failure. Even so, we consider the star topology to be a good choice because it is easier to improve dependability for the unique single point of failure of the star, e.g. the hub can be placed in a more protected area within the system, than for the multiple components that may cause a communication failure in a bus topology. Moreover, replicated star topologies can be used to tolerate either hub as well as link faults. A broader analysis of the communication system dependability aspects can be found in [9] concerning a comparison between TTP/C and Flexray. The specific issue of network topology is also therein discussed, in which the benefits of a star topology over a bus are clear. Such benefits are also the reason for the shift in Ethernet networks, which occurred through the 90s, from a bus to a star topology.

The second objective of our general framework is to exploit the potential dependability advantages the star topology offers to further improve dependability of a CAN network. This dependability improvement can be achieved by taking advantage of an intelligent hub, provided with the necessary capabilities to mitigate the impact on the entire system caused by faults not included in the scope of the first objective, i.e . those which do not prevent nodes from communicating. For instance, the hub could prevent that any node impersonates another node, i.e. masquerading failures, thus restricting the failure semantics of the nodes.

This paper is devoted to the first objective, only, and it addresses the design of a simplex star topology with one active hub. Both the design of a replication scheme for our CAN star topology as well as the achievement of the second objective pointed out above will be the subject of future work.

Two requirements were imposed from the beginning on the star design: the preservation of all the characteristics of the CAN protocol which are related to dependability and the option for mechanisms that may facilitate the future development of the redundant star scheme indicated above. In what concerns the first 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, off-the-shelf CAN controllers can be used in the nodes of the star.

In the following section we discuss the properties of existing solutions to improve the dependability properties of CAN, focusing on the advantages of a simplex star topology with respect to simplex and replicated bus topologies. Moreover, existing work on star topologies for CAN is also presented. Section 3 presents the architecture of the star topology for CAN designed in this work. Section 4 discusses the mechanisms which the hub implements in order to detect errors and to prevent their propagation. Section 5 explains the policy the hub follows in order to enable previously blocked ports due to an error confinement decision. Section 6 addresses issues related to the cabling length. Section 7 considers future work and Section 8 concludes the paper.

2 Problem statement

Despite the good dependability properties exhibited by the CAN protocol, it still presents some drawbacks. In particular, there are multiple components of the network whose faults may cause a global failure of the communication system, as referred in Section 1. Notice that, in spite of the existence of several techniques proposed previously in the literature to confine the effect of such faults in CAN systems (Sub-section 2.2), such techniques are not completely effective in the sense that they still allow for single faults of different types and occurring in different physical points to cause global communication failures.

For the purpose of this paper, we will consider *global communication failures*, or alternatively *severe communication failures*, as those that affect the communication capabilities of two or more nodes in the system. The particular component faults this work focuses on are described in the following subsection.

2.1 Fault model

As referred in Section 1, this work focuses on component faults that the occurrence of a single one may cause a global communication failure. Particularly, these are:

- *Stuck-at node* fault. It occurs whenever a given node is damaged and issues a constant bit value. Two types of stuck-at fault exist: stuck-at-dominant and stuck-at-recessive faults, depending on whether the bit value issued by the faulty node has dominant or recessive value respectively. Since the physical layer of CAN is equivalent to a logic-AND of every node's contribution, only a stuck-at-dominant fault may cause a global communication failure (the recessive bit is interpreted as the logical '1' value).

- *Shorted medium* fault. This occurs whenever the medium is electrically connected to battery or to ground due to a short-circuit. For obvious reasons, this fault prevents any communication. When fault-tolerant cabling is used, as recommended in CAN [8], such a fault requires that both wires are shorted to a fixed low impedance electrical source.

- *Medium partition* fault. It occurs whenever the medium is interrupted in such a way that the network is broken into several subnetworks, which are called *network partitions*. Therefore, any two nodes which are each one in a different partition can no longer communicate with each other. Moreover, signal reflections at the open extremities may cause channel errors that prevent nodes in the same partition from communicating properly [8].

- *Bit-flipping* fault. This is a particular case of a *babbling idiot* fault. A babbling idiot fault occurs whenever a component of the network (either a node or a medium) exhibits a *fail-uncontrolled* behavior and starts sending erroneous information with no restrictions either in the time domain or in the value domain. Some potential causes of this fault are: a node that enters into an infinite loop sending messages continuously; a damaged node that sends random bit values; a bad welding on the medium connector that generates random bit values, etc. Some babbling idiot faults will not cause a global failure of the communication system as defined in this paper, but just a degradation of its performance. Therefore, in the context of this paper, we will refer only to the case of a faulty component that continuously generates random bit values, i.e. a bit-flipping fault, which leads to a global communication failure.

2.2 Potential solutions

Some of the faults presented above can be confined in bus systems, up to a certain extent, using techniques that are already known. These techniques rely on the use of replicated transmission media as well as on the use of *bus guardians*. However, they do not guarantee that a single component fault never causes the global failure of the communication system because of characteristics that are inherent to the bus topology.

The use of replicated transmission media generally allows nodes to detect a faulty medium by comparing the values received from each of the replicas [13]. In this way, nodes can disable the faulty medium so that the communication system can still provide a correct service. Nevertheless, this solution does not prevent a faulty node (e.g. a stuck-at-dominant node) from causing a failure of the whole communication system by sending erroneous information to all replicated media. Moreover, this solution has a more subtle weakness; regardless of the routing of the replicated media, they have to come together near every node of the system. This spatial proximity is a potential cause of *common-mode* failures of the replicated media system. For instance, a smash near the node may cause a partition of all media, thus leading to a global communication failure [10].

