Dependability of Distributed Control System Fault Tolerant Units

Juan R. Pimentel
Department of Electrical and Computer Engineering
Kettering University
Flint, Michigan 48504
jpimente@kettering.edu

Mario Salazar
Department of Electrical Engineering
Universidad de los Andes
Bogota, Colombia

Abstract –We investigate two types of fault tolerant units (FTU's) suitable for dependable distributed control systems and numerically evaluate their reliability and mean time to failure (MTTF). A simple simulation-based methodology to numerically evaluate dependability functions of a wide variety of fault tolerant units is presented. The method is based on simulation of stochastic Petri Nets. A set of 15 FTU configurations belonging to five groups is analyzed. Groups 1 and 2 belong to the node oriented category whereas groups 3 through 5 belong to the application oriented category. The methodology allows a quick and accurate evaluation of dependability functions of any distributed control system design in terms of the type of FTU (i.e., node or application), replicas per group, replicas per FTU, and shared replicas.

Keywords: Distributed control systems, fault tolerance, dependability, real-time systems, reliability, simulation, stochastic Petri-Nets.

I. INTRODUCTION

Distributed control systems are becoming important because of their potential to effectively support the implementation of complex control systems with stringent requirements involving fault tolerance and flexibility that will be of prevalence in the future. One example of this kind of systems are embedded control systems such as those used in automotive control for applications such as drive-by-wire steer-by-wire, brake-by-wire, or in general X-by-wire [1]. Distributed control systems support several applications with specific services in a high performance fashion. The advantages of distributed control systems have been documented in a number of sources. For example, the advantages of using advanced digital control in automoviles include increased stability and safety of the vehicle, improved fuel efficiency, reduced pollution, more confortable vehicle operation, increased performance, etc. [5].

Because of their unique requirements, distributed control systems use specific real time networks and protocols with the main examples being CAN (controller area network) [3], TTP/C [4,5] (time triggered protocol), and FlexRay [9]. Whereas CAN uses an event triggered medium access mechanism, TTP/C uses a time-triggered mechanism. The medium access mechanism used by FlexRay is mixed, time-triggered and event triggered. In the above protocols, the time-triggered mechanisms used are based on the TDMA (time division multiple access) principle. Perhaps the main advantage of the TDMA scheme is its deterministic behavior making it possible to perform exact calculations of message delay, control system period, worst case execution delay, jitter, etc.

The design and implementation of networks and protocols that support distributed control systems is different from the design and implementation of other non-real time networks (e.g., Internet based) because of the unique characteristics of the intended applications. A good design must start with special attention to the specific application requirements to produce a set of unambiguous design specifications. Two important requirements identified as important for distributed control systems are dependability and flexibility. Dependability typically involves availability, reliability, safety, and security. In this paper we are concerned with reliability evaluation of a set of fault tolerant units defined from the viewpoint of applications. Carvalho and Portugal [11] have studied the dependability of fieldbus networks in general without regard to FTU's using Markov chains. In the above referenced paper, the entire network is considered as the system under study whereas in this paper we consider applications as the system of interest.

The motivation for our study is the desire to develop a relatively simple tool to evaluate the dependability attributes of various fault tolerant designs of distributed control systems in a relatively easy fashion. The goal of this research paper is to study the relationships between the level of reliability obtained in terms of the degree of fault tolerance offered by various fault tolerant configurations.

II. DISTRIBUTED CONTROL SYSTEMS

As the name implies, distributed control systems involve a set of control systems implemented in a distributed fashion using an appropriate communication protocol. A real time distributed control system is a control system whose functions are partitioned into separate modules each implemented in a distributed fashion on separate nodes interconnected by a real-time computer network. Each node in the computer network must perform computations in a bounded amount of time to meet sampling constraints of the control system. The computing system and the communication network has a number of features designed to meet the requirements and constraints of the control system.

A control system is characterized by one or more feedback control loops, and associated control algorithms, sensors, and actuators. The various controllers, sensors, and actuators can reside in different nodes of the communication network and communicate among one another using the network services. Thus in a distributed control system the applications involve controllers, sensing functions, actuation functions, and communication functions in the context of control loops.

