A New Slack Reclaiming Algorithm for Real-time Systems

Wenzhi Chen
College of Computer Science and Technology, ZheJiang University, HangZhou 310027, China
Email:chenwz@zju.edu.cn

Qingsong Shi, Weifang Hu, Wei Hu and Sha Liu
College of Computer Science and Technology, ZheJiang University, HangZhou 310027, China
Email:{zjsqs, huwf, huwei,liusha}@zju.edu.cn

Abstract: Real-time applications are ubiquitous in general-purpose computing environments, while the real-time systems are growing in complexity. Thus in these hybrid real-time systems, schedulers must guarantee that all hard real-time jobs be completed before their deadlines and improve QoS of soft real-time tasks as much as possible. Towards this goal we have proposed a new slack reclaiming algorithm for server-based real-time systems, and have also implemented it in a real time emulator (RTSIM). This algorithm, named HBASH, which enhances the Constant Bandwidth Server (CBS) by slack reclaiming, allocates slack generated from the running process to the task that needs the slack most, and then this selected task will be scheduled immediately. Hence the algorithm is able to make full use of slack and reduce the response time of soft real-time tasks as much as possible. In this paper, we proved that our algorithm does not violate the schedulability of tasks, and we also evaluated the performance of this algorithm. The experimental results demonstrate that HBASH outperforms other slack reclaiming algorithms and improves soft real-time performance significantly.

Index Terms: server-based; scheduling algorithm; slack reclaiming

I. INTRODUCTION

Modern operating systems always support applications with a variety of timing constraints including hard real-time, soft real-time and best-effort. In recent years, to guarantee performance, many researchers have proposed some effective solutions. For example, the hierarchical HLS scheduler [1], the integrated RBED scheduler [12], and two-level hierarchical scheme [3], server-based schedule, etc. And the server-based scheduling algorithm is becoming a hot subject of research. A server is similar to a virtual processor with a certain speed, and is dispatched by scheduler in EDF (Earliest Deadline First) [2]. Every real-time task is served by a dedicated server. Common server-based algorithms include CUS(Constant Utilization Server)[3],TBS(Total Bandwidth Server)[4]、CBS(Constant Bandwidth Server)[5], etc. From the perspective of system utilization, CUS and TBS are able to isolate logical irrelevant applications, and solve scheduling problem with coexistence of periodic and sporadic real-time task. But we must provide task's precise running parameters for these algorithms, which makes them not suitable for soft real-time tasks, such as multimedia tasks. CBS algorithm, raised by L. ABni in 1998, doesn't need to know task's precise parameters. So that it is more suitable for soft and aperiodic real-time tasks. In addition, the algorithm recharges the server immediately when the server exhausts its budget, therefore it is able to reduce the average delay as much as possible.

In CBS algorithm, when a task budget is less than its execution time, it may be delayed for a long time. On the contrary, the system will waste a lot of budget if without reclaiming. Recent researches have begun to address this problem, raising some slack reclaiming algorithms including IRIS[9], GRUB[8], CASH[6], BASH[7], etc. IRIS enhances CBS with fairer slack reclaiming, but it is unable to recharge the server immediately when server exhausts budget[13]. Slack is not reclaimed until all current jobs have been serviced and the system will otherwise be idle[14]. GRUB is a CBS-like algorithm that dynamically allocates excessive capacity to active servers, but these dynamic operations will cause a large overhead. CASH extends CBS with slack reclaiming algorithm[15]. When a server becomes idle with residual budget, the slack is inserted to the cash queue ordered by deadline. Whenever a new server is scheduled for execution, it will firstly use any queued budget whose deadline is less than or equal to its own. CASH has the shortcoming: the extra budget is only allocated to the earliest deadline task, and it will be unfair to the other tasks which may have more needs of the slack. BASH enhanced CASH, in which it operates better; however, it also has the same drawback.

