Quantitative comparison of the error-containment capabilities of a bus and a star topology in CAN networks

Manuel Barranco, Julián Proenza, *Member, IEEE,* and Luís Almeida, *Member, IEEE,*

Abstract— **There has been an increasing interest in using star topologies in field bus communications, e.g. in TTP/C, FlexRay or CAN, due to increased fault resilience and potential error-containment advantages. In this context, an innovative CAN-compliant star topology has been developed, CANcentrate, whose hub includes enhanced fault-treatment mechanisms. However, despite this interest towards stars, it is still necessary to quantify their real dependability benefits. For this purpose and for the particular case of CAN, this paper presents models for the dependability features of CAN and CANcentrate using Stochastic Activity Networks (SANs). It quantitatively compares their reliability and error-containment capabilities under permanent hardware faults. Models rely on assumptions that ensure that results are not biased toward CANcentrate, which in some cases is too detrimental for it. Thus, despite not reflecting the full CANcentrate potential, results quantitatively confirm the improvement of error-containment it achieves over CAN. Additionally, the way in which the nodes' ability to contain their own errors affects the relevance of using a star topology has been quantified. Although this paper refers to the case of CAN, conclusions regarding the justification of using a star depending on this ability can be extrapolated to other field-bus technologies.**

Index Terms— **Reliability modeling, Stochastic Activity Networks, fault tolerance, error containment, system analysis and design, communication system reliability, communication system fault diagnosis, field buses, CAN protocol, star, topology, hub.**

I. Introduction

Bus topologies have been extensively used in distributed control systems due to their electrical robustness and low cost. However, the suitability of this topology for dependable control systems has been controversial due to some limitations [1]. A number of them arise from the fact that a bus topology includes limited error-containment mechanisms [2] [3]. In fact, a bus topology presents multiple points where a single fault in any of them can prevent more than one node from communicating to any other node of the system, possibly causing a general failure. These are called *points of severe failure* [4], which include the commonly referred single points of failure.

M. Barranco and J. Proenza are with the Systems, Robotics and Vision group (SRV), Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears (UIB), 07122 Palma de Mallorca, Spain (e-mail: manuel.barranco@uib.es; julian.proenza@uib.es)

L. Almeida is with the Departamento de Engenharia Electrotécnica e de Computadores (DEEC), Faculdade de Engenharia da Universidade do Porto (FEUP), 4200-465 Porto, Portugal, and also with the IEETA, Universidade de Aveiro, 3810-193 Aveiro, Portugal (e-mail: lda@fe.up.pt)

Copyright (c) 2009 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

Solutions based on redundant or reconfigurable transmission media, as well as on bus guardians have been proposed to cope with this problem. Nevertheless, due to the characteristics that are inherent to the bus topology, these techniques do not prevent the existence of multiple points of severe failure [4].

Therefore, the interest in using star topologies has been growing, given their better error containment capabilities and their resilience to proximity and common-mode failures [5] [6] [7] [8] [9] [10]. For example, the LAN domain has long ago moved to star topologies with Ethernet, a technology that is now extensively used in the industrial automation and large embedded systems domains. Particularly concerning in-vehicle systems, we can find other examples of transition to star topologies such as with newer technologies like TTP/C [2] and FlexRay [11].

Different star topologies have also been proposed for improving the dependability of Controller Area Network (CAN) [12], e.g. [13] [14] [5]. These can have a significant impact since CAN is one of the most widely used field-bus networks in distributed embedded control systems. Even more significant if it is considered that there is a growing interest on using CAN for new systems that require a higher performance and dependability, as it is reflected in the plenty of work that has been done to improve some of CAN's features, e.g. [15] [16] [17] [18] [19] [20] [21], to integrate it with other communication technologies, e.g. [22], as well as to develop tools for fast verification of CAN network prototypes, e.g. [23].

Among the star topologies that have been developed for CAN, there is a recent proposal for a simplex star topology, called CANcentrate, which incorporates an active hub that prevents errors from propagating across the star. CANcentrate's current design and implementation [4] only deal with hardware faults that manifest as syntactically incorrect logical values in the communication channel. However, its hub could be provided with a bus guardian to deal with faults that manifest at a semantic level, e.g. babbling-idiot. As far as is known, CANcentrate is the simplex star topology that provides the highest degree of fault-treatment for CAN networks.

Despite this growing trend towards the use of star topologies, it is not so clear whether stars achieve a higher degree of dependability than buses. In fact, stars can provide better error-containment but they also include more hardware components, thereby increasing the probability that faults and errors occur. Furthermore, the enhancement of dependability that can be achieved when using a star instead of a bus

©2011 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. doi:10.1109/TIE.2009.2036642

has never been appropriately quantified. On the one hand, previous quantitative analyses that rely on mathematical or statistical models [24] [25] [26] [27] abstract away many important details: the capacity of the hub for containing errors; differences between the failure rates of the nodes of a star and a bus; different component failure modes; the error-detection coverage the hub provides for different failure modes, etc. On the other hand, fault injection experiments carried out in some technologies, e.g. in TTP/C [3], quantitatively demonstrate that a star is better suited to prevent error propagation than a bus. However, no study has been conducted so far to assess whether this actually implies a dependability improvement.

The lack of an appropriate comparison poses doubts about the practicality of using stars for highly dependable embedded systems. Thus, the main aim of the framework here discussed is to mathematically assess the dependability improvement that stars can achieve in the context of field-bus communications in general, taking into account technological aspects of electronic and mechanical components, which have not been considered in the literature yet.

Particularly, this paper quantifies the improvement of dependability achieved when using CANcentrate instead of CAN as a general field-bus network. The dependability features of a typical CAN bus and a CANcentrate network have been modeled using *StochasticActivity Networks* (SANs). Then, the reliability, i.e. the probability that no node is permanently prevented from communicating, as well as the *probability of not suffering a k-severe failure* (PNS) of CAN and CANcentrate were compared. A *k-severe failure* occurs when the communication is restricted to few than $N - k$ nodes. Particularly, the PNS was assessed when $k = 1$ only. By comparing their PNS when $k = 1$, the potential benefits of CANcentrate for systems that can tolerate (or accept) that at most one node cannot communicate, have been studied. Note that the concept of *k-severe failure* is equivalent to the one of *k-terminal reliability*, which is a well-known metric widely used in dependability evaluations of network topologies.

The present framework's second objective is to quantify how some design and implementation aspects, e.g. the type of cabling, affect the dependability of bus and star topologies. The models proposed in this paper are parametrizable, thereby allowing sensitivity analyses of some of these aspects to be carried out. On the one hand, this can guide engineers in the process of developing an adequate bus or star-based network with a view to achieving a specific level of dependability. On the other hand, this allows the use of this paper's proposed models to evaluate whether a given star or bus communication network is more suited for the dependability requirements of a specific distributed embedded application.

Notice that since star topologies were mainly proposed to address the issue of permanent hardware faults, this paper focuses on this kind of faults. Although there is an increasing interest on providing mechanisms to deal with transient faults, an evaluation taking them into account is necessarily application dependant. Therefore, transient faults are beyond the scope of this paper and will be addressed in future work.

This paper's intention is not to provide a definitive assessment in the presence of permanent faults. Firstly, it is hardly possible to find exact values for fundamental dependability parameters such as, for example, the failure rate of the components and the coverage of some error-containment mechanisms. Moreover, there are many ways of implementing a CAN bus and a CANcentrate star, each yielding different dependability results. In order to avoid biased results towards CANcentrate, choices for any of these aspects were always taken in favor of CAN. In some cases, this may have been too detrimental for CANcentrate. Secondly, this paper assumes that semantically incorrect frames issued by babbling-idiot nodes are not contained. However this type of failure could be treated by simply including a bus guardian within the hub, whereas in the CAN bus it would be necessary to include extra hardware in each node. Thus, the potential benefits that CANcentrate could yield when dealing with these failures are not reflected in the results. Furthermore, the failure rates of nodes do not take into account software failures, thereby favoring the CAN bus in the comparison, since the hub cannot exhibit them. Thirdly, other CANcentrate advantages that are inherent to its star topology are not included in the model, e.g. minimization of the impact of spatial proximity faults, avoidance of common-mode failures between a possible hub guardian and the nodes it would supervise. Therefore, the results presented here are just an indication of how CANcentrate improves the dependability of distributed control systems; but they do not reflect the full potential of CANcentrate.

