Framework for Real Time Heterogeneous Multiprocessor System using DYTAS Algorithm

Joel Josephson P[1], Vaibhav Meshram[2]

[1]Assistant Professor, Dept of ECE, Narsimha Reddy Engineering College, Maisammaguda(V), Dhulapally, Kompally, Medchal Road, Secunderabad – 500100 Telangana, India
[2]Professor, Department of Electronics and Communication Engineering, Narsimha Reddy Engineering College, Maisammaguda(V), Dhulapally, Kompally, Medchal Road, Secunderabad – 500100 Telangana, India

---***---

ABSTRACT *- Task Scheduling for real time heterogeneous multi-processor system is one of the most important challenges in embedded real time systems. Scheduling and resource allocation problems require numerous constraints and objectives. The scheduling of tasks in real time heterogeneous multiprocessor environment is still a demanding problem. The objective of this paper is to design a Framework and to demonstrate the scheduling process in real time heterogeneous Multiprocessor systems. This paper gives a clear explanation of the scheduling mechanism in Multiprocessor environment, in particular the paper demonstrates the allocation of tasks by the scheduler and execution of the tasks by the processors. The main idea in developing the framework here is to illustrate the scheduling algorithm with three different applications. The Framework is developed in Microsoft visual studio, tested and verified.*

Keywords: Multiprocessor, Framework, Heterogeneous systems, Scheduling, Real time tasks, Directed Acyclic Graph(DAG).

1. Introduction

In a real-time computer system, the timeliness of the availability of results is also a constraint in addition to their correctness. Multiprocessors have become powerful computing means for running real-time applications and their high performance depends greatly on parallel and distributed network environment system.

Consequently, several methods have been developed to optimally tackle the multiprocessor task scheduling problem which is called NP hard problem. Some examples of real time systems are real time data base systems, process control systems, flight control systems and web browsers (audio and video). As current real time systems become more and more complex, there is an increasing need for them to be implemented on multiple processor platforms.

Real time scheduling on a uniprocessor and Multiprocessor has been an extensively studied problem in the literature. [Anderson, J.H, Srinivasan, A, 2000 Bud, V, Devi, U et al, 2005] discussed Early release Fair scheduling. [Anderson, B,Bletsas K, 2008] explained Multi-processor scheduling for sporadic jobs tasks with pre-emption[Anderson, B., Tovar, E, 2006]emphasized static priority scheduling[Baruah, S, Jonsson, J, 2001]Pre-emptive scheduling and resource allocation[B, Bletsas, K, Baruah, S, 2008] discussed Aperiodic task scheduling and deadline sporadic tasks [B. Sprunt, L. Sha and J. Lehoczky 1989]worked with fixed priority and edf scheduling for hard real time. [Baker, T.P., 2005] analysed a condition for the feasibility of sets of sporadic hard-deadline tasks [Baker, T.P., Cirinei, M., 2006] has done with a scheduling analysis [Baker, T.P., Cirinei, M., Bertogna, M., 2008] and [Baruah, Bertogna et al 2009] explains the necessity of scheduling by resource allocation with the help of EDF Scheduling and[Bertogna Cirenei 2007]response time time and improved schedulability analysis. [Cheng, Cho Stoyenko et al 2006] gives a clear idea on multi-processor scheduling with different scheduling algorithms. [Davis, Easwaran et al 2006] outlines the importance of cluster-based multi-processor scheduling. [Funaoka et al 2008] explains about dynamic scheduling of multiprocessor systems. [G. Fohler and K. Ramamritham 1997] explained Static scheduling of pipelined periodic tasks in distributed real-time systems. [G. J. Joyce Mary and Dr.D.I.Amalarethinam 2011] explained Directed acyclic graph based Task scheduling algorithm. [H. Aydin et al 1999] Integrated scheduling of multimedia and hard real-time tasks. [H. Chetto 1990] explains Dynamic scheduling of real-time tasks under precedence constraints and [I. Bate and A. Burns 1999] explains task attributes of uniprocessor systems. [Kopetz, H.et al, 2008] discussed non-pre-emptive periodic and sporadic tasks scheduling and online multiprocessor scheduling. [K. Jeffay, D. Stanat et al 2010] outlined Schedulability analysis on multiprocessor platforms. [Levin, G., Funk, S., Sadowski, C., Pye, I., Brandt, S., 2010]has given a clear idea on Scheduling algorithms for multi-programming in a hard real-time environment. [M. Spuri and G. Buttazzo 1994] explained distributed systems for the hard real time environment. [Stavrinides, G.L., Karatza, H.D., 2011] discussed Heterogeneous distributed real time systems.

