 doi:10.1109/ETFA.2011.6059166

A Model for Quantifying the Reliability of Highly-Reliable Distributed Systems based on Fieldbus Replicated Buses

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

Abstract

Despite the efforts devoted to increase the dependability of highly-reliable distributed fieldbus systems by means of simplex stars and replicated stars/buses, literature lacks of appropriate analyses that quantify the system reliability these topologies yield. In previous work, we proposed models to adequately quantify the system reliability benefits of simplex buses and simplex/replicated stars. However, a model for replicated buses is an open issue that needs to be addressed, as they normally include less components than stars and, thus, can be more reliable and cost-effective. To fill this gap, this paper presents a model that makes it possible to appropriately quantify the reliability that a highly-reliable distributed system can achieve when using a replicated bus.

1. Introduction

During the last few years different network topologies have been proposed to improve the dependability of fieldbuses targeted to critical control applications. Special effort has been made to offer the possibility of using either replicated buses or simplex / replicated stars in fieldbus communications [9]. Certainly, simplex stars provide better error containment than simplex buses and, in addition, replicated topologies further provide tolerance to hardware faults. However, these topologies include more hardware components than simplex buses and, thus, they have a negative impact on dependability. In this sense note that surprisingly, despite the efforts made to enhance fieldbuses by means of these topologies, literature lacks of appropriate analyses that quantify the degree in which they actually improve system dependability in general and reliability in particular (as reported in [5]).

This has leaded us to devote our efforts to modeling and quantifying the system reliability achievable by fieldbuses that rely on different communication topologies. Specifically, in previous work [5, 4] we proposed models for quantifying the reliability of systems based on the Controller Area Network (CAN) [1] when they use a simplex bus, a simplex star and a replicated star. We are interested in CAN because it is one of the most mature, low-cost and widespread fieldbus technologies, and its application is also expected to grow in domains where criticality is an issue [8]. In this sense also note that the stars our models focus on are CANcentrate [5] and ReCANcentrate [4], since they are the CAN star topologies that provide the best error-containment and fault-tolerance so far.

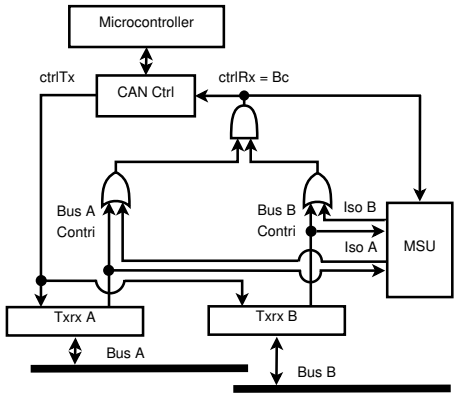

Figure 1. Node architecture proposed in [10]

The results obtained with these models [5, 4] showed that simplex and replicated stars can improve the system reliability when compared with a simplex bus. Moreover, these models are parameterized so that they allow to carry out sensitivity analyses for assessing how different dependability-related features of the system and the network topology influence reliability.

Despite these models are an important step towards elucidating the system reliability that can be achieved with topologies other than simplex buses, the literature still lacks of appropriate models for the other mentioned topology of interest, i.e. the replicated bus. In fact, to our best knowledge, the only work that quantifies the reliability of the replicated bus in the context of fieldbuses is [2]. However, as many other dependability models of communication networks, this work abstracts away details that can strongly influence dependability, e.g. the coverage of the fault-tolerance mechanisms.

Therefore, the objective of the present paper is to provide a model that allows to appropriately quantifying the reliability that can be achieved by means of a replicated bus topology in the context of highly-reliable distributed systems. In this sense, this model must be built up in such a way that its results can be compared with the ones that can be obtained by means of our previous models. Moreover, the model must be parameterized so that it can be used to carry out sensitivity analyses as well. Finally note that, as in our former models, we focus on permanent hardware faults. Software and transient faults are out

of scope, as they are usually addressed using techniques other than specific network topologies.

The rest of the document is organized as follows. First it explains the basics of the replicated bus architecture that has been chosen for the model. Second, it describes the modelling strategy, highlighting the similarities between it and the strategy followed in our former models, and focusing on the modelling issues that are specific to the replicated bus. Third, it presents the results of some sensitivity analyses that demonstrate the feasibility of the model. Finally, it draws the conclusions and points out future work.

2. Replicated bus basics

The first step to appropriately model the reliability benefits of a replicated bus is to specify a replicated-bus architecture that manages the redundant buses in the easiest way, i.e. by increasing as less as possible the hardware/software complexity when compared with a simplex bus, while benefiting as much as possible from the fault-tolerance potential of the bus redundancy.