As a reference for the comparison, typical values of in-vehicle communications for x-by-wire systems [28] were chosen for different parameters, e.g. for failure rates, despite the fact that this paper targets general field-bus communications.

The following section describes CANcentrate fundamentals. Section III introduces the modelling strategy, explaining the assumptions the proposed models rely on. Next, Sections IV and V, describe how the dependability of CAN and CANcentrate was modeled, focusing on the most representative aspects of the models. Then, Section VI shows the dependability results obtained when solving the models. Finally, Section VII concludes the paper and points out future work.

II. CANcentrate basics

An important feature of CAN is the wired-AND function of every node contribution when accessing the bus, thereby providing a dominant/recessive transmission. This property guarantees that whenever a node transmits a dominant bit value, i.e. a logical '0', this value is received by all the nodes in the network. In contrast, a recessive bit value, i.e. a logical '1', is only received as long as every node issues a recessive bit. Moreover, CAN communication relies on a complex bit synchronization mechanism that guarantees that nodes have a quasi-simultaneous view of every single bit on the medium. This bit synchronization allows defining a number of additional mechanisms [12] that significantly improve the dependability properties and real-time response of CAN [4]

However, as explained before, CAN relies on a bus topology and presents multiple *points of severe failure*. This paper's CANcentrate star topology is devoted to preventing severe communication failures. The fault model herein studied [4] takes into account that a node can exhibit a stuck-at-recessive, a stuck-at-dominant and a bit-flipping failure; and that, in addition to these failures, a medium can also suffer from a physical disruption.

In order to improve the dependability of CAN, several approaches have been proposed, e.g. [13] [14] [17] [18] [20] [21] [5]. Special attention should be paid on the most recent of those proposals: ESCAPE CAN [5]. It includes a switch that emulates the arbitration and the acknowledgement mechanisms of CAN [12], in order to address babbling-idiot and masquerading faults. In this sense, it represents a complementary contribution to that of CANcentrate. However, ESCAPE CAN raises some concerns: the feasibility of its implementation has not been already demonstrated, it reduces the compatibility with existing CAN-based applications, and the complexity of its switch should be quite high when compared with a hub. The benefits of CANcentrate over the other solutions mentioned above are thoroughly described in [4].

In CANcentrate, each node is connected to the hub by a dedicated link that contains an uplink and a downlink. For each bit, the hub receives each node contribution through the corresponding uplink, couples all non-faulty contributions with a logical AND function and broadcasts the resultant coupled signal through the downlinks.

The use of an uplink and a downlink for each node allows separating the contribution of each node from the coupled signal, so that the hub can monitor each node contribution separately and detect faulty transmissions. This feature allows the hub to diagnose the location of faults with more precision than the typical error counters of CAN [12]. Permanently faulty contributions are disabled, thereby not propagated to the coupled signal, thus being confined to the port of origin.

Since CANcentrate is fully compliant with CAN, it keeps all CAN dependability features, it can be built using CAN COTS components and it is possible to use it with any CAN-based protocol (e.g. FTT-CAN [16]).

III. Modelling rationale

The models were built using the *Stochastic Activity Network* (SAN) formalism, a stochastic extension to Petri Nets [29]. Petri Nets have long ago been used to model and analyze communication protocols, real-time, highly-dependable and automated manufacturing systems, e.g. [30] [31].

The SAN formalism was used to built the models herein presented following a strategy similar to those proposed in [32] and [33]. A SAN includes *tokens*, *places*, *activities*, *input gates* and *output gates*. The number of tokens located in each place, i.e. the marking of the places, determines the state of the modeled system. Activities are used to change the marking of places, thereby modelling the system transitions through different states. An input gate defines an enabling rule for an activity, whereas an output gate specifies actions to be performed after its related activity fires. Additionally, the SANs formalism provides two primitives to build a model as a hierarchical composition of submodels: the *Join* primitive, which allows interconnecting different submodels, and the *Rep* primitive, which can be used to replicate a given submodel in order to model different instances of the same submodel. Once specified, a whole SANs model can be automatically transformed into a Markov Chain that is analytically solved. In particular, the Moebius software was used [29] to build and analytically solve all the models.

Moreover, some dependability features, e.g. the failure rates and the proportions of the different failure modes, are parameterized, which allows sensitivity analyses to be carried out. This parametrization allows these models to be easily adapted to assess the error-containment improvement achieved when using a star in technologies other than CAN.

A. Implementation assumptions

Firstly, it is necessary to establish what features of the different CAN physical layer standards are included in the implementation of the CAN bus and CANcentrate. This is because different CAN physical layer standards exhibit different degrees of fault tolerance and electrical robustness and, hence, a fair comparison between a CAN bus and a CANcentrate star must consider that both use the same physical layer standard. The ISO 11898-2 High-speed CAN standard [34], which is the most widespread one, was chosen. This standard specifies a two-wire differential bus line terminated at both ends with impedances of 120 Ohm. High-speed CAN does not mandate the tolerance of faults affecting any of the two wires or any of the bus line terminations. Thus, it is assumed that a fault affecting any of the wires will not be tolerated. Conversely, the results experience has yielded in the use of 82C250 CAN transceiver [35] have been used to assume in this paper that the loss of up to one line termination can be tolerated.

Secondly, it is necessary to address the length and layout of both the bus and the star, how nodes are attached to the medium, and the quantity of wires included in each cable. A total length of 70m for the CAN bus is assumed, as well as a length of 35m for each link of the star [4]. Note that such layout reduces the dependability results of CANcentrate, since it includes more cable than what is actually needed for interconnecting an ensemble of nodes in which the two farthest ones are separated 70m.

In order to attach the nodes to the medium in CAN, a daisy chain configuration was chosen so that each pair of adjacent nodes is connected to each other using a CAN cable with a straight connector at both its ends. This is the most reliable option for the CAN bus since it includes no stubs and minimizes the number of connectors. For CANcentrate, each link (including an uplink and a downlink) connecting a node to the hub could also have one straight connector at each end. However, these links would need extra wires and straight connectors with more pins than for CAN. Using different types of connectors and cables for CAN and CANcentrate could pose some uncertainties concerning the fairness of the comparison. Hence, each uplink is implemented separately from its downlink, so that each one of them uses a CAN cable and a pair of connectors equal to those used for CAN. A CAN cable is assumed to include four wires, so that it carries the communication signals CAN_H and CAN_L, plus the power GND (or V-) and V+. Note that each link of the star includes an extra V+/GND pair that is unnecessary in practice, thereby biasing the results in favor of the CAN bus.

As for the terminations, the proposed approach is the typical one, in which special connectors that already have the terminations inside are attached at both ends of the bus or of each uplink/downlink.

Thirdly, it is necessary to take into account the way in which the hub is built and which of its features are to be modeled. For the former point, a CANcentrate prototype hub [4] , which consist of a core that implements fault-treatment capabilities and includes an interface, was selected. The core is implemented in a dedicated IC whereas the interface is basically composed of a set of COTS transceivers. For the latter point, all hub functionalities were considered except CANcentrate's reintegration policy [4] because only permanent faults are taken into account here.

B. Constituent parts

To model the dependability of CAN and CANcentrate it is necessary to decide which are their constituent parts, choosing a level of abstraction that keeps the complexity of the models within reasonable limits.

The components of the CAN bus and CANcentrate were gathered into different assemblies or *network entities*. The first entity is the *Node*, which includes an oscillator; a microprocessor; one CAN controller; a 32Kb SRAM and a 32Kb EEPROM; one or two transceivers, depending on whether the node is a CAN or a CANcentrate node respectively; the corresponding integrated circuit (IC) sockets, and a printed circuit board (PCB), in which the amount of connections depends on the number of components attached to it. The second one is the *Link*, which comprises a CAN cable and a pair of straight connectors. The third one is the *Termination*, which is basically a resistor. Finally, in CANcentrate the hub is divided in two types of entities: the *Hub core* which is composed of a dedicated IC, the hub's PCB, an oscillator, and a socket; and the hub interface, composed by one *Hub Input* and one *Hub Output* per hub port. Each one of these two entities includes a CAN transceiver, a portion of the hub's PCB and a socket.

