Journal of Information Technology Research
Volume 15 • Issue 1

A Multi-Budget-Based Approach to Enhance the Responsiveness of Aperiodic Task for a Bandwidth-Preserving Server in Real-Time Systems

Ajitesh Kumar, AKTU, Lucknow, India & GLA University, Mathura, India*
https://orcid.org/0000-0003-0196-5640

Sanjai Kumar Gupta, Bundelkhand Institute of Engineering and Technology, Jhansi, India

ABSTRACT

Within the advanced computation time, real-time application pulled in much more attention. Implementing a better high-quality real-time system requires to improve the responsiveness of the tasks set. This research work aims to achieve the best quality of service (QoS) in terms of improving the responsiveness of aperiodic tasks and also improved acceptability domain, by accepting to execute multiple aperiodic functions while maintaining the feasibility of periodic tasks in a real-time system. The functional analysis with simulation shows that the proposed algorithm is highly effective in terms of task sets deemed schedulable and also by allowing aperiodic tasks that were rejected by existing approaches. The simulation results indicate that it reduces overall average response time of aperiodic tasks approximately 13% at lowest periodic load (35%), 7% at 60% periodic load, and 4% at 80% periodic load, and in all observed circumstances, the proposed novel algorithm received 7%-10% improvement over the existing one.

KEYWORDS

Aperiodic Task, Budget, MLBBPS, Real-Time System

INTRODUCTION

Real-Time applications have the requirement of both fixed arrival patterns as well as random patterns in nature. The ‘fixed arrival sequence’ is termed as a periodic task, whereas term, ‘aperiodic task’ is used for a random one. For example, a mobile video phone application has stringent regular computing requirements for the number of frames received per second along with aperiodic requirements generated by a user such as a volume control and play-list editing. These user-generated requests are event-driven. Similarly, the flight control system executes pilot control commands such as control of sudden rise of temperature, and speeds are aperiodic requirements along with routine computation

DOI: 10.4018/JITR.299917 *Corresponding Author

requirements. Besides these requirements patterns, time criticality classifies the system as hard and soft in nature. For hard real-time system failures to met one condition may lead to catastrophe; however, degraded quality is received even missing to meet many requests for the case of a soft real-time system.

The algorithms described in this paper determine when aperiodic tasks are executed in presence of periodic task set. They are solutions to the following challenges:

1. Based on the execution time and deadline of each newly arrived aperiodic task, the scheduler decides whether to accept or reject the task. If it accepts the task to execute, it schedules the task so that the task completes its execution in time without causing periodic tasks and previously accepted aperiodic tasks to miss their deadlines. The problems are how to do the acceptance test and how to schedule the accepted aperiodic task set.
2. The scheduler tries to complete each aperiodic task as soon as possible. The problem is how to do so without causing periodic tasks and accepted sporadic jobs to miss their deadlines.

The novel two-phase approach proposes for multi-level budget bandwidth preserving server (MLBBPS) is utilizing the concept of the multi budget with the construction of the budget sample on or after hyper-period. It consumes budget at a multi priority level for execution of aperiodic tasks while ensuring periodic one. The first phase performs offline feasibility analysis for periodic tasks and fixes up the static budget renovated at regular intervals. Apart from the static budget, slack based budget is computed, forming a budget pattern from hyper period to hyper period. Over the above budget pattern formed in the first phase, the calculated budget tuned in the second phase. The tuning has achieved through utilizing the online slack availability that arises due to execution of the task with less execution time (as compared to worst-case time considered at the time of feasibility analysis), and accumulating the small budget fragments into a larger one. Abeni.L. et al. (2015) proposed the improvement in the responsiveness of aperiodic tasks to enhances the acceptance execution ratio of aperiodic tasks, rejected by a deferrable servers. The complexity of the proposed novel algorithm is very similar to the deferrable server that intended to make sure of the correctness of the server. MemGuard: Memory bandwidth reservation system discussed by Yun, H. et al. (2013, April) gives efficient performance under heavy memory workloads in the multiprocessor system but lacking on throughput under time-varying memory workloads.

The significant contribution of proposed research work is to develop a novel approach to enhance the responsiveness of aperiodic task for multi-level budget bandwidth preserving server in Real-Time System. The proposed algorithm developed in two phases, the first phase is for the construction of budget1, budget2, and proposal for MLBBPS algorithm; the second phase is for refinement of budget to increase the responsiveness of aperiodic task. An illustrative example of a functional analysis of the algorithm shows the effectiveness of the proposed approach. The performance of the proposed algorithms evaluates for both synthesized task sets used in Erciyes K. et al. (2019) and data available for different applications in Hamann A. et al. (2018). The study indicates that the proposed multi-level budget bandwidth preserving server receives better responsiveness with an increased number of completed aperiodic tasks over a wide range of variations.

The rest of the research paper organizes as follows. Section 2 describes the system model and related work. The proposed approach multi-level budget bandwidth preserving server has been explored in section 3, while section 4 deals in results and analysis work. In the end, section 5 concludes the paper.

TASK MODEL AND RELATED WORK

This proposed real-time system deals with applications consisting of a preset coming pattern of the tasks along with arbitrary arrival tasks. The fixed arrival pattern and random arrival task are known as periodic tasks and aperiodic tasks, respectively. The periodic tasks are time-driven initiated at the

specified time, whereas occurrences of specific events are initiated at aperiodic one, as stated by Hilman MH.et al. (2020). The full guarantee can be given for periodic tasks in well advance, whereas completion of aperiodic one can be ensured after its arrival online. The proposed system has $\left(n\right)$ independent periodic tasks $\tau_1, \tau_2, \tau_3 \ldots \tau_n$. Each task τ_i has the attributes, worst-case execution time $\left(e_i\right)$, period $\left(p_i\right)$, and relative deadline $\left(d_i\right)$. The author assumes the relative deadline of a periodic task is less than or equal to its period. In addition to these n periodic tasks, there is another periodic task τ_s characterized by an ordered pair (q_s, p_s) that used to provide execution budget for aperiodic one. Here, q_s is the amount of budget released with period p_s. Execution of these tasks is fully preemptive, and preemption overhead is considered negligible for the worst execution time.

Here, Summarizes the symbol used, followed by a definition of key terms used.

Terms Used

- **Finish Time $\left(ft_i^j\right)$:** It is the sum of its release time, own requirement, the requirement of the higher priority tasks executing between its release and its completion:

$$ft_i^j = rel_i^j + e_i + \sum_{k=1}^{k=i-1} \left(t \,/\, p_k\right) * e_k \;\; where \;\; rel_i^j \le t \le p \tag{1}$$

- **Relative response Time of $R_i^j\left(\tau_i^j\right)$:** It is the difference between the finish time and the release time of a τ_i^j. Mathematically, $R_i^j\left(\tau_i^j\right) = ft_i^j - r_i^j$
- Critical instance release of the task τ_i: It defines as a time when a task τ_i release, along with all higher priority tasks.
- **Hyper-period (L):** It can define as the point of time after which all the tasks in the task set T are in phase and schedule pattern for each task restart, i.e., the release pattern of tasks at time $t = 0$ repeat at an integral multiple of hyper period:

$$L = LCM\left(p_1, p_2, p_3 \ldots p_n, p_s\right) \tag{2}$$

- **First Budget:** Fixed budget of the amount q_s fixed to the server and replenish at every integral multiple of p_s call budget 1.

