# Enhanced Aperiodic Responsiveness by Multi Budget Bandwidth Preserving Server

Ranvijay[1], Rama Shankar Yadav[1], Smriti Agrawal[2]
[1]Department of Computer Science and Engineering,
Motilal Nehru National Institute of Technology, Allahabad. India.
[2]Department of Computer Science and Engineering,
Jaypee University of Information Technology, Solan, India.
ranvijaymnnit@gmail.com, rsy@mnnit.ac.in, smriti.agrawal@juit.ac.in

## Abstract

***This paper presents multi budget bandwidth preserving server (MBBPS) to improve the quality of service in terms of better responsiveness of the aperiodic task by utilizing the concept of multi budget and multi priority of the server while ensure the feasibility of periodic task at the same time. Feasibility analysis is done at offline by assigning the priority of periodic tasks as well as server to service the aperiodic tasks by fixed priority rate monotonic first algorithm. Theorem has been formulated to ensure the feasibility of the periodic tasks with enhanced budget. Besides providing better responsiveness to the aperiodic tasks, this work also improves the quality of service (QoS) by accepting aperiodic tasks that were rejected by the existing deferrable server. The complexity of proposed MBBPS algorithm is same as deferrable server. The extensive examples and simulation results illustrate that our approach can effectively reduce the average response time of aperiodic tasks as well as reduce the rejection ratio while guaranteeing the periodic tasks at the same time.***

*Keywords:* Real time systems, schedulability, aperiodic, rate monotonic first, response time.

## 1. Introduction

A real-time system is a system in which computations must satisfy stringent timing constraints besides providing logical correctness, i.e., a correct computation of the result must finish before its specified deadline. Failure to meet the specified deadline in such systems leads to catastrophic loss in case of hard real-time systems. A system may be classified as static or dynamic in nature. Static system follows fixed arrival pattern [1] examples of such applications include process control automated manufacturing system. Feasibility of static system can be determined through offline whereas feasibility of applications with random arrival patterns (termed as aperiodic tasks) are determine through online. For example, in aircraft control application system has to respond the pilot's command while continuing to execute the tasks from air traffic controller for flying the airplane. The pilot commands are random in nature and are activated as a result of certain event such as variation in pressure, cloudiness, emergency landing etc. however the command issued by air traffic controller regarding of altitude, position velocity etc are periodic in nature. The periodic tasks typically arise from sensor data, radar or control loops. Failure to complete the tasks within their deadline can have catastrophic loss while the aperiodic tasks generally arise from operator actions (pilot action) or random events. The problem is to service the aperiodic tasks as quickly as possible without jeopardizing the deadlines of the periodic tasks. Full guarantee can be given to fixed arrival pattern task while, no guarantee can be achieved for the case of random arrival pattern. The predictability can be used to measure the performance of dynamic system. More the predictability betters its performance. Offline feasibility analysis is used for static system consider worst case requirement in terms of execution time, arrival pattern, critical instance release (a task is released along with all higher periodic tasks) task. Thus, there is a need of recourse manager that gives full guarantee for static system while receiving better predictability for dynamic one. On the way to design such a system, there are two major design objectives. The first objective is to maintain the feasibility of periodic tasks set in the presence of aperiodic tasks. The second objective is to minimize the average response times for aperiodic tasks while reducing the rejection ratio of aperiodic task. The authors [5, 7, 8, 10] have formulated the scheduling policies for both periodic as well as aperiodic task, but they have not considered additional budget apart from assigned budget at offline to the server.

The next section deals with overview of existing work available to tackle system having both periodic and aperiodic tasks.

## 2. Related work

This section summarises the scheduling algorithm used for system having a mixture of aperiodic tasks and periodic hard real-time tasks. Full guarantee for periodic tasks are gives using offline rate monotonic first [2], deadline monotonic first [2] over the full guarantee assigned to periodic tasks, aperiodic tasks are scheduled at run time. The simplest way of scheduling aperiodic tasks is background approach [6] where aperiodic task executing only when there is no periodic tasks available at that time. Besides honouring feasibility of periodic task assigned at offline, the background approach suffer greater set back of having either longest response time for

aperiodic one or lesser number of aperiodic tasks completes within their respective deadlines. In contrast of assigning lower priority for aperiodic tasks than that assigned to periodic one, a group of researcher [1, 4, 5] assign highest priority to aperiodic tasks. This gives best response time for aperiodic tasks. However, it may leads to failure of feasibility given to periodic task at offline. Over these two extreme priority assignment approach a group of researchers [6, 7, 8] schedule aperiodic tasks through utilizing the concept of periodic budget allocation. That is, they allocate budget at fixed point of time and aperiodic task if available execute by consuming the budget. This is termed as server composed of budget replenishment and consumption rules. The server is characterized by an ordered pair $(q_s, p_s)$, where $q_s$ is the budget replenished at interval of $p_s$ with utilization ratio, $\text{Us} = q_s/p_s$. The simplest server is the polled server (PS) [7] which replenishes the budget of amount $q_s$ at integral multiple of $p_s$ and process any pending aperiodic requests within the limit of its budget $q_s$ . However, if no aperiodic requests are available, the sever immediately suspends itself until next replenishment time. The aperiodic request arrives just after the server has suspended itself (lapse the budget) has to wait till the next replenishment time. To incorporate the above shortcoming authors [7] extended polled server as deferrable server. Here, consumption rules of polled server has modified through the utilizing the concept of preserving the budget (if not consumed by execution of aperiodic tasks) up to next replenishment time. At next replenishment time fresh budget of amount $q_s$ is released. Though deferrable server provides opportunities for execution of aperiodic task all the time provided budget is available however, it may accumulate budget of amount twice the $q_s$ in some interval of length $p_s$ . This may leads to infeasibility of periodic task that were feasible without deferment of budget. Although the Deferrable Server algorithms do significantly improve aperiodic responsiveness, but they have several important scheduling issues for aperiodic tasks are not addressed by this algorithm. A lower priority periodic task could miss its deadline even if the total utilization of the system with n tasks is not greater than 0.693. in other word a lower-priority periodic task could miss its deadline even if the total utilization of the system with n tasks is not greater than $\text{n}(2^{\frac{1}{\text{n}}} - 1)$.