We characterize a distributed control system as one having:

Multiple control loops

Wide range of activation frequencies or control periods for each loop

Multiple sensors and actuators

Variables of small size

Each control loop is characterized by a set of controllers, a set of reference signals, a set of output signals, and a set of state signals. The controllers are implemented in the host portion of the nodes of the communication system. The various signals are encoded as messages for transmission on the network. It is further assumed that the control systems have stable and high performing control algorithms organized in modules, implemented as tasks with worst case execution times to meet stringent requirements involving determinist, dependable, and flexible behavior. In a traditional control system the functions of sensing, control computation, and actuation are strictly sequential. In a distributed control system such functions can be performed in parallel or in an overlap fashion thus introducing a sysnchronization problem that must be properly addressed at the design phase. Perhaps the main impact on the use of a communication system within a control system is the need of such synchronization mechanism so that the functions of sensing, control computation, and actuation are performed and communicated to the physical locations they are needed in a sequence and fashion that is in accordance to control system principles.

One of the main advantages of distributed control systems is the potential to operate even under the presence of some faults through the use of redundant units configured as special fault tolerant units (FTU). This feature is of tremendous advantage for applications requiring a high level of safety and reliability (e.g., passenger cars, airplanes, etc.)

III. DEPENDABILITY ISSUES

The notion of dependability involves the reliance of a system on the quality of services it provides during an extended interval of time [4]. Important attributes of dependability include availability, reliability, and maintainability. **Availability** is the probability of finding the system in the operating state at some time into the future, **reliability** is the probability of a system staying in the operating state without failure, and **maintainability** is the probability of beign able to repair a system. In the following we provide more precise definitions of the dependability attributes of availability, reliability, and maintainability.

Let S = $\{s_i,\ i \in I\}$ be the set of all possible states of a system. We can divide S into two disjoint subsets S_s and S_f where S_s denotes a subset of states where the system is operating sucessfully and S_f where the system has failed. Thus

$$S = S_s \cup S_f$$

$$S_s = \{ s_i,\ i \in I_s\}$$, and

$$S_f = \{ s_i,\ i \in I_f\}$$

Fig. 1 depicts the previous definitions. Based on the previous definitions we are now in a position to provide precise definitions of the various dependability concepts.

Availability:
$$A(t) = \Pr\{s(t) \in S_s \,/\, s(\tau) \in S,\ \tau \in [0,t>\}$$

Reliability:
$$R(t) = \Pr\{s(t) \in S_s \,/\, s(\tau) \in S_s,\ \tau \in [0,t>\}$$

Maintainability:
$$M(t) = \Pr\{s(t) \in S_s \,/\, s(\tau) \in S_f,\ \tau \in [0,t>\}$$

MTTF (Mean time to failure):
$$\text{MTTF} = \int_0^\infty R(t)dt$$

MTTR (Mean time to repair):
$$\text{MTTR} = \int_0^\infty [1 - M(t)]dt$$

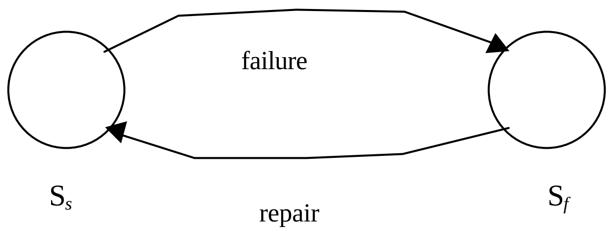

Fig. 1. Failure states (S_f) and success states (S_s) for dependability calculations.

A. Redundancy for fault tolerance.

Most approaches to fault tolerance rely on extra elements introduced to detect and recover from faults.These elements are *redundant* in that they are not strictly required for the system to operate. Care must be taken when introducing redundant components to ensure that they do not lead to a *less* reliable system. A fault tolerant system typically go through the following phases: error detection, damage assessment and confinement, error recovery, and fault treatment. No fault tolerant scheme can start to operate until the fault has manifested itself as an error, which can subsequently be detected, thus the need for *error detection*. Once an error has been detected, the consequences of that error must be assessed through *damage assessment and confinement*; the longer the time delay between occurrence and detection, the greater the possibility of system corruption. The aim of *error recovery* is to transform the system into a state where it can continue to provide full, or degraded, functionality. Finally under *fault treatment*, errors are viewed as the symptoms of faults that unless the cause is treated, errors may be repeated.