Table 1. Symbol Used

Symbol	Meaning(Description)
r_i^j	j th release time of task τ_i
D_i^j	Absolute deadline of j th release of task τ_i
τ_i^j	j th release of task τ_i
ft_i^j	j th release, Finish Time of task τ_i
R_i^j	Response time of j th release of task τ_i
L	Hyper-period

- **Second Budget:** The dynamic budget generates on account of slackness available is called Budget 2.

Related Work

Abeni, L. et al. (2019) proposed a scheduling algorithm for a real-time virtual machine that practically implemented hierarchical scheduling with the vanilla Linux platform. According to Ashjaei, M. et al. (2017), resource reservation technique ensures the predictability and allows flexibility during the task execution in a real-time system. This paper also provides a method for an end to end timing analysis of executed task sets. Stated by Ashmawy A. et al. (2018), a total bandwidth server (TBS) works with the periodic and aperiodic task set very efficiently. In this system, a job is divided into the subpart, and each one has an individual deadline. The simulation results show that average response time reduces the aperiodic task, and when processor utilization is high. The reduction rate of the aperiodic job reaches more than 50%. Brandenburg B. et al. (2016) proposed a scheduling algorithm that can reduce run-time overheads. This algorithm also avoids 50% migration in the worst-case execution time analysis. Hilman M. H. (2020) proposed a novel approach for budget-constrained multiple workflows resource provisioning and scheduling in the context of the WaaS cloud platform. This work also explains the different issues in resource sharing, performance evolution, and unusual activity related to networks. According to the proposed framework of Hosseinimotlagh, S. (2019) significantly reduces sporadic task response time. Simulation results show that randomly-generated task sets demonstrate the performance characteristics of the proposed structure with different configurations. Hussien H. et al. (2019) proposed an adaptive framework for a set of application which is independent of each other and runs on a single processor. This scheduling approach has a less miss ratio with minimal overhead. According to Kim H. et al. (2015), modern processors need virtualization in the field of real-time processes. In this work, a vINT scheme for real-time application benefits in reducing the interrupt handling time and also protecting from interrupt storms. Khan A. A. et al. (2019) proposed a two-phase scheduling process where aperiodic task executes at full speed. Periodic jobs are migrating in another processor if the deadline is earlier and complete the aperiodic task at a lower rate if slack available. Manthalkar et al. (2018) proposed a scheduler framework simulator called PATSAS to predict the behavior of the system and allow scheduling different periodic and aperiodic algorithm. According to EDF based approach by Nascimento F. et al. (2019), hard tasks assigned a dedicated processor. They do not migrate during execution, but the soft job is allowed for migration. Schlatow J.et.al.(2019) provides a novel method for run-time monitoring in a mixed-critical real-time system. A job-shifting approach for online admission of non-preemptive aperiodic tasks in the partitioned time-triggered environment proposes by Syed A. et al. (2018). Wu J.et al. (2019) introduced an RTDS based scheduler to provide an additional amount of PCPU capacity. Zhang Y. et al. (2019) compute the optimal speed of periodic task and introduce a new novel scheduling algorithm for static mixed tasks. The simulation results show that the novel algorithm reduces 53.66% energy consumption response time when compared with the SMTS algorithm.

PROPOSED NOVEL APPROACH

The proposed novel two-phase approach is to work with a fixed budget, termed as budget 1. The additional budget is known as budget-2. This excess budget is restraining to slack available for the lowest priority periodic task. Both types of budgets are determined in phase 1. However, budget 2 is being refined in phase 2 online.

In the next subsection, phase 1 is discussed, followed by phase 2 of the proposed approach.

Phase I: Proposed MLBBPS

The major responsibility of phase 1 is to determine the possibility of periodic tasks server budget (T_s), characterized by execution budget in terms of time and renovation period of budget 1. It also decides budget 2, based on slack available in the schedule of the periodic task set. The fixed amount of Ti rests budget 1 of equal to E_s allocates at each time instant same to the integral multiple of P_s. The budget released will be retained up to the next due time of the allocation in case it not consumed; the server retains an unconsumed budget is known as a deferrable server. The server executes aperiodic tasks against budget 1 at the priority decided at the feasibility analysis done in offline. The server consumed budget 1, on the occurrence of either condition a) sever is executing aperiodic tasks or b) system idle. The next paragraph deals with the computation of budget 2. Over the above budget 1, another kind of dynamic budget compute based on slack available in the schedule of the task set T_s. Here, we decided on the amount of progressive budget available for a specific time interval and consumed at a different priority level. The amounts of the budget available for consumption at different priority levels are different. The amount of the budget available for consumption at priority level (i) is equal to the minimum of the slack available for the task having priority less than or equal to T_i. The detailed steps for the computation of budget 2 offline, along with its consumption window, will discussed. The consumption window time, the interval in which this computed budget is available for execution of aperiodic tasks along with executing priority.

The priorities for the execution of budget 2 with the amount of budget consumed at that level decided offline.

Offline Priority Estimation for the Consumption of Additional Budget

Apart from consumption and replenishment rules and forming a budget pattern for budget 2 from hyper-period to hyper-period, priority of server also plays an essential role in achieving responsiveness and improves acceptance ratio for aperiodic tasks. It always desires to operate the server at the highest priority and have the best quality in terms of responsiveness and acceptance ratio. However, on the way to increase server priority, maintaining the feasibility for the periodic tasks is a significant problem. That is, with an increased preference of periodic server task may become infeasible, which was feasible without priority improvement of the server. So in this section, discuss growth in priority of server to enhance the response time of aperiodic tasks keeping in account the feasibility of periodic tasks. The main problem occurs, at which priority level budget 2 will consume to improve the responsiveness of aperiodic tasks while ensuring the feasibility of periodic tasks at the same time.

In offline priority, estimation decides how much available budge consumed at which priority level in the worst case. Firstly, estimate the laxity of each release of the task that is released within the expiry interval of budget 2 and then take a minimum of them. That termed as laxity of the task in that interval. After computing the laxity for each task, trying to find out at which priority level how much budget 2 consumed without affecting the likelihood of periodic tasks.

The priority estimation algorithm computes the amount of budget 2 consumed at each priority level for budget 2.

Over the above estimation of both budget1 and budget 2 along with the priority of its execution at different priority levels. In phase1, budget calculated on considering the worst-case requirement in terms of both esteemed release time and worst-case execution requirement. However, in online execution a task may not be released in the estimated time, it takes lesser time to complete than that considered in phase 1, and budget 1 may not consume up to the next replenishment time. Thus, online tuning of budget2 is discussed on algorithm-2.