This paper proposed a scheduling algorithm provide fast responsiveness to aperiodic tasks while still ensure the feasibility of the periodic tasks at a high level of periodic tasks utilization. The proposed algorithm, called multi budget bandwidth preserving server (MBBPS), is extension of existing deferrable server which is able to substantially reduce the response time of aperiodic tasks by additional budget apart from assigned budget and improve the priority of the server for the case when server is consuming additional budget. Besides providing better responsiveness to the aperiodic tasks, this work improves the quality of service (QoS) by accepting aperiodic tasks previously rejected by the existing techniques due to lacking of budget and their replenishment and consumption rule. Theorem 1 has been formulated to ensure the feasibility of the periodic tasks. The proposed multi budget bandwidth preserving server has additional feature to provide better quality of service (QoS), predictable behaviour, and ease of implementation. The rest of the paper is organized as follows: in section 3, we describe our system model, terminologies used. Section 4 elaborates our proposed approach with feasibility analysis followed by results and analysis in section 5. Finally, paper concludes with section 6.

## 3. SYSTEM MODEL

This system deals with fixed arrival pattern periodic tasks along with random arrival pattern of aperiodic tasks. The feasibility of periodic tasks is determined as offline using static priority assignment technique, Rate monotonic first [2].The rate monotonic algorithm assigns priorities in inverse relation to task periods, that is the shorter the task's period, the higher the task's priority. In case two tasks have same priority any one can be given higher priority. The priority assigned to server is the priority for execution of aperiodic tasks limiting to availability of budget.

Following considerations are taken same as considered in [7]

1. System consists of n independent periodic tasks$\tau_1,\ \tau_2,\ \tau_3 \ldots \tau_n$. Each task $\tau_i$ has the attributes, worst-case execution time($e_i$), period($p_i$), and relative deadline($d_i$).
2. Relative deadline of a periodic task is less than or equal to its period.
3. In addition to periodic there is another periodic task $\tau_s$ characterized by an ordered pair $(q_s, p_s)$ that is used to provide execution budget for aperiodic one. Here, $q_s$ are amount of budget released with period $p_s$.
4. All overhead for scheduling, context switching considered negligible.

The terms used in this paper are summarized as follows

**Terms used:**

***Finish Time*** $(ft_i^j)$***:*** For $j^{th}$ release of a task $\tau_i$, $\tau_i^j$ it is the sum of its own requirement and requirement of the higher priority tasks released between $\tau_i^j$ and its completion time.

$$ft_i^j = rel_i^j + e_i + \sum_{k=1}^{k=i-1} (\lceil t/p_k \rceil) * e_k \quad where\ rel_i^j \le t \le p$$

***Response Time*** $(\tau_i^j)$***:*** It is difference between the finish time and the release time of a release $\tau_i^j$. Mathematically, $Res_i^j(\tau_i^j) = ft_i^j - rel_i^j$

***Critical instance release of task*** $\tau_i$***:*** It is defined as time when a task $\tau_i$ is released along with all higher priority tasks.

***Periodic task set (***T***):*** Periodic task set T is the set of union of n periodic task and server ($\tau_s$).

Mathematically, $\text{T} = \{\tau_1,\ \tau_2,\ \tau_3 \ldots \tau_n\} \cup \{\tau_s\}$

***Hyperperiod (L):*** it can be defined as the point after which all the task in the task set T are in phase and schedule pattern for each task is restarted i.e. the release pattern of tasks at time $t = 0$ is repeated at integral multiple of hyperperiod.

Mathematically, $L = LCM\ (p_1, p_2, p_3 \ldots \ldots \ldots p_n, p_s)$

***Budget1:*** budget 1 is the original fixed budget ($q_s$) allotted to the server and replenished at every integral multiple of $p_s$.

***Budger2:*** budget 2 is the budget generated by laxity of a release of lowest priority periodic task.

## 4. PROPOSED MULTI BUDGET BANDWIDTH PRESERVING SERVER

The proposed MBBPS algorithm is extension of deferrable server [7] through utilizing the concept of providing additional budget over the (budget 1) supported by deferrable server. This additional budget is limiting to slack available for lowest priority periodic task. The amount of excess budget (termed as budget 2) is computed for each release of lowest priority periodic task up to hyperperiod ($L$). Hence, budget 2 may not same for each release and form budget pattern of length equal to hyperperiod. This budget 2 pattern is repeated from one hyperperiod to another hyperperiod. The server is executed at two priority levels (assigned at offline and highest priority). For consumption of budget1, server executes at assigned priority whereas it priority is raised to highest priority for the case of budget2. Server loses any unused budget 1 at the end of the period of server and its full capacity budget1 ($q_s$) is restored. However, unused budget 2 drop at the deadline of lowest priority release and new budget 2 will be computed to the laxity of the laxity of the next release. The detailed consumption and replenishment rules for two types of budgets are given below.

