ISSN: 2581-4311
DOI: doi.org/10.51219/IJCRSET/T.S.M-Priyankaa/3

URF PUBLISHERS
connect with research world

International Journal of Current Research in Science, Engineering & Technology

https://urfpublishers.com/journal/ijcrset

Vol: 6 & Iss: 4

Research Article

Simulation of Priority Driven Algorithms to Schedule Real-Time Systems

T.S.M. Priyankaa*, S.M.K. Chaitanya

Department of Electronics and Communication Engineering, G.V.P. College of Engineering (Autonomous), Visakhapatnam, Andhra Pradesh, India

Citation: Priyankaa TSM, Chaitanya SMK. Simulation of Priority Driven Algorithms to Schedule Real-Time Systems. *Int J Cur Res Sci Eng Tech* 2023; 6(4), 7-9. DOI: doi.org/10.51219/IJCRSET/T.S.M-Priyankaa/3

Received: 03 October, 2023; **Accepted:** 07 October, 2023; **Published:** 09 October, 2023

***Corresponding author:** T.S.M. Priyankaa, Department of Electronics and Communication Engineering, G.V.P. College of Engineering (Autonomous), Visakhapatnam, Andhra Pradesh, India, Email: priyatutika@gmail.com

A B S T R A C T

Real-time task scheduling is the process which decides the order of execution in which various tasks are to be executed by the Operating System (OS). Every OS employs task schedulers for the purpose of scheduling. Every scheduler operates according to a certain scheduling algorithm. An optimal scheduling algorithm is the one which has the ability to schedule all the feasible task sets. A number of scheduling algorithms have been employed so far. But since the aim of this work is to schedule real-time systems, priority is the key factor that should be considered. So, this work is focused on simulating the priority driven algorithms and evaluating their performance with respect to each other. These algorithms are simulated by a tool called YARTISS which is a real-time multiprocessor scheduling simulator.

Keywords: Data flow graph (DFG); Directed acyclic graph (DAG); Real-time systems; Scheduling algorithms; Task scheduling

1. Introduction

Real-time systems mostly operate by scheduling algorithms that are priority driven. Priority driven scheduling algorithms are categorized into static and dynamic priorities. An example of static priority is the Rate Monotonic Scheduling (RMS) algorithm. The Dynamic priority algorithms are again classified into job level fixed priority and job level dynamic priority. An example of job level fixed priority is the Earliest Deadline First (EDF) algorithm and an example of job level dynamic priority is the Least Laxity First (LLF) algorithm.

During the fundamental stage of designing a system, the dependency between the components is represented by a Data Flow Graph (DFG). The Dataflow Graph is a natural model for describing signal processing systems. It consists of actors called as nodes or vertices and arcs called as edges. Each node in the DFG is considered as a periodic task that has to be scheduled and edges specify the computational flow of the tasks. The algorithms proposed to schedule the system having periodic task dependencies are Rate Monotonic Scheduling (RMS) and Earliest Deadline First (EDF). But the cycles present in the DFG makes it impossible for the scheduling of overall system. So, the system has to be represented in the model of a Directed Acyclic Graph (DAG). The algorithm used to schedule DAG task model is Least Laxity First (LLF). A tool called as YARTISS is used to test, compare and evaluate real-time scheduling algorithms.

Imane Hafnaoui et al[1] presented a methodology for scheduling real-time tasks by transferring the system's dataflow graph into a task graph. He also explains that the objective metrics can be optimized easily by scheduling a system under preemptive RMS. Y.Chandarli et al[2] presented the YARTISS tool and explained the functionality and practical approach to schedule different algorithms. C.L.Liu et al[3] explains the static and dynamic scheduling algorithms and proved that the processor utilization can be obtained by assigning the priorities dynamically. E.Bini et al[4] explains how the different methods proposed for evaluating the performance of priority scheduling algorithms affects the results. A.Saifullah et al[5] explains the scheduling of parallel real-time tasks. Different scheduling algorithms are then applied with some parameters like deadline, release times etc. to the real-time tasks. F.Fauberteau et al[6] explained the Global scheduling of sporadic task sets with constrained deadlines for homogeneous processors. This approach assigns an offset value and deadline for all the subtasks. R.I.Davis et al[7] outlines real time scheduling results independently without considering any scheduling algorithms. It also provides the classification of scheduling methods and various evaluation metrics for comparison. M.G.Harbour et al[8] presented analysis for hard real-time tasks using fixed-priority methods. A schedulability determination method for all the tasks is also presented. This method is applicable to uniprocessor systems which have complex priority structures. Sangchul Han et al[9] concluded that the finish time can be predicted if the actual execution times are time can be predicted if the actual execution times are Known. S.K.Baruah[10] presented the scheduling tests for sporadic task systems on multiprocessor platforms. Shortcomings of the present tests leading to performance degradation are identified and new tests are proposed to overcome the limitations.