Among the different replicated bus architectures that have been proposed for CAN so far [13], we believe that the most suitable is the CANEly's one [10]. Fig. 1 sketches the structure of a CANEly node, which basically includes a microcontroller, a CAN controller *CAN Ctrl*, two transceivers (*Txrx A* and *Txrx B*) and additional logic. CANEly manages the bus redundancy in a simple way by taking advantage of the dominant/recessive transmission and the in-bit response properties of CAN [1], i.e. a dominant bit 0 prevails over a recessive bit 1 and nodes quasi-simultaneously observe every single bit on the channel. Specifically, for transmitting, the node sends the same bit stream (*ctrlTx*) to all buses; whereas for receiving (*ctrlRx*), it simply couples bit by bit (in an AND gate) the streams it observes at each one of them (*Bus A Contri* and *Bus B Contri*). This makes it possible for each node to transparently communicate through the buses as if they were a single logical channel and, thus, eliminates the need of increasing even more the nodes hardware/software complexity to deal with the more complex management of several different channels.

To tolerate a permanently faulty bus, each node needs to isolate that bus and rule it out for communicating. In order to diagnose when a bus is faulty, each CANEly node includes a *Media Selection Unit* (MSU) that monitors the resultant coupled signal, B_c, and the contribution received from each bus. The MSU diagnoses a given bus as being permanently faulty when it detects that the contribution of the bus is either stuck-at (recessive or dominant) [10], or frequently disagrees with the contributions of the other buses and corrupts the frame observed at the coupled signal. When so, the MSU isolates the bus (masks its contribution) from the AND gate by driving a logical 1 (*Iso A* or *Iso B*) into the corresponding OR gate.

After this brief introduction about CANEly, it is even more easy to see why it is appropriate to model a replicated bus based on its architecture. The reason is that CANEly presents a lot of similarities with respect to the replicated star we have modeled, i.e. ReCANcentrate. First, both topologies use a similar AND-based strategy to couple different media and create a single communication domain. Second, both of them isolate faults by masking permanently faulty bus/port contributions by means of OR gates. Finally, they provide tolerance not only to faults affecting the cables and connectors, but also the transceivers each node uses to connect to the media.

3. Modelling rationale

In order to compare the replicated bus with the simplex bus and the simplex/replicated stars, we modelled the replicated bus using the same modelling strategy we proposed for CAN, CANcentrate and ReCANcentrate [5, 4]. However, we had to extend this strategy to cope with additional modelling difficulties posed by some bus replicated issues. Next, the most relevant aspects of the modelling strategy are summarized, highlighting the modelling problems that are specific to the replicated bus, as well as the way in which we have solved them.

3.1 Reliability metric

The reliability is one of the dependability attributes of main concern in critical systems, and it is defined as the probability with which a system continuously provides its service during a given interval of time [12].

To properly reflect the contribution of the underlying communication infrastructure to the reliability of a distributed control system, it is necessary to use a metric that takes into account both, the probability with which the nodes operate, and the probability with which they communicate among them.

Moreover, to quantify the reliability of a highly-reliable distributed system, the metric must also consider the system's ability to continue providing its service despite the loss or disconnection of nodes. For instance, a system constituted by several node replicas can correctly operate while tolerating the failure or the disconnection of up to k out of N nodes. Thus, in order to quantify in our models the reliability of those systems, which we refer to as *fault-tolerant-accepting* (FTA) systems, we introduced in [3] a metric called FTAR_k. This metric is defined in terms of k as the probability with which at least $N - k$ of the N nodes of a system can correctly operate and communicate among them throughout a given interval of time.

3.2 Modelling assumptions

The assumptions our models rely on are chosen to reach the major equanimity as possible between the different topologies; but paying special attention to not favor the stars in the comparison, as CANcentrate and ReCANcentrate were proposed in the context of our work. In any case, most assumptions are parameterized to make it possible to analyze the sensitivity with respect to them.

In our models the system is considered to be composed of the following componentes: microcontrollers,

CAN controllers, transceivers, memory ICs, oscillators, PCBs, segments of cable, connectors, network terminations, and ASICs (in the case of the hubs and the MSU). Each component is supposed to independently fail in a permanent manner, and faults are considered as not being near-coincident in time. The *Time To Failure* (TTF) distribution of each component is considered to be exponential and Non-Defective, with mean $1/\lambda$, where λ is the failure rate expressed in hour^{-1}. The failure rates are calculated using the MIL-HDBK-217F prediction standard [6].

Components can fail, *from the channel point of view*, by transmitting stuck-at-recessive (STR), stuck-at-dominant (STD) or flipping (FLIP) bits [5]. To not disfavor the replicated bus, we assume that the MSU can fail by either stopping its error-containment activities, or by permanently isolating all bus contributions (which leads the node to simply shutdown). Since there is not a real consensus on the components' failure mode proportions, we consider these proportions as equiprobable; excepting the microcontroller, which can only cause a stuck-at-recessive; and the CAN controller, which exhibits a stuck-at-recessive and a stuck-at-dominant/bit-flipping failure with a proportion of 66.6% and 16.6% respectively [3].

For the networks' layout, we assume the length of each star link to be half the total CAN bus length. This is pessimistic for the stars, since to cover the same area as a bus does it can be used a star with a diameter lower than the total bus length. In order to minimize the number of cables and connectors that attach the nodes to the bus line [5], it is supposed a daisy chain configuration for each bus replica.

