A Fault Tolerant Technique- Primary Backup Model For Distributed Control System

Deepti R. Katare, Prof. N. N. Jangle

Dept. of Electrical Engineering, KKWIEER, Nashik, India

***Abstract*—**Distributed control system has been widely used in the recent years. In this seminar brief description is presented about distributed control system. Hence a distributed control sys-tem (DCS) is a computerized control system, in which controller element are not centrally located but distributed throughout the system. That means it is a type of automated control system that is distributed throughout a machine to provide instructions to dif-ferent parts of the machine. Instead of having a centrally located device controlling all machines, each section of a machine has its own computer that controls the operation. For instance, there may be one machine with a section that controls dry elements of cake frosting and another section controlling the liquid elements, but each section is individually managed by a DCS. A DCS is commonly used in manufacturing equipment and utilizes input and output to control the machine .Distributed control concept is a cost effective approach for implementing large systems and networks based on standard low cost processing elements (microprocessor) and components.. But with the increase of the number of processors, a DCS is subject to hardware and software failures. Therefore, a small information fault-tolerant scheduling algorithm based on backward non-preemptive RM (BNPRMFT), which can tolerate both hardware faults and software faults is presented in this seminar. Also i have presented the disign og allocation of primary backup model.

Index Terms*—Distributed control system, Fault tolerant, Pri-mary Backup Technique, Matlab Simulation.*

I. INTRODUCTION

DCSs are increasingly being applied in many fields in recent years, for example, avionic control, nuclear plant control, process control systems, automatic manufacturing control sys-tems and other autonomic systems, because of their attractive advantages, such as the high control performance, reliability and extensibility. With the increasing complexity of a DCS, the possibility of hardware faults and software failures increases. However, a DCS is a kind of hard real-time system, in which the consequences of not executing a task before its deadline may be catastrophic (for instance, threat to human lives or significant economic loss). Thus, a fundamental requirement of DCSs is to complete all real-time tasks within their specified deadlines even in the presence of faults. Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively designed system in which even a small failure can cause total breakdown. Fault tolerance is particularly sought after in high availability or life-critical systems. The ability of maintaining functionality when portions of a system break down is referred to as graceful degradation. A fault-tolerant design enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails. The term is most commonly used to describe computer systems designed to continue more or less fully operational with, perhaps, a reduction in throughput or an increase in response time in the event of some partial failure. That is, the system as a whole is not stopped due to problems either in the hardware or the software. An example in another field is a motor vehicle designed so it will continue to be drivable if one of the tires is punctured, or a structure that is able to retain its integrity in the presence of damage due to causes such as fatigue, corrosion, manufacturing flaws, or impact In order to obtain the high reliability of an distributed real-time system, several different models (techniques) have been developed to realize fault-tolerance in last several decades, namely, (1) Triple Modular Redundancy (TMR) model, (2) Primary Backup (PB) model, and (3) Recovery Block model.

II. FAULT AND FAULT TOLERANCE

A. Types of Faults

Transient Fault : appears once, then disappears

Intermittent Fault :occurs, vanishes, reappears; but: follows no real pattern (worst kind).

Permanent Fault :once it occurs, only the replace-ment/repair of a faulty component will allow the DS to function normally

B. Fault Tolerance

Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively designed system in which even a small failure can cause total breakdown. Fault tolerance is particularly sought after in high-availability or life-critical systems. The ability of maintaining

functionality when portions of a system break down is referred to as graceful degradation A fault-tolerant design enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails. The term is most commonly used to describe computer systems designed to continue more or less fully operational with, perhaps, a reduction in throughput or an increase in response time in the event of some partial failure. That is, the system as a whole is not stopped due to problems either in the hardware or the software. An example in another field is a motor vehicle designed so it will continue to be drivable if one of the tires is punctured, or a structure that is able to retain its integrity in the presence of damage due to causes such as fatigue, corrosion, manufacturing flaws, or impact. A highly fault-tolerant system might continue at the same level of performance even though one or more components have failed. For example, a building with a backup electrical generator will provide the same voltage to wall outlets even if the grid power fails. A system that is designed to fail safe, or fail-secure, or fail gracefully, whether it functions at a reduced level or fails completely, does so in a way that protects people, property, or data from injury, damage, intru-sion, or disclosure. In computers, a program might fail-safe by executing a graceful exit (as opposed to an uncontrolled crash) in order to prevent data corruption after experiencing an error. A similar distinction is made between ”failing well” and ”failing badly”. Fail deadly is the opposite strategy, which can be used in weapon systems that are designed to kill or injure targets even if part of the system is damaged or destroyed. A system that is designed to experience graceful degradation, or to fail soft (used in computing, similar to ”fail safe) operates at a reduced level of performance after some component failures.