On the other hand, the use of bus guardians allows the confinement of erroneous transmissions within one node, enforcing a fail-silent behavior [3]. A bus guardian is a device which supervises the output of a node to its bus interface in order to detect incorrect behavior. In this way, a faulty node, such as a stuck-at-dominant node or a bit-flipping node, can be easily detected and isolated from the rest of the system. Nevertheless, the weak point of this approach is that independence between a node and its corresponding bus guardian is not completely ensured due to potential common-mode failures. These can be caused either by spatial proximity of a node and its bus guardian, or by sharing resources or procedures, e.g. power supply, system clock, clock synchronization algorithm).

Even though the use of replicated media as well as bus guardians significantly improves the dependability characteristics of CAN, these mechanisms still allow multiple components to cause severe failures of the communication subsystem, and therefore they do not fulfill the aim of this work.

Therefore, alternative solutions have been researched, namely those based on a star topology. In this case, each node is connected to a central element, the *hub*, by its own *link*. This provides a natural way to enforce confinement of faulty transmission media by isolating the respective links at the respective hub ports. Furthermore, the hub has a privileged view of the system, as it simultaneously knows the contribution from every node and thus, it can play the role of bus guardian of each node. In this way, spatial proximity between a node and its corresponding bus guardian is avoided. Moreover, the links of a star topology only come into spatial proximity at the center of the star.

It is obvious that the main drawback of a star topology is that the hub represents a single point of failure. In addition, the complexity of the error-detection and fault-treatment mechanisms implemented in the hub implies that its probability of failure is higher than it could be for simpler components, such as a bus guardian. Nevertheless, different strategies can be adopted in order to face this problem. For instance, the hub reliability can be increased by placing it in a well-protected zone inside the physical system or by investing in its quality or even by adopting a replicated star topology. However, as stated in Section 1, this problem is out of the scope of this paper and will be addressed in the future.

Table 1 summarizes the characteristics of the discussed topologies. Note that, despite the improved properties of replicated buses that also use bus guardians, an ideal star topology still exhibits better properties in what refers both to node and to media faults.

Table 1. Severe failure as a consequence of a fault of a single component in different topologies

	Simplex bus	**Replicated bus + bus guardians**	**Ideal simplex star**
Stuck-at-d node	Yes	Yes[a]	No
Stuck-at-d link	Yes	No[c]	No
Stuck-at-r node	No	No	No
Stuck-at-r link	Yes	No[c]	No
Bit-flipping node	Yes	Yes[a]	No
Bit-flipping link	Yes	No	No
Shorted medium	Yes	Yes[b]	No
Medium partition	Yes	Yes[b]	No

Yes: A severe failure of the communication system may occur.

No: A severe failure of the communication system will not occur.

[a] In the case of common-mode failures between the node and its bus guardian.

[b] In the case of spatial proximity failures.

[c] Only if the stuck-at failure does not affect all replicated medias.

2.3 Available star topologies for CAN

Even though the star topology provides a good infrastructure to improve the dependability of the communication system, the adoption of such a topology is not enough. Additional mechanisms should be implemented in the hub in order to detect and isolate faulty components and achieve the behavior of the mentioned *ideal star*. Some star topologies for CAN can be found in the literature [12] [6] [4]. Although these solutions provide some mechanisms to deal with faulty components, none of them includes the minimum mechanisms which are required to prevent the faults described in Section 2.1 from generating severe communication failures.

In [12] [6], a passive star network topology for CAN is presented. This solution relies on the use of a central element, called the *star coupler*, which acts as a concentrator where all the incoming signals are coupled. The result of this coupling is then broadcast to the nodes. In what concerns dependability, the unique advantage that this solution presents is the reduction of the spatial proximity problem between different links. This reduction is possible since the links only come into physical proximity at the center of the star. However, this kind of star couplers shows some technical drawbacks that discourage its use from the practical point of view. On one hand, large coupling loses impose strong limitations on the star radius (5 to 10 meters) and hence force nodes to communicate at low bit rates. On the other hand, the coupling of the incoming signals causes some electrical problems, such as resonances, harmonics or disturbances, which require the use of complex hardware solutions.

Another star topology for CAN has been presented in [12] [6]. This topology relies on an active star coupler, which receives the incoming signals from the nodes bit by bit, implements a logical AND, and retransmits the result to all nodes. In this solution, each node is connected to the star coupler by an independent link composed by two optical paths. In the star coupler, there is a transceiver for each link (a so-called node-coupler transceiver) as well as an internal CAN bus with few centimeters of length. In a first stage, signals from each link are received by the respective node-coupler transceiver and transmitted without any processing into the internal CAN bus. In a second stage, the resultant signals from the internal CAN bus are received by each node-coupler transceiver and retransmitted towards the corresponding node. Although this solution overcomes the technical problems of the solution previously mentioned, it does not include any mechanism to deal with faulty components. Therefore, from the dependability point-of-view it just reduces the spatial proximity problem between different links.