C. Fault assumptions

The models proposed are based on a set of assumptions regarding the statistical properties of faults, failure rates, failure modes and coverages. All faults are assumed to be permanent. Each component may independently fail, following an exponential *Time-To-Failure* distribution (F(t)) with a *Mean Time to Failure* (MTTF) equal to $1/\lambda$; where λ is the failure rate expressed in number of failures per hour. F(t) was also assumed to be *Non-Defective* [32], which implies that the probability with which the component fails at or before time t, F(t) = Prob($X \leq t$), is 0 when $t = 0$, $1 - exp(-\lambda \cdot t)$ when $0 < t < \infty$, and 1 when $t = \infty$.

The possible failure modes are based on the fault model CANcentrate is able to deal with (see Section II). Thus, in principle, an entity can only exhibit a stuck-at-recessive, a stuck-at-dominant, a bit-flipping and, in the case of a *Link*, also a physical disruption. In CANcentrate an uplink that suffers from a disruption delivers a permanent stuck-at-recessive stream to the hub while a disrupted downlink only delivers a stuck-at-recessive value to its node. In contrast, a physical disruption in the bus line can lead to network partition or can just provoke the loss of a termination

Apart from these failures, two new failure modes have been included in the fault model proposed in this paper. First, the resistor that constitutes a *Termination* can also fail leading the medium to lose the termination itself, but without generating a stuck-at or a bit-flipping stream. Second, it is assumed that any fault happening in the *Hub core* causes the failure of all the communication system. This is a pessimistic assumption for CANcentrate since the *Hub core* could suffer from a more benign fault, e.g. it could stop performing fault confinement or it could unfairly isolate a correct port.

Note that in a real system components can actually fail in other manners. This fact can be formalized by means of the concept of *failure mode assumption coverage* [36], defined as the probability that a component failure mode assumption proves to be true in practice. In order to take this coverage into account, it has been considered that each entity can also exhibit a new type of failure, herein called *out-of-fault-model* failure, which gathers all failure modes that are beyond the error-containment capabilities of CANcentrate. For example, a short circuit of the GND wire to the V+ wire that causes a fire that propagates along all the cabling installation.

Finally, in addition to the assumption coverage, it is necessary to recall that although the hub of CANcentrate is devised to deal with a specific fault model, the design and/or the implementation of its error-containment mechanisms cannot be perfect. Therefore, in practice, it is possible that the hub cannot isolate a fault that manifests in a way included in the fault model of CANcentrate. In order to take into account these kinds of situations, the concept of *error-containment coverage* was introduced in the model of CANcentrate. This concept is defined here as the probability that when a failure included in the fault model occurs the hub can detect and isolate it.

Specifically, stuck-at-recessive failures are always isolated, since even if the hub cannot detect them, they have no negative impact in the communication among the nodes. The error-containment coverage of the rest of failure modes depends on the difficulty off their detection. The mechanism used by the hub to detect stuck-at-dominant failures is trivial: a counter that monitors the number of consecutive dominant bits and a specific threshold [4]. Hence it can be said that stuck-at-dominant failures are detected and isolated with a perfect coverage. This is a realistic assumption because if such a failure occurs and the *Hub core* is not faulty, then the related counter will eventually reach its threshold. In contrast, a bit-flipping failure can lead to a huge amount of scenarios and, thus, it is not reasonable to assure a perfect error-containment coverage for it. This paper's model of CANcentrate allows specifying the coverage with which bit-flipping failures are covered as a parameter. In particular, a 90% bit-flipping error-containment coverage is herein assumed, which is not an optimistic assumption for CANcentrate since the various faults so far injected in the presently described CANcentrate prototypes [37] were at all times correctly detected and isolated by the hub.

D. Failure rates and failure mode proportions

Most of the decisions described above concerning implementation assumptions, constituent components and failure assumptions are reflected in the form of specific values for failure rates and failure mode proportions. This allows for both the use of the models proposed in this paper to evaluate different CAN and CANcentrte networks, and the adaptation of said models to technologies other than CAN without great changes in the structure of the models themselves.

A software of prediction of failure rates (called Relex [38] was used to obtain the failure rates of the components, taking into account the MIL-HDBK-217F-2 model [39] and the Tellcordia *Method I Case I* calculation method [38]. The MIL-HDBK-217F-2 is the most widely accepted model for the calculation of failure rates, whereas the Tellcordia methodology is widely used by commercial organizations. In particular, the *Method I Case I* is used when not all specific data regarding components is available, as was the case in this paper. Among all the operating environments provided by the MIL-HDBK-217F-2, the *Ground Mobile* (GM) was chosen, as it is better suited for an in-vehicle system. On the other hand, *Method I Case I* assumes 40 degrees Celsius as the operating temperature, and 50 percent rated stress. The failure rate of a given entity is obtained by adding the failure rates of its components. For all components non-optimistic technological characteristics were assumed e.g. commercial quality level. In order to calculate the failure rate of the *Hub core* for a given number of ports it is necessary to specify its number of logic gates. To this purpose real values obtained from synthesizing said *Hub core* in an FPGA [4] were used.

Regarding the different failure modes' proportions, it is reasonable to assume that all entities exhibit 5% of out-of-fault-model failures. This is because the probability with which a fault in a component causes an unexpected situation, e.g. a fire, is low. On the other hand, all entities, except the *Node*, are supposed to manifest the rest of failure modes equiprobably. In the case of a *Node* it is still necessary to take into account that part of its errors will be contained by the fault-treatment mechanisms of its CAN controller [12]. The CAN controller can detect channel errors and stop transmitting when diagnosing a permanent fault, thereby converting faults within its node to a stuck-at-recessive stream, i.e. fail-silent behavior. However, it cannot be assumed that a node is 100% fail-silent because that is a practical impossibility. Thus a 90% fail-silent (stuck-at-recessive) proportion was chosen. This value is reasonable and maybe slightly overestimated since, for example, a CAN controller cannot prevent its node from being stuck-at-dominant when the oscillator stops [8]; it can do nothing to contain stuck-at-dominant or bit-flipping streams generated by its transceiver; nor is it even clear whether or not a CAN controller can isolate its own faults, specially those that affect the circuits that implement its fault-treatment mechanisms. In any case, due to the relevance of this parameter, an analysis of how the results are affected by varying this proportion has been carried out. Concerning the rest of failure modes, the fact that a *Node* may be equiprobably stuck-at-dominant and bit-flipping has been considered.

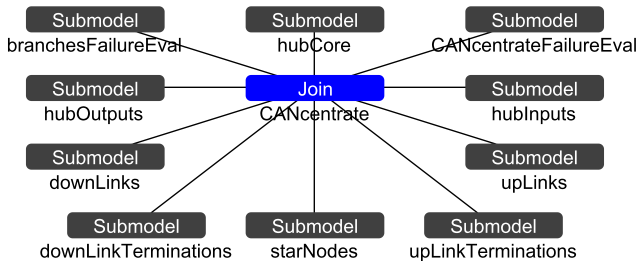

Fig. 1. CANcentrate model

IV. CANCENTRATE MODEL

The models herein presented were built using SANs that have been analytically solved. In a preliminary study [40] CAN and CANcentrate were extensively modeled using the *Join* and *Rep* primitives. Nevertheless, when more details were included in the models, the use of the *Rep* primitive led the state space transformation of the SANs into Markov Chains to become extremely inefficient in terms of computation time. Similar problems have been reported when using Petri Nets to model complex and large systems [41]. In order to overcome this drawback CAN and CANcentrate were modeled following a new approach that does not use the *Rep* primitive.

Figure 1 depicts the model of CANcentrate, which consists of three classes of submodels called *entity*, *entity group*, and *evaluator* submodels. The only *entity* submodel is *hubCore*, which models whether or not the entity *Hub core* is faulty.

There is an entity group submodel for each group of *entities* of a given type, e.g. *Nodes*, that have the same location within its corresponding *branch*. *branch* is defined as the ensemble that is composed of a *Node*; its corresponding downlink and uplink, in which each is a *Link* entity; two pairs of *Terminations*, each pair corresponding to an uplink/downlink; a *Hub input*; and a *Hub output* entity.

When an entity fails, the corresponding entity group submodel decides whether or not the failure leads a previously fault-free branch to be faulty, as well as the specific mode in which the failure manifests at the corresponding hub port. Notice that the location of an entity within its branch determines the way in which the failure manifests at the corresponding hub port. For example, a stuck-at-dominant failure in an uplink manifests as a stuck-at-dominant at the hub port, whereas if it happens in a downlink, it manifests as a stuck-at-recessive. This is because a non-faulty CAN node that receives only dominant bits through the downlink, will enter into the *bus-off* state [12] in which case it stops communicating.