Depending upon the way redundant components are configured or reconfigured there are two types of redundancy, *static* and *dynamic* redundancy. The scheme *static or masking redundancy* is also called **active replication**. Under *static redundancy* several replicas of a node are active (i.e., they are fully operational) simultaneusly. The node and its replicas operate in parallel and their outputs are compared. Redundancy is used to hide errors and no type of reconfiguration of redundant components is necessary; the redundant components operate even when no error exists. Each replica receives a request from the client and simultaneously produce its result and send it to the client which can then vote on the received results. If replicas are fail-silent, then any result can be used (e.g., the first received). The main advantage is its fast response time and its principal disadvantages are that it requires much processing, communication and other resources, and requires some level of replica determinism [2]. Some examples include TMR (triple modular redundancy) and N-version programming.

The scheme *dynamic redundancy* is also called **passive replication**. Under dynamic redundancy a software module is designed to provide detection of errors. It is up to another module to recover from the errors when detected. The recovery procedure involves some kind of reconfiguration and the redundant components are only invoked when an error is detected. Thus a system with dynamic redundancy must be able to recover (through reconfiguration) from a detected error and in the worst case even restart the system. A system which uses recovery will often save its state at specific points and when an error is detected the system will rollback to a saved state. Typically one processor is the primary and executes all service requests while other processors serve as passive backups. In case of failure of the primary, one of the backups will be chosen to become the new primary. This process is repeated in case there are additional backups. If available, a *membership service*[5] could be advantageous for implementing dynamic redundancy schemes. The main advantages of this scheme are that it consumes less resources when compared with active replication and *replica determinism* is not necessary [2]. Their disadvantages are that the state of the primary must be distributed among the replicas continuously, backups must be able to recover their state, and that the primary must be fail-silent.

B. Fault Tolerant Units.

As the name implies, a FTU is a unit that continues to operate even in the presence of some faults. The primary mechanism used by a fault tolerant unit is replication of software, hardware, information, and time. In this paper, we consider only dynamic redundancy (i.e., passive replication) where a normally working node is considered to be a primary node and the redundant nodes are called the secondary nodes. We use the terms redundant node, secondary node, replica, or backup as meaning the same. In addition we are primarily concerned with hardware replication at the node level in the context of a communication network. In this context, we can distinguish two types of FTU's, *node oriented* and *application oriented*. The unit of replication in a node oriented FTU are nodes that are independent from applications. When the primary node fails only nodes that are designated replicas of the primary nodes will take over. A mechanism is needed to allow just one replicated unit to take over to avoid collisions or contention. Thus, the secondary nodes are assumed to have similar or identical functionality of the node they intend to replace. The unit of replication in an application oriented FTU are also nodes but unlike node oriented FTU's, they belong to a common pool of redundant nodes defined for a specific application. When the primary node fails only nodes that are designated replicas of the application to which the failed node belongs will take over. Figure 2 depicts four node oriented FTU's where FTU's 1 and 2 share node 14. In summary, in a node oriented FTU, backups are used in a node oriented fashion, they replace specific nodes independent of any application they may belong. In contrast, in application oriented FTU's, backups are used in a application oriented fashion; nodes that belong ot the same application share a common set of backups. The main advantage of application oriented FTU's is that they save real estate (i.e., backups). Their disadvantages are that they require enough flexibility of the backups to replace any node in an application and a fault management scheme is required in case there are more than one backup [10].

C. FTU Configurations.

Table I shows 5 groups of 15 FTU configurations with each group having 3 FTU's denoted by A, B, and C. We assume that all FTU's in the same group belong to a common application. Groups 1 and 2 contain FTU's that belong to the node oriented category whereas groups 3 through 5 contain FTU's that belong to the application oriented category. For each FTU, the node with an asterisk denotes the primary node whereas the remaining nodes are secondary (i.e., backup or replica) nodes. For example, for FTU A of group 1, node 11 is the first backup and node 12 is the second backup. The main thing to notice between the node oriented category and application oriented category is that while the former may share some replicas, the latter have a common set of replicas shared by all FTU's in the group. One can see that the total number of replicas in group 1 is 6 (two per each FTU) whereas the total number of replicas in group 2 is 3, a saving of three replicas when compared with group 1. In summary, table I shows 15 FTU's belonging to 5 groups (i.e., applications) and 3 FTU categories; the number of replicas per group vary between 1 and 6, the number of replicas per FTU vary between 1 and 3 and the number of shared replicas vary between 0 and 3. It can be noticed that there is a significant saving in the total number of replicas per group of application oriented FTU's (i.e., groups 3, 4, and 5) when compared to node oriented FTU's (i.e., groups 1 and 2).

