Middle-East Journal of Scientific Research 7 (6): 837-843, 2011
ISSN 1990-9233

Unit-Quantum Scheduling for Non-Preemptive Real-Time Tasks

[1]Nasro Min-Allah, [1]Muhammad Mustafa, [1]Sajjad Mohsin and [2]Yongji Wang

[1]Comsats Institute of Information Technology, Park Road, Islamabad, Pakistan
[2]Laboratory for Internet Software Technologies, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, China

Abstract: A lot of efforts have been made to propose variant of scheduling algorithms for real-time systems to achieve higher system utilization on different platforms. From uni-processor perspective, for higher system utilization, dynamic scheduling is the optimal choice and Earliest Deadline First (EDF) is the default algorithm belongs to this class. Under EDF scheduling, existing strategies can be divided into two categories; Preemptive and Non-preemptive. Preemptive EDF scheduling is guaranteed to achieve 100% CPU utilization, whereas Non-preemptive EDF compromises CPU utilization due to its implicit characteristics. In this paper, we address the low CPU utilization issue associated with Non-preemptive scheduling by proposing a unit-quantum scheduling strategy that results in improved CPU utilization on uniprocessor systems. The supremacy of the work over existing counterparts is shown through example task sets.

Key words: Real-Time Systems · Dynamic Scheduling · Quantum-based scheduling · Unit-quantum scheduling

INTRODUCTION

Due to its importance, the scheduling problem has been studied by mathematicians and computer scientists for several decades. Scheduling of tasks has become more critical issue in the filed of real-time systems, where missing a deadline may bring undesired performance in general and ruthless consequences in case of hard real-time systems. An agreed scheduling mechanism in hard real-time systems is priority driven, where a fixed or dynamic priority is assigned to individual tasks. For system utilization point of view, dynamic priority task scheduling is more appealing and is preferred to achieve higher throughput. The most promising scheduling algorithm in dynamic class is the Earliest Deadline First (EDF), which can further be divided into (i) Preemptive EDF scheduling and (ii) Non-preemptive EDF scheduling.

Inherently, preemption is the fundamental property that controls the difficulty of the scheduling problem. If preemption is allowed, the problem is known to be solvable in polynomial time using algorithms such as EDF [1]. Whereas non-preemptive scheduling is an NP-complete problem and it is believed that an optimal polynomial-time scheduling algorithm does not exist. As far as the real-time scheduling is concerned, mostly tasks are assumed to be preemptive, but in some situations non-preemptive execution is preferable. For example, if the system needs to support critical sections, preemption is not allowed. Handling such requirements using a preemptive system, typically involves additional overheads such as using the priority ceiling protocol [2] while a non-preemptive system fulfills the need without a particular mechanism. EDF is optimal in preemptive case [1], while in non-preemptive counter part EDF is no more optimal. The basic problem with EDF is that it is unable to consider future task arrivals. Apart from the fact that EDF is non-optimal for non-preemptive systems, usually it can guarantee to schedule more task sets than other counterparts having the same complexity [3].

In this paper we show that it is possible to enhance the performance of non-preemptive EDF (NP-EDF) by splitting the tasks into smaller non-preemptable subtasks (threads) each of unit quantum size (supposing that unit quantum will be sufficient for critical sections within the tasks). The resulting algorithm is called Unit-quantum based Non-preemptive EDF (UQ-NP-EDF) scheduling strategy and is guaranteed to schedule as many task sets as preemptive EDF.

Rest of the Paper Follows As: In section 2, we provide the related background work. Section 3 describes the problem formulations and notations used in our task model.

Corresponding Author: Nasro Min-Allah, Comsats Institute of Information Technology, Park Road, Islamabad, Pakistan

Section 4, provides the details of the proposed solution. In Section 5, we demonstrate the dominancy of our technique by applying it to task examples and derive a sufficient schedulability condition for component tasks. We conclude in Section 6.