In [4] another star topology is suggested for CAN, namely StarCAN. The main goal of this solution does not address network dependability, but network performance. In particular, StarCAN achieves either an extension more than 10 times longer than a typical CAN network or a bit rate 10 times higher than a typical CAN network. Nevertheless, in order to fulfill this goal, StarCAN sacrifices one of the most important characteristics of CAN, the in-bit synchronization. This decision has an enormous impact on the dependability properties of the network. On one hand, the lack of in-bit synchronization jeopardizes the so-called *data consistency* of the CAN network, since inconsistency scenarios [14] [11] turn out to be more likely. On the other hand, despite the use of some CAN mechanisms, e.g. arbitration and error signaling, off-the-shelf CAN controllers cannot be used raising issues about the practicality of the solution.

Therefore, none of the star topologies for CAN that have been studied fulfills our goal of preventing single faults in different components from generating a severe failure of the communication system. In fact, they do not even address this kind of failures, behaving as a bus with enhanced resilience to spatial proximity faults.

This justifies the design of a new star topology for CAN, with special focus on achieving the dependability properties referred in Table 1 for the ideal simplex star topology.

3 Design of the star topology

In Section 2, it has been shown that CAN networks based on a bus topology do not fulfill the strong dependability requirements of many safety-critical systems. In particular, it has been discussed that the use of a bus topology allows a fault on one network component to cause a severe failure of the communication system, i.e. preventing more than one node from communicating with the rest of the system. In this section we describe the architecture of a star topology for CAN provided with error confinement mechanisms that enforce fault-containment regions for each node and respective link.

3.1 Design basics

Probably the most important characteristic of CAN is the dominant/recessive transmission. This property guarantees that whenever one of the nodes transmits a dominant value, this value is received by all the nodes in the network. In contrast, a recessive value is only received as long as every node issues a recessive value. Moreover, CAN communication relies on a complex bit synchronization mechanism which guarantees that nodes have a quasi-simultaneous view of every single bit on the channel. This mechanism uses the recessive to dominant transitions of the signal on the channel in order to keep the nodes of the network synchronized with respect to the node which is transmitting (the so-called *leading transmitter*). This bit synchronization limits the maximum bit rate of the network, but at the same time allows definition of a number of additional mechanisms (e.g. bit-wise arbitration, ACK bit, error frame), which significantly improve the dependability and real-time properties of CAN networks [8]. Due to the relevance of these mechanisms, it is very important to preserve them even if a star topology is used instead of a bus.

In order to keep the dominant/recessive transmission, the hub must implement a logical AND function of the trans-

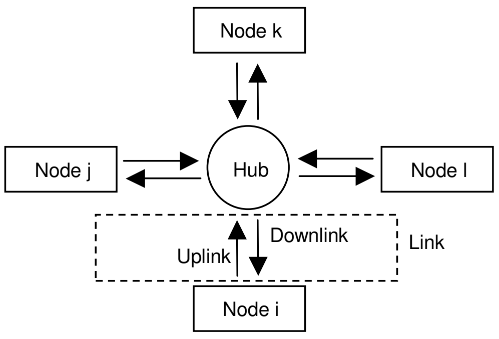

Figure 2. Architecture of the network

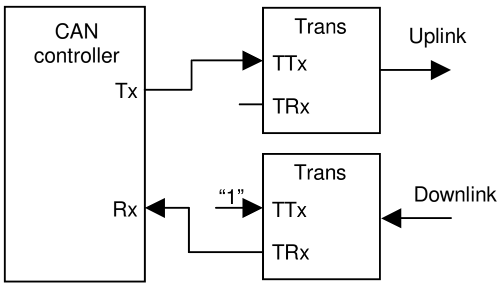

Figure 3. Configuration of the transceivers within one node

missions received from every node. Moreover, and in order to preserve the in-bit synchronization, this logical AND must be performed within a fraction of one bit time, despite the extra delay which the internal circuitry of the hub may cause.

Furthermore, the hub must implement some mechanisms in order to identify faulty components. These mechanisms, which are thoroughly described later on, require the hub to be able to discriminate the signal which any node transmits from the signal which the hub broadcasts to the nodes. A simple way to separate both signals is through the use of two different cables for each link that connects each node to the hub. Figure 2 shows the resulting architecture in which there are only point-to-point unidirectional electrical connections.

The cable which carries the signal from a node to the hub is called the *uplink*, whereas the cable which carries back the resulting signal from the hub to the node is called the *downlink*. Each cable is of the same type as the twisted copper wiring used for implementing typical CAN buses, which have a good resilience against electromagnetic interferences. Moreover, each cable is terminated at both ends, i.e. node and hub.

Therefore, two transceivers are required at the end of each link, i.e. one for the uplink and another one for the downlink. Figure 3 illustrates how the transceivers are connected at the node's end. Note that the data input line (TRx) of the uplink transceiver is left open whereas the data output line (TTx) of the downlink transceiver is forced to have a recessive level (the logical '1' value).

It is important to remark that the architecture presented in this work can be implemented with both off-the-self CAN controllers and off-the-shelf CAN transceivers. This makes the solution practical and relatively low-cost. Nevertheless, the hub requires some specifically designed hardware, as discussed next.

3.2 Internal structure of the hub

The hub plays a crucial role in the star topology since it performs two fundamental functions. On one hand, it implements the logical AND function which allows preservation of the dominant/recessive transmission of CAN as well as the rest of dependability mechanisms of CAN. On the other hand, it implements a number of fault-treatment mechanisms in order to detect and isolate faulty components.