Online Tuning of Budget 2

In the view of online variation in the requirement due to:

Algorithm 1. Priority estimation (PE) algorithm

Input: PE Algorithm for Budget 2 (task set T= $\{\tau_1,\tau_2,\tau_3,\ldots\tau_n \cup \tau_s\}$)

Output: budget 2 pattern along with priority level

Assumption:
- Periodic task set (T) is the union of periodic tasks and server (τ_s). here $\tau_1$is the highest and τ_n is the lowest priority periodic tasks
- Critical instance discharge of tasks in (T) is measured

START:

1: for 1 to LCM $((p_1,p_2,p_3\ldots\ldots\ldots p_n,p_s)/p_n)$ do

2: $B_{2,j} = L_{n,j}$ // maximum budget 2 available in jth interval of p_n

3: for 1……N-1 do

4: a. Compute $(N_i=\lceil(jp_n)/p_i\rceil - \lceil((j-1)p_n)/p_i\rceil)$

// number of releases (N_i) of task τ_i in the interval $((j-1)p_n,Jp_n]$

5: b. Compute L_i^j using equation2

6: c. Compute $L_{i,j}$ =min $(L_i^1, L_i^2, L_i^3,\ldots\ldots L_i^N$ // laxity of task τ_i in jth interval of length p_n ($Laxity_i$) during $((j-1)p_n,Jp_n]$

7: for deciding priority level for execution of budget 2

8: for i = 1……..n do

9: Execution at priority level= i-1

10: $B_{2,j}^{\ i}$ = MIN $(L_{i,j}, L_{i+1,j}, \ldots\ldots\ldots L_{n,j})$

//amount of budget 2 executed at priority level i in j th interval of length p_n

11: for j = i+1 ……n do

12: $L_{i,j} = L_{i,j} - B_{2,j}^{\ i})$

13: End for

14: End for

15: End for

16: End for

END

1. Budget 1 assigned to the server has not consumed up to the next replenishment time.
2. Task releases the resource in case, and it takes lesser time than that considered offline phase 1. That is, the task has executed in the worst case.

Budget 2 needs to recompute. Here, the authors require to answer several questions about what the new dimension of budget 2 is? Whether budget 2 may be updated? What amount of budget is available at different priorities? What will be further laxity for various tasks?

On the way to answer these questions, online tuning divided into two cases:

1. Tuning due to unused budget1.
2. tuning due to release took lesser time than that already booked.

For the case of the remaining budget, the stuffing of the unspent budget from one-time interval to another time interval decided based on idle slots available in the previous execution window of budget1.

The algorithm for scheduling periodic tasks along with the aperiodic by using the proposed multi-level budget bandwidth preserving server (MLBBPS) is potted as shown in Algorithm 5.

Algorithm 2. Function to refine budget 2

//while there is unused budget 1 in any interval [(k-1)Ps, kPs]

START:

1: Initially $B_{2,j}(0) = L_{n,j}$ // // maximum budget 2 available in jth interval of p_n

2: K = 1

3: t = kPs

4: $B_{2,j}(t) = B_{2,j}(t) + \max(\text{unused budget1}(t) - \Sigma^{(k)\,Ps}_{(k-1)\,Ps}\ \text{idle slot}, 0)$

5: Update the laxity of all periodic tasks which priority is less as compared to server

6: Update the laxity of release which arrives or incomplete during the interval [(k-1)Ps, kPs]

7: $L_i^j(t) = d_i - t - reme_i(t) - \Sigma_{k\epsilon\tau\_H\cup\tau\_s}(\lceil (d_i - t)) / p_k \rceil) * (e_k)$

8: $L_{i,j}(t) = \min(L_i^j, L_i^{j+1}, L_i^{j+2}, \ldots\ldots L_i^N)$

9: $B_{2,j}^{\ i} = \min(L_{i,j}, L_{i+1,j}\, L_{i+2,j}, \ldots..\, L_{n,j})$

END

Algorithm 3. Function to fine-tune budget 2 through aperiodic tasks

//while there is unused budget 1 in any interval [(k-1)*Ps, k*Ps]

START:

1: Current time (t_c)=t

2: Compute $k = \lceil t_c / Ps \rceil$

3: $UB1(t_c) = c_s - B1_c\,[(k-1)Ps, kPs)]$

// where $B1_c\,[(k-1)Ps, kPs)]$ amount of budget1 consume in interval [(k-1) Ps, kPs]

4: **If** ($I_l^i \geq (k-1)*Ps$ && $I_u^i \leq k*Ps$) // compute the length of idle slot in interval [(k-1) Ps, kPs] **then**

{

5: $I(i) = I_u^i - I_l^i$ // where, I(i) length of ith interval idle slot

6: IL = I(i) + IL and i = i +1

}

7: $B_{2,j}(t_c) = B_{2,j}(t_c) + \max(UB1(t_c) - IL, 0)$

// Update the laxity of release for all task Ti which priority is less as compared to server and $kP_s \leq a_i^r \leq (k-1)\,P^s$

8: $L_i^r(t) = d_i - t - reme_i(t) - \Sigma_{k\epsilon\tau H\cup\tau s}(\lceil (d_i - t)) / p_k \rceil) * (e_k)$

9: $L_{i,j}(t) = \min(L_i^r, L_i^{(r+1)}, L_i^{(r+2)}, \ldots\ldots L_i^N)$

// $(L_i^r, L_i^{(r+1)},, \ldots\ldots L_i^N)$ are already computed in offline

10: $B_{2,j}^{\ i} = \min(L_{i,j}, L_{i+1,j}\, L_{i+2,j}, \ldots..\, L_{n,j})$

End If

END

Algorithm 4. Function to fine-tune budget 2 through periodic tasks

// Here τ_1 is the highest and τ_n is the lowest priority periodic tasks

START:

1: Current time (t_c) = FT_i

2: $B_{2,j}(t_c) = B_{2,j}(t_c) + e_i^j$ (online(t)) - e_i^j (offline (t)

// Update the laxity of release for all task Ti which priority is less as compared to the task, Ti

// For all task Tk

3: **For** (k=i-1,k--, k≤n) **do**

4: $L_k^r(t) = L_k^r(t) + e_i^j$ (online (t)) - e_i^j (offline(t))

5: $L_{i,j}(t)$= min (L_i^r, $L_i^{(r+1)}$, $L_i^{(r+2)}$,……L_i^N) $L_i^{(r+1)}$,$L_i^{(r+2)}$,……L_i^N are already computed in offline

6: $B_{2,j}^i$ = min ($L_{i,j}$, $L_{i+1,j}$ $L_{i+2,j}$, ….. $L_{n,j}$)

end for

END

Phase II: Refinement of Budget 2

Case 1: When there is unused budget1 in any interval $\left(j-1\right)p_s, Jp_s$ (see Algorithm 6).

Case 2: When periodic task execute in less time online which assigned in offline (see Algorithm 7).

Illustrative Example for Functional Analysis of Algorithm

Example 1: Considering a periodic task set:

$$T = \left\{\tau_1, \tau_2\right\} = \{< e_i, p_i, d_i, >: \left\langle 1.5, 10, 10\right\rangle, \left\langle 2.5, 15, 15\right\rangle\}$$

and server having attribute $\tau_s = (q_s, p_s) = \left(1.5, 3\right)$.