Background: EDF is the most discussed dynamic-priority scheduling algorithm in the research community [1-16]. The absolute deadline of a task is the sum of its arrival time and its relative deadline. At every time instant, EDF selects a ready task for execution having its absolute deadline earliest among all the others. In case, two or more tasks have the same deadline, the choice for next execution is made randomly. Every time a task instance is released the tasks periodicities may change at run-time depending on the closeness of their absolute deadlines, this factor makes EDF a dynamic-priority scheduling algorithm [12]. On a single processor system EDF algorithm is optimal for the preemtable task sets provided that there is no resource, precedence, or mutual exclusion constraints.

***Theorem 1* [1]:** *A set of periodically recurring tasks, sorted in the increasing order keyed by period, can be feasibly scheduled under a preemptive EDF scheduling algorithm iff:*

$$U = \sum_{i=1}^{n} \frac{e_i}{P_i} \leq 1 \tag{1}$$

In practice, tasks may be independent or dependent depending on whether they use to share resources and/or variables. Thus, during execution whenever a task needs to access and modify a shared variable or uses a shared resource it has to contain critical sections that cannot be interrupted. These critical sections are implemented either using the non-preemtable task scheduling or with non-preemtable subtasks scheduling. An important objective of non-preemptive scheduler is to reduce the context-switching time and task waiting time. Non-preemptive version of neither fixed nor dynamic priority uniprocessor algorithms is optimal. However, [5] reported a feasibility test for NP-EDF scheduling algorithm.

***Theorem 2* [5]:** *A set of periodically recurring tasks, sorted in the increasing order keyed by period, can be feasibly scheduled under a non-preemptive EDF scheduling algorithm iff:*

$$\sum_{i=o}^{n} \frac{e_i}{P_i} \leq 1 \tag{2}$$

$$e_i + \sum \left\lfloor \frac{t}{P_j} \right\rfloor e_j \leq t, \forall i, (1 < i \leq n), \forall t, (P_1 \leq t \leq P_i), \tag{3}$$

$$where\ t \in S_i = lP_j, \forall j, (j = 1,...,i), \forall l, \left(l = 1,...,\left\lfloor \frac{P_i}{P_j} \right\rfloor \right)$$

However, it can be seen that feasibility of a task has to be tested at a number of points which makes it impractical for online systems.

Since EDF is optimal only for preemtable task sets. Thus, without preemption, it is impossible to transform a feasible non-EDF based schedule into an equivalent EDF based schedule by interchanging computation blocks of different tasks as described in the proof of EDF optimality. This means that even if some other scheduling algorithm can produce a feasible schedule for a given task set, the EDF scheduler may miss a deadline of the same task set. Similarly, using fixed-priority schedulers for non-preemptive tasks set may cause the priority inversion problem [2]. According to which a higher-priority task is blocked for unlimited or very long period of time by a low-priority task having a critical section. In fact, no priority-based scheduling algorithm is optimal for non-preemtable tasks with arbitrary start times, computations times and deadlines, even on a single processor system [11].

As reported in [4], a number of efforts have been made to improve the schedulability of fixed priority schedulers such as quantum-based scheduling and preemption threshold scheduling. The former sets up a partially non-preemptive region when combined with priority-based scheduling whereas the latter introduces priority inversion [9] during task execution. To avoid the higher utilization issue, involved with Theorem 2, authors in [4] provided an analysis on quantum based scheduling. As defined in [6]: "Under quantum-based scheduling, processor time is allocated to tasks in discrete time units called quanta. When a processor is allocated to some task, that task is guaranteed to execute without preemption for q time units, where q is the length of the quantum, or until it terminates, whichever comes first." ". . . non-preemptive and preemptive systems abstractly can be viewed as the extreme endpoints in a continuum of quantum-based systems: a non-preemptive system results when q = ∞ and a fully preemptive system results when q = 0." Applications of quantum-based scheduling to dynamic priority scheduling have been reported in [6] for efficient object sharing, in [7] for flow protection in communication and in [8] to reduce the scheduling overhead.