Therefore, the hub is divided into three modules, namely the *Input/output Module*, the *Coupler Module*, and the *Fault-treatment Module*. The structure and interconnections of these modules are depicted in Figure 4.

The Input/output Module is made up of a number of transceivers; two for each link. As Figure 4 shows, one transceiver is assigned to every uplink in order to convert the physical signal received from each node into a logical value that the hub can process, $B_{1..n}$. Moreover, one transceiver is assigned to every downlink so that the logical output of the hub, B_0, is converted into a physical signal that is broadcast to every node.

The Coupler Module is made up of an AND gate, which performs the coupling of the uplink signals, and a number of OR gates, one per link, which allow the hub to disable the contribution from a specific uplink to the global AND. This configuration causes an additional delay on the signal that the nodes receive. For the in-bit synchronization of the nodes, this additional delay has to be taken into account as a part of the propagation time. For all purposes it is similar to the extra delay caused by an equivalently longer cable in a bus system.

Note that the output of the AND gate is connected to each and every one of the downlink transceivers. In this way, the output of the hub does not interfere with the signals received through the uplinks, so the contribution of every node remains separated and further mechanisms can be applied in order to identify a faulty component.

The main purpose of the Fault-treatment Module is to detect and isolate the faulty components from the system, so they cannot cause severe communication failures. This function is carried out by performing both *fault diagnosis*, which aims at finding out the port to which a faulty component is connected to, together with *fault passivation*, which aims at isolating the faulty component from the system.

The fault-diagnosis mechanisms of the Fault-treatment Module require identification of the contributions from every uplink as well as knowledge of the current state of the *global frame*, i.e. the frame that is broadcast to all nodes.

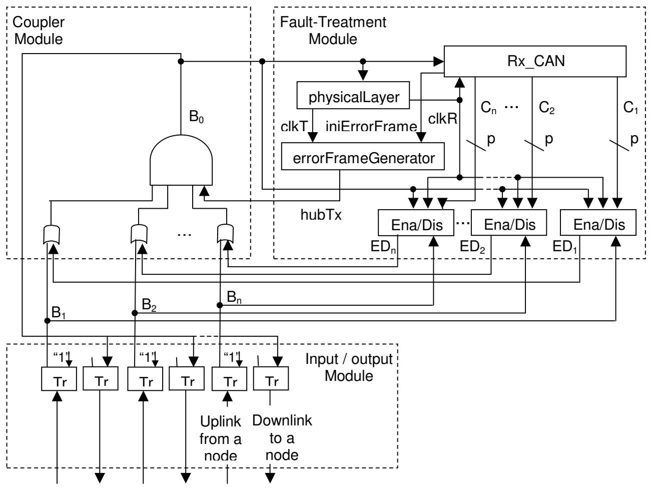

Figure 4. Internal structure of the hub

Fortunately, the use of two cables for each link keeps the contribution from each link separated, and therefore the physical location of the faults can be easily performed. However, this architecture does not allow the hub to discriminate between faults that are caused by a faulty transmission medium and faults that are caused by a faulty node. In fact, each node and its respective link form one fault-containment region which is connected to one hub port. Therefore, from the point of view of the hub, either a faulty medium or a faulty node are viewed as a *faulty port*. The mechanisms that have been defined in order to diagnose a faulty port are thoroughly described in Section 4.

The knowledge of the *global frame* current state is achieved by using two modules that implement similar reception mechanisms of a CAN controller (*Physical layer* and *Rx_CAN* modules). On one hand, the Physical Layer Module synchronizes the hub with the bit stream and generates the transmission and the reception clocks. As in a CAN node, the transmission clock indicates the instant of time that a transmission bit value must be driven into the medium; whereas the reception clock indicates the instant of time that the input signal from the medium (the coupled signal in the case of the hub) must be sampled.

On the other hand, the Rx_CAN Module monitors the bit stream at the coupled signal and generates a set of signals, $C_{1..n}$, that indicates which kind of bit of which kind of frame is being broadcast (e.g. whether the bit is a stuff bit or not; the expected stuff bit value; as well as which frame field of which kind of frame the bit belongs to), i.e. the RX_CAN Module synchronizes the hub with the current state of the global frame. Such information together with the contribution from the different ports, $B_{1..n}$, is used by the *Enabling/disabling* units (*Ena/Dis*) for diagnosing faulty components (Section 4).

Whenever a given Enabling/disabling Unit diagnoses one hub port as being faulty, it removes the contribution of this port from the system by issuing a logical '1' to the *port disable* signal, $ED_{1..n}$, which is connected to the OR gate that corresponds to the faulty port (Figure 4). This effectively removes the contribution from this port to the global AND, being equivalent to disconnecting the link, and the corresponding node, from the star. In general, this mechanism is similar to the one proposed in [13] to manage, locally in each node, the media redundancy in a replicated bus topology.

Finally, to improve the synchronization of the hub with the global frame current state, the *Error Frame Generator Module* within the Fault-treatment Module allows the hub to behave as an active node, i.e. globalizing the error conditions by means of an error active frame. This module receives the order of globalizing error conditions detected in the global frame from the RX_CAN Module and transmits an error frame by means of a dedicated contribution, *hubTx*, driven into the global AND.