The major part of the coverages we model refer to the probability with which the error-containment/fault-tolerance mechanisms of the communication subsystem detect and isolate faults included in our fault model, given that these faults occurs. Broadly speaking, the models include coverages that characterize the ability of the CAN controller to isolate faults happening at the media, itself or the rest of its node; the ability of a hub to isolate faults at its ports; the capacity of a node of ReCANcentrate for communicating using only one star when a fault prevents it from communicating through the other one; and the capacity of an MSU for isolating a faulty bus replica. Basically, we assume that these mechanisms present a coverage of 100% and 95% for stuck-at-recessive and stuck-at-dominant/bit-flipping faults respectively [5].

3.3 Model structure

We built the models using the Stochastic Activity Network (SAN) formalism, which is an extension to Stochastic Petri Nets [11]. Since components' TTFs are exponentially distributed, all our models are transformed into a set of *Continuous Time Markov Chains* (CTMCs) that are analytically solved.

As for the other models, we modelled the replicated bus as a hierarchical composition of different SANs that share specific places by means of the *Join* SAN's primitive [11] (see Fig. 2). We classify the set of SANs of a given model

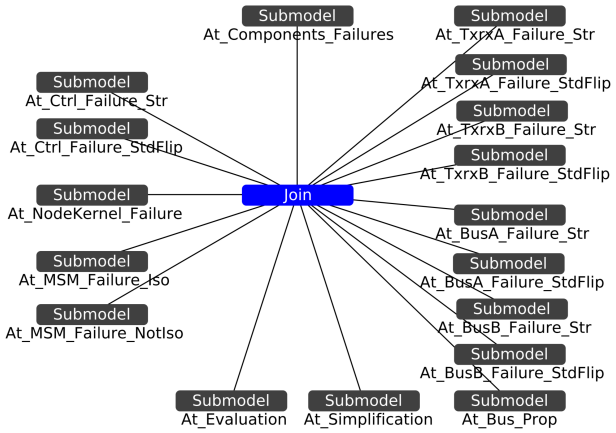

Figure 2. Replicated bus model

into three groups depending on the aspect they represent, namely (1) the occurrence of faults; (2) the way in which errors are propagated/contained and faults are treated by means of the fault-tolerance mechanisms, i.e. the *Coverage Process* [4]; and (3) the evaluation of whether or not the system still delivers its service once the coverage process finishes.

3.3.1 Fault occurrences

As concerns the occurrence of faults, note that the granularity with which a system isolates faulty components to prevent the propagation of errors is limited. Normally when a component fails, the errors it generates are contained by isolating a region of the system that does not only include that component, but also other ones. This means that it is not necessary to model the failure of each individual component, but of the region that is isolated when that component fails. Therefore, in order to reduce the state space of the model, we do not represent the failure of single components but of each one of these regions, which we call *Error-Containment Regions* (ECRs).

Although all our models consider the same kind of components, they are split in different ECRs. This is because each network topology has different error-containment mechanisms and, hence, components are group into different types of ECRs. In ReCANcentrate the types of ECRs and the most relevant components included in each one of them are [4]: (1) the *Node Kernel*, which includes the microcontroller; (2) the *Node Connection*, which comprises all entities a node needs to connected to a given hub, i.e. one CAN Controller, transceivers, cables, etc.; (3) the *Hub Interconnection*, which has all components used to interconnect both hubs by a given sublink; and (4) the *Hub Kernel*, which includes the ASICs that implement the coupling and error-containment mechanisms of the hub.

The replicated bus is divided into similar types of ECRs. However, the error-containment mechanisms of the replicated bus are not centralized at the hubs, but placed in each one of the nodes (see Fig. 1). Thus, the main characteristic of the ECRs of the replicated bus is

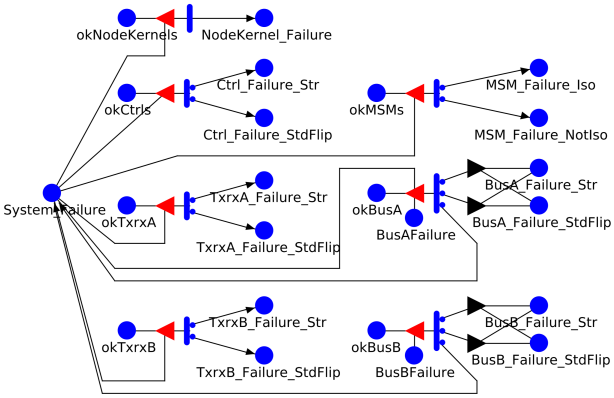

Figure 3. *At_ECR_failure* **submodel**

that almost all of them are located within each node. More specifically, each node includes a *Node Kernel* (as in ReCANcentrate); a *Controller* (Ctrl), which represents the CAN controller; a *Media Selection Module* (MSM), which basically embraces the MSU and the AND/OR gates the node uses for coupling the media; and two *Txrx*, each of which roughly corresponds to a given transceiver. It is also noteworthy that the replicated bus does not include *Node Connections*, *Hub Interconnections* nor *Hub Kernels*. Instead, it includes two *Bus* ECRs, so that each one of them includes the cables and connectors of a given bus replica.