### *4.1 Consumption rules:*

Among two available budget, the consumption of budget is preferred whose expiry time is least. The expiry time of budget1 is next replenishment time whereas, absolute deadline of lowest priority task's release is the expiry of budget2. The rules are as below.

1. The budget of the server is consumed at the rate of one per unit time whenever the server executes aperiodic tasks.
2. Whenever the system is idle (periodic and aperiodic queues are empty) and server has budget then budget is consumed at the rate of one per unit time.

### *4.2 Replenishment rule*

***For budget1:***

The budget1 of the server is set to $q_s$ at time instant $Kp_s$, for $K$ =0, 1, 2, 3,……..and $p_s$ is the period of server.In other word fresh budget1 "$q_s$" is replenished at every period $p_s$ of the server.

***For budget2:***

1. The budget is replenished at every integral multiple to the period $(p_l)$ of lowest priority periodic task.
2. Budget2 of is set to laxity of that lowest priority periodic task released at replenishment time.
3. Laxity of the lowest priority release $\tau_l^j$ is computed using equation (1)

$$\text{laxity } \tau_l^j = d_l - e_l - \sum_{k\epsilon\tau_H \cup \tau_s} (\lceil j d_l/p_k \rceil - \lceil (j-1)d_l/p_k \rceil) * (e_k) \quad (1)$$

Where $\tau_H$ is the set of all higher priority tasks released between release time of $\tau_i^j$ and its deadline and $j = 1, 2, 3, \ldots\ldots\ldots\ldots \lfloor L/p_l \rfloor$.

The proposed multi bandwidth preserving server defers the unused budget up to its expiry time. Theorem has been formulated to ensure feasibility of periodic tasks at run time with enhanced budget.

***Theorem 1:***

*In multi budget bandwidth preserving server having fixed budget governed by periodic replenishment time, and variable budget computed on the basis of laxity of lowest priority periodic task, feasibility of periodic task set is retained at run time iff waiting time $W_l^j$ for $j^{th}$ release of lowest priority periodic task $\tau_l$ is not more than $d_l - e_l$.*

***Proof***: when $j^{th}$ release of task $\tau_l$ is comes with all higher priority periodic release along with the server in this case the finish time for the $j^{th}$release of task $\tau_l$ is maximum and completed within their deadline then all periodic task is also schedulable. Since we are using fixed priority rate monotonic first algorithm so all the higher priority task released between the release time of $\tau_l^j$ and its deadline are must be completed up to their respective deadline if we ensure the feasibility of $j^{th}$ release of task $\tau_l$

The maximum waiting time for $j^{th}$ release of a task $\tau_l$ will be the sum of execution time of all higher priority periodic tasks released between the release time of $\tau_l^j$ and the deadline of $\tau_l^j$.

Mathematically,

$$W_i^j = \sum_{k\epsilon\tau_H \cup \tau_s} (\lceil j * d_l/p_k \rceil - \lceil (j-1)d_l/p_k \rceil) * (e_k) + \text{laxity}\tau_l^j \quad (2)$$

For ensuring feasibility of $\tau_l^j$, wait time $W_i^j$ not more than $d_l - e_l$

Mathematically,

$$W_l^j = d_l - e_l \quad (3)$$

Substitute the value of $W_i^j$ from equation 2 into equation 3, we get

$\sum_{k\epsilon\tau_H \cup \tau_s} (\lceil j * d_l/p_k \rceil - \lceil (j-1)d_l/p_k \rceil) * (e_k) + \text{laxity}\tau_l^j = d_l - e_l$ , put the value of $\text{laxity}\tau_l^j$ from equation 1. We get,

$\sum_{k\epsilon\tau_H \cup \tau_s} (\lceil j * d_l/p_k \rceil - \lceil (j-1)d_l/p_k \rceil) * (e_k) + d_l - e_l -$
$\sum_{k\epsilon\tau_H \cup \tau_s} (\lceil j * d_l/p_k \rceil - \lceil (j-1)d_l/p_k \rceil) * (e_k) = d_l - e_l$

First and fourth terms are cancelled, we get, LHS = RHS, hence, proved.

**Offline approach for consumption of budget 1**

The proposed multi budget band width preserving server improves the quality of service (QoS) in terms of responsiveness and number of aperiodic tasks completed up to their respective deadline. This is a result of laxity based budget pattern and improved priority in case server is consuming budget2. The scheduling algorithm for schedule periodic tasks along with the aperiodic by using the proposed multi budget band width preserving server (MBBPS) is summarized as below.

***MBBPS Algorithm (task set T*** $= \{\tau_1, \tau_2, \tau_3 \ldots \tau_n \cup \tau_s\}$)

*// periodic task set T is union of periodic tasks and server* ($\tau_s$).

*// critical instance release of tasks in T is considered*

*Begin*

*1. Check the feasibility of tasks in T using fixed priority rate monotonic first algorithm.*

*2. for* $J^{th}$ *release of* $T_l$ *where* $T_l$ *is the lowest priority task in the periodic task set T and* $j = 0,1,2 \ldots L/p_l$.