2. Real-Time Scheduling Algorithms

Some of the scheduling algorithms that are applicable in real-time are RMS, EDF and LLF. Each of the three algorithms is explained in detail in this section.

2.1 Rate Monotonic Scheduling Algorithm (RMS)

This is a static priority algorithm with pre-emption. The priorities are assigned as a monotonic function of periodic process i.e., the shortest period has the highest priority. Since the periods are constant, it is called as fixed priority algorithm. The schedulability condition for RMS algorithm isgiven by

$$\sum_{i=1}^{n} \frac{Ci}{Pi} \leq n(2^{\frac{1}{n}} - 1)$$

The LHS of the inequality is the total CPU utilization for ntasks, where

Ci = Execution time (CPU burst) Pi = Period of task Ti

The tasks are schedulable for the above condition assuming that

1. All the tasks should start at t=0.
2. There are no data dependencies between the processes.
3. The execution time is constant for a process.
4. All the tasks have to be periodic.

2.2 Earliest Deadline First Algorithm (EDF)

This is a dynamic priority scheduling algorithm. Here, the priority of tasks changes during the run time. This algorithm assigns priority in the order of deadline. At any instant, the highest priority task is the one that has the closest deadline. This algorithm is optimal for uniprocessor systems. The necessary condition for schedulability is that the CPU utilization must be less than 1 i.e.

$$\sum_{i=1}^{n} \frac{Ci}{Pi} \leq 1$$

Where Ci = Execution time (CPU burst) Pi = Period of task Ti

2.3 Least Laxity First Algorithm (LLF)

This algorithm is also called as Least Slack Time first (LST) or Earliest Deadline As Late As Possible (EDL) algorithm. This is a dynamic preemptive scheduling algorithm. The priorities are dynamic functions of laxities

i.e. the task with the least laxity is assigned the highest dynamic priority.

The laxity or slack of a real-time task Ti at time *t* is given by

$$L_i(t) = D_i(t) - E_i(t) - t$$

Where Di(*t*) = deadline of the task and Ei(*t*) = remaining execution time

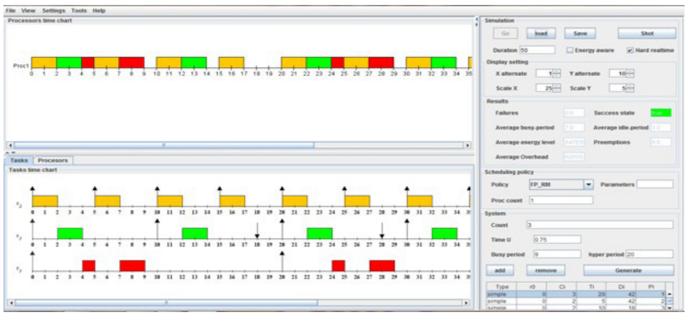
3. Results and Discussion

The simulated results for the above algorithms are shown below. Here, the Least Laxity First algorithm is the optimal algorithm with respect to the other two algorithms as it has 100% schedulability bound.