A new slack reclaiming algorithm based on CBS is proposed in this article. The algorithm will allocate slack to the task with earliest virtual deadline, not always the task with the earliest deadline, and it will schedule the task which has got the slack immediately. Experiments have proved that the algorithm can guarantee that all of hard real-time jobs be completed before their deadlines

This paper is based on "Introduction to Reform on the Course of Fundamental Logic and Computer Design" by Qingsong Shi, Wei Hu, Sha Liu, Tianzhou Chen, Which appeared in the Workshop of the 9th International Conference for Young Computer Scientists(ICYCS 2008), The First International Workshop on Computer System Education and Innovation(IWCSEI 2008), p2528-2532, Zhang Jia Jie, Hunan, China, November 18-21, 2008.
This research was supported in part by National Foundation Research Project No.A1420080190 and No.9140A15040309JW0402

and reduce the average latency of soft real-time tasks as much as possible.

The rest of the paper is organized as follows: Section 2 specifies our notation, definition and basic assumptions. Section 3 describes our scheduling algorithm in detail and analyzes the algorithm. Section 4 illustrates some experimental results achieved on a real-time simulator (RTSIM). Section 5 contains our conclusions and future work.

II. Terminology and Assumptions

A. Task Module

Generally, we identify a specific application in real-time system as task, such as displaying a video clip. Each task consists of a sequence of jobs, for example, displaying video requires decompressing frame data, and decompressing each frame can be seen as a job of the task. We consider that a system is composed of three types of task: hard, soft, and best-effort task.

A task Ti is characterized by the parameters (C_i, P_i, D_i), where C_i represents worst case execution time (WCET) for hard real-time tasks, otherwise, mean execution time for soft real-time tasks, P_i is the minimum inter-arrival time for successive jobs, and D_i is the relative deadline for the task.

The j^{th} job of the task Ti is $J_{i,j}$, characterized by the parameters ($r_{i,j}$, $e_{i,j}$, $d_{i,j}$), $r_{i,j}$ represents the release time for the job , $e_{i,j}$ represents the execution time for job, and $d_{i,j}$ represents the relative deadline for job.

B. Server Module

Each server is associated to a task and is characterized by pair（B_i , P_i）, where P_i is the period of the server, B_i is the budget of the server. For hard real-time task, B_i is set to the task's WCET, and P_i is set to task's period. For soft real-time task, B_i is set to the task's mean execution time, and P_i is set to task's expect period. Both the relative deadline d_k (k represents the number of server's recharging times) and the current budget q are associated with the server at each instant. When the server is recharged, q is set to max budget (q = B), and its deadline is increased（$d_{k+1} = d_k + P$).

The bandwidth for the server S_i is U_i ($U_i = B_i/P_i$), and the total bandwidth of all servers can't exceed 1, i.e. $\sum_{i=1}^{n} U_i \le 1$.

Each server can be in one of the following states:

1. idle: the served task has no pending job.
2. ready: the server is ready to execute, waiting for being scheduled.
3. executing: the server is now being scheduled ,and the served task is running.

III. HBASH Algorithm

In this section, we will introduce this algorithm in detail. The algorithm includes two parts: one is the global scheduling algorithm, the other is the slack reclaiming algorithm. In running process, when the server has extra budget, the latter algorithm will be called by the former. In this algorithm, the slack will be allocated to the task which demands it most. When a task gets the slack, it will be scheduled immediately. Firstly we introduce the related parameters of this algorithm, and then describe the algorithm in detail. At last, we will give the analysis and the theoretical proof for this algorithm.

A. Related parameters

1) Vdeadline

Vdeadline (virtual deadline) represents the original deadline for the task. When $J_{i,j}$ is started to run, we set the task's Vdeadline to the server's d_k. While a server's deadline may be extended upon expiration, Vdeadline remains unchanged until the job completes. Earliest Virtual Deadline First (EVDF) is used to select server to get the slack, and orders servers by Vdeadline. In EVDF, the earlier the Vdeadline, the higher priority the server will get.

2) global_slack

When no server is selected by EVDF, which means that system is in idle state. We identify the residual budget as global_slack which will be given to the next earliest running server. If the idle time interval of the system is greater than global_slack, global_slack is set to 0, otherwise, global_slack subtracts the interval.

B. HBASH algorithm

1) Global Algorithm

The global algorithm is described as follows:

1. while the system is running,

2. when a new task arrives , a new server is created for the task and its parameters (B, P) are initialized as described in section 2.2 . At the beginning, the server state is set to **idle**, $d_0 = 0$, q = B, and q is decreased while the served task is running.