*Do*

*a. allocate budget1 of* $q_s$

*b.expiry_time_budget1=replenishment time + period of replenishment*

*c. next_replanishment _time_budget1 = expiry_time_budget1*

*d Compute the laxity* $\tau_l^j$ *using equation 1*

*e. expiry_time_budget2= replenishment time + deadline of lowest priority task.*

*f. next_ replenishment _time_budget2=next release of lowest priority periodic task.*

*g. Formation of budget pattern from one hyperperiod to another hyperperiod*

*While (aperiodic queue is not empty)*

*If* $(total\ budget > 0)$

*If(expiry_time_budget*$1 \leq$ *expiry_time_budget2 &&* $budget\ 1 > 0$)

*Consume budget 1 first at server assigned priority in offline followed by the budget2 at highest priority as per as the consumption rule*

*Else*

*Consume budget 2 first at highest priority followed by the budget1 at server assigned priority in offline and consume as per as the consumption rule*

*Else*

*Aperiodic task is waiting for the next replenishment of budget 1 and budget 2which is earlier.*

*End if*

*End if*

*End for*

*End*

The effectiveness of proposed approach can be observed in the example 1.

Example1: Considering a periodic task set $T = \{\tau_1, \tau_2\} = \{< e_i, p_i, d_i, >: < 12, 20, 20 >, < 6, 60, 60 >\}$ and deferrable server having attribute $\tau_s = (q_s, p_s) = (6, 30)$. The four aperiodic tasks A1, A2 and A3 A4 arrives at times 12, 34, 72 and 92 with their respective execution time 8.0, 8.0, 2.0 and 12.0. Their respective deadlines are 34, 77, 80 and 118 units.

The schedule for deferrable server (DS) [7] and proposed multi budget bandwidth preserving server (MBBPS) are shown in figure 1 and figure 2 respectively. Here, at time t=0, budget1=6 units is replenished. The laxity of $\tau_2^1$ is 6 units (computed using equation 1) as a result budget 2=6 units is available at time t=0. That is, for MBBPS, the total budget (sum of budget 1 and budget 2) available at t=0 is 12 units against 6 units budget is available in case of existing DS. The expiry times for budget 1and budget 2 are 30 and 60 respectively. Thus, consumption of budget 1 is preferred over budget 2 as expiry time of budget is less than budget 2.

On the arrival of first aperiodic task, A1, at t=12, $\tau_2^1$ and A1 are available in periodic and aperiodic queues respectively. As priority of server is more than priority of $\tau_2$, A1 starts executing with consumption of budget 1 and its budget 1 exhaust at t=18. Now server starts consuming budget 2 with raised (highest) priority and complete A1 at t=20. While, having unused budget2 of amount 4 units. This unused budget is retained up to its expiry time t=60. The next replenishment time of budget 1 is t=30, so in the interval 30 to 60, 10 units of budget is available for the case of MBBPS against only 6 units is available in DS [7]. That is, in MBBPS each interval of length equal to hyperperiod (60) amount of total budget available is 18 units as compared to 12 units' budget in DS. The modified approach not only reduces the average response time of aperiodic tasks but also reduces the rejection ratio of aperiodic tasks to improve the quality of service (QoS). Besides reduction in response time and rejection ratio of aperiodic task it also reduces idle time of the system giving improved system utilization. The effectiveness of proposed approach has been summarized in table 1.

**Table 1: performance measurement of example 1**

| Aperiodic task | Response time with DS [7] / status | Response time with MBBPS |
|---|---|---|
| *A1* | 22 (accepted) | 8 (accepted) |
| A2 | 42 (accepted) | 8 (accepted) |
| A3 | 20 (accepted) | 6 (accepted) |
| A4 | 64 (rejected) | 12 (accepted) |
| **Budget available within hyperperiod**, | | |
| **DS** | | **MBBPS** |
| 12 units | | 18 units |
| **System idle time with or without budget** | | |
| **DS** | | **MBBPS** |
| 6 units without budget | | 6 units with budget |
| **System utilization with in hyperperiod** | | |
| **DS** | | **MBBPS** |
| 54/60= (0.9) | | 58/60= (0.966) |

Thus, it can be observed from example that proposed multi budget bandwidth preserving server improve the quality of service (QoS) in term of better responsiveness of aperiodic task and accept more number of aperiodic tasks while retaining feasibility of periodic tasks. The next section deals with performance measurement of multi budget bandwidth preserving server through simulations.

## 5. SIMULATION RESULTS AND DISCUSSION

In this section simulation of synthesized task set are performed to evaluate the performance of the proposed multi budget bandwidth preserving server, with exiting deferrable bandwidth preserving server. Here, we compare the performance of multi budget bandwidth preserving server refer as MBBPS with exiting deferrable server refer as DS [7]. The key parameters for performance measurment are average response time and acceptence ratio. For that we taken ten

different periodic task sets each containing eight periodic tasks, and it is randomly generated. The other parameter of simulation is summarized in table 2. The aperiodic task are genearted using poision distribution and simulation is run for 10000 aperiodic tasks.

