DOI 10.29042/2017-1218-1223

Helix Vol. 8: 1218-1223

Scheduling Strategy in Fault Tolerant Time Triggered Architectures

B Abdul Rahim*, K Soundara Rajan

*Dept of ECE, Annamacharya Institute of Technology & Sciences, Rajampet, Andhra Pradesh, India

Dept of ECE, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India.

Email: *abraheem@rediffmail.com, soundararajan_jntucea@yahoo.com

Received: 25th April 2016, Accepted: 18th December 2016, Published: 15th March 2017

ABSTRACT

Reliable safety critical systems are designed for fault tolerance and impose high dependability with reliability requirements. Scheduling such systems is hard in traditional event triggered systems and becomes difficult as complexity increases. Time triggered systems with timing guaranteed in task activation, avoiding the risk of missing a hard-critical deadline, overcoming the problem of time predictability. As time triggered pattern is going to take control in safety critical applications, gradually scheduling strategies to be more robust and optimal in resource allocation. Previous work has been on handling the schedule of messages within the time triggered protocol for a given TDMA configuration. The sequence and size of slots in a TDMA round are resolved to reduce the delay. Greedy heuristic and meta-heuristics like Simulated Annealing, Genetic algorithms etc., are investigated for better optimization. The Greedy Algorithm works on arriving at global optima by selecting a local optimum. The worst case here is the visit of all the nodes for best possible solution. Randomly each member is selected from the neighborhood in the simulated annealing process, thereby determining the movement to next state. Simulated Annealing parameters make easy in finding near-optimal solutions within a reasonable time. The selection, crossover and mutation, are the fundamental genetic operations of Genetic Algorithm used to modify the solution to obtain appropriate resultant offspring passing on features to succeeding generations. Genetic Algorithm scheduling in fault tolerant time triggered architectures show significant performance improvement when nodes are large in number and guaranteed for optimal solution.

Keywords

Safety Critical Systems, Time triggered Systems, Scheduling Algorithms.

1. Introduction

A system is said to be critical, if its malfunction leads to economic loss and/or loss of life. Hence the system to be safety critical should possess highest level of safety integrity; malfunction may cause serious consequences [1]. Reliability is the probability of no failures in the interval; this is the measure of performance for fault tolerant systems. Hence for applications of high level of safety, fault tolerant design is must [2]. Dependability is the ability of a system to deliver the service that can be trusted justifiably. The service delivered by a system as it is perceived by other which is interacting with it [3]. So the systems required to function properly while obeying the properties of cost and performance, further providing high dependability despite fault occurrences.

A real-time embedded system is required to satisfy certain nonfunctional parameters like cost, size, power constrains, performance dependability, etc. wherein the timing requirements for hard real time systems is extremely important. A real time system implementing such requirements in order to function correctly had to meet its deadlines. Scheduling of such systems in traditional event triggered systems is hard and becomes more difficult as complexity increases by addition of functionalities. Time triggered systems with timing guaranteed in task activation, avoiding the risk of missing a hard-critical deadline, and thereby overcoming the problem of time predictability. Communication and processing activities in time triggered systems are started at points of time which are already defined. Hence, there will be only one interrupt in the process that is the clock, which partitions the continuous time into sequences of equal space. A time triggered activated task has fixed activation period which makes scheduling of tasks (static) offline. This overcomes the major problem of event-triggered architectures of non-determinism. That is no guarantee of successful transmission of message [4].

As time triggered patterns is gradually introduced in safety critical applications, progressively the scheduling strategies has to be made more robust and optimal in resource allocation. Previous work has been on handling the schedule of messages within the time triggered protocol for a given TDMA configuration. The Sequence and the slots lengths are determined in a TDMA round to reduce the delay. Greedy heuristic and meta-heuristics like Simulated Annealing and Genetic algorithms are investigated for better optimization.

The next section discusses the architecture of time

triggered systems over which the scheduling of tasks for particular nodes is obtained. The scheduling algorithms can be any type, but here we used general optimization algorithms which are discussed in the following section. The results and discussions show the comparative analysis of the algorithms suitability to time triggered approach in the preceding section. Finally, in conclusion section the outcome of the work with summary are presented.