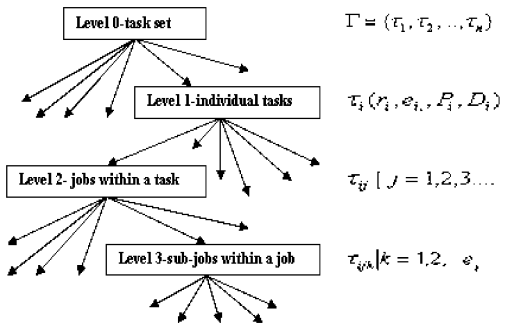

Fig. 1: Hierarchy levels of task set, task, job and sub-job

The quantum-based scheduling techniques discussed in [4, 6-8] result in a suboptimal quantum for task execution, however, the associated complexity is high and impractical for online systems. In this paper, we further address the issue and propose unit quantum based scheduling approach that minimizes the complexity and results in maximum possible system utilization with improved schedulability of dynamic priority non-preemptive periodic tasks. We show that a task set that is not schedulable by NP-EDF scheduling can be schedulable with the proposed technique.

Problem Formulations and Notations

Definition: Under unit-quantum scheduling, every task from a set of non-preemptive periodic tasks set is divided into as many sub-tasks (or threads) as its WCET so that when processor is allocated to a sub-task, it is guaranteed to execute without preemption.

We consider that the system has a task set of n periodic tasks denoted as $\Gamma = \{\tau_1, \tau_2, \ldots \tau_n\}$ to be scheduled on a uni-processor system. We assume that all tasks are independent and non-preemptive following the implicit-deadline model (i.e. for each task period = deadline). The tasks in the system are supposed to enter online having non-zero integral parameters and the task instances (i.e. jobs) can be divided into sub-tasks (or threads) each of unit-quantum length. Also, they need no resource (such as shared data structures) other than a processor for execution. Similarly, the cost associated with context switching etc. is ignored and it is assumed that there is infinite number of priority levels available. All the jobs of a task must be executed sequentially i.e. every job of τ_i is not allowed to run before the preceding job of τ_i completes. Each task τ_i from the task set is characterized by a set of τ_i (r_i, e_i, P_1, D_i) where (i) r_i is its release time (or arrival time) (ii) e_i its predicted worst-case or average execution time (iii) P_i is the period i.e. the task τ_i releases a (potentially infinite) sequence of jobs after every P_i units of time and (iv) D_i is the deadline i.e. each job of τ_i must complete e_i units of execution within at most D_i time units from the release r_i. The task set is intrinsic i.e. $P_i = D_i$ for all tasks.

As unit-quantum based scheduling divides a job into sub-jobs, therefore, the notation used some what differs from normal notation. Figure 1 depicts relation between the terms, where level 0 is the set of all tasks (Γ), level 1 is the individual tasks (τ_i), individual jobs of a task ($J_{i,k}$ k = 1,2,3...) are defined at level 2 and finally level 3 comprises of all the sub-jobs (or threads) within a task instance.

A sub-job is denoted as $S_{i,j,k}$ (where i indicates the i-th task, j the j-th job of i-th task and k the k-th sub-job of the j-th job). Sub-jobs are the entities actually scheduled and run by the scheduler. A job is said to be completed only when all of its threads finish their execution. This makes the system job-level dynamic (or unrestricted dynamic).

Since all sub-jobs are uniformly comprise of unit-quantum with WCET=1 and do not play any other role, therefore, they are characterized by only 2 parameters; release time and absolute deadline i.e.

$$S_{i,j,k}\,(r_{i,j,k}, d_{i,j,}) \;\forall k = 1,2,\ldots., e_i \quad (4)$$

where
$$S_{i,j,k} = r_{i,j,} \text{ for } S_{i,j,k} \quad (5)$$

$$r_{i,j,q} = r_{i,j,p} + 1 | S_{i,j,p} \; Precedes \; S_{i,j,q} \quad (6)$$

and
$$d_{i,j} = r_{i,j} + D_{i,j} \quad (7)$$

Whereas the release time ensures the strict execution order of all sub-jobs within a task instance and absolute deadline associates dynamic priority to the sub-jobs to be used by EDF scheduler. Note that during the process of decomposition, every sub-job inherits its deadline from the parent job (task instance) while having different release time.