For the given task set in the case of phase 1 hyper-period is 30, and deferrable server budget (budget 1) release at 0, 3, 6….with amount equal to 1.5 units. During the hyper-period of 30 units' lowest priority task τ_2, release twice (at 0 and 15) with respective laxities 2.0 and 3.5 units. These laxities are referred to as budget 2 in respective intervals of expiry, i.e., 2.0 and 3.5 units of budget 2 during (0-15] and (15-30] interval. This process forms a budget pattern that repeats from hyper-period to hyper-period. During interval (0,15] τ_s is release 5 times with laxities equal to 1.5 for each release while τ_1 is release 2 times with laxities equal to 2.5 and 4.0, respectively. So 1.5 unit budget budget2 consumed at a priority level 0 and τ_s, 0.5 unit budget 2 is consumed at a priority level τ_s and τ_1 depend upon the availability of aperiodic tasks. However, for the interval (15, 30] 3.5 units of budget 2 are available and the laxity of server τ_s and task τ_1 is 1.5 and 2.5 units, respectively. So 1.5 unit of budget 2 is consumed at priority level 0 or τ_s, 2.0 unit budget consume between priority level τ_1 and τ_s.

It depends upon the availability of aperiodic tasks. For better sake of understanding of budget 2, consumption and distribution are given in table 2.

In this approach, apart from framing the budget pattern and estimation of priority level for consumption of budget2 at offline. In online, the authors improve budget 2 by fine-tuning of budget 2.

Algorithm 5. MLBBPS Algorithm (Task Set T= $\{\tau_1, \tau_2, \tau_3 \ldots \tau_n \cup \tau_s\}$)

// Union of periodic tasks and server (τ_s) is called periodic task set T

START:

1: Make sure the likelihood of tasks (T) by using the fixed priority rate monotonic (RM) first algorithm.

2: **For** J[th] release of τ_n where τ_n is the lowest priority task in the periodic task set T and j=0,1,2…L/pl. t=current time, n=0,k=1

Do

3: **for** ((s=n*(L/p_s), s++, s<(n+1)*(L/p_{ls})) **do**

{

4: a. $B1_s = C_s$

5: b. expiry_time_$B1_s$= (s*p_s) + p_s

6: c. replenishment _time_$B1_s$ = s*p_s

}

7: **for** (j=n*(L/p_l), j++, j<(n+1)*(L/p_l)) **do**

{

8: a. $B_{2,j}$ = laxityτ_l^j // using equation 1

9: b. expiry time_$B_{2,j}$ = (j*p_l) + p_l

10: c. replenishment _time_$B_{2,j}$ = j*p_l

11. d. Formation of budget2 pattern is within the hyper period

}

12: Budget 2 pattern repeated from hyper period to the hyper period

13: Compute the amount of budget 2 consume at various priority level in each length of p_l by priority estimation algorithm

14: **end for**

15: **While** (the aperiodic queue is not empty (t))

16: **if** (total budget (t)>0)

17: **if** (PL ($B_{2,j}$) or priority of server [3]highest priority periodic tasks available at time t **then**

{

18: a. Select the aperiodic tasks on the EDF basis

19: b. Execute the aperiodic tasks and consume the budget 1 or budget2

}

else

{

20: a. Place the aperiodic task in aperiodic queue

21: b. Compute the idle slot / by idle slot algorithm

22: c. Execute the periodic tasks as per their priority level

23: d. Remove the periodic task into the periodic queue

}

end if

24: at the finish time of each periodic task go to step 17

continued on following page

Algorithm 5. Continued

```
25:  if (e_i^j (online(t))<e_i^j (offline(t)) then
 // at any current time (t) periodic job executes in less time as compare to execution time booked in offline
      {
26:    a. t = FT_i^j
27:    b. Call tune budget2 (t,e_i^j (o(online),B_2,j (t_c))
28:    c. Update the amount of budget 2 and update the priority level for consumption of budget2
       }
29:    else Go to step 15
30:  At each time instant t = k*p_s
31:      If (UB1 (k*p_s)>0) then
         {
32:      a. Call tune budget2 (t,c_s)
33:      b. k=k+1
34:      c. Update the amount of budget 2 and update the priority level for consumption of budget2
         }
35: Go to step 15
36:      else then
         {
37:      Aperiodic task wait for the next replenishment of budget 1 and budget 2 (which one is earlier)
38:      Go to step 8
          }
39:     end if
40:           end else
41:  end if
42:          end else
END
```

Suppose aperiodic A1 arrives at time t=1, and its execution time is 0.75 units. At the end of time t=3, the aperiodic queue is empty, and there is a 0.75 unit of budget1 unused. This unused budget uses to improve budget 2 by algorithm fine-tuning of budget2. At time t=3, budget 2 is 2.75; however, offline, it is only 2 units.

The usefulness of the proposed approach (MLBBPS) can be pragmatic in example 2.

Example 2: Considering a periodic task set:

$$T = \left\{\tau_1, \tau_2\right\} = \{< e_i, p_i, d_i, >: \left\langle 1.5, 10, 10 \right\rangle, \left\langle 2.5, 15, 15 \right\rangle\}$$

and server having attribute $\tau_s = (q_s, p_s) = \left(1.5, 3\right)$.

Algorithm 6. Refinement of budget 2: Case 1

```
// Here τ1 is the highest and τn is the lowest priority periodic tasks
START:
1:   for j=1 to n do
     {
2:     When there is unused budget1 in any interval ((j-1)ps, Jps
3:     Pre-pone the schedule of lowest priority periodic task as compare to server
4:     Unused budget 1= remaining budget at time t = Jps
5:     Budget 2= Budget 2 + unused budget -Σ(J-1)^j idle slot
     }
6:   end for
7:  for i=k to n do
     {
8:    Laxity of lowest priority periodic tasks as at time t=Jps compare to the server is increased by amount unused
      budget except all the job which finish before time t=Jps
9:    Laxity of lowest priority periodic tasks as at time t=Jps compare to the server which finishes before time t=Jps
10:   Laxity K = laxity K + unused budget – (Jps- finish time τk )
}
11: end for
END
```

Algorithm 7. Refinement of budget 2: Case 2

```
// Here τ1 is the highest and τn is the lowest priority periodic tasks
START:
1:   for task Ti do
     {
2:     When periodic task execute in less time as compare to assigned in offline
3:     Budget 2= Budget 2 + worst-case execution time Ti-original execution time Ti
4:      j=i+1 to n
5:     Pre-pone the schedule of the periodic task which priority is less than task Ti
6:     Laxity j = laxity j + worst-case execution time Ti-original execution time Ti
     }
7:   end for
END
```

Aperiodic tasks A1, A2, A3, A4, A5, A6 and A7 arrive at times 0, 3, 4.5., 6, 7.5, 8.5, and 12 with their respective execution time 0.75, 2, 1, 2, 1, 1 and 1.5. Their respective deadlines are 2, 6, 9, 10, 14, 17, and 19 units.