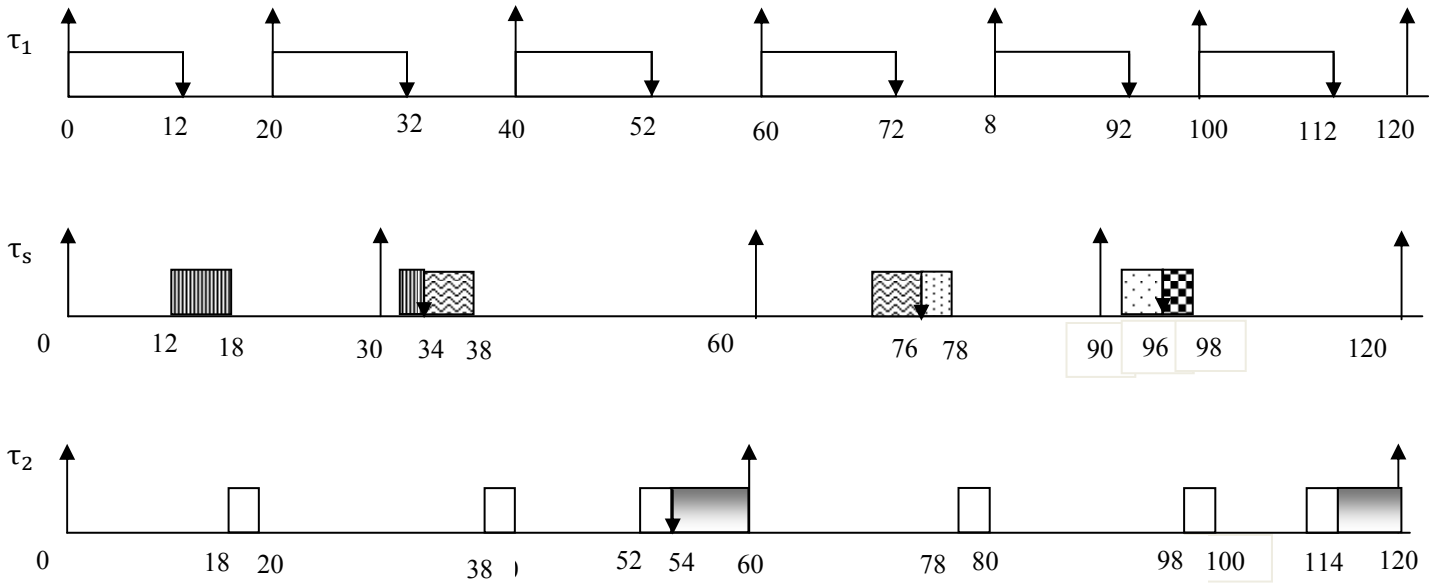



Figure 1: Scheduled for existing deferrable server

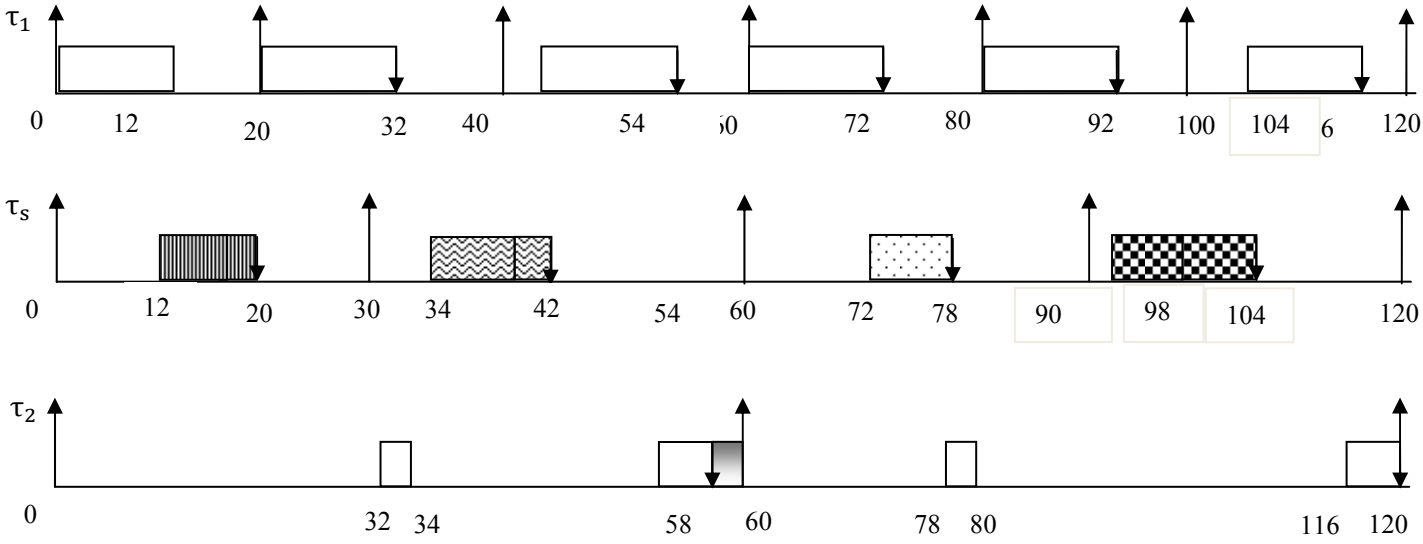
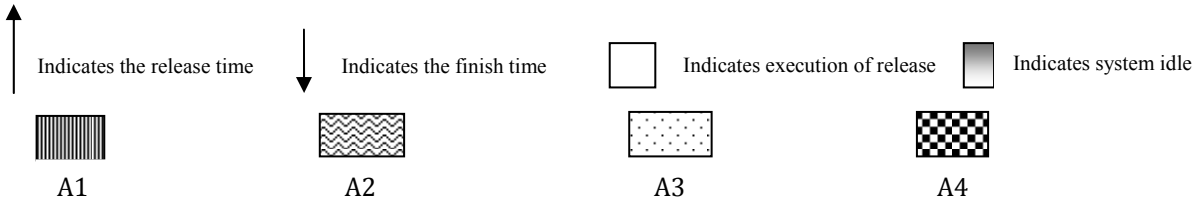



