First quantitative results of the dependability improvement achieved by ReCANcentrate

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

Luís Almeida
IEETA / DEEC-FEUP
Universidade do Porto, Portugal
lda@fe.up.pt

Abstract

There is a growing interest in using star topologies instead of buses as the communication infrastructure for highly-reliable distributed control systems, given the better dependability stars are supposed to provide. For the Controller Area Network (CAN), we developed a simplex and a replicated star called CANcentrate and ReCANcentrate respectively. In a previous work we modelled the dependability of the CAN bus and CANcentrate using Stochastic Activity Networks (SANs). There we presented the first quantitative analysis of the error-containment benefits of a simplex star when considering permanent hardware faults. This paper quantitatively analyzes, for the first time, how a replicated star such as ReCANcentrate can improve both error-containment and reliability, also considering permanent hardware faults. We explain our modelling strategy using SANs and show some first and novel results.

1 Introduction

Controller Area Network [1] (CAN) is a field bus communication protocol widely used in distributed control systems, mainly due to its electrical robustness, low cost and good real-time properties. Nevertheless, CAN presents some shortcomings that discourage its use in highly-reliable applications, e.g. x-by-wire control systems. One of its major limitations is that it relies on a non-redundant bus topology with scarce error-containment and fault-tolerance mechanisms. To overcome this limitation, we developed two CAN-compliant star topologies called CANcentrate and ReCANcentrate [2]. The first one is a simplex star aimed at improving error containment. Its active hub includes novel mechanisms to contain errors at their ports of origin. The second one, ReCANcentrate, is a replicated star that includes two hubs, similar to the one of CANcentrate, to further provide fault tolerance.

Given the stars potential dependability benefits [3], the interest in using them instead of buses has also been growing in other technologies, e.g. in in-vehicle systems, such as with TTP/C [3] and FlexRay [4]. However, despite this interest, it is not a priori evident that stars are more dependable than buses: stars include more hardware, thereby increasing the probability that faults and errors occur.

For this reason, the project within which this work is included aims at modelling different bus and star networks to quantitatively asses the improvement of dependability stars can achieve. In a previous work [5] we quantified the improvement of dependability achieved when using CANcentrate instead of CAN when permanent hardware faults can occur. Results quantitatively corroborate the error-containment benefits that a simplex star was supposed to yield; but they also show that it slightly reduces the communication subsystem reliability.

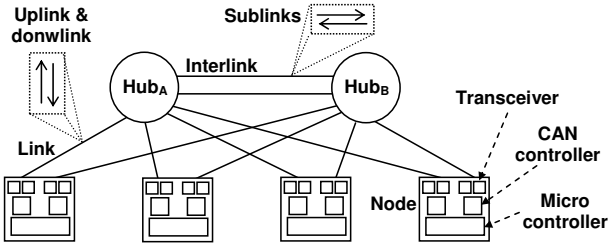

Figure 1. ReCANcentrate architecture

The current paper goes further than [5] and presents the work we are carrying out to quantitatively evaluate, for the first time, how a replicated star topology such as ReCANcentrate can improve not only error containment, but also reliability when considering the possibility of permanent hardware faults.

To our best knowledge, before this paper and [5], the enhancement of dependability that can be achieved when using a replicated and a simplex star instead of a bus had never been appropriately quantified. Even fault injection experiments that quantitatively demonstrate that stars are better suited to prevent error propagation than a bus, e.g. [6], do not clarify if this better error containment actually implies a dependability improvement. Moreover, this paper also shows, for the first time, a quantitative comparison between the dependability that can be achieved with a replicated and a simplex star topology.

2 ReCANcentrate basics

CAN relies on two fundamental properties: the *dominant/recessive transmission* and the *in-bit response* [1]. The former means that the medium implements a wired-AND function of all nodes' contributions, so that a dominant bit '0' prevails over a recessive bit '1'. The second one guarantees that nodes quasi-simultaneously observe every single bit on the channel.

These properties are kept in ReCANcentrate [2], whose general architecture is sketched in Figure 1. ReCANcentrate includes two hubs and each node is connected to each of them by a dedicated link containing an uplink and a

 doi:10.1109/ETFA.2009.5347228

downlink. Additionally, both hubs are interconnected by at least two interlinks each of which contains two independent sublinks, one for each direction. Each hub receives the contribution to each bit value of each node directly attached to it through the corresponding uplink. It couples all non-faulty node contributions with a logical AND function to calculate what we call the *hub contribution*. Each hub sends to the other one its own contribution, so that the resulting signal that each hub broadcasts to its own nodes is the one that results from coupling its own contribution with the contribution received from the other hub. Hubs perform this coupling within a fraction of the bit time, thereby creating a single logical broadcast domain since both hubs behave like one, transmitting the same value bit by bit in their downlinks.

