Monograph:
Shallof, A. and Bennett, S. (1994) The Effect of Redundancy and Repair on the Performability of Distributed Real-Time Control Systems with RepetItive Task Invocation. Research Report. ACSE Research Report 500 . Department of Automatic Control and Systems Engineering

The Effect of Redundancy and Repair on The Performability of Distributed Real-Time Control Systems With Repetitive Task Invocation

by

A. Shallof and S. Bennett

Department of Automatic Control and System Engineering

University of Sheffield, Mappin Street, Sheffield 1, 4DU, U.K.

Research Report NO. 500

1994

The Effect of Redundancy and Repair on the Performability of Distributed Real-Time Control Systems With Repetitive Task Invocation

A. Shallof and S. Bennett

Abstract. Distributed real-time control systems depend on producing correct software and proper hardware architecture to support the software. Hardware components degrade with the time and this can lead to hardware failure, which in a distributed real-time control system may be manisfested through the failure of some tasks to meet their deadline. The effect of random failure, repair and redundancy on the completion of tasks of a system by their deadline have been investigated. We show how a system with redundancy and with random failure and repair rates can be modelled using the General Stochasic Petri net approach and how the performability of the system can can be calculated using a method based on the modified Laguerre function.

Keywords : Distributed real-time control systems; modified Laguerre functions; reliability; performability; GSPN.

1. INTRODUCTION

A hard real-time system is one in which every system activity has to be completed before a certain deadline. When such system is implemented as a distributed real-time control system the activities are performed in different nodes, activities that depend on each other and execute on different nodes must communicate via some mechanism. Since the tasks have deadlines it is essential that the messages between nodes are delivered within bounded time and the message protocol must provide appropriate support [1]. Activities are performed by tasks which are executed repetitively at each of the nodes. A task is a software module that can be invoked to perform a particular function. In a hard real-time system a task is characterized by its timing constraints, precedence constraints and resource requirements.

Tasks can be classified into two types [2][3]: non-perodic and perodic. A non-perodic task is one

which when invoked is expected to execute just once and has an arbitrary arrival time and deadline. A perodic task is defined as one which is invoked exactly once per period and constitutes the normal computation for the process under control. Most perodic tasks co-operate with each other through communication to accomplish the overall control mission. The precedence constraints among a set of tasks specify the sequential relationships between the tasks. A task A is said to precede task B if task A must finish before task B begins.

Tasks may be preemptable and non-preemptable. A task is preemptable if its execution can be interrupted by another task at any time and resumed afterwards. A task is non-preemptable if it must run to completion once it starts. Whether a task is preemptable or not is mainly determined by the nature of the application environment. The completed number of tasks depends on the reliability because real-time constraints cannot achieved if system components are not reliabile. Standard reliability metrics do not provide measures of the performance of degradable distributed real-time systems. Instead a combined performance-reliability metric, referred to as performability has to be used. Performability is defined as the probability that a system reaches a certain accomplishment level over an utilization interval called a mission time [4]. Performability differs from reliability in that reliability is a measure of the likelihood that all of the functions are performed correctly whereas performability is a measure of the likelihood that some subset of the functions is performed correctly.

In the literature there is no work which addresses the effect of redundancy and repair on the performability of distributed real-time control systems. In this paper we describe how a particular algorthim [5] can be used to find the effect of repair and redundancy on the performability of a distributed real-time control system. This algorithm is based on the convolution operation between task invocations starting from the last invocation and working backwards to the first invocation. The convolution operation is performed using modified Laguerre functions. In section 2 we describe the performability concept and modelling techinque and in section 3 we show how a particular system can be modelled in section 4 we give some results.

2. Performability Concepts and Modeling Technique

2.1 Performability Concepts

Performability is defined as the probability that a system reaches a certain accomplishment level over an utilization interval called a mission time. The performability model consists of a reliability model and performance model. Relability is defined as the probability that the system stays in an operational state throughout an interval. So it is clear the reliability model represents the

components of the system that are working and it is based on knowledge of component failure and repair rates. The performance of a distributed real-time system depends on producing correct software which matches the specification of the application and on the proper hardware architecture to support the software. Hardware components degrade with time and hence faults can appear through hardware failure. Also because it is virtually impossible to prove the correctness of large complex distributed real-time systems even after the normal extensive testing and debugging phase of the software, additional program errors may surface after the system has been put into operation.

The performance model (reward model) shows how well the the system works with each possible set of operational components. If the reward rates are defined to reflect the performance levels of the system in different configurations then measures such as the expected total amount of work completed in the mission time or the expected throughput of the system with failure and repair can be computed. Thus the performability is a measure of the likelihood that some subset of the function is performed correctly over a specified mission time. From the above it is clear that the performability model consists of a reliability model, a performance model and a way of combining the results of the two models. In this paper we use modified Laguerre functions to find the performability measures [5]. We define the repairable system by a stochastic process $\{ X_t, t \geq 0 \}$ where X_t describes the structure state of the system at time t and $X_t \in \{ 1, 2, \cdots N \}$ and where N is the size of the state space of the process. We assume f_i be the reward rate (or the performance level) associated with the structure state i and the vector f defines the reward structure. The normal system behavior is analyzed by considering the task invocations during one planning cycle [2]. Let random variable $Y(t)$ represent the total accumulated reward in the mission interval t in a distributed real-time control system thus from the general transform equation of performability [6].