As just said, since it is unnecessary to represent the failure of each individual component, we reduced the state space of the models by modelling the failure of just the ECRs. Moreover, the models do not even represent each individual ECR either. Instead, the models include one SAN for each type of ECR (see for instance the model of ReCANcentrate [4]). In other words, the models include a set of SANs, each of which represents all the ECRs of a given type, e.g. all the *Node Kernels*. As concerns the internal structure of these SANs, each one of them basically includes a place whose marking represents the number of surviving ECRs of a given type, a timed activity that models when any surviving ECR of that type fails, and a set of instantaneous activities that decide what is the failure mode the new faulty ECR exhibits [4].

We followed the same strategy for modelling fault occurrences in the replicated bus. The only difference is that, in order to make the replicated bus model more compact, these SANs are implemented together into a single SAN called *At_ECR_failure* (Fig. 3). As can be seen, there is one place for almost each type of ECR, namely *okNodeKernels*, *okCtrls*, *okMSMs*, *okTxrxA* and *okTxrxB*. Note that the two last places correspond to the transceivers connected to the bus replicas called *Bus A* and *Bus B* respectively (see Fig. 1). Also, just for convenience, there is a place dedicated to each one of the bus replicas, *okBusA* and *okBusB*. The marking of each one of these two last places indicates whether or not the corresponding bus replica is faulty.

The timed activity to which each one of the places mentioned in the former paragraph is connected models both, the time that elapses until any ECR of the corresponding type fails, and the type of errors the fault generates. For instance, the timed activity of *okCtrls* models the time that elapses until any surviving *Controller* fails and, then, it decides what is the type of errors the faulty *Controller* generates. Note that in order to model the time that elapses until a new ECR of a given type fails, the corresponding activity takes into account both the TTF of the components that constitute that kind of ECR and the number of these ECRs that have not failed so far. Similarly, it calculates the proportion with which a faulty ECR exhibits different failure modes by considering the failure mode proportions of the ECR constituent components. The details of how this is done are explained in [4].

Finally, note that when one of these activities fires and decides what is the kind of errors the faulty ECR generates, it sets a token in a place that represent that failure mode. For example, when a *Controller* fails the corresponding activity sets a token in either *Ctrl_Failure_Str* or *Ctrl_Failure_StdFlip*, indicating that the *Controller* that has failed exhibits a stuck-at-recessive or a stuck-at-dominant/bit-flipping respectively. Once a token is set in any of the places of *At_ECR_failure* that represents a failure mode, a series of SANs are activated in order to model how the errors are propagated/contained and the faults are treated, i.e. as said before, what we call the *Coverage Process* (CP). The behavior of these SANs is explained later on in Section 3.3.3.

3.3.2 Node's operational states

As just said, when a new fault occurs, a series of SANs are activated to model the *Coverage Process* (CP). In principle, to correctly carry out that process it is necessary to know what is the state of each ECR and its location with respect to the other ones. Just as an example, let us imagine a *Txrx* ECR that fails in a bit-flipping manner. If the bus replica (the *Bus* ECR) it is connected to is already stuck-at-recessive, then the flipping bits transmitted by the *Txrx* cannot propagate through that replica. Otherwise, if that bus replica is not faulty, then the flipping bits will propagate through it and it is necessary to further analyze whether or not each node connected to the bus replica contains them. Moreover, in order to correctly analyze how the flipping bits propagate from the *Txrx* to the internals of its own node, the first step would be to determine if the *MSM* of that node is already faulty. If so, it would be necessary to determine whether that *MSM* has failed by isolating the *Txrx* or by stopping its error-containment actions. If the *MSM* is not faulty, then it will be needed to evaluate if it succeeds in isolating the *Txrx*. In any case, if the *MSM* does not isolate the errors, then it will be evaluated whether or not the *Controller* (Ctrl) does it, and so on and so forth.

Nevertheless, at this point, it is worth to recall that in order to reduce the state space we model the number of surviving ECRs, but not the state of each individual ECR. Thus, it is necessary to include within the model

additional data related to the state and the location of the ECRs. This data should be as less as possible, but it must suffice the CP.

For instance, for ReCANcentrate we found out that it is enough to include information about two aspects: (1) whether or not each hub is faulty, and (2) whether or not each node can operate and communicate through each one of the hubs, i.e. what is the *Node Operational State* (NOS) of each node [4]. Since the model of ReCANcentrate already has places that keep the state of each hub, it does not include extra places that model the first aspect. Conversely, it does need to include extra places to keep what is the NOS of each node. In this sense it is worth noting that again, in order to reduce the state space, we do not add places to model the NOS of each individual node, but a set of places such that the marking of each one of them represents the number of nodes being in a given NOS.

[4] thoroughly explains why knowing the state of the hubs and the number of nodes in each NOS is enough for carrying out the CP in ReCANcentrate. Basically, when an ECR fails it is decided what is the NOS of the node that is firstly affected by the ECR failure. The probability of choosing a given NOS is calculated by dividing the favorable possibilities by the total number of them. The same strategy is used to decide what is the NOS of each one of the nodes that is subsequently affected by the propagation of errors as the CP progresses.