Proposed Solution: Run-time scheduling refers to the process of determining, during the execution of a real-time application system, which job(s) should be executed at each instant in time. Normally, a run-time scheduling algorithm is implemented as follows: At each time instant, assign a priority to each active job and allocate the available processor(s) to the highest priority job(s). In order to meet the requirements of online real-time systems, real- time scheduling needs not only to satisfy and achieve (at least) the known utilization bounds,

but also design efficient run-time dispatching algorithms. Unit-quantum based scheduling is such an endeavor. Since the proposed solution is supposed to be used for online real-time systems, an admission controller is required to perform an acceptance test on every new task entering in the system.

Under unit-quantum scheduling, every task from a set of non-preemptive periodic task set is divided into as many sub-tasks (threads or sub-jobs) as its WCET so that when processor is allocated to a sub-task, it is guaranteed to execute without preemption. Thus, the scheduling entities are sub-tasks having uniform (i.e. unit) CPU requirement. Upon release of j-th instance of a task τ_i (i.e. $\tau_{ij} \backslash j$ = 1,2,3,..), it is decomposed into a set of sub-jobs $\{\tau_{i,j,k} \backslash k = 1,2,....e_i\}$ which are scheduled using NP-EDF. The completion of a job occurs upon completion of its last sub-job. Any time a job may be in active or passive state. In active state, a job releases sub-jobs (or threads) successively at every time slot t, starting from its own release time until its last sub-job is released, after which it enters into passive state. A job remains in passive state till its completion.

At run-time, a newly arrived non-preemptive task τ_i is admitted into the system only if it is found feasibly schedulable by EDF, means it does not violate the system's utilization bound i.e. $U = \sum_{i=1}^{n} \frac{e_i}{P_i} \leq 1$. The pseudo-code for acceptance test is given in Algorithm 1. The global boolean variable UB (utilization bound) is initialized to false and U (utilization) to 0 when the system starts.

Algorithm 1: Accept Task

procedure AcceptTask(Γ,τ_i)
if (UB = true) then return end if Compute $Temp = U + \frac{e_i}{P_i}$

if ($Temp \leq 1$) then accept the task as a member of Γ U = Temp if $Temp = 1$ UB = true end if end if end procedure

Algorithm 2 depicts the pseudo-code of UQ-NP-EDF scheduler which is invoked after every time slot t. It maintains three queues; an active job queue (AJQueue), a passive job queue (PJQueue) and a thread queue (TQueue) to keep the lists of active jobs, passive jobs and sub-jobs/threads, respectively. Whenever the processor becomes free for assignment, the thread with earliest absolute deadline will be dispatched for execution while ties are broken arbitrarily. Because of unit-quantum length, every thread occupies the processor non-preemptively only for a single quantum.

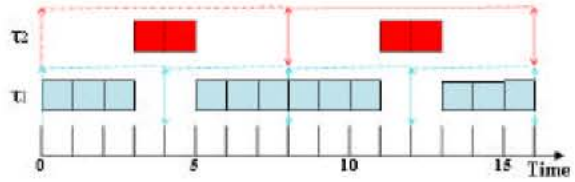

Fig. 2: UQ-NP-EDF schedule of tasks τ1(0, 3, 4, 4) and τ2 (0, 2, 8, 8)

Algorithm 2: UQ-NP-EDF

procedure UQ-NP-EDF( )
for every task $\tau_i \in \Gamma$
if $(\frac{t}{\tau_i.P_i} = 0)$ then Create its jth job τ_{ij}