Moreover, although the ability of globalizing error situations is not used by any fault-treatment mechanisms presented in this paper, it allows the hub to abort the transmission of any frame and, therefore, to implement further fault-treatment mechanisms, e.g. to abort the transmission of a forged message due to a masquerading fault. This will be addressed in future work.

4 Fault-diagnosis mechanisms

The faults that are diagnosed by the Fault-treatment Module are: stuck-at-dominant, stuck-at-recessive and bit-flipping faults (Section 2.1). Notice both shorted media and media partitions manifest themselves at the hub port as stuck-at faults.

The fault diagnosis mechanisms are essentially implemented by the Enabling/disabling units that operate separately over each port. The internals of this units are showed in the Figure 5.

On one hand, the Enabling/disabling Unit has a dedicated *event counter* and an associated *management module* for each type of fault that must be detected: the *Dominant Bit Counter* (DBC) and the *BDC Manager Module* for stuck-at-dominant faults; the *Non-Acknowledge Counter* (NACKC) and the *NACKC Manager Module* for the stuck-at-recessive; and the *Bit-Flipping Counter* (BFC) and the *BFC Manager Module* for the bit-flipping faults. Each management module basically analyzes the port contribution B_i and the control signals C_i, from RX_CAN, in order to decide how to increase or decrease its related event counter.

On the other hand, the Enabling/disabling Unit has a *Threshold Control Module* that diagnoses the port failure, isolates the port contribution by setting the corresponding ED_i signal and resets all the managers, whenever a given counter exceeds its specific threshold.

The Threshold Control Module is also responsible for notifying the user about the state of the port by means of the *portStatus* state signal. Each hub's port can be in three different states: *idle*, *active* and *disable*. The idle state indicates that the port contribution is enabled, but that no activity (only recessive bits) has been detected on it for a specified interval of time. In the active state, the port con-

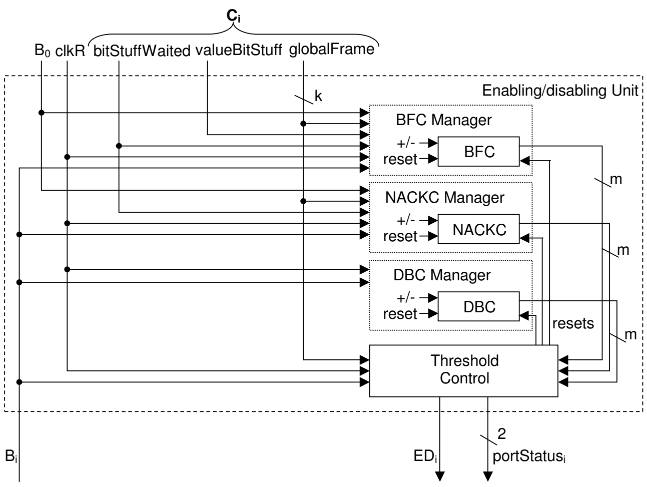

Figure 5. Internals of the port enabling/disabling Unit

tribution is also enabled, but it indicates that some activity is regularly detected on the port. Finally, the disable state indicates that the port contribution is disabled due to a fault diagnosis. Section 5 explains how the Threshold Control Module manages the port states.

4.1 Stuck-at-dominant faults

In order to detect stuck-at-dominant faults, the DBC counts the number of consecutive dominant bits that are received from the respective port. These values are compared with a given maximum number of allowed consecutive dominant bits, T_{nskd}. The T_{nskd} threshold takes into account two different contributions:

$$T_{nskd} = (T_{stuff} + 1) + N * T_{errorFlag}$$

The first term, $T_{stuff} + 1$, specifies the minimum number of consecutive dominant bits that violates the stuffing rule in a CAN network (6 bits). The second term, $N * T_{errorFlag}$, specifies the maximum number of bits of consecutive or overlapped error flags allowed before diagnosing a stuck-at-dominant fault.

The DBC is increased each time its corresponding manager observes a dominant bit value on the uplink, and it is reset as soon as its manager observes a recessive value. Whenever a DBC exceeds the threshold T_{nskd}, the Threshold Control Module sets the port to disabled state isolating it.

Notice that for $N = 1$ the threshold coincides with the one proposed in [13]. In that case, the threshold can be reached with a single bit error in the error flag that follows a dominant stuff violation, leading to an erroneous diagnosis of a stuck-at-dominant fault. Using a higher value of N reduces the probability of performing an erroneous stuck-at-dominant diagnosis.

The value of N can be configured depending on the application, but at this stage we consider that $N = 2$ is flexible enough and does not imply a significant loss of reactivity in diagnosing stuck-at-dominant faults.

4.2 Stuck-at-recessive faults

Due to the AND function the hub implements, a port suffering a stuck-at-recessive fault does not interfere with the communication among the rest of the nodes in the star (the recessive bit is interpreted as the logical '1' value). Therefore, this kind of fault does not generate a severe failure of the communication system. Nevertheless, detection of this kind of faults may still be useful in order to implement additional fault tolerance mechanisms in higher levels of the system architecture, for example to detect a crashed or absent node.