Each node is constituted by commercial-off-the-shelf components only: two CAN controllers, four transceivers and a microcontroller (Figure 1). Each CAN controller is connected to one hub, only, using a transceiver for the uplink and another one for the downlink. The single broadcast domain allows nodes to easily manage the replicated traffic [7]. Basically, each node transmits through one hub, only, while receiving from both hubs simultaneously.

Regarding the fault model, each hub is able to detect faults at nodes, links, interlinks or at the other hub that manifest as stuck-at-recessive, stuck-at-dominant or bit-flipping streams [2]. Each hub disables any permanently faulty contribution, thus isolating it at its port of origin. Moreover, each node can diagnose when a fault, e.g. in a link or a hub, prevents it from communicating through a given star. For that it basically uses the *Transmission Error Counter* (TEC) and the *Reception Error Counter* (REC) [1] of its CAN controllers. Whenever any of these counters reaches a given threshold, the node stops using the corresponding CAN controller for communicating.

3 Modelling rationale

3.1 Metrics and modelling formalism

Among the different dependability properties, we are interested in error-containment and reliability. To compare the error containment, we evaluate the capacity of each network for minimizing the number of nodes that are prevented from communicating when a fault occurs. For that, we measure the probability with which at least $N-k$ of N nodes can communicate among them throughout time. We refer to this as the *probability of not suffering a k-severe failure* (PNS). A high PNS is specially relevant for systems that can tolerate that up to k of N nodes cannot communicate. Particularly, we are interested in the PNS when $k = 1$. Concerning reliability, we measure the probability with which all N nodes can continuously communicate with each other over time.

Regarding the modelling formalism, we used a stochastic extension to Petri Nets called *Stochastic Activity Networks* (SANs) [8], following a strategy similar to [9]. Once specified, a whole SANs model can be automatically transformed into a Markov Chain that is analytically solved. In particular, we used the Moebius software [8] to build and analytically solve all our models.

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 modelled system. An activity is connected to one or more source places and has one or more *cases*, each one connected to one or more destiny places. Each activity *fires* in accordance with a given statistical distribution to change the marking of places, thereby modelling the system transitions through different states, e.g. the failure of a given component. An input gate defines a condition for an activity to fire, which depends on the marking of its source places. This gate also specifies how to change the marking of the source places when the activity fires. When firing, an activity also selects one of its cases to change the marking of specific destiny places. An output gate is connected to a given case and specifies the set of marking changes to be performed in accordance with some conditions.

3.2 Modelling assumptions

Next, we summarize the main assumptions our models rely on. Most assumptions are reflected as model parameters, thereby allowing to perform sensitivity analysis with respect to them. Anyway, so far all assumptions are made favoring whenever possible the CAN bus and always guaranteeing that results are not biased toward stars.

Concerning the networks' layout, we assume the length of each link (and interlink) to be half the total CAN bus length. This is pessimistic for the stars, since a star could use much less cable to cover the area occupied by the nodes the bus interconnects. Additionally, we choose a daisy chain configuration as the way to attach nodes to the bus. This is the most optimistic case for CAN, since it needs no stubs and the least number of connectors.

Regarding what are the components that constitute the networks, we consider the following ones: microcontrollers, CAN controllers, transceivers, memory ICs, oscillators, PCBs, segments of cable, connectors, network terminations, and ASICs (in the case of the hubs).

Concerning component failures, we suppose that they are independent. The *Time To Failure* distribution, F(t), of each component is considered to be exponential with mean $1/\lambda$, where λ is the failure rate expressed in number of failures per hour. More specifically, we assume that F(t) is Non-Defective [9]. If the failure time is X and F(t) is Non-Defective, then the probability with which the component fails at or before time t, F(t) = Prob($X \leq t$), is 0 when $t = 0$, $1 - exp(-\lambda * t)$ when $0 < t < \infty$, and 1 when $t = \infty$. To calculate each component failure rate, we use a software based on the MIL-HDBK-217 standard prediction model [10].

We assume that a component failure can basically manifest, from the channel point of view, in the following modes: stuck-at-recessive, stuck-at-dominant or bit-flipping [2]. Additionally, we consider that a component can also provoke a failure mode we call an *out-of-fault-model* failure. This mode gathers all faults that are be-

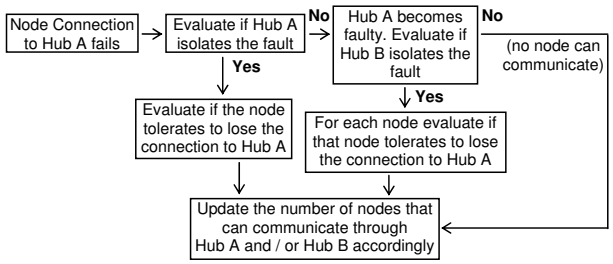

Figure 2. Schema of how to model an example of error propagation and fault treatment

yond our fault model and that, thus, cannot be treated. Since there is not a real consensus on the failure mode proportions of components, we initially assume a 5% of *out-of-fault-model* failures and the rest of failure modes as equiprobable; which can be considered as reasonable [11].