2. Time Triggered Systems

The time triggered paradigm systems are designed to be fully synchronous and are built around time triggered network used for message transmission and also for synchronization [5]. The main advantage of this system is composability and easy implementation of fault tolerance, making it suitable for safety critical applications. The host subsystem is isolated from the communication subsystem by communication controller in the time triggered architecture. The Time Triggered Protocol TTP/C is used for communication between the nodes [6]. The protocol accomplishes the job of exchange of messages between the nodes of cluster in a prescribed manner. According to the static schedule the communication subsystem decides transmission and reception of messages of a particular node is relevant or not.

The TTP/C network consists of two channels, replicating each other are connected to nodes or electronic modules as shown in figure 1(a). The electronic module or node consists of a host, communication network interface and a TTP/C controller. Basically, the host subsystem inferred to execute the application, which may be an actuator or a sensor. The communication subsystem is a communication controller executing the TTP/C protocol. This node or electronic module is replaceable in case of fault, hence referred to as Smallest Replaceable Unit (SRU). A fault tolerant unit (FTU) is the combination of one or two electronic modules and the electronic module and the network is called the cluster [7][8][9]. A particular service is delivered by the FTU even if an electronic module fails form the group.

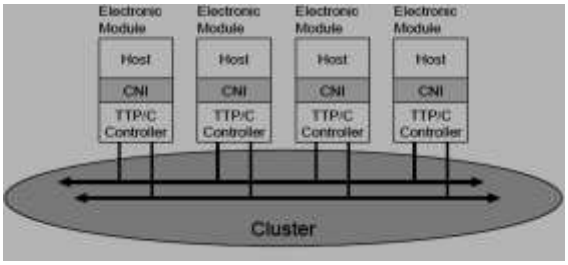


Figure 1. (a) TTA scheme with Electronic modules connected by replicated communication channels.

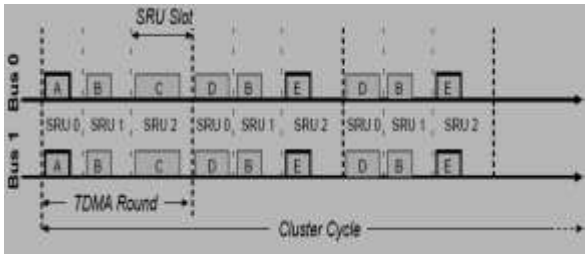


Figure 1. (b) Cluster Cycle Representation.

The bus accessing scheme is derived from the global time notion controlled by cyclic time division multiple access, where every active module has its own TDMA slot. Each electronic module sends its message in the TDMA slots allocated to it. The sequence of TDMA slots forms a TDMA round as illustrated in Figure 1(b). After completion of one TDMA round another TDMA round is initiated with the same access pattern but probably with different message. The number of TDMA rounds determines the cluster cycle length. When this cluster cycle execution ends, the transmission pattern starts again in new cluster cycle. The system does two types of schedules; communication schedule and scheduling of individual node. All the nodes in the system share same network for message transmission and reception which is basically broadcast type. As different tasks are performed by the node, the node scheduling is different and it should be predefined. The operating system executes this schedule in time triggered manner.

The Communication Network Interface (CNI) between the host subsystem and the TTP/C controller acts as a temporal firewall i.e, preventing the propagation of errors. Any single hardware failure is tolerated and if any malicious host produces erroneous data can never interferes in correct operation of the cluster. That is fail silence condition in temporal domain is guaranteed by the TTP/C controller. Thus system becomes fault tolerant and applicable to safety critical systems like automotives, avionics, railways etc, [10].

3. Scheduling algorithms

Greedy Algorithm

The Greedy Algorithm takes decision incrementally in small steps without going back to the same state. The decision on each step is to improve the current state in a myopic (narrow minded) fashion without thinking of global situation. The decision could be fixed and often based on some simple priority rules. Here, the node selection based on the priority that a node is ideal and not blocked for malicious data transfer. Let N={1,2….n} be the set of nodes available of use and each node is selected for task execution with

a starting time of s_i and finishing time f_i, i.e, between $\{s_i, f_i\}$. Here, the execution times all the tasks are sorted from least to the maximum. Then the task allocation to the nodes is selected greedily going down from the list by picking which node is suitable for the current task. The running time depends on what type of sorting algorithm is adopted. The sorting part can be as small as *O(n log n)* and the other part is *O(n)*.