$$P_i(x,t) = prob[\, Y(t) \leq x \,|\, X_0 = i \,], \quad i = 1, ..., N \tag{1}$$

and

$$p^{+*}(\gamma, \delta) = (\delta I + \gamma F - Q)^{-1} e \tag{2}$$

where $p^{+*}(\gamma, \delta)$ is the Laplace-Stieltjes transform (shown by +) of $P_i(x,t)$ in x variable and Laplace transform (shown by *) in t variable, F is the a diagonal matrix with entries f_i representing the reward structure; Q is the Markov generator matrix of the reward model , I is the identity matrix and e is the column vector of all ones.

Proposition [5] : Let $p^{+*}(\gamma,\delta)$ be the column vector representing the double Laplace transform of the performability then

(i) the performability density is given by

$$p(x,t)=e^{ct}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}p_{ij}\hat{l}_{ij}(x,t) \tag{3}$$

where

$$p_{ij}=R_{ij}D \tag{4}$$

and

$$R_{ij}=\left[R_{i-1,j}\,A+R_{i,j-1}\,B+R_{i-1,j-1}\,C\right] \tag{5}$$

and the $N\ X\ N$ matrices A, B, C and $N\ X\ 1$ matrix D as the follwing:

$$\begin{aligned}
S &= \left\{\left\{KI+KF-2Q+2KCI\right\}\right\}^{-1}\\
A &= \left\{K(I-F)-2(Q-KCI)\right\}S\\
B &= \left\{K(-I+F)-2(Q-KCI)\right\}S\\
C &= \left\{K(I+F)+2(Q-KCI)\right\}S\\
D &= 2K^2Se
\end{aligned} \tag{6}$$

(ii) the performability distribution is given by

$$P(x,t)=1-(e^{ct}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}P_{ij}\hat{l}_{ij}(x,t)) \tag{7}$$

where

$$P_{i,j}=2K\sum_{m=i+1}^{\infty}\sum_{n=0}^{j}(-1)^{i+m}p^{\#}{}_{mn} \tag{8}$$

$$p^{\#}{}_{i,j}=\begin{cases}\dfrac{1}{K^2}(p_{i,j}-p_{i-1,j}) & i>0\ ,\ j=0\\ \dfrac{1}{K^2}(p_{i,j}-p_{i,j-1}) & i=0\ ,\ j>0\\ \dfrac{1}{K^2}(p_{i,j}-p_{i-1,j}-p_{i,j-1}+p_{i-1,j-1}) & i>0\ ,\ j>0\end{cases} \tag{9}$$

furthermore

$$p^{\#}{}_i = K \sum_{j=0}^{\infty} \hat{l}_j(t) \sum_{n=0}^{j} p^{\#}{}_i \; n \tag{10}$$

$$p^{\#}{}_{ij} = \sum_{k=0}^{i} g^{\#}{}_{kj} a^{\#}{}_{i-k} \tag{11}$$

$$i \; \hat{l}_i(x) = (2i-1-Kx)\hat{l}_{i-1}(x) - (i-1)\hat{l}_{i-2}(x) \tag{12}$$

2.1.2 Algorithm :

The structure of the overall algorithm as follows:

1. Create the state space of the system during each invocation.
2. Compute the Laguerre coefficients .
3. Compute the Laguerre difference coefficients .
4. Choose the suitable value of K .
5. Compute the performability measures of the system .

2.2 Modelling Technique

We have used the General Stochastic Petri Net (GSPN) technique to develop the performability model. A Petri net comprises of a set of places, a set of transitions and a set of directed arcs. In the graphical representation of a Petri net the places are drawn as circles and the transitions as bars. Arcs connect transitions to places and places to transitions. Places may contain tokens which are drawn as black dots. The state or marking of a Petri net is defined by the number of tokens contained in each place .

A formal definition of a PN is thus the following [7].

$$PN = (P, T, A, M) \tag{13}$$

where P is a finite set of places, T is finite set of transitons, A is a set of arcs and M is a marking. A transition is enabled when all of its input places contain at least one token. Enabled transitions can fire, thus removing one token from each input place and placing one token in each output place. Each firing of a transition modifies the distribution of tokens on places and thus produces a new marking for the Petri net.

The reachability graph of a PN with initial marking M is defined as the graph composed of all markings that can be reached from M by means of sequence of transition firings.

Another type of arc in a Petri net is the inhibitor arc. An inhibitor arc drawn from a place to a transition means that the transtion cannot fire if the place contains at least as many tokens as the multiplicity of the inhibitor arc.

A stochastic Petri net (SPN) is a Petri net where each transition is associated with an exponentially distributed random variarable that expresses the delay from the enabling to the firing of the transition. A formal definition of a SPN is thus [8][9]:

$$SPN = (P, T, A, M, R) \tag{14}$$

where P, T, A, M as in (13) and R is the set of firing rates associated with the PN transition. The SPNs are isomorphic to continuous-time Markov chains (CTMC) due to the memoryless property of the exponential distribution of firing times . The sojourn time in each marking (state) is an exponentially distributed variable with the average value $[\sum_{i \in H} r_i]^{-1}$, where H is set of transitions that are enabled by the marking . The transition rate from marking M_i to marking M_j is obtained as $\sum_{k \in H_{ij}} r_k$.