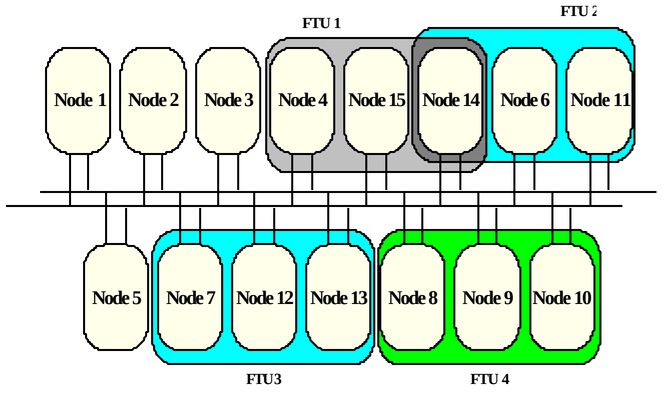

Fig. 2. Node oriented FTU's.

Table I. Distribution of nodes per groups and FTU's.

G	**FTU A**	**FTU B**	**FTU C**	**Rpl per group**	**Rpl per FTU**	**Shared Rpl**
1	5*, 11, 12	6*, 13, 14	7*, 15, 16	6	2	None
2	5*, 11, 12	6*, 11, 13	7*, 12, 13	3	2	11, 12, 13
3	5*, 11, 12, 13	6*, 11, 12, 13	7*, 11, 12, 13	3	3	11, 12, 13
4	5*, 11, 12	6*, 11, 12	7*, 11, 12	2	2	11, 12
5	5*, 11	6*, 11	7*, 11	1	1	11

(*) Denotes primary node within an FTU. G: Group, Rpl: Replica.

D. Dependability performance measures.

In this paper we are interested in the relative comparison of reliability functions of FTU configurations with various degrees of fault tolerance. The degree of fault tolerance will be determined by the number of backups (i.e., replicas), whether backups are shared, and how backups are shared (node oriented or application oriented).

For comparison reasons we review the fundamental results of the reliability and availability functions for the simplest case, an FTU with no replicas with and without repair [8].

We assume a single node per FTU with an exponential failure density function given by

$$f(t) = ?e^{-?t}$$

The reliability function R(t) for such a single node FTU without repair is

$$R(t) = e^{-?t}$$

and the MTTF = 1/?.

IV. MODELS AND EXPERIMENTS

We have chosen stochastic Petri-Nets (SPN) as the mathematical framework to evaluate the reliability functions. The advantages of SPN for performance and reliability analysis of communication networks have been widely documented in the literature [6,7]. In addition, we have chosen a SPN simulator as the computational tool to evaluate the reliability functions. The main advantages of such tool is that it supports modeling a wide variety of probability distribution functions and Petri Net extensions (e.g., the test arc) and that the solution is obtained through simulation thus allowing the analytical evaluation of any model regardless of its complexity.

As noted, the reliability function of an FTU is the probability that the FTU continues to operate given that it is operational. The event denoting that an FTU continues to operate is the same as the event that the primary node of the FTU or any of its backups continue to operate, according to the error detection and recovery scheme of the fault tolerant mechanism in question. We have configured a generic Petri-Net model that follows the structure of Fig. 3 where a token in the FTU-test place represents the initial operating state of an FTU. The Petri-Net model has been designed to simulate an experiment to observe the behavior of the system. Although the distribution of tokens in the model (i.e., its marking) represent the state of the primary node and all of its backups, we are only interested in two behavioral states indicating whether the FTU under test has failed or not. Depending upon the distribution of node failures, and the nature of the fault tolerant mechanism, after a certain simulated time interval T, the FTU may find itself in either of two states, a success state (the FTU continues to operate) or a failure state (the FTU is not able to provide its services). The interval T is controlled by the timing control block of Fig. 3 and the experiment is repeated N times. In each trial of the experiment, one FTU is tested (one token leaves the FTU-test place of Fig. 3) and either of two events happen; it continues to operate (one token is added to place Ns) or it fails (one token is added to place Nf). After N trials of the experiment, the number of tokes in place Ns correspond to the number of times the FTU under study did not fail (i.e., a success) and the number of tokes in place Nf correspond to the number of times the FTU failed. A reliability value for a fixed value of T is simply the probability of finding the system in place Ns after a simulated time of T. The reliability function over time is obtained by varying T.