On the other hand, there are two *evaluator* submodels. The first one, called *branchesFailureEval*, evaluates whether or not a failure happening in a branch can be confined by the hub. The other, called *CANcentrateFailureEval*, takes into account failures occurred at branches and at the *Hub core* to determine when a generalized failure has occurred.

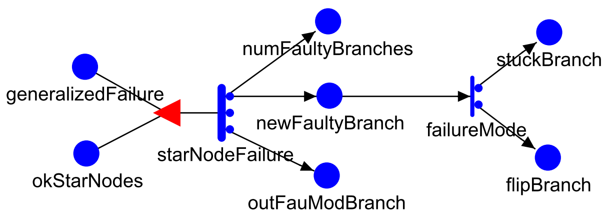

Fig. 2. StarNodes submodel

Additionally, this submodel indicates to the rest of submodels that such a failure occurs, so that all of them can stop evolving in order to reduce the state space. It is important to note that *CANcentrateFailureEval* allows specifying the number of *Nodes* that must be prevented from communicating before a generalized failure can be considered to have occurred. In this way, it is possible to study the reliability as well as different degrees of severe failure.

Finally, notice that the *join* primitive is used to interconnect different submodels by specifying the places they share. Due to space limitations, it is not possible to describe all submodels and their interconnections. Nevertheless, below follow more detailed explanations of the most representative submodels.

A. StarNodes submodel

The *starNodes* submodel, which is depicted in Figure 2, is a representative example of an entity group. The marking of place *okStarNodes* denotes the number of *Nodes* that are not faulty; whereas a token in any of the places *stuckBranch*, *flipBranch*, and *outFauModeBranch* indicates to the *branchesFailureEval* submodel that a *Node* has failed leading a branch to suffer from the corresponding type of failure. On the other hand, place *numFaultyBranches* is shared with all the entity group submodels and its marking denotes the number of branches that are faulty.

More specifically, the activity *starNodeFailure* models the time that elapses until any non-faulty *Node* fails. Since the Time-To-Failure distribution of each *Node* is exponentially distributed and each *Node* can independently fail, the time elapsed until a surviving *Node* fails is also exponentially distributed, with a failure rate $\lambda_{starNodes} = \lambda_{starNode} \cdot okStarNodes \rightarrow Mark()$. $\lambda_{starNode}$ is the failure rate of a *Node*, and $okStarNodes \rightarrow Mark()$ is the marking of place *okStarNodes*, i.e. the number of surviving *Nodes* at a given instant of time.

When the activity *starNodeFailure* fires, a token is erased from *okStarNodes* and one of the three cases, represented by circles at the right edge of the activity, is chosen. The first case represents a *Node* that exhibits a failure mode included in the fault model and that, additionally, provokes the failure of a branch that was not faulty. This case is selected with probability $(1-nodeOutFauMod)\cdot(nBranches-numFaultyBranches \rightarrow Mark())/okStarNodes \rightarrow Mark()$; where *nodeOutFauMod* is the proportion in which a *Node* fails exhibiting an out-of-fault-model failure mode; *nBranches* is the total number of branches; and $numFaultyBranches \rightarrow Mark()$ is the number of branches that have failed so far. When this case is chosen, the marking of place *numFaultyBranches* is increased in one token in order to track the number of faulty branches. Additionally, a token is placed in *newFaultyBranch*, thereby enabling activity *failureMode*. This activity instantaneously fires, transferring the token to one of the two places: *stuckBranch* and *flipBranch*, which respectively represent that the fault manifests as a stuck-at and as a bit-flipping failure at the hub port. The probability of choosing the first case is calculated by adding up the proportions with which a *Node* exhibits stuck-at-recessive (fail-silent) and stuck-at-dominant failures; whereas the probability of the second one is the proportion of *Node* bit-flipping failures.

The second case of the activity *starNodeFailure* also models a situation in which the failure mode is included in the fault model. But this case corresponds to a situation in which the *Node* that fails is located in a branch that was already isolated. Since the reintegration policy of CANcentrate is not considered, the *Node* failure does not further impact the communications. Thus, no more actions are performed within the model when this case is chosen.

Regarding the third case, it models a *Node* that fails in a way not included in the fault model and that leads to a generalized failure. Thus, it is selected with probability *nodeOutFauMod*. When this occurs, a token is placed in *outFauModeBranch*, thereby indicating that a branch suffers from such a kind of failure. Notice that in this case it is not necessary to evaluate whether or not the *Node* is placed in a branch that was previously non-faulty. This is because, an out-of-fault-model failure will lead the whole communication subsystem to fail in spite of the presence of the hub.

Finally, it is worth noting that the place *generalizedFailure* is shared among all submodels. As will be explained later, submodel *CANcentrateFailureEval* places a token at this place to indicate that a generalized failure occurred, so that all submodels can stop to reduce the state space. As can be seen in Figure 2, place *generalizedFailure* is connected to the input gate corresponding to the activity *starNodeFailure*, in order to stop the *starNodes* submodel by disabling this activity when a generalized failure occurs.

B. BranchesFailureEval submodel

Submodel *branchesFailureEval* shares with all entity group submodels the places: *stuckBranch*, *flipBranch* and *outFauModeBranch* (Figure 2). When a token is set in any of these places, *branchesFailureEval* decides whether or not the corresponding failure is successfully confined by the hub, and sets a token in one of three possible places: *faultyCoveredBranches*, *faultyNonCoveredBranch* and *outFauModFailure*, removing the original token. Specifically, the marking of place *faultyCoveredBranches* indicates the number of branches that are faulty but successfully isolated by the hub. Conversely, a token in place *faultyNonCoveredBranch* means that a branch has failed and that the hub was unable to isolate it. Finally, a token in *outFauModFailure* indicates that a branch presents an out-of-fault-model failure.

Since the error-containment coverage of stuck-at failures is 100%, a token in *stuckBranch* is always transferred to *faultyCoveredBranches*. In contrast, the error-containment coverage

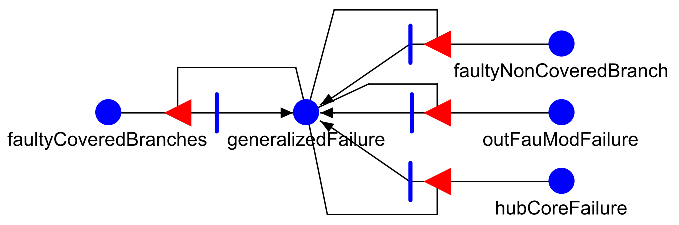

Fig. 3. CANcentrateFailureEval submodel

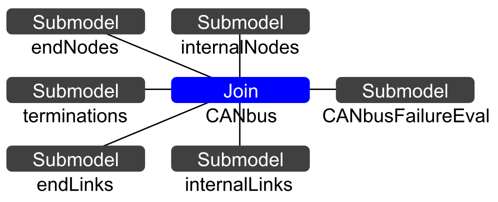

Fig. 4. CANbus model

of a bit-flipping failure is not perfect. Submodel *branchesFailureEval* models this fact by transferring any token set in *flipBranch* to either *faultyCoveredBranches* or *faultyNonCoveredBranch*. As explained in Section III-C, 90% is not an optimistic assumption for this coverage. However, the probability of choosing one or the other place is parameterized, thereby allowing a sensitivity analysis with respect to it.

C. *CANcentrateFailureEval submodel*

CANcentrateFailureEval submodel, which is depicted in Figure 3, is devoted to deciding when the network fails as a whole, and informs all submodels about such situation by placing a token in place *generalizedFailure*. On the one hand, it becomes aware of failures happening at branches by means of places *faultyCoveredBranches*, *faultyNonCoveredBranch* and *outFauModFailure* which are shared with the *branchesFailureEval* submodel. On the other hand, it shares place *hubCoreFailure* with submodel *hubCore* to be able to detect when the *Hub core* is faulty.

Each one of these places is connected to *generalizedFailure* by means of an input gate and an *instantaneous activity* [29]. Each input gate enables the activity it is connected to, depending on the marking of its incoming place. In particular, whenever a token is received in any of the places *faultyNonCoveredBranch*, *outFauModFailure* or *hubCoreFailure*, the corresponding gate triggers its activity, so that a token is placed at *generalizedFailure*. This is because a fault that cannot be isolated propagates preventing all *Nodes* from communicating.