3. When a new job arrives at time t, we insert the job into the server's waiting queue.

If the server is idle

(a) if $t > d_k - q*P/B$, then recharge the server. ($q= B ; d_k= \max(t , d_{k-1})+P$;)

(b) otherwise, use the remaining time. q and d_k remain unchanged.

Vdeadline is set to the deadline d_k of the server. The server is set to **ready**

4. The ready server with the earliest deadline becomes **executing**. If there is no ready or executing server, the system becomes idle. Otherwise, if there is global slack time, we allocate it to the server (q = q + global_slack; global_slack =0;)

5. An executing server will not stop executing its pending jobs on the CPU until it has finished its jobs or

consumed its budget and decreases its budget q by the actual amount of CPU consumed.

(a) If it has consumed its budget, the server is recharged with full budget q = B, and its deadline is incremented $d_{k+1} = d_k + P$

(b) If it has no pending task,

i if vdeadline <d_k, the server is set to **idle.** Goto step 4

ii otherwise, goto **Slack Reclaiming algorithm** , donates any remaining budget q to the task of another server with the earliest virtual deadline (instead of earliest deadline). Vdeadline is set to d_k + P, and the server is set to **idle**. Goto step 4.

6. end of while

2) Slack Reclaiming algorithm

The variable slacktime describes the extra budget generated in running process, and slack reclaiming algorithm is as follows:

```
begin
1. if slacktime is greater than 0 , select a server $S_i$ in EVDF.
2. if no server $S_i$ is selected, we add the slacktime to global_slack (global_slack = slacktime; slacktime = 0;).
3. If $S_i$ is in ready state, we allocate slacktime to it, and schedule it immediately. Otherwise goto step 5.
4. $S_i$ does not stop executing its pending job on the CPU until it has finished its job or consumed the slacktime and decrease slacktime by the actual amount of CPU consumed.
    (a) if $S_i$ have completed the job , goto step 1.
    (b) if slacktime is equal to 0, $S_i$ stops running and is set to ready . Return to global algorithm
5. if $S_i$ is in idle state and q <B and q!=0, slacktime is used to make up the server's budget time until q to the max budget B, goto step 1
end
```

3) Algorithm Analysis

In this algorithm, hard real-time task, as its execution time is no more than its budget, won't postpone deadline. Moreover, it maybe yield slack. Soft real-time task, as its execution time may be more than its budget, may delay deadline, i.e. Vdeadline is less than deadline.

In terms of the EVDF, slack may be allocated to three kinds of server as follows:

(1) The server whose Vdeadline is less than deadline. According to the above analysis, we can see that the server serves a soft real-time task. If this task can be completed within the slack time, which can avoid task delaying, and improve its quality of service. Otherwise, the task demands less running time than before, hence it can be completed as soon as possible.

(2) The server whose Vdeadline is equal to deadline, and the server is idle. According to the above analysis, we can see that the server serves a soft real-time task too. Then the slack will be used to make up the budget of the server until server's budget reaches maximum budget B, which can decrease the possibility of task's postponing deadline.

(3) The server whose Vdeadline is equal to deadline, and the server is ready. Now it allocates slack to the earliest deadline task the same as BASH algorithm, and it doesn't influence other tasks.

From the analysis, we know that the algorithm doesn't allocate slack to the task which has earliest deadline, but to the task which have more need of the slack, i.e. the task which should be completed earlier. In addition, the task which gets the slack will be scheduled immediately. As the two advantages, the algorithm can improve the soft real-time performance.

4) Theoretical Validation

In this section, we analyze the schedulability condition for a hybrid task set consisting of hard periodic and soft tasks.

Each task is scheduled using a dedicated server. If each hard periodic task is scheduled by a server with maximum budget equal to the task WCET and with period equal to the task period , it behaves like a standard hard task scheduled by EDF. The difference is that each task can gain and use extra budget and yield its residual budget to other tasks. The new algorithm HBASH is able to improve the average responsiveness of soft tasks by performing slack reclaiming .The runtime exchange performed by HBASH, however, does not affect schedulability. The periodic task set can be guaranteed using the classical Liu C L and Layland J W condition[2]: $\sum_{i=1}^{n} u_i \le 1$. Each server is similar to a special periodic task in system, and is scheduled by EDF. The total bandwidth of all servers is no more than 1. i.e. $\sum_{i=1}^{n} B_i / P_i \le 1$. (B is the maximum server budget and P is the server period). So it satisfies the schedulability condition, and the slack reclaiming algorithm does not affect the schedulability.