Finally, we also model two coverages. On the one hand, the *error-containment coverage*, which can be defined as the probability of detecting and isolating a fault included in our fault model, given that this fault occurs. In fact, we consider different error-containment coverages: the one related to the capacity of each CAN controller to isolate faults affecting its node; the coverage with which each CANcentrate and ReCANcentrate hub isolates faults at its uplink ports; and the coverage with which each ReCANcentrate hub isolates faults at its interlink ports. On the other hand, only for the case of ReCANcentrate, we model what we call the *fault-tolerance coverage* of the node. When a fault prevents a node from using one star, this coverage is the probability with which the node can continue communicating using the other star.

3.3 Modelling strategy

In order to keep a reasonable model complexity, we do not model single components, but groups of them we call *network parts*. We differentiate the following types of network parts: (1) the *Node Core*, which basically includes the node microcontroller, memory ICs and sockets; (2) the *Node Connection*, which comprises all the components a node needs to be connected to the bus line or to a given hub, i.e. one CAN controller and one transceiver in the case of the bus or, in the case of the stars, one CAN controller, two node transceivers and all cables, connectors, terminations and the hub transceivers corresponding to the link; (3) the *Hub Interconnection*, which includes the cables, connectors, terminations and hub transceivers of a given sublink of ReCANcentrate; (4) the *Bus Section*, which contains the cable and connectors that constitute the section that connects two adjacent nodes in a daisy chained CAN bus; and (5) the *Hub Core*, which includes all the hub components except its transceivers.

We model each one of the networks, i.e. CAN, CANcentrate and ReCANcentrate, as a composition of two types of submodels, we call *parts submodel* and *coverage submodel*. A *parts submodel* represents all network parts of a given type, e.g. all *Node Connections*. Each *coverage submodel* corresponds to a specific fault-treatment or fault-tolerance mechanism and models whether or not this mechanism isolates or tolerates a given fault.

A *parts submodel* basically has a place whose marking represents the number of parts that are not faulty; an activity that models the occurrence of a fault in any of those parts; places that represent how the fault manifests; and some places it shares with a given *coverage submodel* to indicate to it when a part has failed. When a fault occurs in a given part, the corresponding *parts submodel* decides the failure mode with which the fault manifests and compels *coverage submodels* to carry out a sequential process that models how the errors generated by that fault propagate, how they are contained and, if appropriate, how the fault is tolerated. Depending on whether or not the fault is successfully isolated and/or tolerated, *coverage submodels* update a set of places that represent the number of nodes that can communicate among them. Figure 2 summarizes, as an example, how this process is carried out when a *Node Connection* part fails in a ReCANcentrate network where hubs are called *Hub A* and *Hub B*.

4 First dependability results

Figures 3 and 4 compare the reliability and the probability of not suffering a k-severe failure (PNS) with $k = 1$ in the CAN bus, CANcentrate and ReCANcentrate. Specifically, they show the *mission time* achieved by these networks, which here is understood as the time of operation during which they present a reliability or a PNS ≥ 0.99999 (0.99999 is the value of reliability required for a throttle-by-wire control system during 10 hours [12]).

Some dependability parameters are: coverages of 95%; and failure rates (failures/hour) of 3.2531245E-6 for the *Node Core*, 1.4044118E-6 for the *Node Connection* in the CAN bus, 2.2617741E-6 for the *Node Connection* in the stars, 1.3092281E-6 for a ReCANcentrate *Hub Core* with 4 ports, and 2.1193189E-6 for a ReCANcentrate *Hub Core* with 20 ports. Note that most parameters reflect modelling assumptions that may have been too detrimental for CANcentrate and ReCANcentrate, so that results are likely to be lower bounds to stars dependability.

Figure 3 shows that given a fixed number of nodes, ReCANcentrante is more reliable than the CAN bus and CANcentrate. With 4 nodes they respectively achieve mission times of 0.68, 0.53 and 0.43 hours; whereas with 20 nodes the mission times are 0.13, 0.10 and 0.08 hours. As expected, the benefits are not outstanding in absolute terms. This is because ReCANcentrate improves the communication subsystem reliability, but here we measure the reliability of the overall system. To reflect the full potential of ReCANcentrate it would be necessary to consider fault-tolerance mechanisms at other parts of the system, i.e. at nodes, which are the least reliable elements. This is in fact what we partially do when measuring the PNS for a specific value of k, since this metric is not only useful to see the gain provided by the error containment capabilities, but it also corresponds to the reliability of a system that is able to tolerate the fault of k nodes.