In contrast, the number of the tokens located in *faultyCoveredBranches* that are accepted before enabling the activity this place is connected to is parameterized. In this way, it is possible to vary the number of *Nodes* that must be unable to communicate before the communication system is considered to be faulty. In particular, if one token is enough to enable this activity, then the probability of not having a token in *generalizedFailure* is the reliability of the communication subsystem. Conversely, if more than k tokens are necessary to enable the activity, then the probability of not having a token in *generalizedFailure* constitutes the probability of not suffering a k-severe failure.

V. CANbus model

The CAN bus has been modeled following the same approach as for CANcentrate. As shown in Figure 4, the CAN bus is modeled by joining different *entity group* submodels with an *evaluator* submodel, called *CANbusFailureEval*.

Analogously to the CANcentrate model, an entity group represents all entities of a given type that have the same location. Again, the consequences that an entity failure provokes on the service delivered by the communication subsystem depends on where the entity is located within the bus. For instance, a disruption in the *Link* that connects a *Node* placed at an end of the bus with the next *Node* in the bus line will not cause a severe failure. This is because that failure will only prevent the *Node* located at the end of the bus from communicating. In contrast, a disruption in a *Link* that interconnects any pair of *Nodes* not located at the ends of the bus will cause that fewer than $N-1$ *Nodes* can communicate among themselves, thereby provoking a k-severe failure with $k=1$.

More specifically, the two *Nodes* meant are each located at one of the ends of the bus line, as *End nodes*, and were modeled with the *endNodes* submodel. Likewise, in reference to the two *Links* located at the ends of the bus line, as *End links*, which were modeled with the *endLinks* submodel. The rest of the *Nodes* and the *Links*, i.e. the *Internal nodes* and the *Internal links*, are respectively modeled with the *internalNodes* and *internalLinks* submodels. Finally, the two *Terminations* are modeled with the *terminations* submodel.

As concerns the consequences that the failure of an entity can provoke in the service delivered by the communication subsystem, a difference between a *node exclusion* and a *blocking failure* has been established. The former refers to a situation in which the failure leads only one *Node* to be prevented from communicating; whereas the second occurs when the errors generated by the faulty entity prevent all *Nodes* from communicating.

On the other hand, the evaluator submodel *CANbusFailureEval* takes into account all faults occurred, decides when a generalized failure takes place, and notifies this to the rest of submodels, so that all of them can stop. Recall that the *CANcentrate* model allows the specification that k of N *Nodes* can be prevented from communicating before diagnosing a generalized failure, where $0 \leq k < N-1$. In contrast, the *CANbus* model only allows k in the range $0 \leq k < 2$ to be specified.

The next section explains the most illustrative submodels that constitute the *CANbus* model.

A. *EndNodes submodel*

The *endNodes* submodel models the two *End nodes* that are placed at both ends of the bus line. An *End node* that suffers from an out-of-fault-model failure causes a generalized failure of communication. However, conversely to what happens with

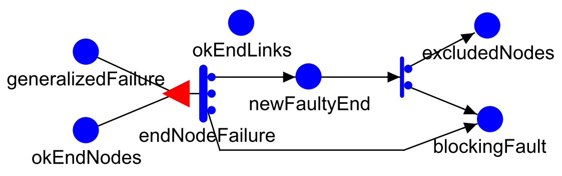

Fig. 5. EndNodes submodel

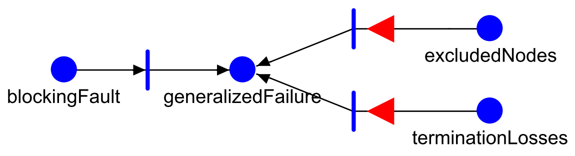

Fig. 6. CANbusFailureEval submodel

Internal nodes, if the failure suffered by an *End node* is included in the fault model, it is possible that this failure does not cause any impairment because the *End node* could already be isolated. In particular, an *End node* is already isolated when the *End link* that connects it to the rest of the bus has suffered a previous disruption For this reason, besides the place *okEndNodes*, whose marking indicates the number of surviving *End nodes*, the submodel includes the place *okEndLinks*, which denotes the number of non-faulty *End links* (see Figure 5).

On the other hand, places *excludedNodes* and *blockingFault* are shared with the other submodels. The marking of the former specifies the number of *Nodes* that have been excluded from the communication so far; whereas a token in the second one means that an entity has failed provoking a blocking failure.

Activity *endNodeFailure* models the time that elapses until any of the surviving *End nodes* fails. Thus, this time is exponentially distributed, with a failure rate $\lambda_{endNodes} = \lambda_{endNode} \cdot okEndNodes \rightarrow Mark()$. Where $\lambda_{endNode}$ is the failure rate of an *End node*; and $okEndNodes \rightarrow Mark()$ is the number of surviving *End nodes* at a given instant of time.

When this activity is triggered, three cases are possible. The first and the second cases, respectively located at the top and at the middle of the activity, refer to situations in which the *End node* exhibits a failure mode included in the fault model; whereas the third case models that an *End node* fails exhibiting an out-of-fault-model failure. The difference between the two first cases is that while the first one corresponds to a case in which the *End node* that fails was able to communicate before failing, the other one coincides with a situation in which the *End node* had already been excluded from the communication because its *End link* was disrupted.

The probability of the third case is always the proportion with which a *Node* exhibits an out-of-fault-model failure, i.e. *nodeOutFauMod*. In contrast, the probability of selecting one of the two first cases depends on both the number of surviving *End nodes* and surviving *End links* at a given instant of time. If no *End node* and no *End link* were previously faulty, the probability of excluding a new *End node* is $(1 - nodeOutFauMod)$. But, if one *End link* was faulty, then this probability is $(1 - nodeOutFauMod) \cdot 0.5$, since there is only a 0.5 probability that the *End node* does not correspond to that faulty *End link*. Conversely, if one of the *End nodes* was faulty, then the probability of excluding a new *End node* is also $(1 - nodeOutFauMod)$, independently of whether or not an *End link* has also failed. This is because if there is a faulty *End link*, it must correspond to the already faulty *End node*. Otherwise it would imply that the already faulty *End node* and the already faulty *End link* are located at a different bus end, which would have provoked a generalized failure (a severe failure), thereby stopping all the submodels.

B. *CANbusFailureEval submodel*

The structure of submodel *CANbusFailureEval* is shown in Figure 6. In order to know what faults have happened, it shares places *blockingFault*, *excludedNodes* and *terminationLosses* with other submodels.

CANbusFailureEval receives a token in place *blockingFault* whenever an *Internal link* fails or when an *End link* exhibits a failure other than a physical disruption. When an *Internal node* exhibits a stuck-at-dominant or a bit-flipping failure a token is also received in *blockingFault*. In what concerns the failure of an *End node*, it only leads a token to be received at this place if it suffers an out-of-fault-model, or if it suffers a stuck-at-dominant or a bit-flipping and had not previously been excluded due to the disruption of its corresponding *End link*.

On the other hand, the marking of place *excludedNodes* is increased when an *End link* corresponding to a non-faulty *End node* suffers a physical disruption; as well as when an *Internal node* or an *End node* that is not already isolated suffers a stuck-at-recessive failure.

Finally, the marking of *terminationLosses* denotes the number of *Terminations* that are lost. Recall from Section III-A that a *Termination* is placed within the connector that attaches an *End link* with an *End node*. Thus, a given *Termination* can be lost not only when it fails, but also when its corresponding *End link* suffers a physical disruption. Thus, the marking of *terminationLosses* is increased when an *End link* fails and its corresponding *Termination* was non-faulty and viceversa.

A token in *blockingFault* is instantaneously transferred to *generalizedFailure*. In contrast, since up to one *Termination* loss is tolerated, two tokens are needed in place *terminationLosses* to provoke a generalized failure. Finally, it is possible to specify whether one token or two tokens are needed in *excludedNodes* to provoke such a failure. In the first case, the probability of not having a token in *generalizedFailure* is the reliability of the CAN bus, whereas in the second case it is the probability of not suffering a k-severe failure with $k = 1$.