Enqueue(AJQueue, τ_{ij}) endif endfor for every job τ_{ij} in AJQueue i.e. $\forall \tau_{ij} \in AJQueue$
Create its kth sub-job $\tau_{ij,k}$, ($\tau_{i,j,k}$ $d_{i,j}$), such that

$r_{i,j,1} = r_{i,j} otherwise r_{i,j,k} = r_{i,j,k-1} + 1, and\ d_{i,j} = r_{i,j} + D_{i,j}$

Enqueue(TQueue ($\tau_{i,j,k}$))
if $\tau_{i,j,k}$ is the last sub-job of τ_{ij}, (i.e. k=e_i), then
Dequeue(AJQueue (τ_{ij}))
Enqueue(PJQueue (τ_{ij}))
endif
endfor
if (TQueue $\neq \phi$)
Find a sub-job $S_{ij,k} | S_{ij,k}.d_{ij} = \min\{\tau_{ij,k}.d_{ij} \backslash \tau_{ij,k} \in TQueue$
Dequeue (TQueue, $S_{ij,k}$)
if $S_{ij,k}$ is the last sub-job of its parent job, then
Enqueue(PJQueue, $S_{ij,k}$)
endif
Run $S_{ij,k}$
endif end procedure

Motivating Examples: In order to demonstrate the working of UQ-NP-EDF algorithm, consider a periodic task set comprising of two tasks τ1(0, 3, 4, 4) and τ2 (0, 2, 8, 8). The task set is feasibly schedule because $U = \frac{3}{4} + \frac{2}{8} = 1$.

Figure 2 demonstrates the corresponding schedule of the task set where the release times and deadlines of jobs/sub-tasks (or threads) are represented by up and down arrows, respectively.

Theorem 3: A non-preemptive EDF schedulable task set is also schedulable using unit-quantum based non-preemptive scheduling on a single processor.

Proof: As stated in [10], "....owing to the general optimality of the preemptive of the preemptive EDF

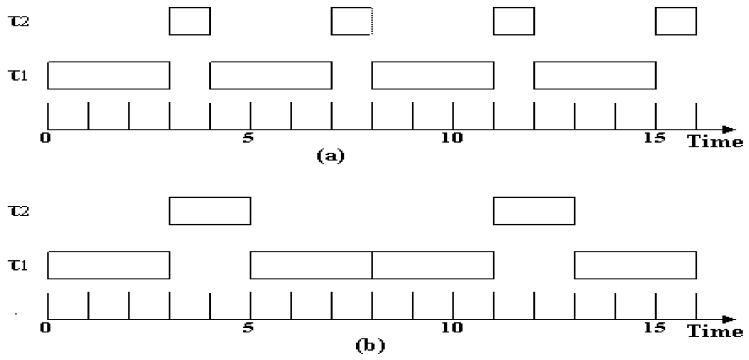

Fig. 3: (a) EDF (b) NP-EDF schedule of tasks τ1(0, 3, 4, 4) and τ2 (0, 2, 8, 8)

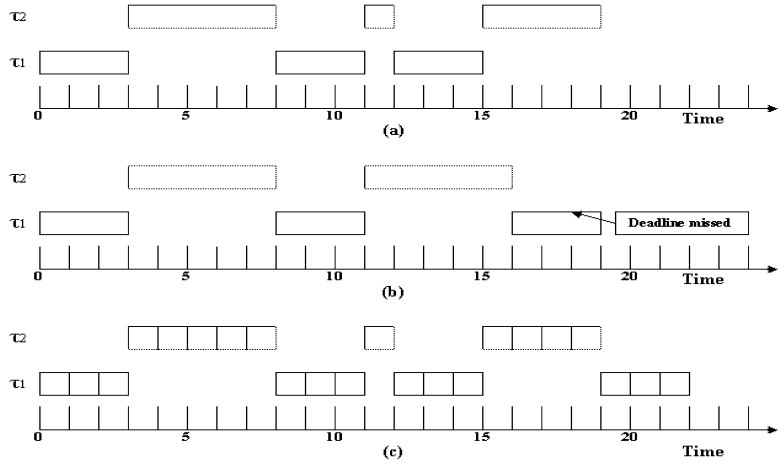

Fig. 4: (a) EDF, (b) NP-EDF and (c) UQ-NP-EDF schedules of tasks τ1(0,3,6,6) and τ2 (0,5, 10,10)