3.1 RMS Algorithm

3.1.1 Results for a Uniprocessor

The RMS algorithm is simulated using YARTISS software and the results are presented below for a uniprocessor and a multiprocessor system (**Figures 1& 2**):



Duration 50 ☐ Energy aware ☑ Hard realtime

Display setting: X alternate 1, Y alternate 10, Scale X 25, Scale Y 5

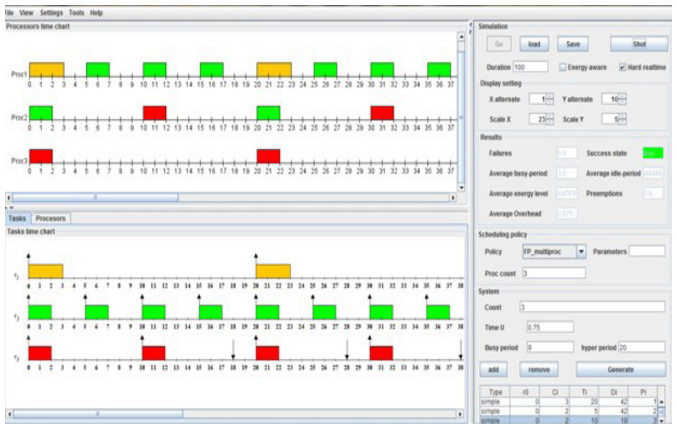
Results: Failures 0.0, Success state true, Average busy-period 7.0, Average idle-period 1.0, Average energy level [illegible], Preemptions 0.0, Average Overhead [illegible]

Scheduling policy: Policy FP_RM, Parameters, Proc count 1

System: Count 3, Time U 0.75, Busy period 9, hyper period 20

Type	r0	Ci	Ti	Di	Pi
simple	0	3	20	42	1
simple	0	2	5	42	2
simple	0	2	10	18	3

Figure 1: RMS results for a uniprocessor.



Duration 100 ☐ Energy aware ☑ Hard realtime

Display setting: X alternate 1, Y alternate 10, Scale X 23, Scale Y 5

Results: Failures 0.0, Success state true, Average busy-period 3.0, Average idle-period [illegible], Average energy level [illegible], Preemptions 0.0, Average Overhead 0.075

Scheduling policy: Policy FP_multiproc, Parameters, Proc count 3

System: Count 3, Time U 0.75, Busy period 9, hyper period 20

Type	r0	Ci	Ti	Di	Pi
simple	0	3	20	42	1
simple	0	2	5	42	2
simple	0	2	10	18	3

Figure 2: RMS results for a multiprocessor.

3.2 EDF Algorithm

3.2.1 Results for a Uniprocessor

The EDF algorithm is simulated using YARTISS software and the results are presented below for a uniprocessor and a multiprocessor system (**Figures 3&4**).

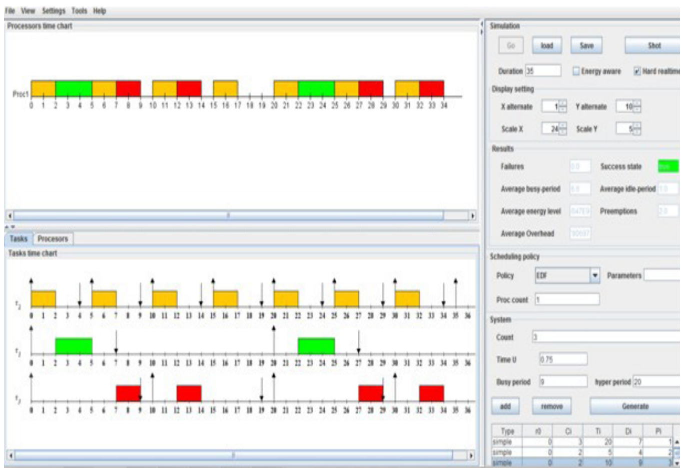


Figure 3: EDF results for a uniprocessor.

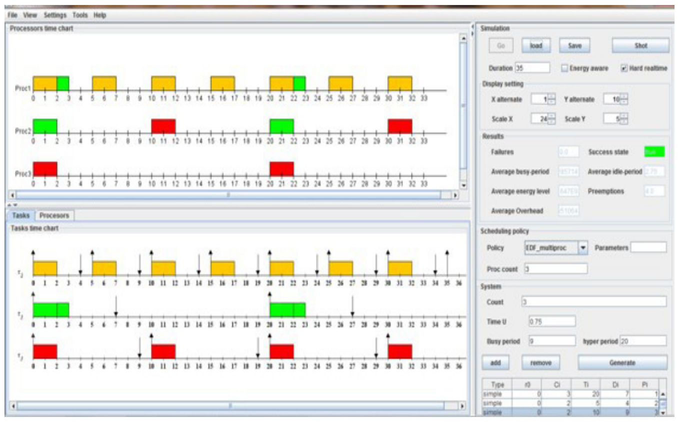


Figure 4: EDF results for a multiprocessor.