VI. Quantitative assessment

In order to assess the advantages and disadvantages of using CANcentrate instead of a CAN bus, a comparison was carried out on the reliability and the probability of not suffering a k-severe failure (PNS), when $k = 1$. Particularly, the focus is on

Parameter	Value
Hub bit-flipping error-containment coverage	90%
Node fail silent proportion	90%
CAN bus internal node failure rate	$4.11569 \cdot 10^{-6}$
CAN bus end node failure rate	$4.11426 \cdot 10^{-6}$
CANcentrate node failure rate	$4.26449 \cdot 10^{-6}$
CAN wire failure rate (per Km)	$1.00000 \cdot 10^{-7}$
Connector failure rate	$2.07774 \cdot 10^{-8}$
Termination failure rate	$7.38299 \cdot 10^{-8}$
Hub core failure rate (in the case of 4 ports)	$1.04499 \cdot 10^{-6}$
Hub Input/Output failure rate (per port)	$1.50232 \cdot 10^{-7}$

TABLE I
SOME DEPENDABILITY PARAMETERS

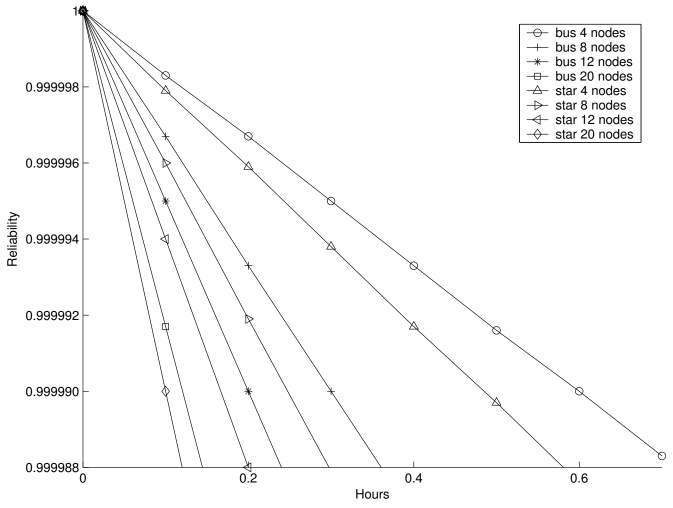

Fig. 7. Reliability vs number of nodes

the *mission time* achieved by CAN and CANcentrate, which is defined as the expected length of operation of one mission for a system [28].

Despite targeting general field-bus communications, typical values of reliability of x-by-wire systems were taken as a means of reference. Specifically, a comparison was made of the mission times during which CAN and CANcentrate present a reliability or a PNS ≥ 0.99999, which is the value of reliability required by the less demanding x-by-wire systems, e.g. for throttle-by-wire, during a mission of 10 hours [28].

The comparison is carried out considering from 4 up to 20 nodes, which are typical node numbers for CAN networks in body and powertrain electronics applications in vehicles [42].

Finally, remember that, as stated in Section III, all dependability parameters were determined, and options taken, with specific care not to favor CANcentrate, which indicates that the results herein presented are likely to be lower bounds to the dependability achievable with CANcentrate.

A. Reliability vs number of nodes

The main dependability parameters used for comparing the reliability of both CAN and CANcentrate are shown in Table I. This table also shows the computed values of those parameters following the techniques, assumptions and options explained in the previous sections.

As depicted in Figure 7, the CAN bus is more reliable than CANcentrate for any number of nodes. This is an expected result given the extra hardware of CANcentrate with respect to CAN. The difference of reliability grows as the time or the number of nodes increases.

If the mission times during which the reliability is ≥ 0.99999 are studied, CAN and CANcentrate provided with 4 nodes, respectively achieve 0.6 and 0.48 hours of mission time. This implies that CANcentrate reduces the mission time to around 20%. If 20 nodes are considered, then CAN and CANcentrate respectively achieve 0.12 and 0.1 hours of mission time. In this case CANcentrate reduces the mission time to around 16.6%. Note that although a higher number of nodes implies a bigger difference between CAN and CANcentrate in terms of reliability, the drop of mission time provoked by CANcentrate decreases as the number of nodes grows. This is because the reliability of both infrastructures decreases faster as the number of nodes increases.

B. Error-containment vs number of nodes

Here error-containment is assessed in systems that are robust to the loss of communication with at most 1 out of N nodes, i.e. systems that can operate as long as a k-severe failure, where $k = 1$, does not occur. As mentioned before, PNS is used as metric of error containment. In particular, the mission times during which CAN and CANcentrate present a PNS ≥ 0.99999 are analyzed.

Figure 8 depicts the PNS of CAN and CANcentrate for different numbers of nodes. Results are, to some extent, the opposite of those obtained in terms of reliability: CANcentrate is better than CAN for any number of nodes, and the difference between their PNS grows as the time or the number of nodes increases. This was also expected, given the enhanced fault-treatment mechanisms of CANcentrate.

Conversely to what happens with reliability, a higher number of nodes does not only imply a bigger difference in terms of PNS, but also a bigger difference in the achievable mission times. If the mission times during which the PNS is ≥ 0.99999 are analyzed, CAN and CANcentrate provided with 4 nodes, respectively achieve 3.8 and 4.4 hours of mission time. This implies that, for 4 nodes, CANcentrate increases this time to around 16%. But if 20 nodes are considered, then CAN and CANcentrate respectively reach 0.78 and 1.28 hours; which means that CANcentrate improves the mission time by 64%. The benefit that CANcentrate yields in terms of mission time increases as the number of nodes grows because the CAN bus is much more sensitive to this number than CANcentrate, e.g. a CANcentrate network provided with 12 nodes achieves a higher mission time than a CAN bus that includes 8. This means that CANcentrate supports an increase in the number of nodes that can be included in the network while ensuring

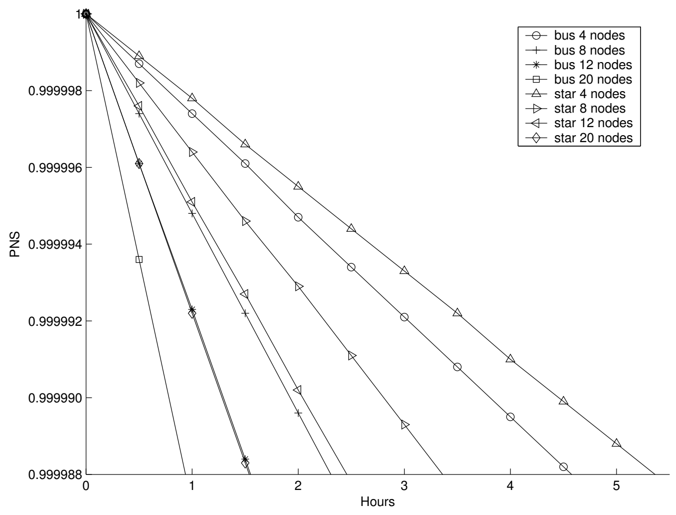

Fig. 8. PNS vs number of nodes

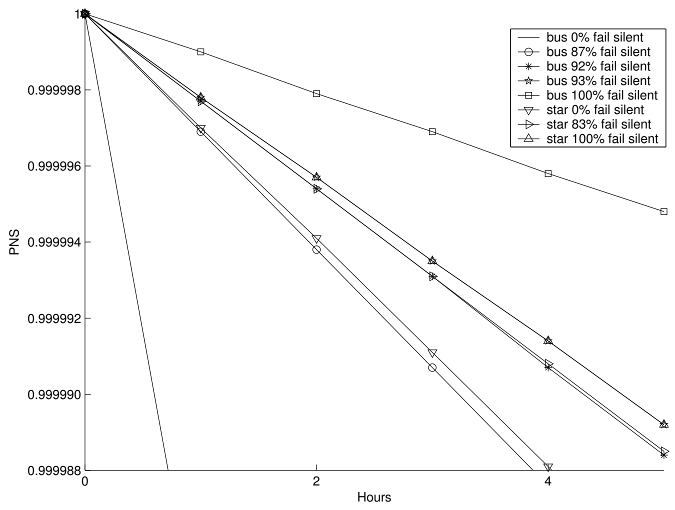

Fig. 9. PNS vs fail-silent node proportion for 4 nodes

PNS ≥ 0.99999 during a given mission time (approx. 50% in the previous example).