The detection of stuck-at-recessive faults poses an additional difficulty because a CAN node may be a long time without transmitting and it would be thus theoretically impossible to discriminate between a stuck-at recessive node and an operational but silent node. However, the CAN protocol specifies that every CAN controller must transmit a dominant bit in the ACK field of every frame that is correctly received [8]. Therefore, the absence of this bit can be used to detect stuck-at-recessive ports.

For port i such detection is carried out by the *Non-Acknowledge Counter* (NACKC) and its management module, *NACKC Manager*. The NACKC Manager Module compares the state of the global frame (*globalFrameState* signal) with the port contribution (B_i), in order to increase the NACKC whenever an ACK omission is detected, and to decrease it when the port receives a dominant ACK bit. It is important to note that by decreasing the counter, instead of resetting it when detecting a dominant bit value, the hub can detect not only stuck-at-recessive failures, but also nodes that tend to be stuck-at-recessive (e.g. a node in the error passive state that usually detects local errors during frame receptions).

When the NACKC exceeds a predefined threshold, the Threshold Control Module does not disable the port, but sets it to the idle state, which notifies the user about the inactivity of the port.

4.3 Bit-flipping faults

In the context of this work, we only consider a special case of a babbling-idiot fault, i.e. the bit-flipping fault. A bit-flipping fault occurs whenever one port receives an erroneous arbitrary sequence of bits, without any restriction.

The CAN standard specifies a mechanism that the nodes can use in order to detect such kind of faults: a *Transmission Error Counter* (*TEC*) and a *Reception Error Counter* (*REC*). These counters are increased and decreased following some rules established in the CAN specification, and may cause a node to reduce its impact on the communication process by going into the *error passive* state or to disconnect itself from the network [8] in order to prevent further propagation of local errors.

But, since faulty nodes may perform the operations related to the TEC and REC in a wrong way or simply crash and links can also be the source of bit-flipping faults, we decided to implement in each Enabling/disabling Unit a dedicated *Bit-Flipping Counter*, *BFC*, and its associated *BFC Manager Module*. The error confinement strategy is implemented as follows. On one hand, the correct error confinement strategy for increasing the BFC implies that the hub knows which node is generating errors. Specifically, the hub must be able to detect when a node is generating a primary error.

A node can generate errors either during a *normal* (data/remote) frame or during an error frame. Each node contribution has a predefined valid format during a normal global frame. However, there are many different error scenarios that lead to many combinations of different error flags and error delimiters that can be sent by the nodes, e.g. under error situations a given pattern of dominant and/or recessive bits sent by a node can be an erroneous bit followed by an error flag, the rest of an error flag, a corrupted error flag, a corrupted error delimiter, etc. Therefore, the detection of a primary error during a normal global frame is straight, while such detection during an error global frame cannot be ensured.

For detecting a primary error during a normal global frame, the BFC Manager Module only needs to monitor when a node's contribution is erroneous. The detection of an erroneous contribution depends on the type of node (transmitter or a receiver), on the actual state of the global frame and on the agreement with the stuffing rule. Whenever a bit value transmitted by node i is erroneous, its respective BFC_i is increased.

Nevertheless, under error conditions many different situations can occur concerning the nodes contribution with respect to the global frame. Thus, the detection strategy performed by the BFC Manager Module needs using more complex rules that just a simple comparison:

- If an erroneous contribution occurs through the port i, but it does not cause an error in the global frame, the BFC Manager Module assumes that it was an erroneous bit not detected by any node or that the node connected to the port is sending an active error frame that still has not generated a global error. Thus, if the contribution after the error does not match with a correct contribution or with an error frame, the BFC Manager Module increases the BFC_i.

- When a global error is detected, the BFC Manager Module expects that each node (except the node/s who caused the error) starts to transmit an error flag. If an error flag is not received from the port i after the global error detection, the BFC Manager Module increases the BFC_i.

- A correct active error flag sent by a node is formed by six consecutive dominant bits. Nevertheless, if a node detects an extra error during its own error flag, it starts to transmit the error flag from the beginning. Thus, a node can send a correct dominant value in any bit of its own error flag, but monitors an incorrect recessive bit, leading to the transmission of overlapped or consecutive error flags. To cope with such a situation, each time that the BFC Manager Module monitors a number of consecutive dominant bits bigger than a multiple of an active error flag's length, the BFC_i is increased.

- After a global error flag, the hub waits to receive from all the nodes a correct error delimiter composed only by recessive bits. If a node i transmits a dominant bit value during its own error delimiter, the BFC Manager Module would increase the respective BFC_i. This rule tries to cope with situations where either a node sends a corrupted active error flag, or generates an additional primary error during the error delimiter.

On the other hand, the strategy for decreasing the BFC is based on the detection of a relevant event as a good indication of correct operation. Specifically, the BFC Manager decreases the BFC whenever the global frame reaches the *bus idle* CAN state.

Finally, it is important to discuss the value of the BFC threshold and the number of units that the BFC has to be increased or decreased, depending on the BFC's management rule that is applied. These values are specific to each application and should be configured by the user. However, we consider a good choice to adopt a general strategy based on the standard CAN. This strategy holds as follows.

- When a node i is responsible of an error during a normal frame, the BFC_i is increased in +8.

- When a node i generates an error during its own error frame contribution, the BFC_i is increased in +16. The hub penalizes such situations because errors during the error frame are a good indication of a bit-flipping behavior.

- When the state of the global frame reaches the *idle* state, all the BFC are decreased in -1. This asymmetric approach of increments and decrements is intended to require high reliability of the nodes and links.