A. A case for study

To understand the proposed approach better, we will describe a simple example which shows how our reclaiming algorithm works, and compare new algorithm with BASH algorithm which is another reclaiming algorithm.

Consider a task set consisting of three periodic tasks, taskA, taskB, taskC, and their tasks parameters and server parameters are given in the Table 1. Cavg is the average execution time of the task. We have implements HBASH in RTSIM[11]. Fig.1 and Fig.2 are the running charts of the two algorithms in RTSIM . All tasks running for 100,000 time units, and they release synchronously. The light marks the deadline miss.

From the running charts, it can be seen that the number of deadline miss of the new algorithm HBASH is less than BASH, and the finish time of every instance is no later than BASH. The average response time of HBAS

TABLE I.
TASK SET AND SERVERS PARAMETERS

Task	Cavg	WCET	P	Server	B	P
TaskA	2	3	8	Server1	2	8
TaskB	2	3	9	Server2	3	9
TaskC	5	5	12	Server3	5	12

is also shorter than BASH (HBASH is 10.0909, BASH is 11.3636, reduced by about 12%)

We take the first instance of the taskA as an example, other instances run similarly. At the beginning, taskA（q=2, d_0=8, Vdeadline=d_0=8）, taskB (q =3, d_0=9, Vdeadline=d_0=9) ,and taskC(q = 5, d_0 = 12, Vdeadline=d_0=12).

HBASH: At time t=0, taskA is scheduled with earliest deadline. When taskA consumes the residual budget at time t = 2, it is recharged with full budget (q=2, d_1= d_0+8=16). Now execute taskB, and taskB is completed at time t = 4 with one extra time unit. In principle of EVDF, the one time unit is given to the earliest virtual deadline taskA. TaskA runs at time t = 4, and finishes the job at time t = 5. At last, execute taskC, and it finishes at time t = 10.

BASH: At time t=0, taskA is scheduled with earliest deadline. When taskA consumes the residual budget at time t = 2, it is recharged with full budget (q =2, d_1 =d_0+8=16). Now execute taskB, and taskB completes at time t=4 with one extra time unit. In principle of EDF, the one time unit will be given to taskC with earliest deadline. TaskC runs at time t = 4, and finishes the job at time t = 9. Then it will give the extra one time unit to