Likewise, for carrying out the CP in the replicated bus, it is also necessary to know whether each bus replica is faulty or not, as well as how many nodes are in a given NOS. Moreover, as in ReCANcentrate, it is only necessary to add extra places for representing the number of nodes in each possible NOS. Nevertheless, the definition of the NOS is much more complex in the replicated bus. Certainly, like in ReCANcentrate, each NOS must be defined in such a way that it reflects whether or not the node operates and can still transmit and/or receive through each one of the bus replicas. However, the fault-tolerance mechanisms of the replicated bus are not centralized at any hub, but located in each one of the nodes. Therefore, in the replicated bus each NOS must be defined so that it additionally reflects what are the error-containment capabilities of the node depending on the state of the nodes's internal ECRs. The example outlined at the beginning of this section illustrates this necessity.

An important consequence of this need is that, conversely to ReCANcentrate, the amount of possible NOSs in the replicated bus is huge. This poses two major difficulties. First, it is needed to exhaustively identify what are all the possible NOSs. For that purpose, we generated a tree that explores all the combinations of the possible states of the ECRs of a node. We obtained a number of NOSs of the order of some hundreds. Second, once all NOSs are identified, it is necessary to analyze if it is possible to reduce their number in order to prevent the explosion of the state space. Fortunately, we found out that many NOSs are equivalent from the point of view of the node's capacity for both communicating and containing errors. Thus, after exhaustively examining the tree of NOSs we reduced their number down to 53.

3.3.3 Coverage process

The major part of the SANs of our models are devoted to represent the *Coverage Process* (CP). They are located at the left and right sides of Fig. 2. Each one of these SANs (CP SANs) models the result of the actions carried out by one or more error-containment or fault-tolerance mechanism of the system, e.g. the ability of each MSU to contain a faulty bus, the capacity of a CAN controller to do so when the MSU does not succeed, etc.

Fig. 4 sketches the steps that compose the CP. The CP starts once a fault occurs in an ECR and the corresponding SAN sets a token in one of the places that indicates the failure mode of the ECR. As already explained in Section 3.3.1, in the replicated bus this is done by the SAN *At_ECR_failure* (Fig. 3). When this occurs, it is activated the CP SAN that corresponds to both the type of ECR that fails and its failure mode, e.g. *At_Ctrl_Failure_StdFlip* (left side of Fig. 2) is activated if the ECR that fails is a *Controller* (Ctrl) and it exhibits a stuck-at-dominant/bit-flipping failure.

As specified in the first square of Fig. 4, the CP SAN that becomes active determines what are the NOSs affected by the fault; or in other words, what are the nodes that, due to their NOS, are affected by the fault. Specifically, if the fault happens in any of the ECRs that are defined within the node, then only one node is directly affected by the errors the fault generates and, thus, the CP SAN only needs to determine what is the NOS of that node. Conversely, if the fault happens in any of the *Bus* ECRs, then multiple nodes will be affected by the errors the bus propagates. Thus, the corresponding CP SAN, e.g. *At_BusA_Failure_BitFlip*, has to determine what is the NOS of each one of these nodes.

In any case, for choosing a given NOS the CP SAN follows the strategy sketched in Section 3.3.2, in which the NOS is selected with a probability that is calculated by dividing the favorable possibilities by the total number of them. For instance, if the ECR that fails is located within a node, then the NOSs that are candidates to be chosen by the CP SAN are those in which the ECR is not faulty. Let $\text{NOS}_{\text{canditates}}$ be the set of candidate NOSs, and let $\text{NOS}_{\text{canditates\_i}}$ be one of these NOSs. Then, the probability with which the CP SAN selects $\text{NOS}_{\text{canditates\_i}}$ as the NOS of the node where the ECR has failed is calculated just by dividing the number of nodes in $\text{NOS}_{\text{canditates\_i}}$ by the number of total nodes in $\text{NOS}_{\text{canditates}}$.

Once the CP SAN elucidates what is/are the affected NOS/s and how many nodes are in that NOS/s, it evaluates if each one of these nodes isolates the errors. The decision of whether or not a node contains the errors depends on its NOS, i.e. it is based on the coverage of the error-containment mechanisms that a node being in that NOS

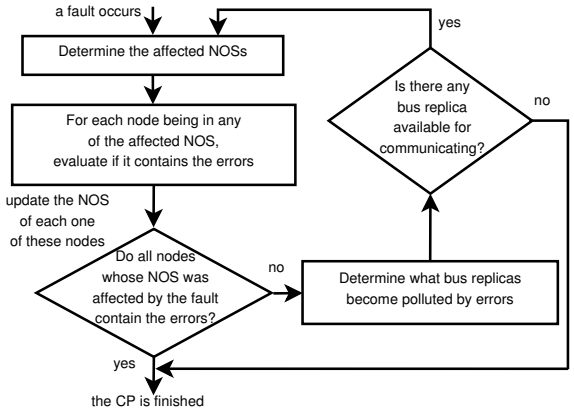

Figure 4. Coverage Process (CP)