III. PRIMARY BACKUP MODEL

This section presents simulation results of our proposed algorithm BNPRMFT for different task sets. We demonstrate the strength of our algorithm by comparing its simulation results with the case the backward preemptive RM is employed named BPRMFT.A set of n periodic tasks p = T1, T2 Tn is to be scheduled on a number of processors, task i will be represented as Ti until and unless specified. For each task Ti, there are a primary copy and a backup copy associated with it. The computation time of a primary copy is denoted as c, which is the same as the computation time of its backup copy

. The tasks may be independent or dependent of each other.In this section, we first present our basic fault-tolerant algorithm and the corresponding schedulability analysis. Then, we point out some problems existing in the basic algorithm and propose two more ideas to improve the percentage of the successful primaries.

As mentioned earlier, our algorithm uses the last chance philosophy If there are primaries pending for execution, al-ternates will not be scheduled until the latest possible time, called the notification time, on or before which if alternates are not scheduled, they will not be completed in time. Here we represent two algorithmic task in which the first job is to assign the primary and backup task to uniprocessors and then it will be preallocate and to each task notification time is given. Our proposed algorithm has two main objectives: 1) guarantee either primary or alternate of each task (job) to be successfully completed before their corresponding deadlines;

(2) complete as many primaries as possible to achieve better computation quality. The first objective is achieved by using an offline fixed priority scheduling algorithm (such as RM) to ensure the successful accommodation of all alternate jobs. This offline schedule is constructed backward from time T and the alternates are executed as late as possible, thus leaving the largest possible room for the execution of the primaries to accomplish the second goal.

A. Details of Simulation

1)Assigning the Primary Backup copies: Therefore starting with the first algorithm that assigns the Primary and Backup copies to the uniprocessor, following is the table which gives us the four input as discussed in the algorithm 1 i.e. No of task denoted as i, the commputation time both for primary and backup copy, primary copy tp of task i, backup copy tb of task i. The table is shown below

t	c	tp	tb
1	2	1	2
2	5	2	3
3	10	3	4
4	15	4	5
5	20	5	6

This table gives us the values which will be put up into a formula given below

$$S = \sum \frac{\tau i : tp : c}{\tau i : T} \qquad (1)$$

by putting the values in this formula the Primary copy will be assign.

Similarly there is another formula for to assign Backup copy which is presented as follow

$$S = \sum \frac{\tau i : tb : c}{\tau i : T} \qquad (2)$$

and this will assign backup Copy.

2) Result: Using the above content of the table and putting it into the formulas we obtain primary and backup copy which are assigned to the uniprocessor. The result will be given in form of table in which the assigned Primary Backup copy is obtain by summing all the values given above. Table represent

four columns S is for Primary Copy, P is for Backup copy, sumS is the summation of Primary copy, and sumP is the summation of Backup copy.

S	P	sumS	sumP
0.20	0.40	0.20	0.40
1.00	1.50	1.20	1.90
3.00	4.00	4.20	5.90
6.00	7.50	10.20	13.40
10.00	12.00	20.20	25.40

3) Allocation of Primary Backup copy: Given a real-time periodic task set, we first use any fixed priority-driven schedul-ing algorithm to reserve time intervals as late as possible for all the backup in a planning cycle before runtime. At runtime, if there are primaries pending during the time intervals that were not reserved by backup, the scheduler chooses the primaries to execute first. The primaries can be scheduled by any online scheduling algorithm, such as a (fixed or dynamic)priority-driven preemptive scheduling scheme with the RM or EDF priority assignment. A primary may fail (because of software bugs or taking too long to complete) at any time during its execution. If a primary fails, its corresponding alternate must be executed. Moreover, when the notification time, of alternate is reached, yet its corresponding primary has not been completed or has failed, is activated (thus preempting the execution of any primary, including, or other lower-priority alternates). The primary, if it has not been finished, will be aborted since its backup is now chosen to be executed. Every backup, if activated on or after its notification time, has higher priority than all primaries and the activated backup are executed according to their priorities assigned by the offline fixed-priority algorithm.

Note, however, that an alternate need not be activated if its corresponding primary has been successfully completed before its notification time. That is, if backup finishes its execution successfully before the notification time, the alternate need not be activated and, hence, the time interval(s) allocated to can be reallocated to other primaries or backup. In this case, the notification time is no longer needed and is thus cancelled. Moreover, the notification times of other backup need to be adjusted since the time reserved is now freed The Simulink Model has six blocks namely

1) Primary Block - It is the first priority block which when fault occurs comes into picture for the execution.

2) Backup Block- It is the alternate block for the primary block which comes into action only when primary fails

. If primary is in running condition without occurrence of fault then backup will not execute.

3) Notification Time - The system has two task which is also called as job i.e. Primary task and Backup task. Each task has been given its notification time whose other name is deadline. This is given so that the task should complete its job within the notification time otherwise it is consider as a fault.

4) End Time- End time is the time for which the Primary and Backup copy finishes before the given notification time. For eg if Primary copy finishes before the notifica-tion time, so that time will be called as End time. Again the time between the End time and Notification time is calculated which is called as Available time on which the allocation of primary and backup task is depended.

5) Switch - This is the Main block for the simulation model which controls all the other blocks. This is the condition based switch .So pass through input 1 when input 2 satisfies the selected criterion; otherwise, pass through input 3. The inputs are numbered top to bottom (or left to right). The first and third input ports are data ports, and the second input port is the control port.