3.3 LLF Algorithm

The LLF algorithm is simulated using YARTISS software and the results are presented below (**Figure 5**):

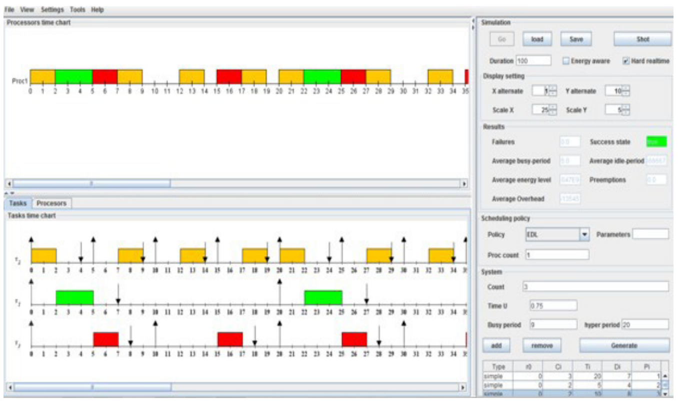


Figure 5: LLF results.

4. Comparison

The comparison of the above three scheduling algorithms with respect to different performance metrics is shown in the table below (**Table 1**).

It is evident from the table that the LLF algorithm is the optimum algorithm compared to RMS and EDF algorithms. This is because LLF algorithm has higher busy period, less idle period, less number of preemptions, more CPU utilization and less number of overheads which are required for an optimal scheduling algorithm.

Table 1: algorithms with respect to different performance metrics.

ALGORITHM	SCHEDULING CRITERIA	CPU UTILIZATION	BUSY PERIOD	IDLE PERIOD	OVERHEADS	PREEMPTIONS
RMS	Period	Less	0.33333	0.81819	66668	10
EDF	Deadline	Full	0.33333	0.81819	91342	5
LLF	Laxity	Full	5	0.66667	63745	0

5. Conclusion

In this paper, three algorithms were presented to schedule real-time systems. Among those, RMS and EDF are applicable when the system is represented as a Data Flow Graph (DFG) i.e. during the fundamental stage of its design and LLF algorithm is applicable after remodeling the system's DFG into a Directed Acyclic Graph (DAG).

6. References

1. Hafnaoui I, Ayari R, Nicolescu G, Beltrame G. Scheduling real-time systems with cyclic dependence using data criticality. Design Automation for Embedded Systems 2017;1(2):117-136.
2. Chandarli Y, Fauberteau F, Masson D, Midonnet S, Qamhieh M. Yartiss: A tool to visualize, test, compare and evaluate real-time scheduling algorithms. In WATERS 2012;21-26.
3. Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. JACM 1973;20(1):46-61.
4. Bini E, Buttazzo GC. Measuring the performance of schedulability tests. Real-Time Systems 2005;30(2):129-154.
5. Saifullah A, Ferry D, Li J, Agrawal K, Lu C, Gill CD. Parallel real-time scheduling of DAGs. IEEE Transactions on Parallel and Distributed Systems 2014;25(12):3242-3252.
6. Qamhieh M, Fauberteau F, George L, Midonnet S. Global EDF scheduling of directed acyclic graphs on multiprocessor systems. In Proceedings of the 21st International conference on Real-Time Networks and Systems 2013;287-296.
7. Davis RI, Burns A. A Survey of Hard Real- Time Scheduling Algorithms and Schedulability Analysis Techniques for Multiprocessor Systems. University of York Technical Report. YCS 2009;443.
8. Harbour MG, Klein MH, Lehoczky JP. Timing analysis for fixed-priority scheduling of hard real- time systems. IEEE Transactions on Software Engineering 1994;20(1):13-28.
9. Han S, Park M. Predictability of least laxity first scheduling algorithm on multiprocessor real- time systems. In International Conference on Embedded and Ubiquitous Computing. Springer 2006;755-764.
10. Baruah S. Techniques for multiprocessor global schedulability analysis. In Real-Time Systems Symposium. IEEE International 2007;119-128.