By using these rules it is possible to devise an algorithm that automatically derives, from the SPN description, the state-transition rate matrix of the CTMC. Thus it is possible to compute the steady state probability distribution of markings by solving the usual matrix equation.

$$\Pi Q = 0 \tag{15}$$

with the additional constant

$$\sum_i \pi_i = 1 \tag{16}$$

where Q is the infinitesimal generator whose elements are obtained from the model and Π is the vector of the steady state probabilities.

From the steady state distribution Π it is possible to obtain quantitative estimates of the behavior of the model as given below.

1. The probability of a particular condition of the SPN.

If in the subset A of the reachability set $R(M)$ the particular condition is satisfied the required probability is given by

$$P\{ A \} = \sum_{i \in A} \pi_i \tag{17}$$

2. The expected value of the number of tokens in a given place.

If A(i,x) is the subset of $R(M)$ and x is the k-bound for place p_i then

$$E\ [\ m_i\ ]= \sum_{n=1}^{k} [\ n\ P\ A\ (\ i, n\ )\ ]. \tag{18}$$

3. The mean number of firings in unit time.

If A_j is the subset of $R(M_0)$ in which a given transition t_j is enabled the then mean number of firings of t_j in the unit time is given by

$$f_j = \sum_{M_i \in A_j} \left[\pi_i \left[r_j \ / \sum_{t_k \ enabledin \ M_i} r_k \right] \right] \tag{19}$$

Generalized Stochastic Petri Nets[11] are logical extenstion of Stochastic Petri Net. In GSPN there are two classes of transitions, timed transitions T_t and immediate transitions T_i . Timed transitions have firing times distributed exponentionally, immediate transitions fire in zero time. If the set of enabled transitions H comprises only timed transitions then transition t_i ($i \in H$) fires with probability $r_i \ / \sum_{k \in H} r_k$ where r_i is the firing rate of transition and H is set of transitions that are enabled by the marking. If H comprises both immediate and timed transitions then only immediate transitions can fire. If H comprises zero or more timed transitions and only one immediate transition then this is the one that fires. When H comprises several immediate transitions it is necessary to specify a probability density function on the set of enabled immediate transitions according to which the firing transition is selected. The subset of H comprising all enabled immmediate transitions togther with associated probability distribution is called a switching distribution [9]. A GSPN can be solved by using the fact that all states for which immediate transtions are enabled are reached and left at the same instant of time. So before constructing the Markov graph the reachability graph has to be modified.

3. System Modeling

Real-time systems are characterized by the fact that both logical and timing properties of the system must be satisfied. There are two types of real-time systems: soft real-time systems and hard real-time systems. In soft real-time systems tasks are are not constrained to finish by specific time on each and every invocation. In hard real-time systems tasks have to meet specific deadlines on every invocation. That is they have to be performed not only correctly but also in a timely fashion. A general model for determining that real-time tasks meet their deadlines on every invocation must include information on task and component redundancy. Assuming the system consists of three nodes (A , B , C) . Two tasks may interrupt the main task in node B (tB) , one from

the same node (tD) and one from node A (tA). Task tB will decide to either accept or queue the query from tD or tA. When tB accepts the query it stops its current thread of control and starts executing the response routine; it returns to where it left off after the response routine is completed and the repy tD or tA is made. Also we assume another two tasks can send messages to task B, one from the same node (tE) and another from node C (tC) . When tasks tE or tC send a message they remain blocked until they receive a reply from task B . If task B executes "receive" before a message arrives it becomes blocked. Tasks tC or tE remain blocked until a reply is recived from task tB . Assuming all tasks (tA ,tB ,tC) are first invoked at t_0 with period 90 msec, 180 msec and 90 msec respectively and any task fnished before it is next invocation waits. If any task unable to complete it is current invocation before its deadline it will discrarded. Also assume node B has two processors which have random failure and repair rates and their operation is :

(i) two processors are operational (fault free).

(ii) one processor operational (fault free) and failed processor under repair.

(iii) both processors are failed one failed processor is under repair and other waiting for repair

(iv) both processors are failed and both are under repair.

Fig. 1 together with tables 1 and 2 show a GSPN model based on the above assumptions.

4. Results and Discussion

In this section we will use the algorithm presented previously to analyze the model described in section 3.