6) Scope - It will give us the result of the implemented inputs.

IV Results: Results are based on three different conditions which are:-

End Time < Notification Time - In this condition if the end time is less than notification time than available time is tested. And obviously the primary has been completed its work before deadline so we can obtain available time. Due to which the backup will not come into picture and its primary will continue to execute. According to the simulation and the condition the result obtain is

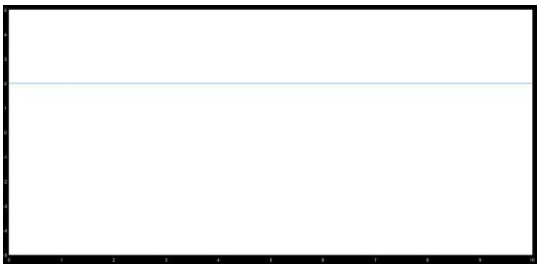

Fig. 1. Primary waveform

End Time > Notification Time - In this condition if Notification time is less than end time, it means the deadline of the given task has been finished and it still have not been completed therefore it is considered as a fault and its alternate i.e. Backup will execute. According to the simulation and the condition the result obtain is

Fig. 2. Backup waveform

End Time = Notification Time In this condition if the End time and the Notification time is equal means the system is working in the normal condition and after the primary task completes at its notification time then immediately its alternate i.e. Backup will execute. According to the simulation and the condition the result obtain is

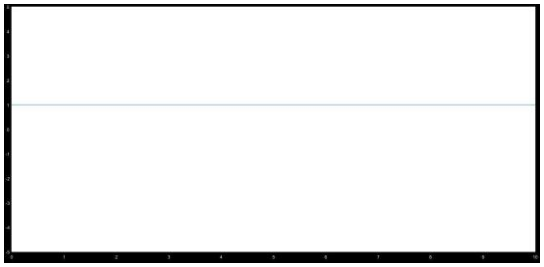

Fig. 3. waveform

IV. CONCLUSION

Considering that DCSs are subject to hardware and soft-ware faults, I have presented a fault-tolerant scheduling algorithm named BNPRMFT. Compared with other fault-tolerant scheduling algorithms, BNPRMFT can tolerate not only hardware faults, but also software faults. In our fault-tolerant scheduling algorithm, every task has a primary copy and a backup copy which are independent and assigned to different processors according to a heuristic algorithm which can balance the loads of primary copies and backup copies on each processor. A backup copy is executed only when its corre-sponding primary copy fails due to a fault. A notification time (NT) is set for a task, before or at which backup copy must start, otherwise it cannot be finished before its deadline. Unlike other fault-tolerant scheduling algorithms for hardware faults, BNPRMFT can execute as many primary copies as possible due to their high control performance. Unlike other algorithms for software faults, BNPRMFT can tolerate hardware faults by executing backup copies assigned to different processors. In order to lower the cost of the algorithm, non-preemptive RM has been employed to schedule primary copies and backward non-preemptive RM has been applied to calculate notification times of tasks in order to leave more time for executing pri-mary copies.Finally, computer simulation has been carried out to testify BNPRMFT. Compared with BPRMFT, BNPRMFT can gain a higher success rate in executing primary copies and lower the runtime overhead for the algorithm implementation.

REFERENCES

[1] Y. Rui, C. Qinqin, L. Zengwu, S. Yanmei, Multiobjective evolutionary design of selective triple modular redundancy systems against SEUs, Chinese Journal of Aeronautics 28,(2015)

[2] R. Al-Omari, A. K Somani, and G. Manimaran. An Adaptive Scheme for Fault-Tolerant Scheduling of Soft Real-Time Tasks in Multi-processor Systems. Journal of Parallel and Distributed Computing, 2005,65(5):595-608.

[3] R. Mahmud Pathan. Fault-Tolerant Real-Time Scheduling Algorithm for Tolerating Multiple Transient Faults. 4th International Conference on Electrical and Computer Engineering ICECE 2006, Dec. 2006, Dhaka, Bangladesh, 577-580

[4] C. C. Han, K. G. Shin, and J. Wu. A Fault-Tolerant Scheduling Algorithm for Real-Time Periodic Tasks with Possible Software Faults. IEEE Transactions on Computers, 2003, 52(3): 362-372.

[5] H. Beitollahi, S. G. Miremadi and G. Deconinck. Fault-Tolerant Earliest-Deadline-First Scheduling Algorithm. IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2007, Long Beach, CA, 1-6.

[6] D. Liu, C. Y. Zhang, R. Li, et al. Fault-Tolerant Real-Time Scheduling Algorithm in Software Fault-Tolerant Module. Journal of Computer Research and Development, 2007, 44(9) : 1495- 1500.

[7] W. F. Ding, R. F. Guo, C. G. Qin, et al. A Fault-Tolerant Scheduling Algorithm with Software Fault-Tolerance in Hard Real-Time Systems. Journal of Computer Research and Development, 2011, 48(4) : 691-698.