algorithm with respect to uniprocessor scheduling, the feasibility of a task set under non-preemptive EDF scheduling implies the feasibility of the same task set under preemptive scheduling. The opposite is not true." Since unit-quantum based scheduling reduces the continuum scheduling space of every task from e_i to 1, thus if a non-preemptive task set is feasibly schedulable by NP-EDF, naturally it will be more feasibly schedulable using unit-quantum.

Figure 4 and Figure 5 show schedules of the task sets given in their caption. In both cases, the NP-EDF misses a deadline and conveys its non-schedulability whereas EDF and UQ-NP-EDF feasibly schedule the task sets.

Keeping in view these schedules, we can have the following theorem.

Theorem 4: If an implicit deadline task set $\Gamma = (\tau_1, \tau_1, \ldots \tau_n\}$ where τ_1 for every task $\tau_1 \in N$ is unschedulable using NP-EDF on a single processor, can be feasibly scheduled by UQ-NP-EDF iff: $U = \sum_{i=1}^{n} \frac{e_i}{P_i} \leq 1$

Proof: Since under UQ-NP-EDF scheduling every task is subdivided into sub-tasks each of unit quantum size, Q=1. As e_i is integer

$\Rightarrow Q_i \leq e_i$ where $j = 1,2,\ldots|e_i|$

$$\Rightarrow \frac{Q_j}{P_i} \leq \frac{e_i}{P_i}$$

$$\Rightarrow u_i^{/} \leq u_i$$

Also $Q_j \leq \sum Q_j \leq e_i$

This shows our algorithm presents lower load than the older scheme i.e. $u_i \leq \frac{e_i}{P_i}$. Even our test never fails when compared with optimal test, as it generates 100% CPU utilization. While, according to Theorem 2, the existing NP-EDF test is not optimal and the second part always restricts its utilization less than 100%. This shows our test schedules the task set which is not schedulable with NP-EDF. This completes the proof.

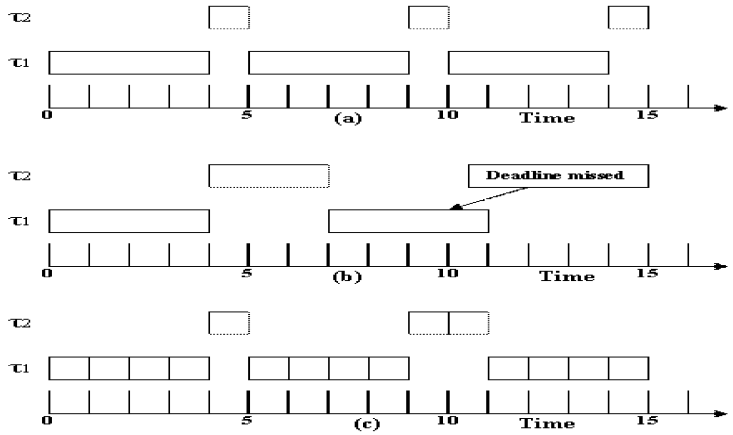

Fig. 5: (a) EDF, (b) NP-EDF and (c) UQ-NP-EDF schedules of tasks τ1(0,4,5,5) and τ2(0,3, 15,15)

Corollary 1: Under unit-quantum scheduling, the blocking time due to lower priority tasks equals zero.

Proof: Since the scheduling decision is made at each time instant by firstly releasing the sub-jobs and then selecting the one with the highest deadline for dispatching. Thus, there is no blocking time by a lower priority job for a higher priority one.