Figure 2 to 7 show the complementary distribution ($prob(Y(t) > x$) plotted against number of tasks x with mission time t as a parameter. Plots are shown for mission times between 30 msec and the planning cycle time (180 msec). For example we can see from figure 2 that with only one processor in node B the probability of five tasks completing thier first invocation within a mission time of 90 msec in 0.111063. A summary of the results extracted from figures 2 to 7 is given in the following table.

mission time+	one processor		two processors*		two processors**	
	number of tasks	probability	number of tasks	probability	number of tasks	probability
30	1.7 as shown in fig.2	0.111964	1.9 as shown in fig. 4	0.134081	2.0 as shown in fig. 6	0.116142
60	2.9 as shown in fig.2	0.126059	4.0 as shown in fig. 4	0.115532	4.3 as shown in fig. 6	0.106807
90	5.0 as shown in fig.2	0.111063	5.0 as shown in fig. 4	0.257497	5.0 as shown in fig. 6	0.288036
120	5.9 as shown in fig.3	0.103254	6.9 as shown in fig. 5	0.103704	7.1 as shown in fig. 7	0.106209
150	8.1 as shown in fig.3	0.101661	8.9 as shown in fig. 5	0.107961	9.2 as shown in fig. 7	0.101504
180	9.9 as shown in fig.3	0.107028	10.0 as shown in fig. 5	0.114666	10.0 as shown in fig. 7	0.125227

+ time in millisecond, * one repair,** two repairs

A comparison of the performance with one and two processors in node B is shown in figure 8. It is assumed that the processors serve the tasks independently and on a first-come first-served basis. As might be expected there is a higher probability that two processors complete the job within the mission time than one processor. Figure 9 shows the repairability of the system which depend on the failure and repair of the processors, the figure shows that when the repair rate is increased, that is the transition time for t11 decreased, the system repairability increases. Similiarly the effect of increasing the failure rate can be found by decreasing the transition time of t10 and this will result in a decrease in repairability.

Thus the repairability depends on the probability of changing the token from place (p16) to place (p15). Figure 10 shows the availability of one processor and two processors with one repair and two processors with two repairs.

From the above it is clear the the availability is equal the probability p15 contains token is 1.0 plus the probability p16 contains token is 1.0 or may be found as 1.0 mines probability p16 contains token is 1.0 and probability p14 contains token is 1.0 . Thus the system work if there is no tokens in palce p16 and place p14 in the same time. Thus figure 10 shows the system has more availabilty and less unavailability incase two processors with two repairs than one processor or

two processor with one repair. Figure 10 shows that the system is more reliable with two processors than with one processor.

CONCLUSION

In this paper we have shown how the effects of redundancy and random failure and repair on the performance of a simple distributed real-time system can be evaluated. The performability is then computed by numerical evaluation. The algorithm used is one wich is based on a modified Laguerre function. Distributed real-time control systems with redundancy give successful applications by reduced job completion time, higher probability of successful job complete and the system meet the time by which a task must finsh.

Transitions	Comments
t1, t4	task A query delay for first and second invocations respectively.
t2, t5	task A query response routine executed for first and second invocations respectively.
t3, t6	task A reply delay for first and second invocations respectively.
t7, t8	task D query response routine executed for first and second invocations respectively.
t9	task B processing the local job.
t10	failure of processor.
t11	end of the repair of one processor.
ti	start of the repair of the processor.
t12, t13	processing the message from task E for the first and the second invocations.
t14, t17	task C message delay for first and second invocations respectively.
t15, t18	processing the message from task C for the first and the second invocations.
t16,t19	task C reply delay for first and second invocations respectively.

Table 1 : Description of the transitions of the Petri net.

Places	Comments
p1, p5	start task A query for first and second invocations respectively.
p2, p6	query ready for first and second invocations respectively.
p3, p7	reply ready for first and second invocations respectively.
p4, p8	received task A reply for first and second invocations respectively.
p9, p11	start task D query for first and second invocations respectively.
p10, p12	received task A reply for first and second invocations respectively.
p13	local job in node B available.
p14	failed processor, waiting for a repair.
p15	both processor fault free.
p16	processors under repair.
p17	repair resource available .
p18	end the local job in node B.
p19	new local job in node B.
p20	end of the planning cycle.
p21	sart of planning cycle.
p22	a receive from task E executes.
p23, p26	start task E issues a send to task B in the first and second invocations respectively.
p24, p28	task E reply receive for first and second invocations respectively.
p25, p27	end of processing the message from task E in the first and second invocations respectively.
p29	task B executes a receive from task tC.
p30, p31	end of processing the message from task A in the first and second invocations respectively.
p32, p36	start task A issues a send to task B in the first and second invocations respectively.
p33, p37	task C message ready for first and second invocations respectively.
p34, p38	task C reply ready for first and second invocations respectively.
p35, p39	task C reply receive for first and second invocations respectively.

Table 2 : Description of the places of the Petri net.