can use. For instance, a node being in a NOS in which the MSM has failed by stopping its error-containment activities will not be able to isolate the errors generated by one of its transceivers, whereas a node in a NOS where the MSM is non-faulty will isolate the errors with a given probability of success. Note that as a result of this evaluation the CP changes the marking of the NOSs' places accordingly. The next step of the CP after this evaluation is to check if the errors have been contained. If so, the CP finishes. Otherwise, if any node does not contain the errors, then it will pollute the bus replicas it can transmit to. The CP finishes if both bus replicas were non-faulty and due to this situation both are polluted with errors. Similarly, the CP finishes if the errors affect the only bus replica that was non-faulty. Finally, if both bus replicas were non-faulty and the errors only pollute one of them, it is necessary to evaluate if the errors that propagate through that replica are contained by each one of the nodes connected to it. For doing so, the CP is reinitialized as if that bus replica would have failed (see Fig. 4).

It is important to note that the CP process of the replicated bus is more complicated than the one of ReCANcentrate [4]. Each node of the bus is a potential point from which errors can propagate to any of the bus replicas and, then, to the other nodes. Hence, the number of ways in which errors can propagate through the system is greater in the replicated bus. Moreover, since each node has more error-containment mechanisms (and hence the number of NOS is greater) in the replicated bus, it is also more difficult to analyze what are the nodes that are affected by the errors, and what are their NOSs.

3.3.4 System evaluation

Every time the CP finishes, it is necessary to determine whether or not the whole system is still operational or fails. Normally, this decision is taken by a SAN submodel we refer to as the *System Evaluator Submodel* (SES). In the case of the replicated bus this SAN is called *At_Evaluation*, see the bottom of Fig. 2.

The SES analyzes the marking of the places that represent the number of nodes being in the different NOSs; calculates how many nodes can still operate and communicate among them; and then elucidates if the system is operational. The number of nodes that must operate and communicate among them in order to consider that the system is operational is parameterizable in all our models, making it possible to measure different degrees of reliability, i.e. FTAR_k for different values of k.

Additionally, in order to reduce the computation time of the model, some CP SANs are provided with the capacity of determining when, due to a certain condition, the system is faulty. For instance, as said before, *At_Ctrl_Failure_StdFlip* is activated when a *Controller* ECR fails in a stuck-at-dominant or bit-flipping manner. If this SAN determines that the NOS of the node that is affected by the fault is such that the node was communicating through both bus replicas, then it directly diagnoses that the system fails as both buses will be polluted with the dominant/flipping bits.

In any case, when the SES or any CP SAN detects that the system is faulty, it writes a token at a place called *System_Failure*. This place is shared among all submodels, so that all them stop evolving when observe that token. This allows reducing the size of the state space of the underlying CTMC. Moreover, in order to reduce the state space further, the replicated bus model includes a SAN called *At_Simplification* (bottom of Fig. 2). When a token is set at *System_Failure* this SAN immediately clears the marking of all the places of the whole model (except of *System_Failure*), thereby ensuring that the CTMC represents the failure of the system by means of just one state.

Finally, the presence of the token in *System_Failure* is used to easily quantify the reliability. Specifically, we define a *rate reward variable* [11] as $1.0 - \textit{System\_Failure} \rightarrow \textit{Mark()}$, so that its value at a particular point in time is the system reliability itself.

4. Reliability analyses

Next we describe some results that show the capacity of the model proposed in this paper for characterizing the reliability benefits that a replicated bus topology can bring to a highly-reliable distributed system. Since the results of any dependability model cannot be used per se, we compare the system reliability obtained when using the replicated bus with the one that would be achieved when using a simplex CAN bus and a ReCANcentrate star. Moreover, in order to show the potential of the model to carry out sensitivity analyses, the reliability of the replicated bus and these two other topologies is analyzed with respect to two dependability-related features. Each one of these analyses takes as a starting point a case of reference [3], in which all models' parameters were set to favor buses over stars, and then varies the value of the parameters that characterize the dependability-related feature under study. Table 1 shows some values of the case of reference.

We assume that faults affecting the hardware of a node replica can lead it to exhibit an *Authentification detectable (non-malicious) Byzantine failure*. Given this failure semantic and in order to consider a cost-effective highly-

Table 1. Some models parameters

Parameter	Default value	Meaning
stdFlipCov	0.95	STD/FLIP error-containment coverage of the CAN controller, MSU and hub
connectFR	$2.07774 \cdot 10^{-8}$	Connector failure rate
wireFR	10^{-7}	Wire failure rate per kilometer
txrxFR	$6.73258 \cdot 10^{-7}$	Transceiver failure rate
ctrlFR	$1.25537 \cdot 10^{-6}$	CAN controller failure rate
microFR	$3.25312 \cdot 10^{-6}$	Node's microcontroller failure rate
hubElecFR	$1.275596 \cdot 10^{-6}$	Hub electronics failure rate for 3 nodes

reliable system, we assume a system of 3 node replicas, which is the minimum number of replicas needed to tolerate the failure of one of them (no matter which one).