Figure 2: Scheduled for proposed multi budget bandwidth preserving server



| Table 2: Simulation Parameters | | |
|---|---|---|
| **Parameter** | **Condition** | **Range** |
| $UTh$ Utilization Threshold | Is assigned | 0.01 |
| $u_i$ Utilization | If $U - \sum u_{i-1} \geq UTh$ the select a uniform random number | $(0, U - \sum u_{i-1}]$ |
| | If $U - \sum u_{i-1} < UTh$ then assign | $u_i = U - \sum u_{i-1}$ |
| $e_i$ worst case execution time | select a uniform random number | (0,100] |
| $p_i$ period | select a uniform random number | (0,1000] |
| $d_i$ deadline | select a uniform random number | $[e_i, p_i]$ |
| *For each periodic task set, three periodic loads were chosen for simulation* | assigned | 40%, 60%,80% |
| *Aperiodic task arrival time* | Using a Poisson arrival process | |
| *Aperiodic service times* | Using exponential service time distribution | 2%, of the server's period |

In the following section we measure the effect of variation in aperiodic load, periodic load and server utilization on the average response time of aperiodic task and rejection ratio of aperiodic task.

***Effect of load on Average response time of aperiodic task:***

The effect of load on the average response time of aperiodic task ca n be seen from the figure 3 figure 4 and figure 5. Figure 3 compare the performance of proposed multi budget bandwidth preserving server MBBPS with existing deferrable sever DS when periodic load is 40% of total load and server is of utilization of 0.2 and aperiodic load varies from 10% to 60%. We observe from the figure as total load increase average response time of aperiodic task increases. when the aperiodic load is 30% to 60% MBBPS approach have significant reduction almost 15% in average response time of aperiodic tasks over existing DS[7]. This is because budget 2 will be better utilized by occurrence of more aperiodic task. While when the aperiodic load is varied from 10% to 20% MBBPS approach have almost 5% reduction in average response time of aperiodic tasks This is due to the most of the time aperiodic will complete within its allotted budget i.e. budget 1 (Qs). Because periodic load is less (40%) so it will less interference to aperiodic task. Figure 4 compare the performance of proposed multi budget bandwidth preserving server MBBPS with existing deferrable sever DS when periodic load is 60% of total load and aperiodic load varies from 5% to 40%. In figure 3 the periodic load is fixed and it ig 40% but in figure 4 it is 60% so as the periodic load increase the amount of budget 2 will be deceases. As aperiodic load varied from 5% to 20% MBBPS perform almost 7% better to the existing one. While in the figure 3 it is 5 % better because more periodic interfere the aperiodic task so it may be budget 1 is not properly utilize so most of the time our approach utilizes the budget 2 while when aperiodic varied from 20% to 40%, 10% improvement is received over existing one. While in figure 5 compare the performance of proposed multi budget bandwidth preserving server MBBPS with existing deferrable sever DS when periodic load is 80% of total load and aperiodic load varies from 5% to 20%. Our proposed approach is almost 3% better over exiting one due to the amount of budget 2 is less as compare to figure 3 and figure 4.