C. *Error-containment vs fail-silent node proportion*

As said before, these results are based on a set of assumptions that may have been too detrimental for CANcentrate. One of such assumptions is the proportion with which a node fails in a fail-silent manner, i.e. the proportion with which it fails delivering a stuck-at-recessive stream. As explained in Section III-D, although a proportion of 90% is herein considered reasonable, this value is maybe slightly overestimated, and thus biased in favor of the CAN bus. In fact, the real value for this parameter is unknown and, thus, a fair comparison between CAN and CANcentrate requires a sensitivity analysis with respect to it. Moreover, this analysis is also interesting to estimate the relevance of using star topologies in communication protocols in general, depending on the ability of nodes to contain their own errors.

To carry out this sensitivity analysis the percentage of stuck-at-recessive failures was varied. The proportion of out-of-fault-model failures is kept 5%, whereas stuck-at-dominant and bit-flipping failures are assumed equiprobable again, so that the sum of their proportions is equal to 95% minus the percentage of stuck-at-recessive failures.

Figure 9 depicts the PNS of the CAN bus and CANcentrate provided with 4 nodes. Note, again, that this is the number of nodes most unfavorable to CANcentrate as the advantages of CANcentrate are reinforced when the number of nodes is increased. The figure shows that when the proportion of node silent faults is 100%, the CAN bus is better than CANcentrate. This is obvious since with such a proportion the fault-treatment mechanisms of the hub become irrelevant. However, it can be seen that a drop of the proportion of silent faults dramatically affects the CAN bus, but not CANcentrate. This is due to the fact that the hub of CANcentrate is able to confine faults that, otherwise, would lead to a severe failure. More specifically, it is enough that the proportion of silent faults of nodes in a CAN bus drops to 87% to turn the bus into a worse option than a CANcentrate network in which the proportion is 0%.

On the other hand, notice that the PNS presented by a CAN bus where node failures are 92% fail-silent is lower than the PNS yielded by CANcentrate when this proportion is 83%. The CAN bus only becomes definitively better than CANcentrate when the proportion of fail-silent failures is $\geq 93\%$.

VII. Conclusions

In this paper a quantitative comparison has been carried out on the dependability benefits of using a star topology instead of a bus in CAN networks. In particular SANs were used to compare the reliability and the PNS when $k = 1$ in CAN and CANcentrate, and permanent faults occur. As far as it has been possible to determine, this is the first formal (quantitative) comparison between a star and a bus that takes into account the capacity of the hub and nodes to contain errors, different failure modes of the components, and implementation aspects.

Moreover, the models presented here are parameterizable so that they can throw light on how some design and implementation choices affect network dependability. This has a significant practical interest, since these models can help engineers in designing and implementing a bus or star-based network that fulfills a specific level of dependability, e.g. models can help them choose adequate electronic components depending on their complexity. Furthermore, the models can be used to elucidate when a star is preferable to a bus, based on the dependability requirements of a specific distributed embedded application. Specifically, these models allow carrying out sensitivity analyses with respect to the number of nodes, the proportion of fail-silent node failures, the failure rate and the failure mode of components, and the hub error-containment coverage for each one of these failure modes.

In this paper, from 4 to 20 nodes were considered, this being the typical node number in CAN-based body and powertrain electronics applications in vehicles. Results show that when reliability is the issue, the maximum decrease of mission time that CANcentrate can cause is around 20%. However, when PNS is the main concern, the improvement in mission time yielded by CANcentrate ranges from 16% to 64%. Moreover, the PNS of the CAN bus is much more sensitive to the number of nodes than the PNS of CANcentrate, allowing the latter to include more nodes while presenting a PNS above a specific threshold during a given mission time. This indicates that the use of a star is more and more justifiable as the number of nodes grows, even though including a node in a star requires more hardware than including a node in a bus does.

Results demonstrate that CANcentrate is more suited than CAN for systems that accept or tolerate up to 1 communication-failed node, thereby quantitatively corroborating the advantage that star topologies intuitively yield in terms of error-containment. Moreover, results are likely to be lower bounds to the dependability of CANcentrate, given the special concern taken to ensure it was not favored in the analysis and the fact that several other advantages, such as the minimization of the impact of spatial proximity faults, were not even modeled, as explained in Section I. Therefore, it can thus be inferred that even in the case that our analyses should yield a result whereby the PNS of CAN is equal to the one of CANcentrate, the actual fact is that CANcentrate is superior.

An analysis of the sensitivity of the PNS with respect to the proportion with which the nodes exhibit fail-silent failures was carried out, using for that purpose the developed parametrizable models of CAN and CANcentrate. As far as it can be ascertained, this is the first time that the ability of nodes to contain their own errors has been studied in terms of how it affects the relevance of using a star topology. Results show that this ability dramatically affects the PNS presented by the CAN bus, but not that of CANcentrate. However, they also show that when the nodes of a bus have a similar failure rate than the nodes of a star (which is the case presented in this work), then the use of a star topology is only justified if the probability that nodes are fail-silent is $\leq 92\%$.

Although these results refer to the case of CAN, and other technologies, such as TTP/C or FlexRay, deal with different failure modes, conclusions regarding the justification of using a star topology depending on the fail-silent node proportion can be extrapolated to these technologies. On the one hand, other technologies use similar components with similar failure rates. On the other hand, failure modes can be abstracted so that what really matters is the proportion of failures that can be covered by the hub and the nodes.

Future work will perform additional sensitivity analyses with respect to dependability parameters such as the error-containment coverage of the hub; the failure rate of components whose amount is bigger in CANcentrate; and the value of k for different degrees of k-severe failure. Future work is also planned to include other technologies and topologies.

Finally, as explained in Section I, a quantitative assessment including transient faults has been postponed to a later work. Notice that transient faults, by definition, cannot prevent nodes from communicating indefinitely but cause temporary unavailability of the communication system thereby negatively affecting its performance, e.g. deadline violations, increased average response times, packet losses. The particular impact on dependability is, necessarily, application dependent and thus, it is beyond the scope of this paper. For example, deadline violations can lead to a generalized failure in hard real-time systems; but this strongly depends on the specific set of messages and the scheduling [43]. Besides, response times can also affect the dependability of some types of control system, e.g. of life-critical ones [44]. But similarly to what happens with deadline violations, the way in which performance affects dependability depends on the application.

It is nevertheless possible to assess the performance that can be achieved with a bus and a star topology thus leading to an indirect comparison of the dependability they can yield in the presence of transient faults. Therefore, a study will be carried out in the short term in order to quantitatively compare the performability of the CAN bus and CANcentrate under these faults.

Acknowledgement

This work was supported in part by the Spanish Science and Innovation Ministry with grant DPI2008-02195, and in part by FEDER funding.

References

[1] J. Rushby, "A Comparison of Bus Architectures for Safety-Critical Embedded Systems," SRI International, Menlo Park, California," Contractor Report, 2003.

[2] H. Kopetz, "Time-Triggered Protocols for Safety-Critical Applications," Presentation, Vienna University Of Technology, TU Wien, Karlsplatz 13, 1040 Vienna, Austria, March 2003.

[3] A. Ademaj, G. Bauer, H. Sivencrona, and J. Torin, "Evaluation of Fault Handling of the Time-Triggered Architecture with Bus and Star Topology," *IEEE International Conference on Dependable Systems and Networks (DSN 2003), San Francisco*, Jun. 2003.

[4] M. Barranco, J. Proenza, G. Rodríguez-Navas, and L. Almeida, "An Active Star Topology for Improving Fault Confinement in CAN Networks," *IEEE Transactions on Industrial Informatics, vol. 2, issue 2*, vol. 2, pp. 78–85, May 2006.

[5] B. Hall, M. Paulitsch, K. Driscoll, and H. Sivencrona, "ESCAPE CAN Limitations," in *SAE 2007 Transactions Journal of Passenger Cars: Electronic and Electrical Systems, Detroit, USA*. Society of Automotive Engineers, USA, August 2008.

[6] G. Bauer, H. Kopetz, and W. Steiner, "The central guardian approach to enforce fault isolation in the time-triggered architecture," in *Proceedings of the The Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03), Washington, DC, USA*. IEEE Computer Society, 2003, p. 37.

[7] G. Cena, A. Valenzano, and S. Vitturi, "Advances in automotive digital communications," *Computer Standards & Interfaces, vol. 27, issue 6*, pp. 665–678, June 2005.

[8] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in Automotive Communication Systems," *Proceedings of the IEEE*, vol. 93, no. 6, 2005.

[9] *Fault Tolerant Services For Safe In-Car Embedded Systems*. CRC Press. The Embedded Systems Handbook. Edited by Richard Zurawski, 2005.