Finally, note that since the system is considered to tolerate the failure/disconnection of up to 1 node, the metric used to measure the reliability is the FTAR_1. Moreover, in order to make results as visually clear as possible, we do not plot the evolution of the FTAR_1 in time, but how these values affect the *Mission Time* (MT). The MT is the maximum amount of time during which a system exhibits a reliability equal to or greater than a certain threshold [7]. In particular, the analyses presented here use a reliability (FTAR_1) threshold of 0.99999. Note that since the MT has a direct relationship with the achievable system reliability, for the sake of clarity results will be discussed in terms of the MT only.

4.1 Reliability vs cabling failure rate

As said in Section 1, replicated topologies include extra hardware when compared with simplex ones. Thus, one important aspect that must be analyzed is if the better fault-tolerance capabilities of a replicated bus and a replicated star compensate their extra hardware in terms of reliability. This is specially, important when comparing the replicated star with the simplex and replicated bus, as the replicated star is the one that provides the better error-containment and fault-tolerance, but also the most quantity of hardware, e.g. wires, connectors, transceivers, CAN controllers and the hubs.

Particularly, this section compares the sensitivity of the three mentioned topologies with respect to the failure rate of the wires and connectors, i.e. the *cabling*. In this sense, to keep the relative weight of wires and connectors as in the case of reference, the failure rate of the connectors is always considered one order of magnitude lower than that of the wires. Specifically, note from Table 1 that in the case of reference the failure rates are $2.07774 \cdot 10^{-8}$ hour^{-1} per connector and 10^{-7} hour^{-1} per km of wire.

For the sake of clarity, the x-axis legend of Fig. 5 refers to the order of magnitude of the wire failure rate only. The figure considers these failure rates ranging from 10^{-1} to 0 hour^{-1}, so that they are arranged in descendent order in the x-axis. The failure rate of 0 hour^{-1} corresponds to the ideal case in which the cabling cannot fail.

Fig. 5 shows that the replicated CAN bus and ReCANcentrate lead to a higher MT than CAN for almost any

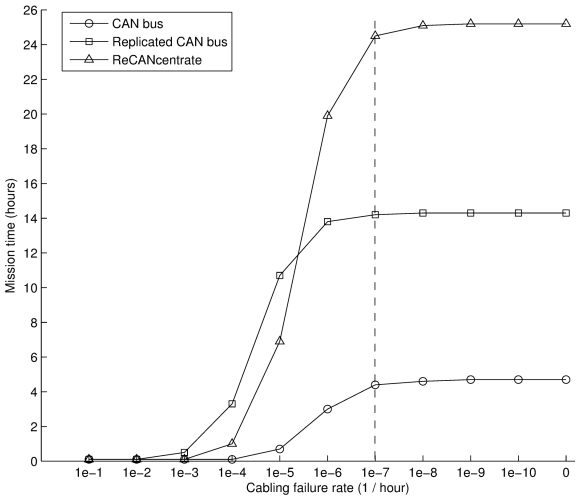

Figure 5. MT vs Cabling failure rate

failure rate of the cabling, which demonstrates that the better error-containment and fault-tolerance of both replicated topologies largely compensate their extra cabling. In fact, only when the quality of the cabling becomes unrealistically pessimistic, the system reliability that is achieved with the replicated topologies is similar to the one that can be achieved with a simplex bus.

Another important result is that the replicated star is much better than the replicated bus when the cabling failure rate is equal or greater than 10^{-6}, which is one order of magnitude higher (lower quality) than in the case of reference. However, the replicated bus shows to be more resilient than the replicated star to the decrease of the cabling quality. In fact, the replicated bus becomes better than the star as the cabling failure rate comes closer to a too high (non-realistic) failure rate of 10^{-5} or 10^{-3} hour^{-1}.

Finally, results show that, in practice, it is not worth to improve the reliability of the cabling in any of the topologies beyond the case of reference, as the system MT cannot be improved much further when doing so.

4.2 Reliability vs transceiver failure rate

Next we analyze the influence on the reliability of the transceiver, as its number is larger in the replicated topologies; specially in ReCANcentrate, which doubles and quadruplicates the number of transceivers of the replicated and the simplex CAN bus respectively [3]. Also note that conversely to wires and connectors, a transceiver is an electronic component, thus it is much more easy to improve its reliability by investing in its quality.

Fig. 6 depicts how the transceiver reliability affects the system MT achieved with each one of the topologies. Specifically, the x-axis shows the order of magnitude of the transceiver failure rate, whose value in the case of reference is of $6.73258 \cdot 10^{-7}$ hour^{-1} (see Table 1).