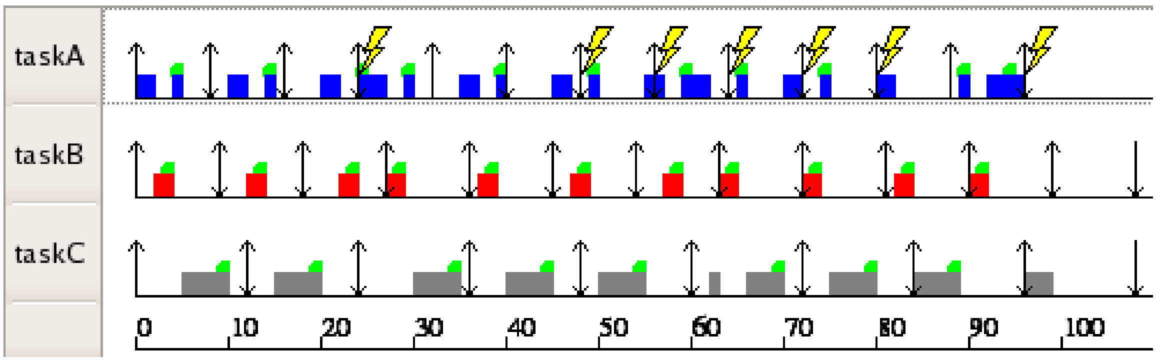

Figure 1. HBASH's running chart

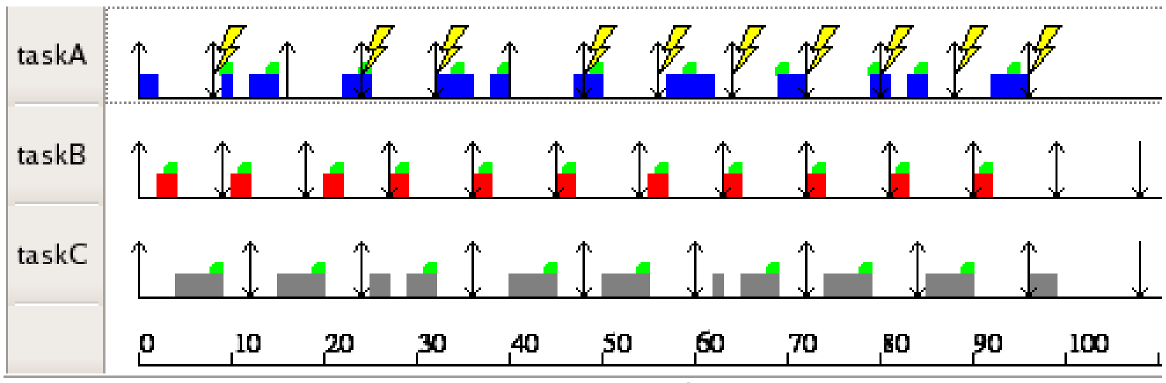

Figure 2. BASH's running chart

taskA. TaskA finishes at time t = 10. But taskA could have completed earlier.

IV. PERFORMANCE EVALUATION

The HBASH algorithm has been implemented in the real-time simulator RTSIM to measure the performance. In this section, we present the experimental results of the simulations that have been conducted: In particular, HBASH has been compared with the BASH and CBS algorithms. We have done two sets of experiment to investigate the effect of load and period on the performance of soft-time tasks. The first set shows the performance as varying the load of the soft real-time server, and the second shows the performance as the period of the soft real-time server. The performance of the algorithms was measured by computing the soft task's average response time.

A. Task's execution time

In all of our experiments, we apply the approach in paper [10]. The actual execution time c of a task is a random value drawn from the following distributions:

$$NW(\mu)=\begin{cases}\frac{1}{\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\delta^2}}, 0<x\le\mu \\ 0, x\le 0 \,\|\, x>\mu\end{cases} \quad (1)$$

A normal distribution (with mean μ and standard deviation $\sigma = 0.1\mu$) except for the values that are non-positive or greater than˙ . Random values drawn from this distribution could simulate hard real-time task's execution time, and task's WCET is˙ .

$$NA(\mu)=\begin{cases}\frac{1}{\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\delta^2}}, 0<x \\ 0, x\le 0\end{cases} \quad (2)$$

A normal distribution (with mean μ and standard deviation $\sigma = 0.1\mu$) except for the values that are non-positive. Random values drawn from this distribution could simulate soft real-time task's execution time, and task's mean execution time is˙ .

B. Task's response time

A task's response time has been normalized with respect to the average response time of all the jobs.

$$RT_i=\frac{\sum_{j=1}^{n}(f_j-r_j)}{n} \quad (3)$$

(f_j is the finish time of job j, and r_j is the release time of the jth job)

Since we are focusing on the performance of real-time applications in a mixed environment, we reserve a minimum of 2% of the CPU for best-effort tasks.

C. Experiments and Results

1) First experiment

The first experiment shows soft real-time performance as a function of soft-time task's load. The workload consists of 1 periodic soft real-time task and 5 hard real-time tasks. The soft real-time task's parameters are given in Table 2. We change the two parameters U and $\overline{c}$, while keeping the other parameters fixed. Periods of hard tasks are chosen to be uniformly distributed in the interval [100,300], while their computation times are randomly generated such that their total utilization is equal to 0.98-Us. Each point in Fig.3 has been computed over 50 runs, and has duration of 100,000 units of time. All the tasks release synchronously.

In each discrete point, HBASH outperforms CBS, but BASH is similar to CBS. It demonstrates that HBASH performs better than BASH. If the load of the soft real-time is higher, the load of the hard real-time will be lower, so that the amount of available slack for reclaiming will be less. From the Fig3, we know that HBASH still perform better than other algorithms in the case of few slack .Because it not only reclaim the slack fully, but also schedule the task which gets the slack in advance.