A. Nodes without repair

This model category corresponds to a fault tolerant mechanism that involves an error detection and recovery scheme consisting of the primary node and a number of secondary nodes where none of the nodes can be repaired after they fail. The degree of fault tolerance offered depends on the number of backups (i.e., replicas), whether the backups are shared, and how backups are shared (node oriented or application oriented). As noted, the reliability function of an FTU is the probability that the FTU continues to operate given that it is operational. For the models in this category, the event denoting that an FTU continues to operate is the same as the event that the primary node of the FTU or any of its backups are operating subject to the constraint that some (if any) backup nodes may be shared with other FTU's as shown in Table I.

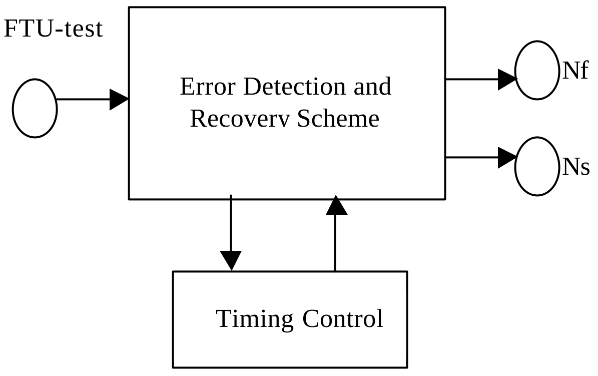

Fig. 3. Structure of a generic Petri Net model for reliability calculations.

The Petri Net model for FTU A of group 1 is depicted in Fig. 4 and corresponds to an FTU with one primary node and two backup nodes. The places on the top of the figure represent states indicating when the primary node is active and when it fails, when the first backup is active and when it fails, and when the second backup is active and when it fails. The transitions simulating failures follow exponential distributions with mean node failure rate of 1 failure per 10^5 hours (i.e., 1 failure per 11.446 years). The transitions in black represent instantaneous transitions. The number of tokens in the places Nf and Ns on the far right represent the number of failures and successes in N trials of the experiment where N is the initial number of tokens in place FTU-test. Places P15 and P16 along with transitions T21 and T22 represents the timer control block and model a timer that regulates a time window that controls the length of the experiment. The model is independent of the actual protocol used (e.g., CAN, TTP/C, or FlexRay) because of the relative high values of mean time to failures relative to the timing parameters of any specific protocol (e.g., TDMA slot time of TTP/C).

V. RESULTS

The results of the experiments are a set of reliability functions and the mean time to failure (MTTF) for all FTU configurations. Because of the symmetry of the FTU configurations defined in Table I, the reliability of any of the 3 FTU's per group (i.e., A, B, or C) are the same. Thus it is only necessary to show five reliability functions corresponding to each of the five FTU groups of Table I. The reliability functions of FTU's in all groups are depicted in Fig. 5, whereas their corresponding MTTF are listed in Table II. As expected, it can be noticed that the reliability functions of FTU's with a higher number of backups decay more slowly when compared with FTU's with a smaller number of backups. Likewise, the MTTF of FTU's with a certain number of replicas are longer when compared to the MTTF of FTU's with a smaller number of replicas.

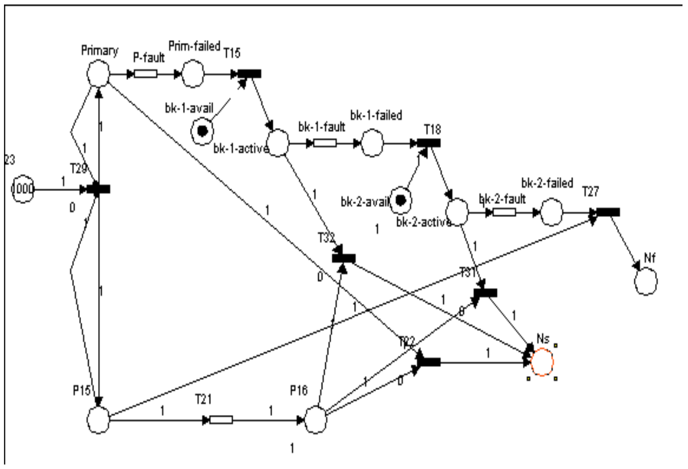

Fig. 4. Petri Net model corresponding to FTU A, group 1.