Figures 1 and 2 show, the proposed novel MLBBPS approach reduces the response time of aperiodic jobs, minimizes the rejection ratio of aperiodic tasks, reduces average response time, and

Table 2. Budget 2 consumption distribution for interval (0, 15]

The priority level for execution of budget 2	Budget2 consumption distribution starting from priority above τ_s	Budget2 consumption distribution starting from priority between the level τ_s and τ_1	Budget2 consumption distribution starting from priority between the level τ_1 and τ_2
Between priority level τ_s and 0	1.5	Nil	Nil
Between priority level τ_s and τ_1	1.5	2	Nil
Between priority level τ_s and τ_2	1.5	2	2

Table 3. Budget 2 consumption distribution for the interval (15, 30] in online at time t=3

The priority level for execution of budget 2	Budget2 consumption distribution starting from priority above τ_s	Budget2 consumption distribution starting from priority between the level τ_s and τ_1	Budget2 consumption distribution starting from priority between the level τ_1 and τ_2
Between priority level τ_s and 0	1.5	Nil	Nil
Between priority level τ_s and τ_1	1.5	2.5	Nil
Between priority level τ_s and τ_2	1.5	2.5	2.75

also improves the quality of service (QoS). The success of the proposed approach has summarized in table 2. Table 2 observation shows that the intended multi-level budget bandwidth preserving server improves the quality of service (QoS) in terms of enhancing responsiveness of aperiodic tasks, accepting more aperiodic tasks for execution, and increased system utilization while maintaining the feasibility of periodic tasks.

RESULT ANALYSIS

This section utilizes the simulation of the proposed algorithm, and it uses the task set to performs and evaluates the performance of the proposed novel approach, multi-level budget bandwidth preserving server, with available deferrable bandwidth preserving server.

Here, the authors compare the performance of multi-level budget bandwidth preserving server is referred as MLBBPS with exiting deferrable server with background approach is referred as DSWB by Yoder J.et.al (2017).

Load Effect on Average Response Time of Aperiodic Task

The critical parameters for performance measurement in Figures 3, 4, and 5 are response time (avg.) and acceptance quotient.

Here, the authors measure the variation of system load on aperiodic average responce time. Figures 3, 4, and 5 are comparing the proposed approach with the available deferrable server approach when the load of periodic task 40%, 60%, and 80%. In Figure 3, the regular task load is 40% of the total capacity of the system, and the aperiodic task load is varied from 10% to 60% with server utilization 0.2. The authors observe that in this satuation the aperiodic task response increases when the total

Table 4. Shows the performance evaluated in Example 2

Aperiodic task	Time Response of Aperiodic task with DS and background by Yoder J.et.al (2017) (DSWB)	Response time with Proposed MLBBPS
A1	0.75 (accepted)	0.75 (accepted)
A2	3.5 (rejected)	2 (accepted)
A3	3.0 (accepted)	1.5 (accepted)
A4	3.75 (accepted)	2 (accepted)
A5	5.25 (rejected)	2.25 (accepted)
A6	4.75 (rejected)	1.75 (accepted)
A7	3.25 (accepted)	1.75 (accepted)
A8	4.5 (accepted)	3.0 (accepted)
A9	6.5 (rejected)	4.0 (accepted)
A10	8.0 (accepted)	4.5 (accepted)
A11	6.5 (rejected)	3.0 (rejected)
A12	5.5 (accepted)	3.0 (accepted)
A13	5.0 (accepted)	3.5 (accepted)
Average response time for 13 aperiodic tasks within the interval [0, 30]		
DS + background		MLBBPS
4.634		2.538
Acceptance ration of aperiodic tasks within time interval [0,30]		
DS + background		MLBBPS
8/13 (61.5%)		13/13(100%)
Budget available within hyper-period		
DS + background		MLBBPS
21.75 units		21.75 units

Figure 1. Schedule for proposed novel approach MBBPS

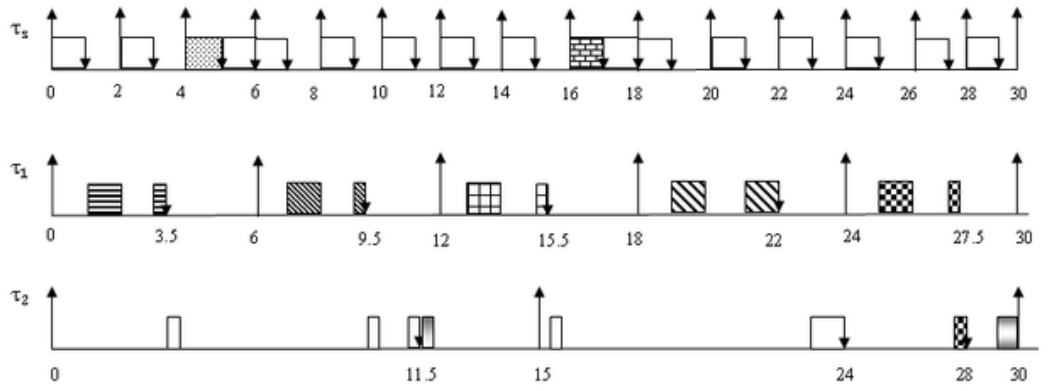


Figure 2. Schedule for existing DS and Background

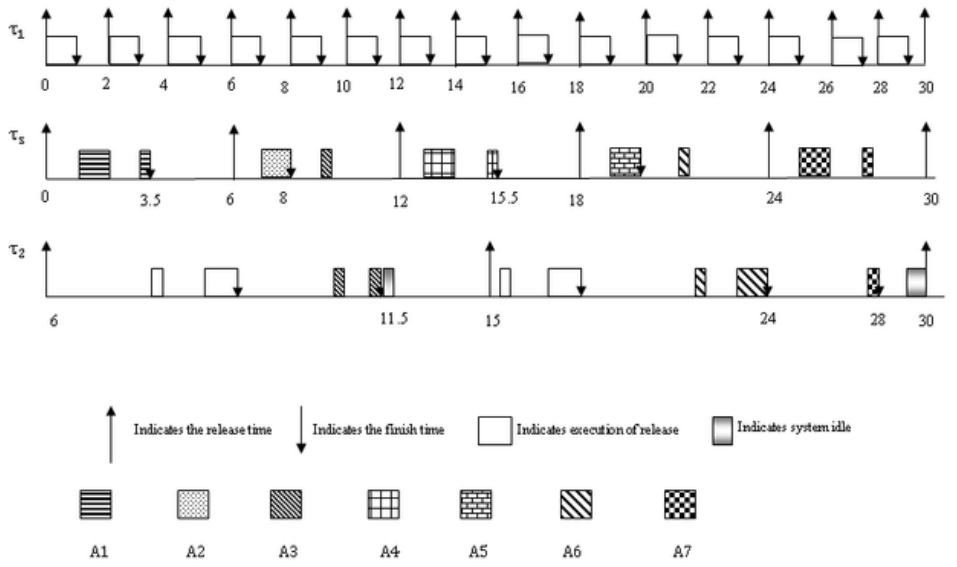


Table 5. Shows some assumptions which consider during the simulation

About Task Parameter	Assumption	Variation
Utilization Threshold	assigned	0.01-0.02
Utilization	If Utilization-$\Sigma u_{(i-1)} \geq$Uth then any random number selected	$(0,U_i-\Sigma U_{(i-1)}]$
	If Utilization-$\Sigma u_{(i-1)}$ < Uth then assigned	Utilization = $U_i-\Sigma u_{(i-1)}$
WCET	Any random selection	(0,90]
Execution Period	Any random selection	(0,900]
Task Deadline	Any random selection	$[e_i, p_i]$
Periodic task load	assigned	40%, 60%,80%
Aperiodic task arrival time	By using a process(Poisson Arrival Method)	