Results are very similar to the ones obtained for the cabling, yet there are substantial differences. First, the system MT can be boosted by investing in the quality of the transceivers when using ReCANcentate or the CAN

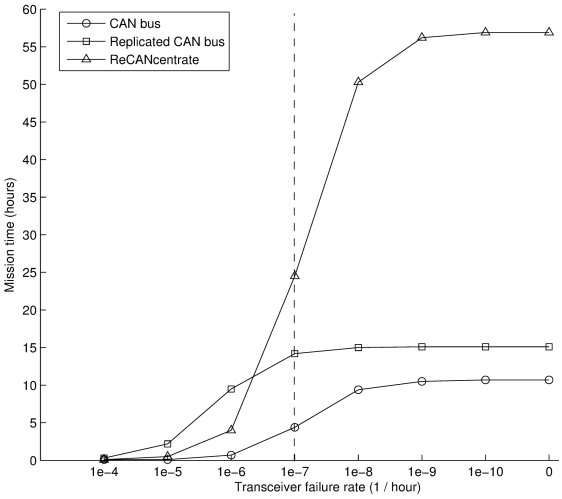

Figure 6. MT vs Transceiver failure rate

bus, but not when the underlying topology is the replicated CAN bus. In fact, the MT benefits of the replicated bus when compared with the simplex one become less evident when the transceiver quality is improved beyond the case of reference. The other important difference with respect to the former analysis is that if we compare the robustness of the topologies to both the cabling and the transceiver, the replicated bus becomes even more robust than the others in the second case. For instance, note that it is enough that the order of magnitude of the transceiver failure rate increases in less than one unit, i.e. from 10^{-7} to near 10^{-6} hour^{-1}, to make the replicated bus much better than the star in relative terms; in the former analysis the replicated bus becomes better than the star only if the order of the cabling failure rate decreases in more than one unit.

5. Conclusions and future work

Although replicated buses have been traditionally used to improve reliability of fieldbus-based systems, no study appropriately quantifies the reliability this topology can actually yield. To fill this gap, this paper proposes a model that makes it possible to adequately quantify the reliability achievable by highly-reliable CAN-based systems that rely on a replicated bus topology.

The paper explains how to model such systems with SANs following the same strategy we proposed to model field-bus systems that rely on simplex buses and simplex/replicated stars. We reveal that, surprisingly, to model a replicated bus poses additional difficulties. These are due to the fact that, unlike in stars, the major part of the replicated bus error-containment mechanisms are implemented in each one of the nodes.

The paper uses the model proposed here, and the ones we proposed formerly, to carry out some sensitivity analyses that compare the reliability attainable by a replicated CAN bus with the one what would be achieved by a simplex CAN bus and a replicated CAN star. Results demonstrate the suitability of the model to study the reliability benefits of replicated bus topologies.

We plan to use the models we have proposed so far to carry out more sensitivity analyses that bring light into which is the most suitable topology for improving reliability in fieldbuses. In this sense, note that the models we proposed so far can be adapted to consider fieldbus technologies other than CAN. Also, this work is currently being extended to tackle the issue of temporary faults, as well as the negative impact that external events such as collisions have on the benefits of the different topologies.

Acknowledgements

This work was supported by project DPI2011-22992 (Spanish *Ministerio de economía y competividad*) and by FEDER funding.

References

[1] ISO11898-1. Controller Area Network (CAN) - part 1: Data link layer and physical signalling., 2003.

[2] D. Aza-vallina, B. Denis, and J.-m. Faure. Communications reliability analysis in networked embedded systems. In *European Conference on Safety and Reliability - ESREL 2011*, 2011.

[3] M. Barranco. *Improving Error Containment and Reliability of Communication Subsystems Based on Controller Area Network (CAN) by Means of Adequate Star Topologies*. PhD thesis, DMI, Universitat de les Illes Balears (UIB), 2010.

[4] M. Barranco, J. Proenza, and L. Almeida. Reliability Improvement Achievable in CAN-based Systems by Means of the ReCANcentrate Replicated Star Topology. In *8th IEEE International Workshop on Factory Communication Systems, Nancy, France*.

[5] M. Barranco, J. Proenza, and L. Almeida. Quantitative Comparison of the Error-Containment Capabilities of a Bus and a Star Topology in CAN Networks. *IEEE Transactions on Industrial Electronics*, 58(3):802–813, Mar. 2011.

[6] DOD. MIL-HDK-217f-2 Military Handbook, Reliability Prediction Of Electronic Equipment, 1995.

[7] J. Morris and P. Koopman. Representing design tradeoffs in safety-critical systems. In *Proc. WADS, St. Louis, MO.*, pages 1–5, 2005.

[8] J. Munoz-Castaner, R. Asorey-Cacheda, F. Gil-Castineira, F. Gonzalez-Castano, and P. Rodriguez-Hernandez. A Review of Aeronautical Electronics and Its Parallelism With Automotive Electronics. *IEEE Trans. on Industrial Electronics*, 58(7):3090–3100, 2011.

[9] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert. Trends in Automotive Communication Systems. *Proceedings of the IEEE*, 93(6), 2005.

[10] J. Rufino, P. Verissimo, and G. Arroz. A Columbus' egg idea for CAN media redundancy. *Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, pages 286–293, 1999.

[11] W. H. Sanders and J. F. Meyer. Stochastic Activity Networks: Formal Definitions and Concepts. *Lecture Notes in Computer Science*, 2090:315–343, 2001.

[12] M. Shooman. *Reliability of Computer Systems and Networks*. 605 Third Avenue, New York, USA, 2002.

[13] V. Silva. *Flexible Redundancy and Bandwidth Management in Fieldbuses*. PhD thesis, 2010.