CONCLUSION

Real-time scheduling theory is very rich from scheduling perspective and a variety of scheduling techniques have been proposed for uniprocessor systems. For online periodic real-time systems, EDF scheduling is optimal scheduling technique for preemptive case. However, no optimal scheduling algorithm is available for non-preemptive counterparts. Keeping in view the importance of non-preemptive EDF scheduling we have presented a novel concept of unit-quantum based scheduling for hard real-time non-preemptive task sets on a single processor system. Under the proposed technique every task instance is subdivided into sub-tasks each of unit-quantum at run time and executes these sub-tasks based on EDF scheme. The proposed technique is a blend of quantum-based scheduling and EDF. Based on the experimental results we conclude that the proposed technique guarantees 100% system utilization.

REFERENCES

1. Liu, C.L. and J.W. Layland, 1973. Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment, J. the A.C.M., 20(1): 40-61.
2. Sha, L., R. Rajkumar and J.P. Lehoczky, 1990. Priority Inheritance Protocols: An Approach to Real-Time Synchronization, IEEE Transactions on Computers. Sept., 39(9): 1175-1185.
3. Cecilia Ekelin, 2006. Clairvoyant Non-Preemptive EDF Scheduling, Proceedings of the 18th Euromicro Conference on Real-Time Systems.
4. Park, M., H.J. Yoo and J. Chae, 2009. Analysis on Quantum-Based Fixed Priority Scheduling of Real-Time Tasks, 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC-09). Jan 2009. Suwon, S. Korea. pp: 15-16.
5. Jejurikar, R. and R. Gupta, 2005. Energy Aware Non-preemptive Scheduling for Hard Real-time Systems, Proceedings of the 17th Euromicro Conference on Real-Time Systems, pp: 21-30.
6. Anderson, J.H., R. Jain and K. Jeffay, 1998. Efficient Object Sharing in Quantum-based Real-time systems, Proceedings of the 19th IEEE Real-Time Systems Symposium, pp: 346-355.
7. Baruah, S., 2005. The Limited-preemption Uniprocessor Scheduling of Sporadic Task Systems, Proceedings of the 17th Euromicro Conference on Real-Time Systems, pp: 137-144.
8. Lim, T.M., B.S. Lee and C.K. Yeo, 2007. Quantum-Based Earliest Deadline First Scheduling for Multiservices. IEEE Transactions on Multimedia. 9(1): 157-168.
9. Wang, Y. and M. Saksena, 1999. Scheduling Fixed-Priority Tasks with Preemption Threshold. Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications, pp: 328-335.

10. George, L., N. Riviere and M. Spuri, 1996. Preemptive and Non-preemptive Real-time Uniprocessor Scheduling. Technical report, September 1996. INRIA, France.
11. Mok. A., 1983. Fundamental Design Problems of Distributed Systems for the Hard Real-time Environment, Ph.D. thesis, Massachussets Institute of Technol.,
12. Albert, M. and K. Cheng, 2002. Real-time Systems: Scheduling, Analysis and Verification, A John Wiley and Sons, Inc. Publication Hoboken, New Jersey,
13. Min-Allah, N., I. Ali, J. Xing and W. Yongji, 2010. Utilization Bound for Periodic Task Set with Composite-deadline, Journal of Computers and Electrical Engineering, 36(6): 1101-1109.
14. Min-Allah, N. and S.U. Khan, 2010. Wang Youngji, Optimal Task Execution Times for Periodic Tasks Using Nonlinear Constrained Optimization, Journals of Supercomputing, DOI: 10.1007/s11227-010-0506-z,
15. Khan, S.U., 2011. Mosaic-Net: A Game Theoretical Method for Selection and Allocation of Replicas in Ad Hoc Networks, J. Supercomputing, 55(3): 321-366.
16. Min-Allah, N., S. Islam and W. Yongji, 2010. Enhanced Time demand Analysis, World Applied Science J., 9(1): 01-13.