Figure 3. Shows the aperiodic Avg. response time with 60% periodic load

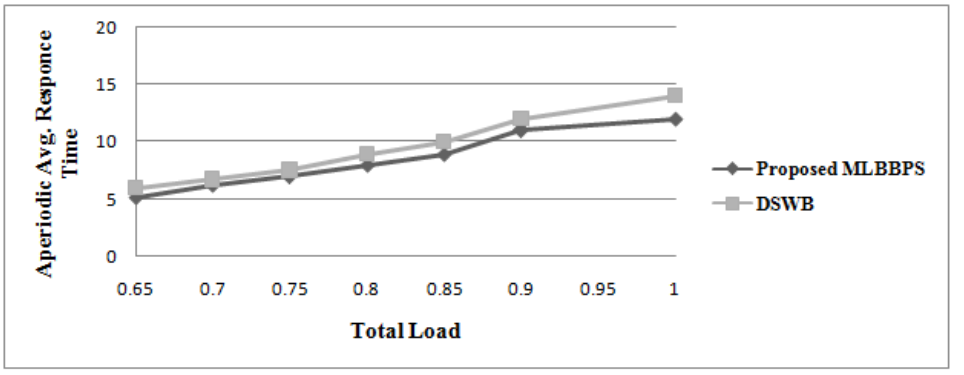


Figure 4. Shows the aperiodic Avg. response time with 80% periodic load

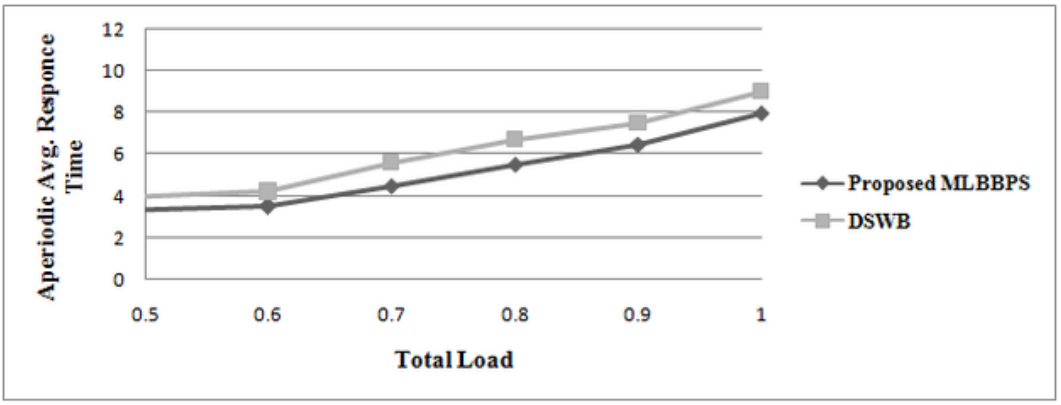
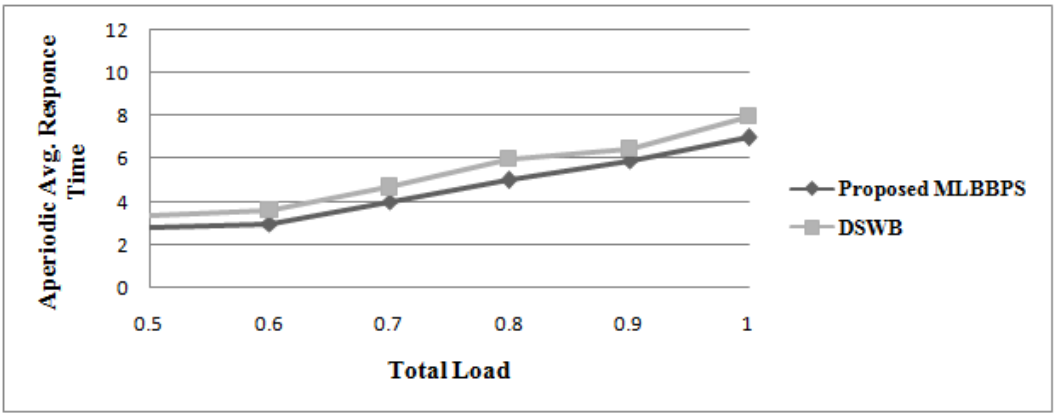


Figure 5. Shows the aperiodic Avg. response time with 40% periodic load



load increases. When the aperiodic task load varies from 30% to 60%, the proposed approach gives a 15% reduction in average response time of aperiodic task over the existing method. That will happen due to a more number of aperiodic jobs that occurred; budget 2 utilized better, and most of the time, the aperiodic task complete execution within the assigned budget. In Figure 4, periodic task load increases up to 60% of total capacity, and aperiodic task varies from 5% to 40%, so that budget 2 decreases and the performance of the proposed algorithm is better only 7% in comparison with deferrable server approach. In Figure 5, periodic task load increases up to 80% of total capacity, and aperiodic task varies from 5% to 20%, so that budget 2 is meager, and performance of the proposed algorithm is almost 3% better in comparison with deferrable server approach.

Load Effect on Rejection Ratio of Aperiodic Task

In this simulation, the authors measure the result of periodic and aperiodic load variation, server utilization on average response time, and aperiodic task rejection ratio.