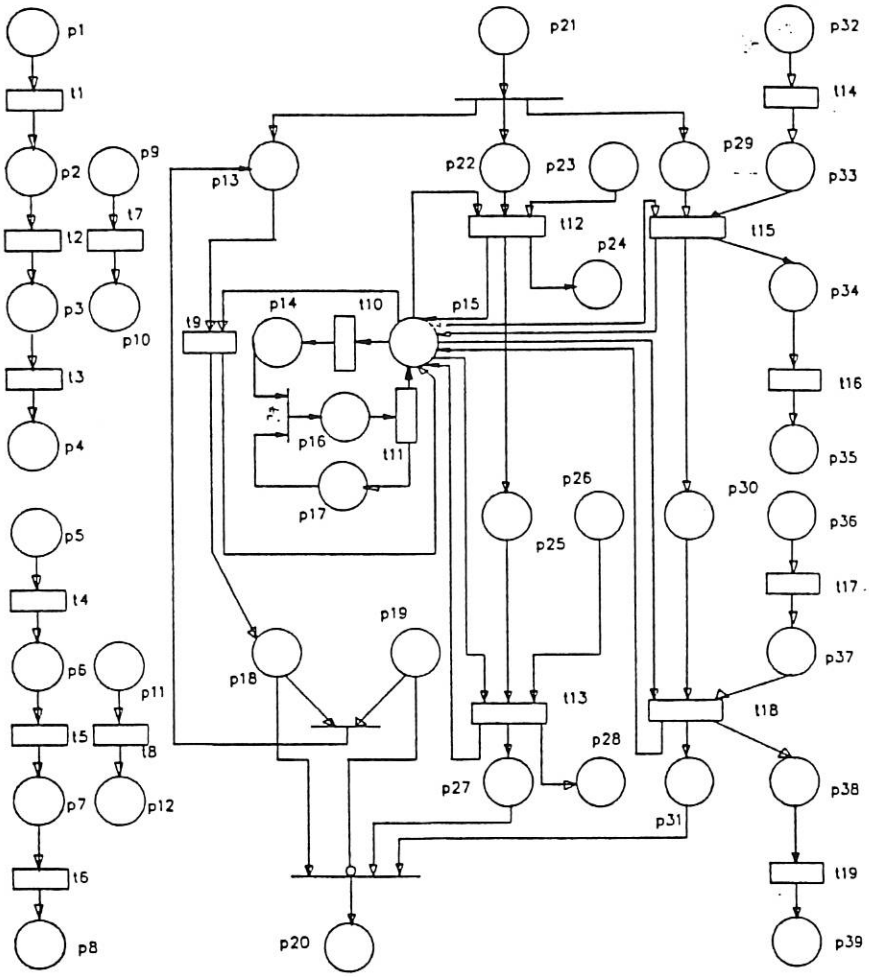

Fig. 1. GSPN model of distributed real-time control system description.

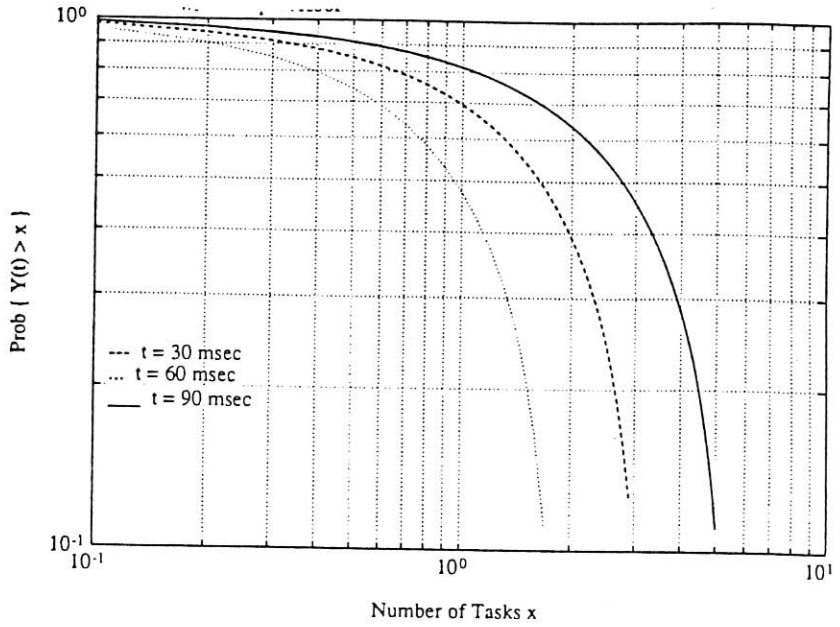

Fig. 2. The complementary distribution of the number of tasks completed during the first invocation by one processor.

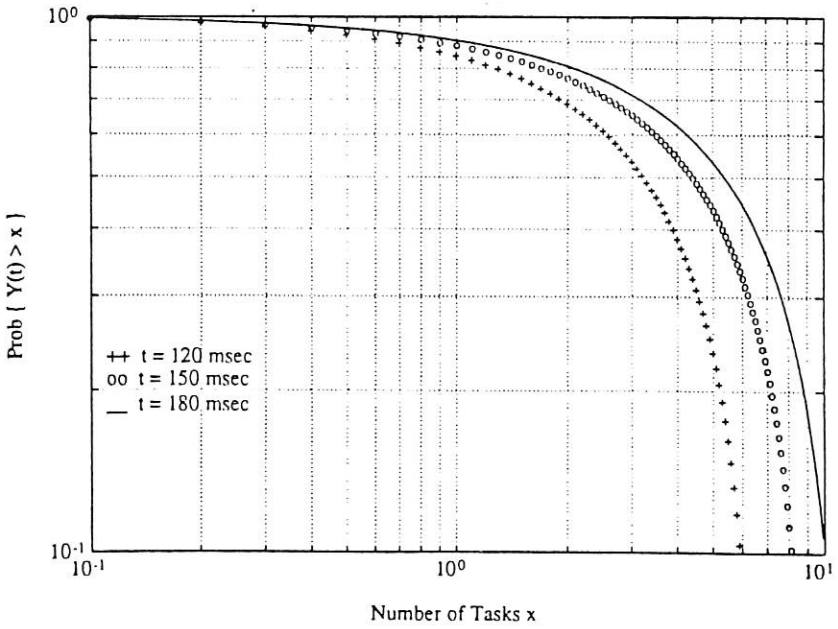

Fig. 3. The complementary distribution of the number of tasks completed with the ending of the planning cycle by one processor.