2) Second experiment

The second experiment shows soft real-time performance as a function of soft-time task's period. The workload consists of one periodic soft real-time task and5

TABLE II.
TASK AND SERVER'S PARAMETERS

Task	Task parameters		Server parameters			Parameter adjustment	
	f($\overline{c}$)	p	B = $\overline{c}$	P = p	U=B/P	△($\overline{c}$)	△(U)
SRT1	NA(20)	200	20	200	10%	+4	+2%

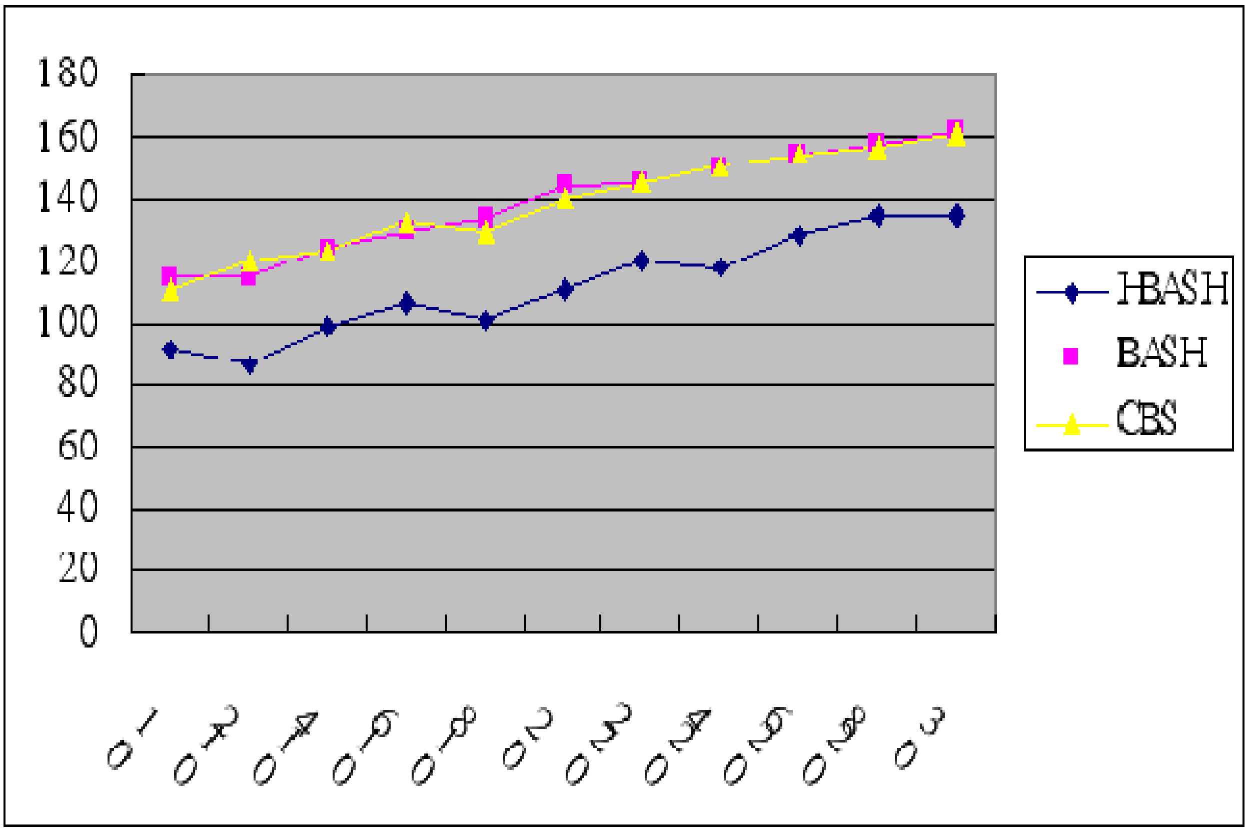

Figure 3. Average response time as load

hard real-time tasks. The soft real-time task's parameters are given in Table 3. We change the two parameters p and $\bar{c}$, while keeping the other parameters fixed. Periods of hard tasks are chosen to be uniformly distributed in the interval [200,600], while their computation times are randomly generated such that their total utilization is equal to 0.98-U. Each point in Fig.4 has been computed over 50 runs, and has duration of 100,000 units of time. All the tasks release synchronously.