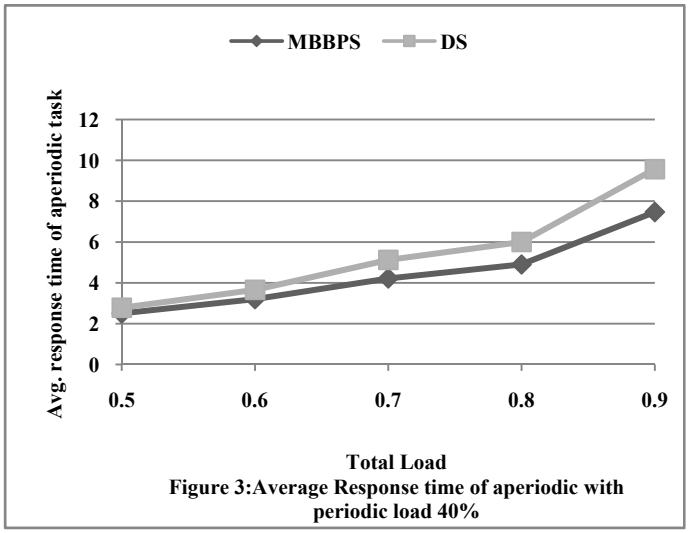


**Figure 3:Average Response time of aperiodic with periodic load 40%**

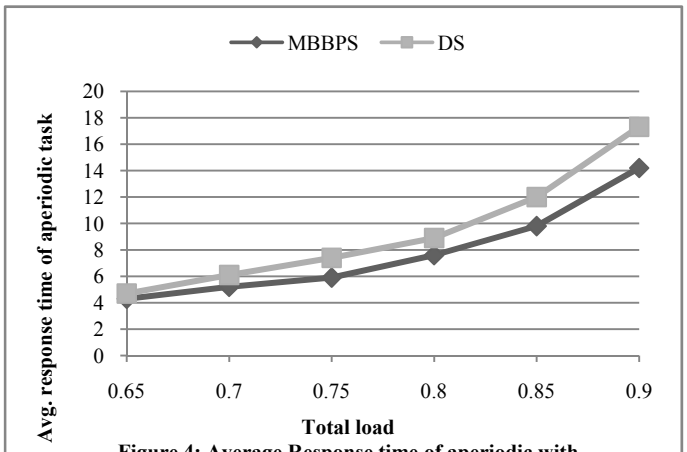


**Figure 4: Average Response time of aperiodic with periodic load 60%**

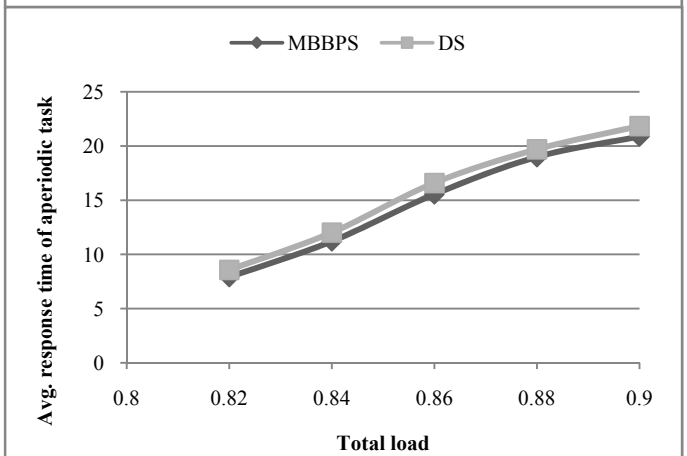


**Figure 5: Average Response time of aperiodic with periodic load 80%**

***Effect of load on rejection ratio of aperiodic task:***

The effect of load on the rejection ratio of aperiodic task can be seen from the figure 6 figure 7 and figure 8. Figure 6 ,figur7, figure 8 compare the performance of proposed multi budget bandwidth preserving server MBBPS with existing deferrable sever DS when periodic load is 40%, 60% and 80% of total load and aperiodic load varies from 10% to 60% , 5% to 40% and 5% to 20% respectively. In figure 6 when periodic load is 40% and aperiodic varied from 10 to 20%, only 3% more aperiodic task are accepted because most of the time aperiodic will service by budget 1 hence, most aperiodic will be accepted by both the approach. But when the aperiodic load varied from 30% to 60% our approach accept 10% more aperiodic task by assigning higher value to budget 2 as well as better utilization of budget 2. While, in figure 7 and figure 8 almost 5% and 3% more aperiodic tasks are accepted respectively over existing one.