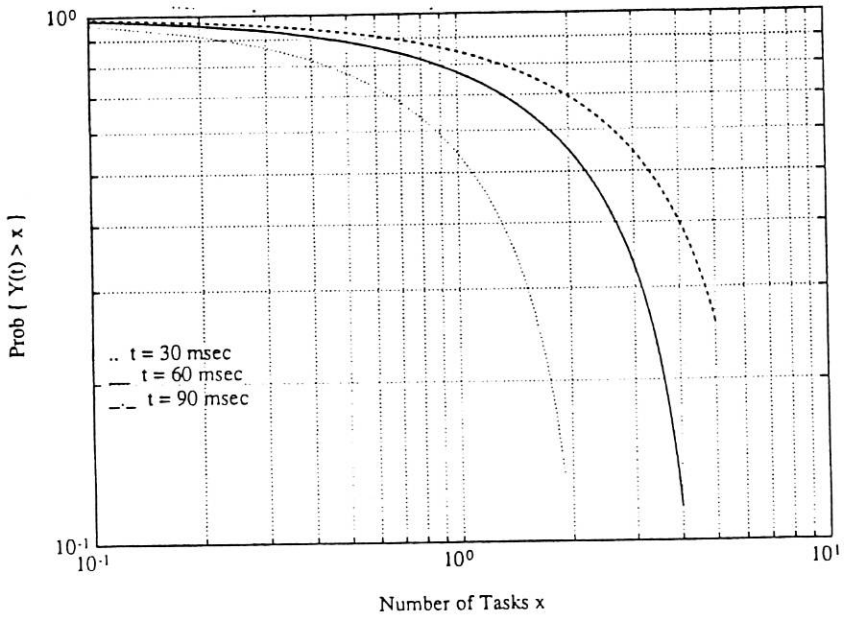

Fig. 4.The complementary distribution of the number of tasks completed during the first invocation by two processors with one repair.

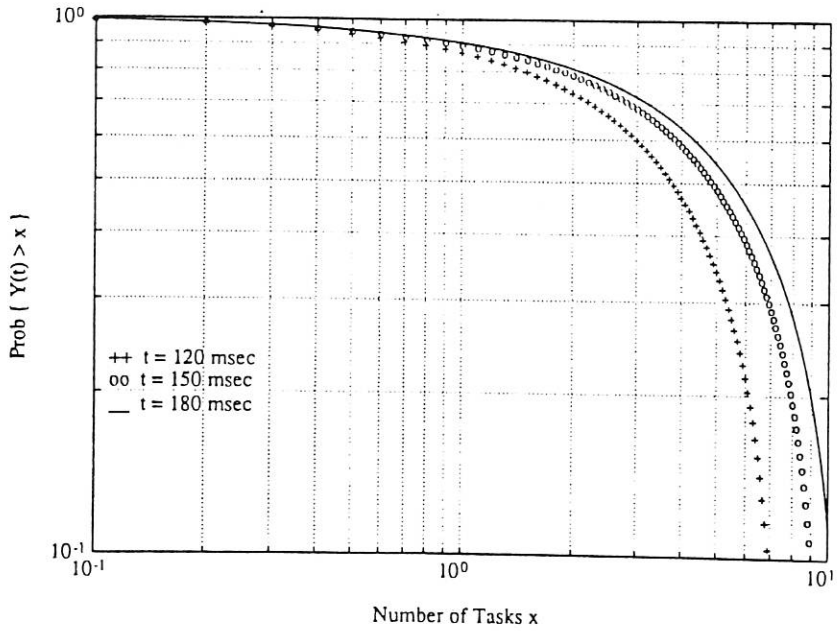

Fig. 5. The complementary distribution of the number of tasks completed with the ending of the planning cycle by two processors with one repair.