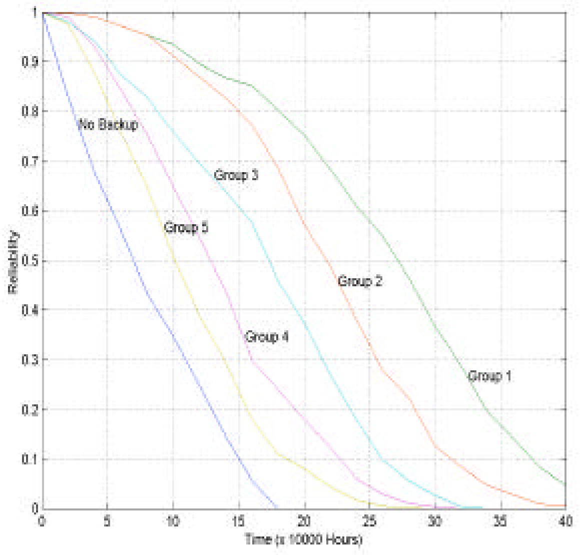

Fig. 5. Reliability functions for FTU's with no repair.

VI. SUMMARY AND CONCLUSIONS

One of the main advantages of distributed control systems is the potential to operate even under the presence of some faults through the use of redundant units configured as special fault tolerant units (FTU). Fault tolerant units are effective means to improve the dependability of distributed control systems. A simple method to numerically evaluate dependability functions of a wide variety of fault tolerant units has been developed. The method is based on simulation of timed Petri Nets with extensions such as the test arc. A set of 15 FTU configurations belonging to two categories has been analyzed. The degree of fault tolerance is determined by the number of backups (i.e., replicas), whether the replicas are shared, and how replicas are shared in each category (node oriented or application oriented). The reliability functions improve as the number of backups increase. The reliability of FTU's that share backups do not vary significantly when compared to the reliability of FTU's that do not share backups. The reliability of FTU's in groups that have a common set of replicas shared by all FTU's in the group do not vary significantly when compared to the reliability of FTU's having replicas that are not shared. The main advantage of FTU's that share replicas is that real estate is saved (i.e., less replicas are used) while the level of dependability is about the same when compared with FTU's that do not share replicas. The methodology presented in this paper allows the evaluation of dependability functions of any distributed control system design to be performed quickly and accurately.

Table II. MTTF of Petri Net models.

FTU	**MTTF, nodes without repair**
Group 1	2.59x 10^5 hours
Group 2	2.15 x 10^5 hours
Group 3	1.66 x 10^5 hours
Group 4	1.32 x 10^5 hours
Group 5	1.07 x 10^5 hours
No backups	0.76 x 10^5 hours

VII. ACKNOWLEDGEMENTS

The first author wishes to thank the Fulbright Commission of the U.S. and Colombia for funding the grant under which this research was performed. Additional thanks go to the Department of Electrical Engineering at the Universidad de los Andes, Bogota, Colombia where the research was performed.

VIII. REFERENCES

[1] Bretz, E.A., "By-Wire Cars Turn the Corner," IEEE Spectrum, Vol. 38, No. 4, pp.68-73, April 2001.

[2] Poledna S., *Fault-Tolerant Real-Time Systems*, Kluwer Academic Publishers, 1996.

[3] L. Lawrenz, *CAN System Engineering: From Theory to Practical Applications*, Springer, 1997.

[4] H. Kopetz, G. Grunsteidl, "TTP – A protocol for Fault-Tolerant Real-Time Systems". IEEE Computer, pages 14-23, January 1994.

[5] Kopetz H., *Real-Time Systems, Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, 1997.

[6] Molloy, M., "Performance Analysis Using Stochastic Petri Nets," IEEE Trans. On Computers, Vol. C 31, Sept. 1982.

[7] Narahari Y., and Viswanadham N., "Performance Modelling of a Fault-tolerant real-time Multiprocessor Using Stochastic Petri Nets", in *Reliability and Fault-Tolerant Issues in Real-Time Systems*, Proceedings in Engineering Sciences, Vol. 11, pp. 187-208, 1987.

[8] Billinton R., and Allan R. N., *Reliability Evaluation of Engineering Systems, Concepts and Techniques,* 2nd Ed., Plenum Press, New York, 1992.

[9] http://www.flexray-group.com

[10] Pimentel, J.R, and Sacristan, T. "A Fault Management Protocol for TTP/C," Proc. IECON'01 (Int. Conf. On Industrial Electronics), pp. 1800-1805, Denver, CO., Nov. 2001.

[11] Carvalho, A., and Portugal, P., "On Dependability Evaluation of Fieldbus Networks: A Permanent Fault Analysis," Proc. IECON'01 (Int. Conf. On Industrial Electronics), pp. 299-304, Denver, CO., Nov. 2001.