- The bit-flipping threshold can be set to 127 units.

5 Reintegration policy

To increase the tolerance to sporadic errors, the hub implements an automatic reintegration policy of disabled ports, which state machine is depicted in Figure 6.

A port is set to the *idle* state whenever the hub is initialized or a stuck-at-recessive failure is detected. In such state, the port's contribution is enabled.

As soon as the hub receives a *meaningful* contribution from a port, e.g. an ACK signaling, an error flag transmission or a dominant bit contribution during the arbitration, the corresponding Threshold Control Module sets that port to the *active* state. The single difference between the *idle* and *active* states is that the second one indicates that the

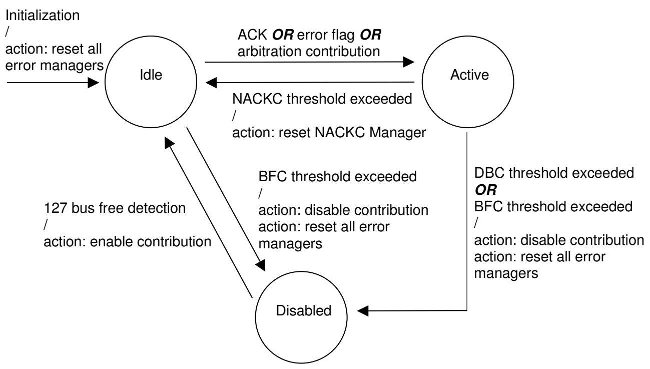
Initialization
/
action: reset all error managers
Idle
ACK OR error flag OR arbitration contribution
Active
NACKC threshold exceeded
/
action: reset NACKC Manager
BFC threshold exceeded
/
action: disable contribution
action: reset all error managers
127 bus free detection
/
action: enable contribution
DBC threshold exceeded
OR
BFC threshold exceeded
/
action: disable contribution
action: reset all error managers
Disabled

Figure 6. Reincorporation strategy schema

node is regularly participating in the communication process.

Whenever the stuck-at-dominant or the bit-flipping thresholds are exceeded, the Threshold Control Module sets the port to the *disabled* state and resets the respective error managers. Once a port is in that state, the corresponding Threshold Control Module waits to monitor a constant recessive contribution during 127 CAN bus free occurrences. After detecting this period of *inactivity*, the port will be set again to the *idle* state and its contribution will be enabled.

The reintegration policy allows an autonomous performance of the hub because it is able to return to normal operation by itself.

6 Considerations on the cable length

The length of the cabling is an important factor in a distributed embedded system. In CAN, due to the synchronization at the bit level among all nodes, there is an inverse relationship between the bit rate and the maximum bus length. In our star topology, these relationship is preserved as the bit level synchronization of CAN is maintained. However, since the signals travel to the hub and then in parallel in all links back to the nodes, the maximum length applies only to every pair of links. This feature may represent a substantial increase in the capacity to interconnect nodes when compared with the bus topology. Consider a system with N nodes separated in space. The total length of the bus that interconnects such nodes is Lb (see Figure 7b). On the other hand, consider all nodes interconnected by means of a hub with link i having length L_i (see Figure 7a). Despite depending on the nodes placement, for the general case, $Lb >> L_i + L_j, \forall_{i,j}$ (see Figure 7). This is a major benefit of the star topology. On the other hand, also for the general case, $Lb < \sum_i (L_i)$ meaning that the total length of the cabling system is longer in the star topology. Nevertheless, the superior connectivity of the star may allow using higher bit rates than with a bus due to the stronger limitation on the bus length.

In what concerns the length of each star link, the bit level synchronization imposes a limitation on the sum of the lengths of every pair, as stated above. Let this limitation be $Lmax_s$, the star diameter. In order to have the lengths of all links independent of each other, the previous constraint implies that $\forall_i L_i < Lmax_s/2$.

To derive $Lmax_s$, the maximum diameter of the star, we need to analyze the propagation of the electrical signals from end-to-end. With respect to a bus topology, the star presents an extra delay caused by the hub (additional transceivers and internal gates). This delay is dominated by the former factor since the gate delays are negligible (order of 1ns or less using modern technologies) when compared with the transceiver delay (around 150ns for fast transceivers, including bus to reception pin and transmission pin to bus [7]). For a given bit rate B, the bit time $1/B$ has now to account for both propagation effects as in a bus plus hub delay. For the former aspect, consider all the parts that contribute to establish the bit time in CAN using the normal bus topology. Let this be t_{pb} (notice that $t_{pb} = 1/B$ by definition). In a star, all these parts related to propagation effects also have to be considered, taking t_{ps}. However, the bit time now also includes the hub delay t_h, thus $t_{ps} = 1/B - t_h$. Therefore, from the point of view of signal transmission, we can define a star equivalent bus, with propagation effects taking t_{ps} and operating at a bit rate B' so that

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

The previous equation shows that a star is, from an electrical signal transmission point of view, equivalent to a bus operating at a higher bit rate. Moreover, the higher the bit rate, the larger the difference. Therefore, the maximum diameter of the star $Lmax_s$, operating at bit rate B, is the maximum length of standard CAN operating at bit rate B'. For example, given the 150ns figure of hub delay referred above, a star operating at $B = 1Mbit/s$ has a maximum diameter equal to the length of a bus operating at 1.18Mbit/s. On the other hand, if $B = 125Kbit/s$ then the maximum diameter of the star equals the length of a bus operating at 127.4Kbit/s which implies a negligible reduction in length. To calculate the effective bus length for these transmission rates refer to [5]