From Fig.4, we can see that HBASH is similar to CBS at first, and when the period is longer, HBASH outperforms CBS better. But BASH performs no better than CBS. In terms of EDF, the longer the soft real-time tasks' period, the lower task’s priority, so that tasks' response time is longer too. From Fig.4, we know that HBASH still outperforms other algorithms, and as period is longer, it performs better.

V. Conclusion

This paper presents a new slack reclaiming algorithm for server-based real-time systems. Not only can it allocate slack to task more reasonably, but also the task will be scheduled immediately using greedy method, which reduces the response time of soft tasks by great

TABLE III.
Task and Server‘s parameters

Task	Task Parameters		Server Parameters			Parameter Adjustment	
	f($\bar{c}$)	p	B = $\bar{c}$	P = p	U=B/P	△($\bar{c}$)	△(p)
SRT1	NA(50)	100	50	100	50%	+20	+40

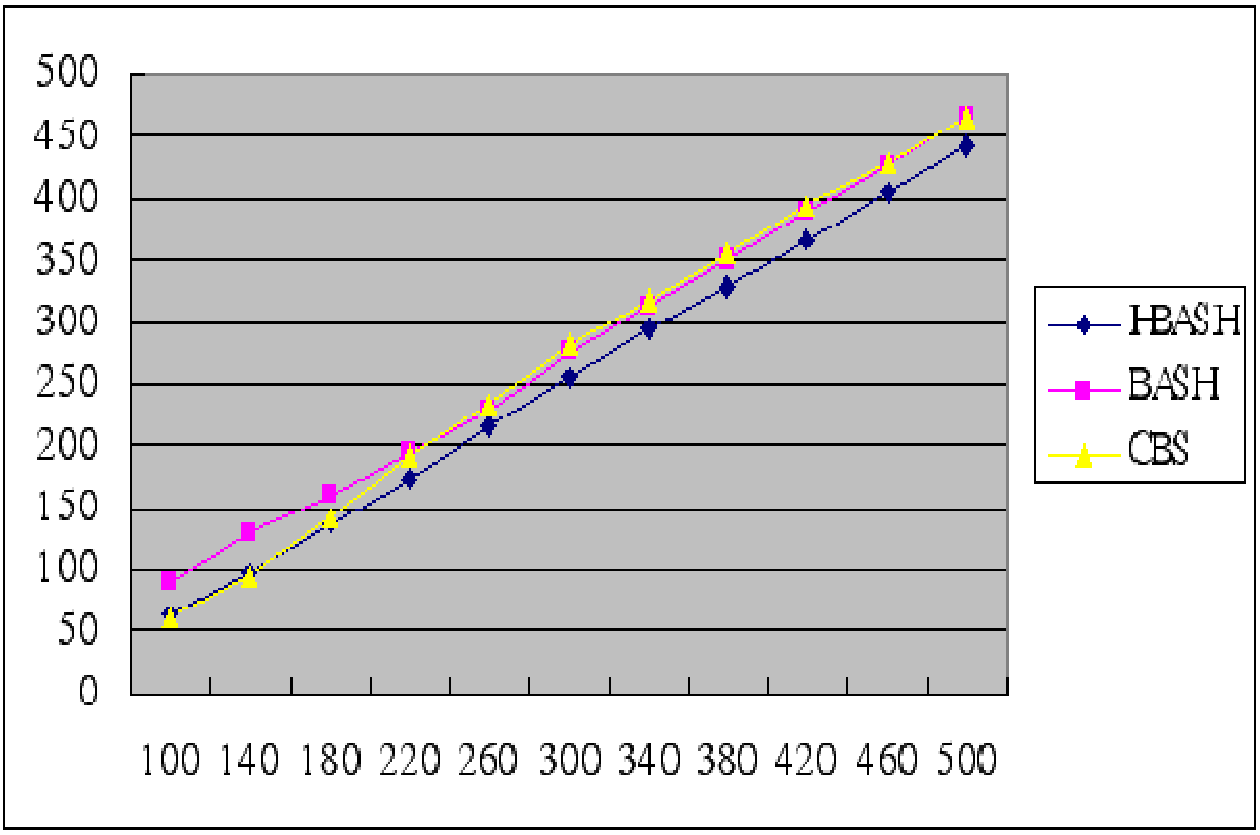

Figure 4. Average response time as period

extent. In order to evaluate the algorithm, we have implemented it in RTSIM (a real-time simulator). Experimental results show that the algorithm performs better than other algorithm, and can improve the performance of soft real-time task.