R*set* is the set of all requests
X*set* is empty (X*set* will store all the tasks that will be scheduled)
While R*set* is not empty
Choose $i \in$ R*set* such that finishing time of *i* is least
Add *i* to X*set*
Remove from R*set* all requests that overlap with *i*
Return X*set*

Considering the greedy nearest neighbor heuristic the sorting part can rise to $O(n^2)$. The advantage of this algorithm is easy to design and implement, also runs fast since they are simple. Greedy Algorithm works on arriving at global optima by selecting a local optimum. Thus, the optimal solution for the problem is derived from optimal solution of the sub problem, looking like a dynamic programming. The worst case here is the visit to all the nodes for best possible solution. Even then we cannot expect to obtain the overall optimum from greedy algorithm.

3.1. Simulated Annealing

The Simulated Annealing is a meta-heuristic which relies on local search [11]. The Simulated annealing is metallurgical process of heating and then cooling it to a steady organized state. The algorithm chooses a neighbor randomly and then determines whether to move to the next state or not i.e, solution is generated by a suitable mechanism having the change in cost function. If the cost function is reduced, the existing solution is replaced by the generated neighbor. If it is increased, the generated neighbor replaces the existing solution with an acceptance probability function is given by $e^{-(fj-fi)/T}$ where fj is generated state and fi is the present state and *T* is Temperature, a control parameter. In the process, small increase in *f* is more likely accepted than significant increase, and most of the generated neighbors are accepted when *T* is high, they are rejected if *T* approached zero. The initial temperature is always kept high so that it does not get stuck in local minima; the algorithm thus generates some neighbors at each temperature as it drops. Simulated annealing algorithm is guaranteed to get near optimal solution with a reasonable amount of computation time [12][13].

Generate a random solution *Xm*
For *i*=1 to δ (no of iterations)do
Generate a random solution X_1;
Generate a new solution X_1' for X_1;
If X_1' is better than *Xm*
Xm is updated with X_1'
Endif
Endfor
Return the best solution

3.2. Genetic Algorithm

Genetic Algorithm is a search heuristic employs the processes found in natural biological evolution, introduced by John Holland in 1970[14]. The genetic algorithm operates on a given population of potential solutions to find those that come near some criteria or specification. Survival of fittest is the basic principle adopted by the algorithm to find better approximations. In the problem domain, according to their level of fitness, a set of new approximations are created. These evolutionary steps are called generations and are propagated by operators that are borrowed from natural genetics. This creates evolution of populations of individuals that are better suited to their environment than the individuals (parents), just as in natural adaptation. The genetic algorithm is based on fundamental genetic operations such as selection, crossover and mutation, used to modify the solution to obtain appropriate resultant offspring passing on features to succeeding generations.

Genetic Algorithm considers many points in the search space simultaneously and provides rapid convergence to a near optimum solution. But genetic algorithm suffers from the problem of excessive complexity in too large problems. Genetic algorithm is an iterative procedure that consists of a constant-sized population of individuals represented by a finite linear string of symbols, called the chromosome, encoding a possible solution in a given problem space, referred to as the search space or state space. The search space comprises of all probable solutions to the optimization problem. At every step of evolution, called the generation, individuals of the current population are decoded and are evaluated according to a fitness function set for a given problem. The probable number of times an individual is selected is more or less proportional to its relative performance in the population. Between two preferred individuals, Crossover is performed to form new individuals by exchange of part of their genomes. By introducing mutation operator in between premature convergence can be avoided.


```
t := 0; initialize time
Initialize population N(t);
Evaluate fitness of population N(t);
While not complete do
  t := t + 1;
  N'(t) := select parents N(t);
  recombine N'(t);
  mutate N'(t);
  evaluate fitness of N'(t);
  N := survive N,N'(t);
End while
```

The feature of each candidate solution is measured by the fitness function, according to the given optimization objective. This is used to help determine which chromosomes are retained in the population as successive generations are evolved. The fitness function is calculated for each chromosome, and is determined by the number of tasks in the chromosome that meets their deadlines. In tasks scheduling algorithms, the three main elements that must be included in the chromosome format are:

- the list of the tasks to be scheduled;
- the order in which these tasks should execute on a given processor;
- the list of the processor which these tasks should be assigned to.

In Crossover operation, all the members of the population are grouped (randomly) into subsets of two chromosomes per set. After a randomly selected point the processor part of the gene in the chromosomes pair is swapped. This is equivalent to assigning some tasks to different processors. This operation called single point crossover. There are many alternative crossover methods available, the most well known one is two points crossover. Rather than select one crossover point and swap the tail of the two chromosomes, two point crossovers selects two random points in the chromosomes and swaps the information in between.