Figure 6, 7, and 8 shows the comparison of the proposed two-phase novel approach with the deferrable server algorithm. Here aperiodic task rejection has been measured when a periodic load is 40%, 60%, and 80%. In Figure 6, regular capacity is only 40%, and the aperiodic amount varies in

Figure 6. Shows the aperiodic task rejection ratio (%) with periodic task load 40%

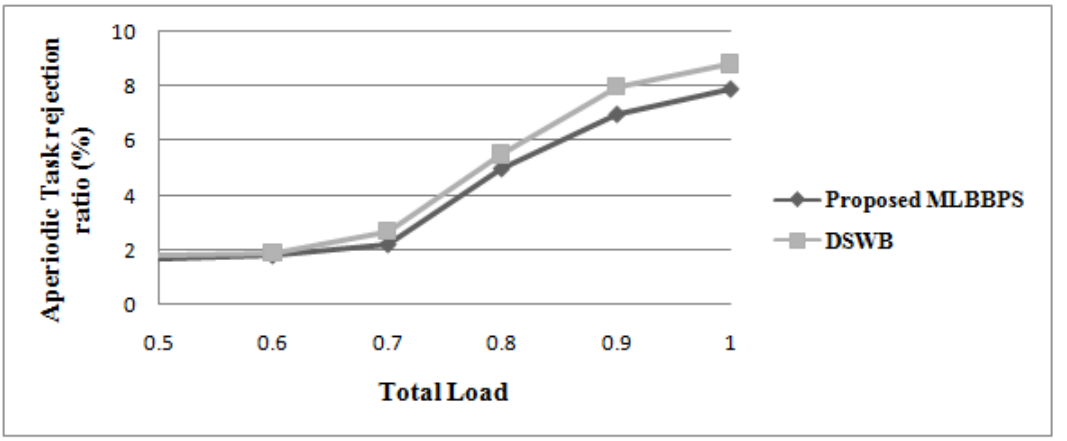


Figure 7. Shows the aperiodic task rejection ratio (%) with periodic task load 60%

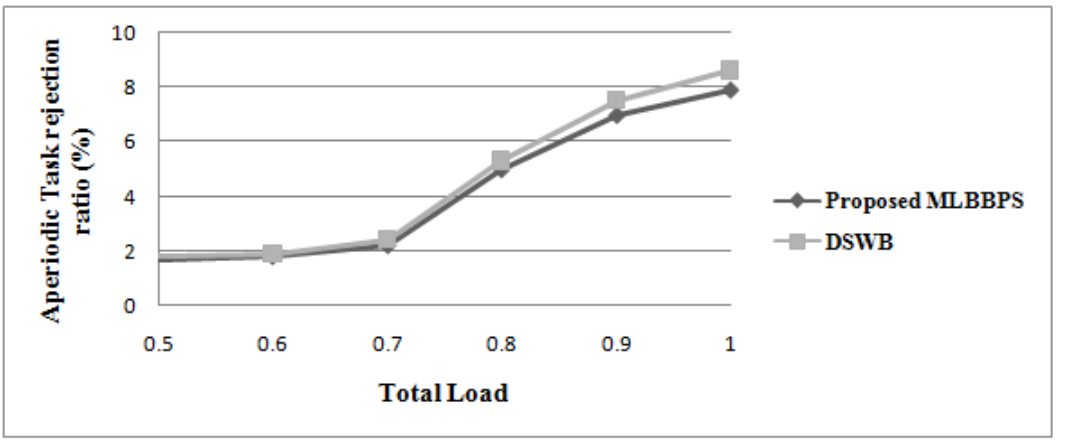
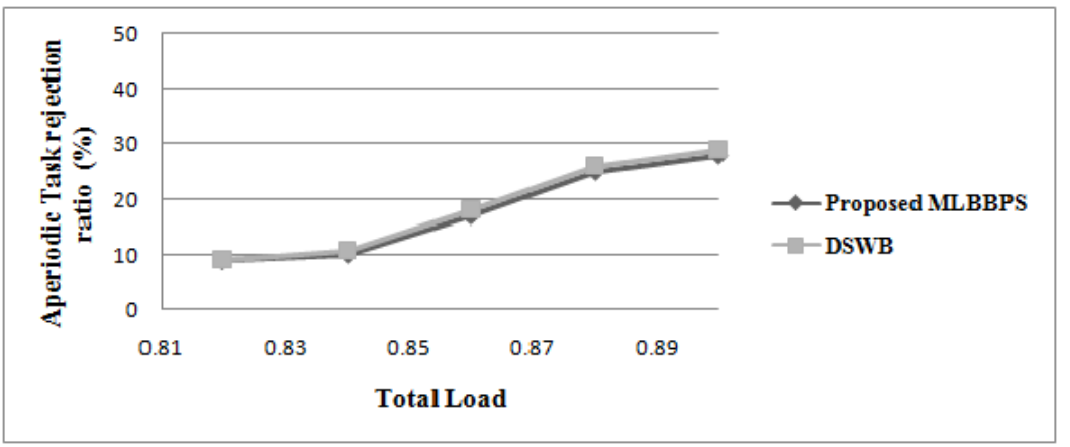


Figure 8. Shows the aperiodic task rejection ratio (%) with periodic task load 80%



between 10% to 20% then only 3% aperiodic task accepted but when aperiodic job varies in between 20% to 40% later in the proposed algorithm acceptance ratio increases up to 10% due to higher value of budget 2. The observation of Figures 7 and 8, almost 5% and 3% aperiodic task accepted correspondingly in comparison with the existing algorithm.

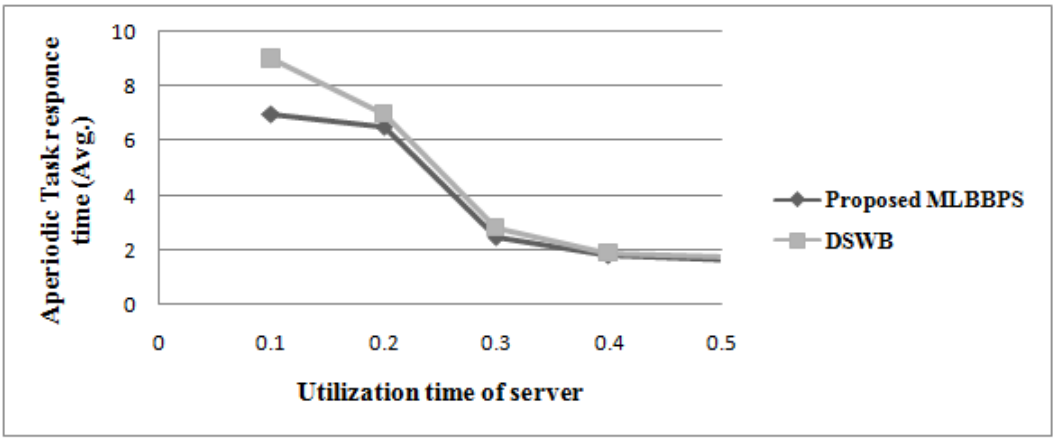
Response Time Disparity of Aperiodic Task in Presence of Periodic Task

In Figure 9, in general, periodic response time is decreased when the utilization of the server increases. For better responsiveness to the user, it is a better opportunity to finish the aperiodic task efficiently when severing utilization increases. When the server utilization raises, for better responsiveness to the user need to be finish aperiodic task earlier. When server utilization is lower almost 8% (0.1-0.2) while improvent in utilization is received 2% as compare to existing server approach in terms of average response time.