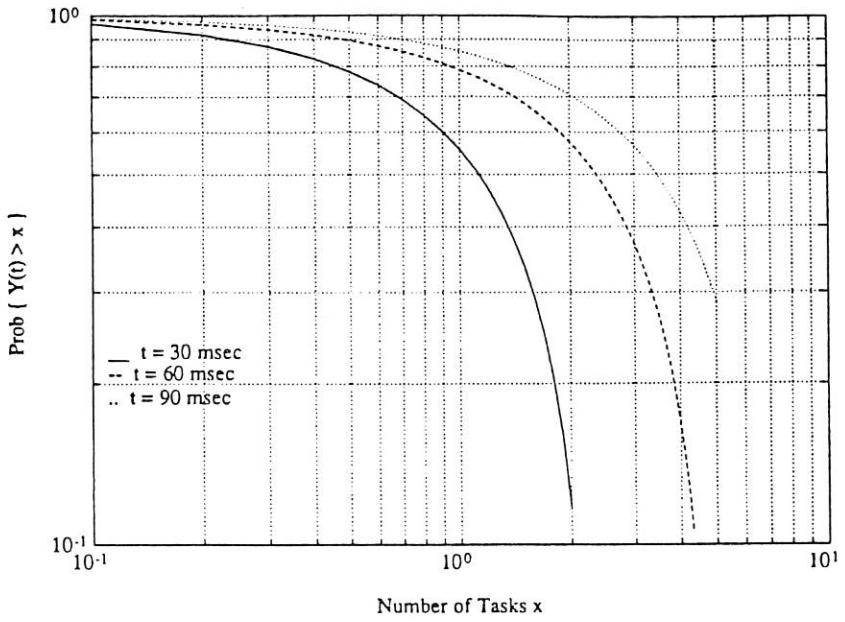

Fig. 6. The complementary distribution of the number of tasks completed during the first invocation by two processors with two repairs.

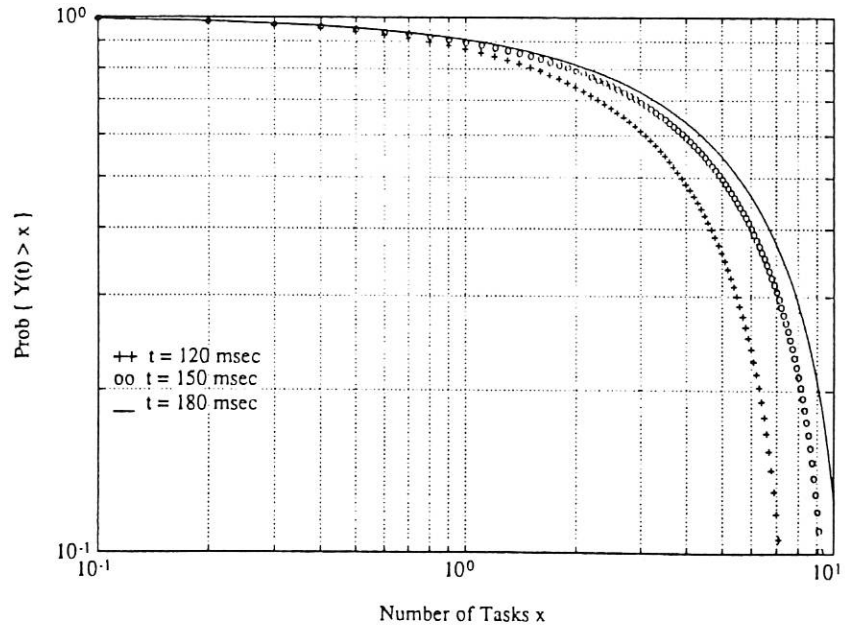

Fig. 7. The complementary distribution of the number of tasks completed with the ending of the planning cycle by two processors with two repairs.

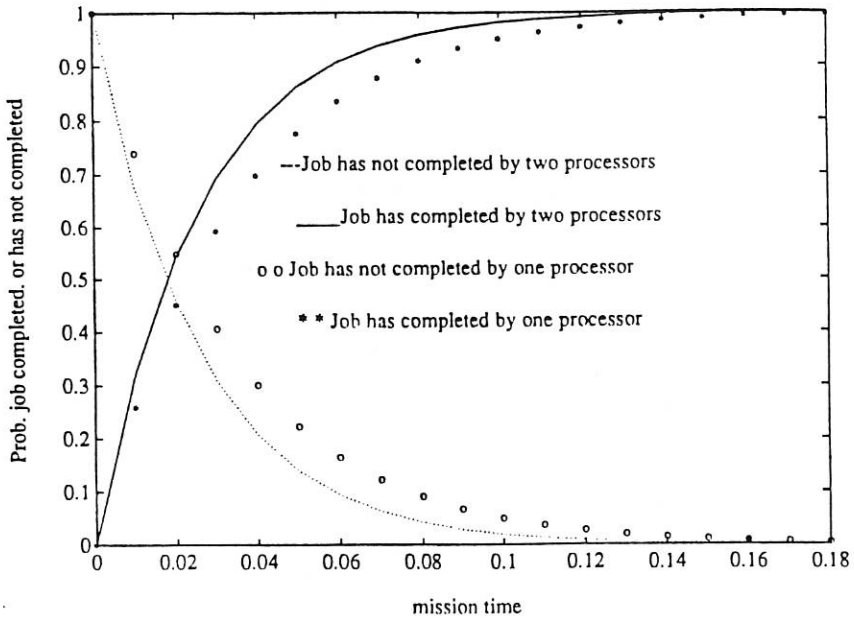

Fig. 8. The performance of one processor and two processors during one planning cycle.