In genetic algorithm, mutation can be realized with a certain probability of random deformation of the strings. The encouraging effect seen is preservation of genetic diversity and, by this effect, stuck up or the local maxima can be evaded. The mutation rate indicates the probability that a cell will be changed. Thus, the expected number of mutations per individual is equal to the mutation rate multiplied by the length of an individual. If a cell is selected to be mutated, then either the processor number or the task of that cell will be randomly changed.

The advantage of GA is it does not require any derivative information, which may not be available for many real-world problems. Compared to traditional methods it is found to be faster and rather more efficient. GA can find application in optimization of both continuous and discrete functions and, also multi-objective problems. GA gives good number of solutions and not just a possible solution hence is useful in system having large search space and involves large number of parameters.

The GA is not suitable for all problems, as in simple and having derivative information. In GA fitness value is calculated repeatedly which might be computationally expensive for some problems. Being a stochastic process, there are no guarantees on the optimality or the quality of solution. Also, if not implemented properly, it may not converge to the optimal solution.

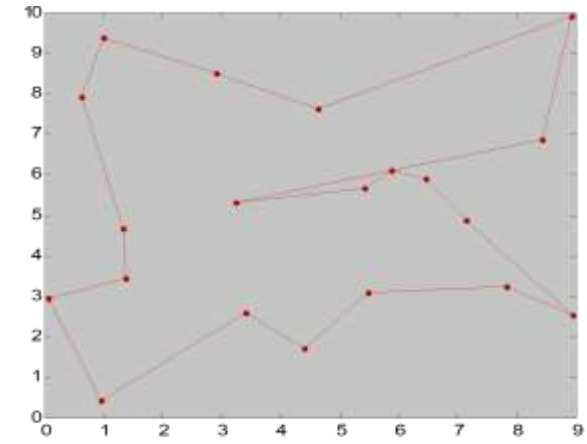
4. Results and discussions

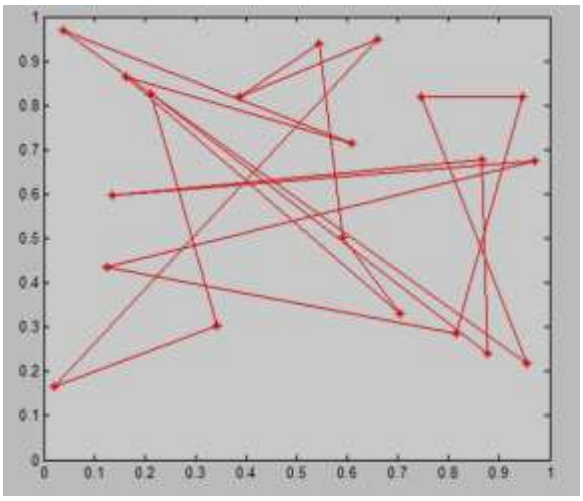
In order to evaluate the heuristic algorithms for task allocation the experiment was run on Intel core 2 duos, 2.6Ghz PC with travelling salesman problem mapping. Here, we considered 20 nodes that are randomly placed over which the tasks were scheduled. The allotment of tasks, which are non-preemptive, over the nodes, was optimized by first with Greedy Nearest Neighbor algorithm and found to be pretty fast in allotment but it was not the optimal solution. Although the best solution is obtained with fixed iterations, and every time new best value is obtained. Thus, we can say that Greedy Nearest Neighbor with good sorting process found to be suitable for less number of nodes, as the nodes increases lesser to the reach of optimal solution.

In the Simulated Annealing process the potential issues were dependent upon computation time, probability of acceptance, local optima, and cooling schedule. For better results the cooling should be slow, here it was kept at 0.97. Thus, the computation time increased progressively as the nodes increased. If the initial temperature is high the iterations have to be increased, so also the computation time increases. The probability of acceptance depends upon the temperature and its cooling schedule, initial temperature of 2000 value was considered and maximum iterations were limited to 2000, and then found that computation time increased as the nodes were increased from 20 to 100. If we add more constraints to the problem like disintegration of faulty nodes and erroneous data of electronic module, more local optima are created. With this problem the probability that solution may be struck in one of the local optima and chance of achieving the best results i.e, global optima is reduced. As we ran progressively

the schedule pattern changed due to the above problem.

When we used the Genetic Algorithm for the above problem, we found the computation time is little bit high and obtained the best solution, which is the optimal solution. The results obtained show that approximately 15 - 60% scheduling time is reduced. This was due to increased number of nodes from 20 to 100 with same constraints. The figure below shows the scheduled distances for 20 nodes.