Resource sharing has not been taken into consideration in this algorithm. But in real system, sharing resources is of great importance. In future, we will add resource constraints to the algorithm, and apply it in a more complex environment, such as an open system environment.

REFERENCES

[1] J.Regehr , J.A.Stankovic. HLS: A framework for composing soft real-time schedulers. In proceeding of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001), 3-14, Dec. 2001

[2] Liu C L, Layland J W. Scheduling Algorithms for Multi-Programming in a Hard-Real-Time Environment. Journal ACM,20(1):46~63,1973

[3] Z.Deng J. W.-S. Liu J.Sun. A scheme for scheduling hard real-time applications in open system environment. Proc. Of 9th Euromicro Workshop on Real-Time Systems. Toledo, Spain: IEEE Computer Society, 1997:191-199

[4] M Spuri, GC Buttazzo. Efficient aperiodic service under the earliest deadline scheduling. Proc. Of IEEE Real-time Systems Symposium. San Juan, Puerto Rico: IEEE Computer Society, 1994:2-11

[5] L.ABni,G.Buttazzo. Integrating multimedia applications in hard real-time systems. In Proceedings of the 19th IEEE Real-Time Systems Symposium(RTSS 2004),Dec.2004

[6] M.Caccamo, G.Buttazzo, L.Sha. Capacity sharing for overrun control. In Proceedings of the 21th IEEE Real-Time Systems Symposium(RTSS 2000),295-304,Dec.2000

[7] M.Caccamo, G.Buttazzo, D.C.Thomas. Efficient reclaiming in reservation-based real-time systems with variable execution times. IEEE Transactions on Computers,54(2):198-213,Feb.2005

[8] G.Lipari and S. Baruah. Greedy reclaimation of unused bandwidth in constant-bandwidth servers. In Proceedings of the 12th Euromicro Conference on Real-Time Systems,193-200,June 2000

[9] L.Marzario, G.Lipari, P.Balbastre, A.Crespo. IRIS: A new reclaiming algorithm for server-based real-time systems. In 10th IEEE Real-time and Embedded Technology and applications Symposium(RTAS04), May 2004

[10] Caixue Lin, Scott A. Brandt: Improving Soft Real-Time Performance through Better Slack Reclaiming. RTSS 2005: 410-421

[11] http://rtsim.sssup.it/

[12] S.A.Brandt, S.Banachowski, C.Lin, and T.Bisson. Dynamic integrated scheduling of hard real-time, soft real-time and non-real-time processes. In Proceedings of the 24th IEEE Real-Time Systems Symposium(RTSS 2003):396-407

[13] Amotz Bar-Noy, Sudipto Guha, Yoav Katz, Joseph Naor, Baruch Schieber and Hadas Shachnai. Throughput maximization of real-time scheduling with batching. ACM Transactions on Algorithms(TALG), 5(2),2009

[14] Rajeev Alur and Gera Weiss. RTComposer: a framework for real-time components with scheduling interfaces. In Proceedings of the 8th ACM international conference on Embedded software(2008):159-168.

[15] Rodolfo Pellizzoni and Marco Caccamo. M-CASH: A real-time resource reclaiming algorithm for multiprocessor platforms. Real-Time Systems ,Springer Netherlands, 40(1), 2008:117-147.

WenZhi Chen was born in 1969. He got his PhD's degree of computer science from College of Computer Science and Technology, Zhejiang University, Hangzhou City, China.

He is a associate professor in College of Computer Science and Technology, Zhejiang University. His current research interests include: embedded real-time systems, distributed computing and virtualization technology.

Qingsong Shi He works as a associate professor in College of Computer Science and technology, Zhejiang University. His current research interests include: embedded systems and distributed computing.

Weifang Hu is a graduate student in College of Computer Science and technology, Zhejiang University. Her current research interest is embedded real-time systems.

Wei Hu is a post- PH.D. in College of Computer Science and technology, Zhejiang University. His current research interest is embedded real-time systems.

Sha Liu is a graduate student in College of Computer Science and technology, Zhejiang University. His current research interest is embedded real-time systems.