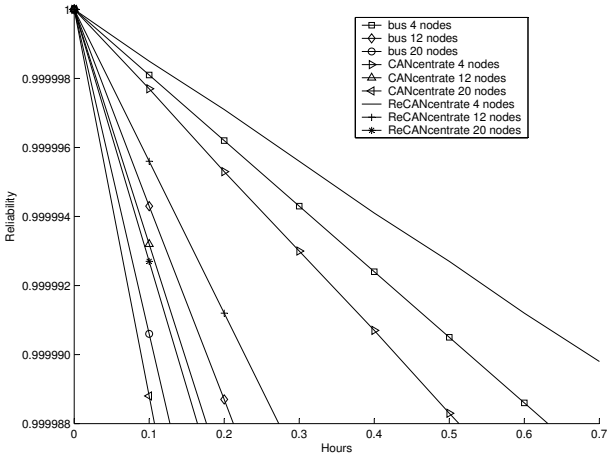

Figure 3. Reliability comparison

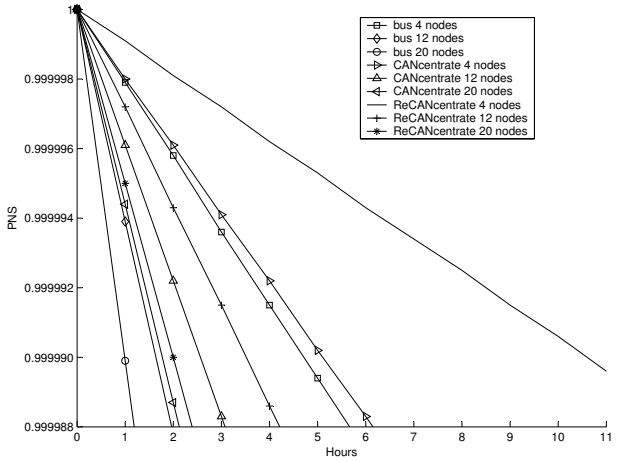

Figure 4. PNS comparison

Figure 4 shows that both stars improve the PNS (with $k = 1$) of CAN. The mission times of the CAN bus, CANcentrate and ReCANcentrate respectively are: 4.7, 5.1 and 10.6 hours with 4 nodes; and 1.0, 1.8 and 2.0 with 20. We also observe that ReCANcentrate improves the mission time of CAN more than CANcentrate does. With 4 nodes, CANcentrate and ReCANcentrate improve the mission time by 8% and 125%, whereas for 20 nodes the improvements are of 80% and 100%. However, it is worth noting that as the number of nodes grows, the mission times of both stars tend to converge. We consider that this is because ReCANcentrate needs more hardware than CANcentrate for a given number of nodes. Thus, as the number of nodes grows, the fault-tolerance mechanisms of ReCANcentrate do not compensate the higher rate with which faults that cannot be isolated, i.e. not-covered and *out-of-fault-model* faults, raise when compared with CANcentrate.

5 Conclusions and future work

This paper presents the last results of our on-going work towards quantitatively assessing the dependability benefits that star topologies can achieve in the field-bus communication domain. Specifically, it focuses on the dependability assessment of a replicated star topology, called ReCANcentrate, we recently proposed for CAN. First results corroborate that the error containment and fault-tolerance capabilities of ReCANcentrate improve the reliability of CAN-based systems that cannot tolerate the failure of any node and, even more, of those that can tolerate the failure of one node. To our best knowledge, this is the first formal (quantitative) comparison between a bus and a replicated star. In the short term we will analyze the sensitivity of the benefits of ReCANcentrate with respect to dependability parameters such as the different coverages, the proportion of *out-of-fault-model* faults, and the components' failure rates.

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

[2] M. Barranco, J. Proenza, and L. Almeida, "Boosting the Robustness of Controller Area Networks: CANcentrate and ReCANcentrate", *IEEE Computer*, vol. 42, no. 3, pp. 66–77, May 2009.

[3] H. Kopetz, "Time-Triggered Protocols for Safety-Critical Applications", Presentation, March 2003.

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

[5] 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.

[6] 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.

[7] M. Barranco, J. Proenza, and L. Almeida, "Designing and Verifying Media Management in ReCANcentrate", in *WFCS'08. IEEE Workshop on Factory Communication Systems, Dresden, Germany*, 2008.

[8] W. Sanders and T. B. of Trustees, "Moebius User Manual Version 1.6.0", 2004.

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

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

[11] M. Barranco, "Improving Error Containment of Controller Area Network (CAN) by means of Adequate Star Topologies", Official preliminary PhD thesis, Departament de Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, Spain, November 2008.

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