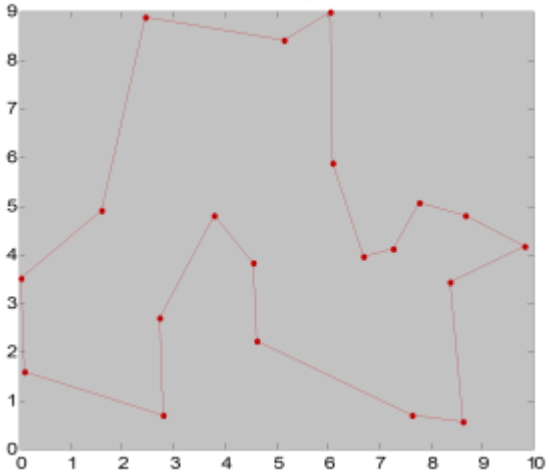


Figure 2: Best solution with (a) Greedy nearest neighbor (b) Simulated Annealing and (c) Genetic Algorithm

5. Conclusions:

From the above discussions it is found that the best solution for optimized allocation of tasks to the nodes depends upon the number of nodes and number of tasks. The experiments were carried out with the nodes providing composability i.e, fail safe property of the architecture. With fewer nodes and less constraints, the scheduling can be simpler with Greedy algorithm. Whereas for large number of nodes the simulated annealing can be used, although constraints increases the computation time increases and it may not yield to global optimum solution. The best results are with use of Genetic algorithm, even though the number of iterations it takes is large but best result can be obtained, which can be the optimal solution. The convenient methods search from a single point, whereas GA operates on whole population of points (strings). This makes it more robust and chances of reaching to the global optimum. Thereby reducing the risk of trapped in a local optimum point. If multiple schedules are used, the scheduling time can be decreased drastically by using Genetic heuristic algorithm.

Acknowledgements

The author gratefully acknowledges Annamacharya Institute of Technology & Sciences, Rajampet, India for the support given to carry this work.

References

1. Mike Falla (edt), Advances in safety critical Systems-Results and achievements from the DTI/EPSRC R&D Program 1997. P.1-2.
2. Martin L Shooman, Reliability of Computer

Systems and Networks. John Wiley 2000. P.14-15.
3. A. Avizienis, J Laprie and B Randell, Fundamental Concepts of Dependability, Proc of 3rd Info Survivability Workshop 2000. P. 7-12.
4. B Abdul Rahim, and K Soundara Rajan, Fault Tolerance in Real Time Systems through Time Triggered Approach, CIIT IJ of DSP, Vol 3(3), 2011. P.115-120.
5. H kopetz and G Gruenstiedl, TTP- A Protocol for Fault Tolerant Real Time Systems, IEEE Computer, Vol 24(1), 1994. P. 14-23.
6. H kopetz, et.al, A Synchronization strategy for a TTP/C Controller, SAE paper 960120, SAE press Warrendale, 1996. P.19-27.
7. SAE: Class C Applicatioon Requirements – J2056/1, SAE Handbook, SAE Press Waarendale, 1994. P. 23.366-23.372.
8. H kopetz and R Nossal, The Cluster Compiler – A Tool for the Design of Time Triggered RTS, ACM SIGPLAN Workshop, 1995.
9. C Scheidler, L J Schafers and O K Fuhrmann, Software Engineering for parallel systems: The TRAPPER Approach, HICCS-28, IEEE CS Press, 1995. P. 349-358.
10. C Scheidler, et.al, Time Triggered Architectures, EMMSEC'97, Advances in information Technologies: The business challenge, IOS Press, 1997. P. 758-765.
11. R W Eglese, Simulated annealing: A tool for operation research, Euro. J of Operation Res., Vol.46, (1990), 271-281.
12. K Krishna, K Ganeshan and D Janakiram, Distributed Simulated Annealing algorithms for job shop scheduling, IEEE Transactions on systems, Man and Cybernetics, Vol 25(7), 1995. P. 1102-1109.
13. Andrew D Martin and Kevin M Quinn, A review of discrete optimization algorithms, The Political Methodologist, Vol 7(2), spring 1996. p.6-10.
14. G R Harik, F G Lobo and D E Goldberg, The compact Genetic Algorithm, IEEE Transactions on Evolutionary Computation, Vol 3(4), 1999. P. 287-297.