CONCLUSION

In this research work, the authors proposed a two-phase novel approach for multi-level budget bandwidth preserving server to schedule periodic and aperiodic task. One of our main objectives in this research work was to enhance the responsiveness of aperiodic task to guarantee that the real-time constraints are easily verified. The author reduces the response time of aperiodic task and provides better responsiveness with improved budget. The innovative approach provides improved budget consumption, and its accessibility achieved through utilizing the conception of multi budget with a multi priority level with deferment. The functional analysis indicates that the proposed multi-level budget bandwidth preserving server receives better responsiveness (Up to 7% - 10%) with an increased number of completed aperiodic tasks over a wide range of variations. Two-phase approach has been performed to analysis the responsiveness of aperiodic tasks, the first phase is for the construction of budget1, budget2, and proposal for MLBBPS algorithm; the second phase is for refinement of budget to increase the responsiveness of aperiodic task. An illustrative example of a functional analysis of the algorithms and simulation results shows that, it reduces overall average response time of aperiodic task approximately 13% at lowest periodic load (35%), 7% at 60% periodic load and 4% at 80% periodic load. The simulation results shows that in all observed circumstances the proposed algorithm received 7%-10% improve over existing one.

Figure 9. Response time of aperiodic tasks (avg.) with periodic task load 40% and aperiodic load 20%



Although the novel approach are quite good and constitute a set of results to guarantee the better performance in terms of average response time of aperiodic task, there are some improvements that can still be made. We may enhance our approach to include the normal aperiodic task set as well as the sporadic arrival of task set.Proposed Novel approach will be enhancing the performance of fixed-priority system with slack stealing concept.

FUNDING AGENCY

The publisher has waived the Open Access Processing fee for this article.

REFERENCES

Abeni, L., Biondi, A., & Bini, E. (2019). Hierarchical scheduling of real-time tasks over Linux-based virtual machines. *Journal of Systems and Software*, *149*, 234–249. doi:10.1016/j.jss.2018.12.008

Abeni, L., Lipari, G., & Lelli, J. (2015). Constant bandwidth server revisited. *Acm Sigbed Review*, *11*(4), 19–24. doi:10.1145/2724942.2724945

Ashjaei, M., Khalilzad, N., Mubeen, S., Behnam, M., Sander, I., Almeida, L., & Nolte, T. (2017). Designing end-to-end resource reservations in predictable distributed embedded systems. *Real-Time Systems*, *53*(6), 916–956. doi:10.1007/s11241-017-9283-6

Ashmawy, A., & Tanaka, K.(2018). *Reinforcing Total Bandwidth Server with Multivalued WCET*. Academic Press.

Brandenburg, B. B., & Gül, M. (2016, November). Global scheduling not required: Simple, near-optimal multiprocessor real-time scheduling with semi-partitioned reservations. In *2016 IEEE Real-Time Systems Symposium (RTSS)* (pp. 99-110). IEEE. doi:10.1109/RTSS.2016.019

Erciyes, K. (2019). Uniprocessor-Independent Task Scheduling. In *Distributed Real-Time Systems* (pp. 151–182). Springer. doi:10.1007/978-3-030-22570-4_7

Hamann, A., Dasari, D., Martinez, J., & Ziegenbein, D. (2018, October). Response Time Analysis for Fixed Priority Servers. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems* (pp. 254-264). doi:10.1145/3273905.3273927

Hilman, M. H. (2020). *Budget-constrained Workflow Applications Scheduling in Workflow-as-a-Service Cloud Computing Environments*. Academic Press.

Hosseinimotlagh, S., & Kim, H. (2019, April). Thermal-aware servers for real-time tasks on multi-core GPU-integrated embedded systems. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (pp. 254-266). IEEE. doi:10.1109/RTAS.2019.00029

Hussien, H., Shaaban, E., & Ghoniemy, S. (2019). Adaptive Hierarchical Scheduling Framework for TiRTOS. *International Journal of Embedded and Real-Time Communication Systems*, *10*(1), 119–135. doi:10.4018/IJERTCS.2019010107

Khan, A. A., Ali, A., Zakarya, M., Khan, R., Khan, M., Rahman, I. U., & Rahman, M. A. A. (2019). A Migration Aware Scheduling Technique for Real-Time Aperiodic Tasks Over Multiprocessor Systems. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 27859–27873. doi:10.1109/ACCESS.2019.2901411

Kim, H., Wang, S., & Rajkumar, R. (2015, August). Responsive and enforced interrupt handling for real-time system virtualization. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications* (pp. 90-99). IEEE. doi:10.1109/RTCSA.2015.15

Li, H., Lu, C., & Gill, C. (n.d.). *Predicting Latency Distributions of Aperiodic Time-Critical Services*. Academic Press.

Manthalkar, R. N. M. D. R., & Vengatesan, K. (2018). PATSAS: Periodic and Aperiodic Real-Time Task Scheduling Algorithms Simulator. *International Journal of Pure and Applied Mathematics*, *118*(20), 2681–2687.

Nascimento, F. M. S., & Lima, G. (2019, November). A Flexible Framework to Schedule Soft Aperiodic Tasks in Hard Real-Time Systems. In *2019 IX Brazilian Symposium on Computing Systems Engineering (SBESC)* (pp. 1-8). IEEE. doi:10.1109/SBESC49506.2019.9046046

Nikolov, V., Wesner, S., Frasch, E., & Hauck, F. J. (2017). A Hierarchical Scheduling Model for Dynamic Soft-Realtime System. In *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Nirmala, H. (2015). Aperiodic task Scheduling Algorithms for Multiprocessor systems in Real Time environment. *International Journal of Engineering and Computer Science*, *4*(08).

Schlatow, J., Möstl, M., & Ernst, R. (2019, April). Self-aware scheduling for mixed-criticality component-based systems. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (pp. 267-278). IEEE. doi:10.1109/RTAS.2019.00030

Syed, A., Pérez, D. G., & Fohler, G. (2018). Job-shifting: An algorithm for online admission of non-preemptive aperiodic tasks in safety critical systems. *Journal of Systems Architecture*, *85*, 14–27. doi:10.1016/j.sysarc.2018.01.005

Wu, J., & Li, J. F. (2018). An Enhanced Real-Time Deferrable Server Scheduler for Xen Virtualization Systems. *IAENG International Journal of Computer Science*, *45*(3), 403–412.

Yadav, R. S., & Agrawal, S. (2010). Enhanced aperiodic responsiveness by multi budget bandwidth preserving server. *ACM SIGBED Review*, *7*(2), 1–8. doi:10.1145/1850820.1850821

Yoder, J., Amaro, L., Hagey, R., & Bankston, M. S. (2017). *U.S. Patent No. 9,679,299*. Washington, DC: U.S. Patent and Trademark Office.

Yun, H., Yao, G., Pellizzoni, R., Caccamo, M., & Sha, L. (2013, April). Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)* (pp. 55-64). IEEE.

Zhang, Y., & Li, H. (2019). Energy aware mixed tasks scheduling in real-time systems. *Sustainable Computing: Informatics and Systems*, *23*, 38–48. doi:10.1016/j.suscom.2019.06.004

Ajitesh Kumar is working as an Assistant Professor in the department of CEA at GLA University Mathura. He has 15 years of experience in teaching and published ten research papers in the reputed journals. His current research interest includes real-time system, IoT, and its application.

Sanjai Kumar Gupta is working as an Associate Professor in the Department of CS&E at B.I.E.T., Jhansi U.P., India . He has been published more than 40 papers in different reputed journals.