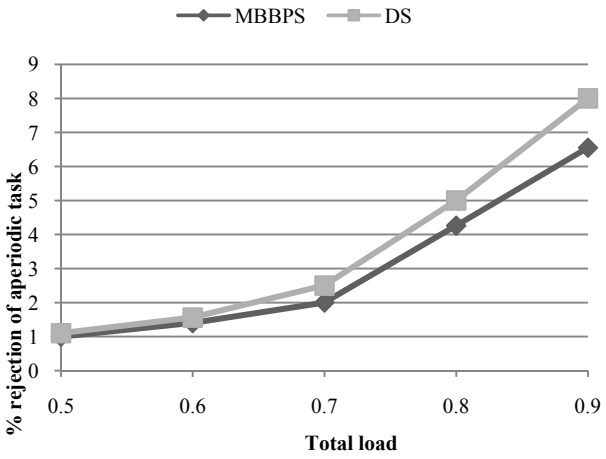



**Figure 6: % rejection of aperiodic task with periodic load 40%**

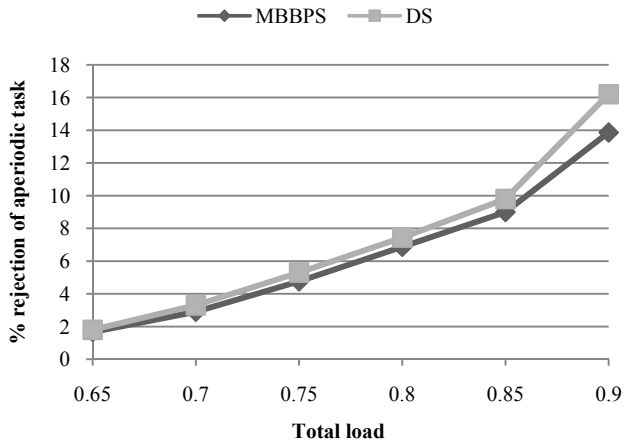



**Figure 7: % rejection of aperiodic taskwith periodic load 60%**

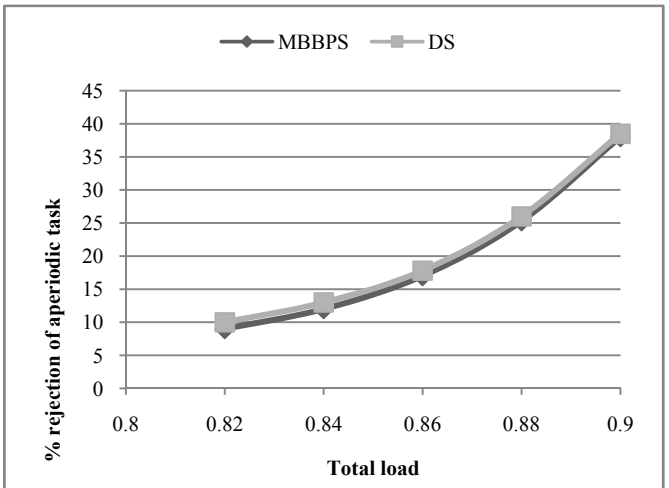



**Figure 8: % rejection of aperiodic taskwith periodic load 80 %**

***Effect of server utilization on Average response time of aperiodic task:***

The effect of server utilization on the average response time of aperiodic task can be seen from the figure 9. Figure 9 compare the performance of proposed preemption control deferrable server PDS with existing deferrable sever DS when periodic load is 40% of total load and aperiodic load is 20%. It is observed that the average response time of aperiodic task of both approaches decreases with the increment in sever utilization. As the server utilization increases aperiodic task have a better opportunity to finish earlier leading to better responsiveness to the user. At lower server utilization (0.1-0.2) almost 10% while at utilization (0.2-0.4) 2% is improvement is received as compare to exiting one in terms of average response time.

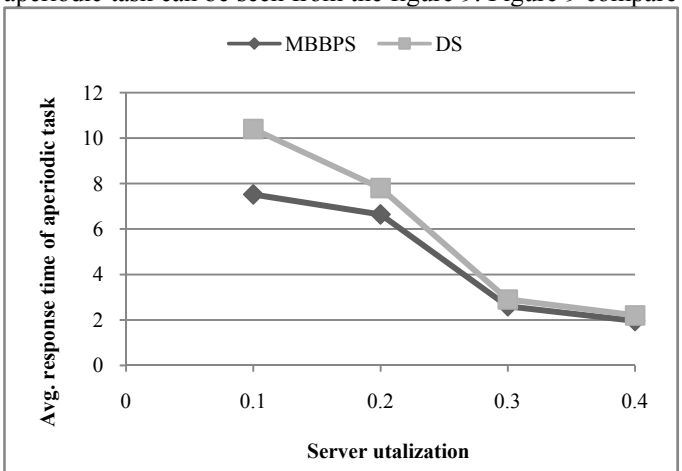



**Figure 9:Average Response time of aperiodic with periodic load 40% and aperiodic load 20%**

## 6. CONCLUSION

In this paper we have proposed multi budget bandwidth preserving server for scheduling of mixed task set. We provide better responsiveness to aperiodic task by reducing the response time by improving the availability of enhanced budget through out. The improved budget and its availability are achieved through utilizing the concept of multi budget with deferment. Here, server maintains two types of budget one follow periodic server while lowest priority periodic task is source for other. The replenishment and consumption rules have thoroughly modified to ensure feasibility of periodic tasks with excess budget pattern. The proposed algorithm has improvement in terms of responsiveness, budget availability and its utilization. The examples and simulation studies has carried out. It has been observed that the proposed scheduling algorithm reduce the overall average response time of aperiodic tasks is approximately 12 % at lower periodic load (40%), 6% at medium periodic load (60%) and 3% at higher periodic load (80%) while 7% at lower periodic load (40%), 5% at medium periodic load (60%) and 2% at higher periodic load (80%) improvement is received for acceptance ratio of aperiodic tasks over existing one. Thus, extensive simulation and illustrative example shows that our proposed approach is capable of performing better in terms of average response time of aperiodic task as well as acceptance ratio of aperiodic task while retaining the feasibility of periodic tasks.

## References:


[1] Lehoczky, J. E, and Ramos-Thuel, S "An optimal algorithm for scheduling soft-aperiodic tasks in real-time systems". In Proceedings of Real-Time Systems Symposium, December, pp. 110-123. 1992.

[2] J. P. Lehoczky. L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," in proc. 10th IEEE Real-Time Syst. Symp., 1989, pp. 166-171.

[3] M. Joseph and P. Pandya, "Finding response times in a real-time system,'' Compur. J., vol. 29, no. 5, pp. 390-394, 1986.

[4] A.Burns, K.Tindell, A.J.Wellings, "Fixed Priority Scheduling with Deadlines prior to Completion", IEEE 1994.

[5] Lin, T. H., and Tarng, W.Scheduling periodic and aperiodic tasks in hard real-time computing systems. In Proceedings ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, May, 1991, pp. 31-38.

[6] P.L., "Fixed Priority Scheduling of Periodic task sets with arbitrary deadlines", Proceedings 11th IEEE Real-Time Systems Symposium, Lake Buena Vista, FL, USA, pp.201-209, December 1990

[7] J. K. Strosnider, J. P. Lehoczky, and L. Sha, "The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments," IEEE Trans. Comput., vol. 44, no. 1, pp. 73–91, Jan. 1995.

[8] B. Sprunt, L. Sha, and J. P. Lehoczky, "Aperiodic task scheduling for hard real-time systems," J. Real-Time Syst., vol. 1, no. 1, pp. 27–60, 1989

[9] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in Proc. IEEE Real-Time Systems Symp., Madrid,Spain, 1998, pp. 4–13.

[10] Jane W. S. Liu, "Real-Time Systems" Prentice Hall, 2000, ISBN-10: 0130996513