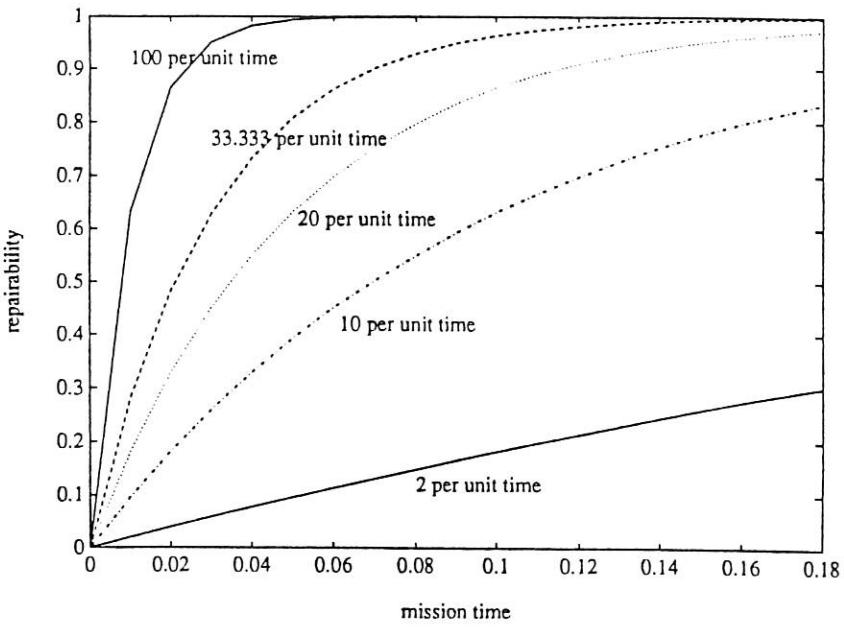

Fig. 9. Repairability of the processors during the planning cycle.

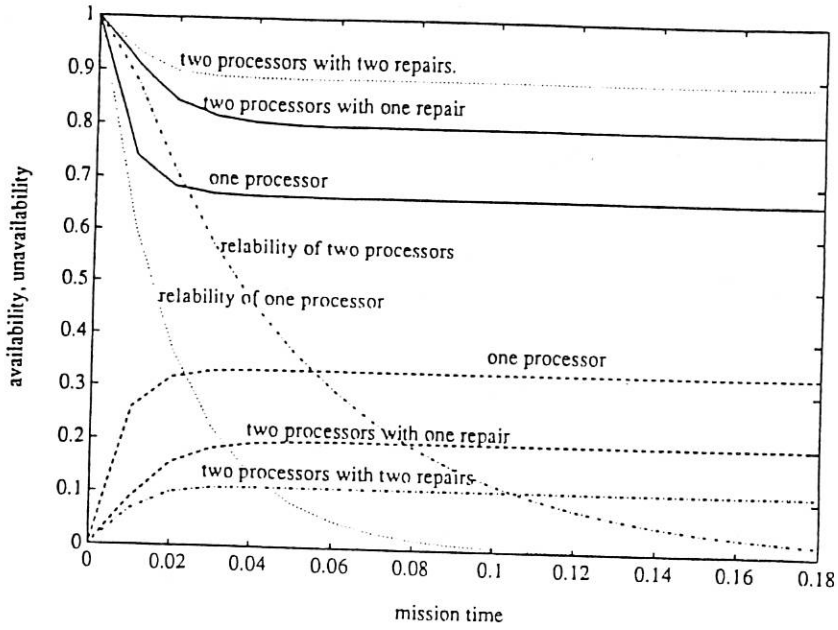

Fig. 10. Availability, unavailability and reliability of one processor and two processors with one and two repairs during one planning cycle.

REFERENCES

[1] K. Ramamritham,“ Channel Characteristics in Local- Area Hard Real-Time Systems,” Computer Network and ISDN Systems, pp. 3-13, 1987

[2] D. Peng and K. G. Shin “ Modeling of Concurrent Task Execution in a Distributed System for Real-time Control ” IEEE Trans. Comput., vol. C-36, no.4, pp. 500-516, Apr.1987.

[3] S.Cheng, J. Stankovic, and K. Ramamritham ''Scheduling Algorithms for Hard Real-Time Systems- A Briev Survey,'' Tutorial Hard Real-Time Systems, J Stankovic snd K. Ramamritham Eds. Computer Socity press, 1988

[4] J.F. Meyer, '' On Evaluating the Performability of Degradable Computer Systems,'' IEEE Trans. Comput. vol.C-20, no.8, pp. 720-731, Aug. 1980. Ramamritham Eds. Computer Socity press ,pp. 150-167, 1988

[5] A. Shallof and S. Bennett, '' Performability Analysis of Distribuited Real-Time Systems, '' Submitted to IEEE Trans. Computer

[6] B. R Iyer, L. Donatiello and P. Heidelberger, ''Analysis of Performability for Stochastic Models of Fault-Tolerant Systems''' IEEE Trans. Comput., vol. C-35, no. 10, pp. 902-907, Oct.1986.

[7] T. Agerwala,''Putting Petri Nets to Work,'' IEEE Computer vol. 12, no.8, pp. 85-94, Dec. 1979

[8] K. Molley,'' Performance analysis using Stochastic Petri Nets,'' IEEE Trans. Comput. Sep. 1982

[9] M. Marson,M.Balbo and G.Conte, ''A Class of Generalized Stochastic Petri Nets for the Performance Evalution of Multiprocessor Systems,'' ACM Trans. Comput. Syst., pp. 93-122, May 1984.