7 Future work

The star topology offers many possibilities to improve dependability and real-time capabilities of a CAN network. In fact, more fault detection mechanisms can be integrated into a central privileged node, the hub, that may allow either further restricting the failure semantics of CAN-based communication systems, as well as notifying and isolating faulty regions. Moreover, the star topology in CAN imposes a cabling length constraint on pairs of nodes, only, and thus more nodes and larger areas can be served for a given transmission rate, or higher rates can be used for a given length constraint. Given that the real-time properties of the CAN medium access control are strictly maintained, the proposed star topology for CAN can thus improve the real-time performance of the network by means of a higher throughput.

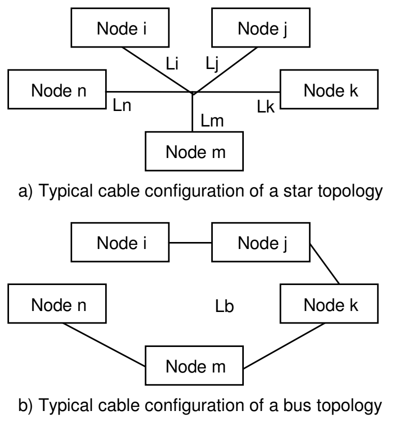
Node i
Node j
Li
Lj
Node n
Node k
Ln
Lk
Lm
Node m
a) Typical cable configuration of a star topology
Node i
Node j
Node n
Lb
Node k
Node m
b) Typical cable configuration of a bus topology

Figure 7. Comparison between the lengths of the cabling system in a star and in a bus

In the short term we will consider some improvements over the simplex star solution namely, the use of a single cable in each link to reduce the costs of the wiring as well as the replication of the hub to eliminate the only single point of failure that remains in the system.

A particular approach that will be considered in future work is the use of switched strategies, which can be used to reduce the overhead introduced by damaged frames and error signaling, as well as to segment the network and further increase the global throughput.

8 Conclusions

Despite being widespread in distributed embedded systems, the use of CAN in safety-critical applications has been a controversial topic. This is due to a few factors such as the bus topology. In fact simplex bus topologies suffer from several impediments to enforce error confinement while replicated buses may exhibit common mode and spatial proximity faults. On the other hand, star topologies may represent a positive step due to the key role that the hub can perform to diagnose and passivate faults. In fact, it allows reducing the number of components whose failure can cause a severe failure of the communication system, to a unique single point of failure, i.e. the hub.

In this paper we discuss the characteristics of bus and star topologies in what concerns the ability to confine errors. We proposed the implementation of an active star topology that is compatible with off-the-shelf CAN controllers and that can be used with any CAN-based protocol (e.g. TTCAN [10], FTT-CAN [1], Timely CAN [2], MajorCAN [11], etc).

We described the architecture of our star central device, a hub, which can be built using off-the-shelf FPGA technology. Moreover, we discussed the fault diagnosis and passivation mechanisms of the hub and addressed the specific issue of link length, which, in CAN, is a particularly important topic due to the bit level synchronization and the resulting coupling between bit rate and link length.

We have shown that for a given bit rate a star may have a diameter generally similar to the length of a bus except for higher bit rates, in which case it is slightly lower. On the other hand, the star may cover a substantially larger area than the bus or, for the same area, to use a higher bit rate.

In general, the simplex star topology proposed in this paper is a further step towards improving both dependability and real-time performance of CAN networks.

References

[1] L. Almeida, P. Pedreiras, and J. A. Fonseca. The FTT-CAN Protocol: Why and How. *IEEE Transactions on Industrial Electronics, Vol. 49, No. 6*, December 2002.

[2] I. Broster and A. Burns. Timely use of the CAN Protocol in Critical Hard Real-time Systems with Faults. *Proceedings of the 13th Euromicro Conferencs and Real-time Systems. IEEE*, 2001.

[3] I. Broster and A. Burns. An Analysable Bus-Guardian for Event-Triggered Communication. In *Proceedings of the 24th Real-time Systems Symposium*, pages 410–419, Cancun, Mexico, Dec 2003. IEEE.

[4] G. Cena, L. Durante, and A. Valenzano. A new CAN-like field network based on a star topology. *Computer Standars & Interfaces*, March 2001.

[5] CiA. CAN data link layer. Technical report, CAN in Automation (CiA), Am Weichselgarten 26.

[6] CiA. CAN physical layer. Technical report, CAN in Automation (CiA), Am Weichselgarten 26.

[7] Infineon(technologies). CAN-Transceiver TLE 6250, 2002.

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

[9] H. Kopetz. Fault Containment and Error Detection in TTP/C and FlexRay. Research Report 23, Technical University of Vienna, August 2002.

[10] H. Kopetz. Time-Triggered Protocols for Safety-Critical Applications. March 2003.

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

[12] M. Rucks. Optical layer for CAN. *1st International CAN Conference*, November 1994.

[13] J. Rufino, P. Veríssimo, and G. Arroz. A Columbus' Egg Idea for CAN Media Reduncancy. *Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, June 1999.

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