[10] *Handbook on Automotive Embedded Systems*. CRC Press. Edited by Nicolas Navet and Françoise Simonot-Lion, 2009.

[11] FlexRay^{TM}, "FlexRay Communications System - Protocol Specification, Version 2.1," FlexRay^{TM}, 2005.

[12] ISO, "ISO11898-1. Controller area network (CAN) – Part 1: Data link layer and physical signalling," 2003.

[13] IXXAT, "Innovative products for industrial and automotive communication systems," 2009. [Online]. Available: http://www.ixxat.de/index.php

[14] H. Saha, "Active high-speed CAN hub," *Proceedings of the 11th International CAN Conference, Stockholm, Sweden*, September 2006.

[15] C. Cena and A. Valenzano, "An Improved CAN Fieldbus for Industrial Applications," *IEEE Transactions On Industrial Electronics*, vol. 44, no. 4, pp. 553–564, August 1997.

[16] L. Almeida, P. Pedreiras, and J. A. Fonseca, "The FTT-CAN Protocol: Why and How," *IEEE Transactions on Industrial Electronics - Special Issue on Factory Communication Systems*, vol. 49, no. 6, December 2002.

[17] L.-B. Fredriksson, "CAN for critical embedded automotive networks," *IEEE Micro, Special Issue on Critical Embedded Automotive Networks*, vol. 22, no. 4, pp. 28–35, July-August 2002.

[18] I. Broster and A. Burns, "An Analyzable Bus-Guardian for Event-Triggered Communication," in *Proceedings of the 24th Real-time Systems Symposium (RTSS)*. Cancun, Mexico: IEEE, Dec 2003, pp. 410–419.

[19] C. Cena and A. Valenzano, "FastCAN: A High-Performance Enhanced CAN-Like Network," *IEEE Transactions On Industrial Electronics*, vol. 47, no. 4, pp. 951–963, August 2000.

[20] G. Buja, J. R. Pimentel, and A. Zuccollo, "Overcoming Babbling-Idiot Failures in CAN Networks: A Simple and Effective Bus Guardian Solution for the Flexcan Architecture," *IEEE Transactions On Industrial Informatics*, vol. 3, no. 3, pp. 225–233, August 2007.

[21] J. Rufino, P. Veríssimo, and G. Arroz, "Enforcing Dependability and Timeliness in Controller Area Networks," in *Proceedings of 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON 2006)*. IEEE, November 2006, pp. 3755–3760.

[22] F. Gil-Castineira, F. J. González-Castano, and L. Franck, "Extending Vehicular CAN Fieldbuses With Delay-Tolerant Networks," *IEEE Transactions On Industrial Electronics*, vol. 55, no. 9, pp. 3307–3314, August 2008.

[23] W. Prodanov, M. Valle, and R. Buzas, "A Controller Area Network Bus Transceiver Behavioral Model for Network Design and Simulation," *IEEE Transactions On Industrial Electronics*, vol. 56, no. 9, pp. 3762–3771, September 2009.

[24] G. A. Ray and J. J. Dunsmore, "Reliability of Network Topologies," in *INFOCOM 1988. IEEE International Conference on Computer Communications. New Orleans, USA.*, March 1988, pp. 842–850.

[25] F. Cao, "Reliability Analysis of Partitioned Optical Passive Stars Networks," in *22nd Annual IEEE International Conference on Local Computer Networks (LCN'97)*, 1997, p. 470.

[26] K. Fitzgerald, S. Latifi, and P. K. Srimani, "Reliability Modeling and Assessment of the Star-Graph Network," *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 49–57, March 2002.

[27] F. Atiparmak, B. Dengiz, and A. E. Smith, "Reliability Estimation Of Computer Communication Networks: ANN Models," in *Proceedings of the Eight IEEE International Symposium on Computers and Communication (ISCC 03)*, June 2003, pp. 1353–1358.

[28] J. Morris and P. Koopman, "Representing Design Tradeoffs in Safety-Critical Systems," *WADS. Workshop on Architecting Dependable Systems, St. Louis, Missouri, USA*, 2005.

[29] W. Sanders and T. B. of Trustees, "Moebius User Manual Version 1.6.0," 2004. [Online]. Available: http://www.mobius.uiuc.edu

[30] R. Zurawski and M. Zhou, "Petri Nets and Industrial Applications: A Tutorial," *IEEE Transactions On Industrial Electronics*, vol. 41, no. 6, pp. 567–583, December 1994.

[31] R. Zurawski, "Petri Net Models, Functional Abstractions, and Reduction Techniques: Applications to the Design of Automated Manufacturing Systems," *IEEE Transactions On Industrial Electronics*, vol. 52, no. 2, pp. 595–609, April 2005.

[32] M. Mahotra and K. S. Trivedi, "Dependability Modeling Using Petri-Nets," *IEEE Transactions on Reliability*, vol. 44, no. 3, September 1995.

[33] P. Portugal and A. da Silva, "A Framework for Dependability Evaluation of Fieldbus Networks," *WFCS'04. IEEE Workshop on Factory Communication Systems, Vienna, Austria*, 2004.

[34] ISO, "ISO11898-2. Controller area network (CAN) – Part 2: High-speed medium access unit," 2003.

[35] PHILIPS, "Data sheet - PCA82C250 - CAN controller interface - Product Specification," 2000.

[36] D. Powell, "Failure Mode Assumptions and Assumption Coverage," *Digest of Papers of the IEEE 22th Int. Symp. Fault-Tolerant Computing FTCS-22, Boston, Massachusetts-USA*, pp. 386–395, July 1992.

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

[38] R. Corporation, "Relex Reliability Software," 2006. [Online]. Available: http://www.relex.com

[39] DOD, *MIL-HDK-217F-2 Military Handbook, Reliability Prediction Of Electronic Equipment*. Department of Defense Washington DC, 1995.

[40] M. Barranco, J. Proenza, and L. Almeida, "First results of the assessment of the improvement of error containment achieved by CANcentrate," in *WFCS'06. IEEE Workshop on Factory Communication Systems, Torino, Italy*, 2006.

[41] M. Abdollahi and A. Movaghar, "Application of Stochastic Activity Networks on Network Modelling," *SoftCOM'02. 10th International Conference on Software, Telecommunications and Computer Networks, Split, Dubrovnik, Croatia*, 2002.

[42] O. Pauk, "Powering Connectivity in Todays Automobiles," *Power Electronics Technology magazine*, 2004.

[43] H. A. Hansson, T. Nolte, C. Norström, and S. Punnekkat, "Integrating Reliability and Timing Analysis of CAN-Based Systems," *IEEE Transactions On Industrial Electronics*, vol. 49, no. 6, pp. 1240–1250, December 2002.

[44] L. Tomek, V. Mainkar, R. Geist, and K. Trivedi, "Reliability Modeling of Life-Critical, Real-Time Systems," *Procedings of the IEEE*, 1993.

Manuel Barranco received the first degree in Informatics Engineering from the University of the Balearic Islands (UIB), Palma de Mallorca, Spain, in 2003. He is currently pursuing the Ph.D. degree in computer science at the UIB.

He is a member of the Systems, Robotics and Vision research group at UIB. His research interests include dependable and real-time systems, fault-tolerant distributed embedded systems, event-triggered and time-triggered communication systems, field-bus networks such as CAN and dependability modeling.

Julián Proenza received the first degree in physics and the doctorate in informatics from the University of the Balearic Islands (UIB), Palma de Mallorca, Spain, in 1989 and 2007, respectively.

He is currently holding a permanent position as a lecturer in the Department of Mathematics and Informatics at UIB. His research interests include dependable and real-time systems, fault-tolerant distributed systems, clock synchronization, dependable communication topologies, and field-bus networks such as CAN.

Dr. Proenza is a Member of the IEEE Industrial Electronics Society.

Luís Almeida received the licenciatura degree in electronics and telecommunications engineering and the Ph.D. degree in electrical engineering from the University of Aveiro, Aveiro, Portugal, in 1988 and 1999, respectively.

He is an associate professor in the Electrical and Computer Engineering Department at the University of Porto, Portugal, and a senior researcher in the Institute of Electronics and Telematics Engineering of Aveiro. Formerly, he was a Design Engineer in a company producing digital telecommunications equipment. His research interests include real-time networks for distributed industrial and embedded systems and control architectures for mobile robots

Dr. Almeida is a Member of the IEEE Computer Society.