The rest of the paper is organized as follows: The importance of Scheduler is presented in section 2, The Framework of uniprocessor Scheduling is discussed in section 3, the Directed Acyclic graph(DAG) and its implementation is given in section 4, the Dynamic Task Scheduling(DYTAS) Algorithm, the Algorithm which the scheduler considers to schedule the tasks is explained in section 5, The Framework model developed for multiprocessor scheduling is presented in section 6, the simulated results of the framework and the output graphs are shown in section 7, Conclusion of this paper is given in section 8.

2. The Scheduler for Heterogeneous System

The scheduler is the core part of the operating systems, which orders the assignment of CPU and the resources to the tasks in a multitasking environment. The function of the scheduling algorithm is to determine, a task set, for a sequence of task step executions (a schedule). The tasks can be classified according to their arrival style periodic tasks and aperiodic tasks. The task set can be composed of a set of independent tasks if their executions are not synchronized, or the task can be composed of a set of dependent tasks, if their executions are synchronized. If a task set can be scheduled to meet given pre-conditions, the task set is termed as a feasible one. A typical pre-condition for hard real-time periodic processes is that they must always meet their deadlines. An optimal scheduler is able to produce a feasible schedule for all feasible task sets confirming to a given pre-condition. For a particular task set an optimal schedule is the.

The purpose of task scheduling is to organize the set of tasks ready for execution by the processor system to meet the performance objectives. The order of these tasks is called a 'schedule'. For real-time embedded systems, the primary objective is to ensure that all tasks meet their deadlines. A schedule can be feasible or optimal: a feasible schedule orders tasks making them to meet all their deadlines; an optimal schedule is one which ensures that failures to meet task deadlines are minimized. The scheduler is responsible for coordinating the execution of several tasks on a processor. The scheduler may be pre-emptive or non-pre-emptive. The scheduler for hard real-time systems must coordinate resources to meet the timing constraints of the physical system which implies that the scheduler must be able to predict the execution behaviour of all tasks within the system. So the basic requirement of real-time systems is predictability. Unless the behaviour of a real-time system is predictable, the scheduler cannot guarantees that the computation deadlines of the system will be met.

3. The Uni-processor scheduling

The basic idea of developing a Framework is to make students familiarize the scheduling process done by the different scheduling algorithms. In other words this Framework can be termed as a Teaching tool. The simulator is GUI based and acts as user interface.

The Framework developed by researchers till now has involved the scheduling process with a single processor. The Framework has a window in which it takes the burst time, period and priority for the tasks as the inputs from the user, and a push down menu is given in the window which scrolls down the names of scheduling algorithms like Round robin, first come first served (FCFS) and shortest Job first (SJF). The user can select the algorithm and click the run button. Then the scheduling process starts by considering the different parameters given by the user with the help of the software written for each algorithm.

Then during the process of execution the output window will pop-up showing the results in the execution in the form of a gnat chart.

The execution of tasks in the above explained framework is completely executed in uniprocessor environment. And this type of scheduling can be considered for the tasks which are independent. If dependent tasks are being considered i.e the tasks in the form of a cluster the tasks cannot be executed with a single processor which takes much time to execute all the tasks with precedence and successor tasks. For this reason the multiprocessor scheduling is being considered for the tasks which are in a cluster.

4. The Real time Heterogeneous Multiprocessor system:

4.1. Heterogeneous Systems:

One fundamental source of heterogeneity is the composition of subsystems with different execution and interaction semantics. At one extreme of the semantic spectrum are fully synchronized components, which proceed in lockstep with a global clock and interact in atomic transactions. Such a tight coupling of components is the standard model for most synthesizable hardware and for synchronous real-time software. At the other extreme are completely asynchronous components, which proceed at independent speeds and interact non-atomically. Such a loose coupling of components is the standard model for most multithreaded software. Between the two extremes, a variety of intermediate and hybrid models exist (e.g., globally-asynchronous locally-synchronous models).

Another fundamental source of heterogeneity is the use of models that represent a system at varying degrees of detail and are related to each other in an abstraction (or equivalently, refinement) hierarchy. A key abstraction in system design is the one relating application software to its implementation on a given platform. Application software is largely untimed, in the sense that it abstracts away from physical time. The application code running on a given platform, however, is a dynamic system that can be modelled as a timed or hybrid automaton. The run-time state includes not only the variables of the application software, but also all variables that are needed to characterize its dynamic behaviour, such as time variables and other quantities used to model resources. We need tractable theories to relate component-based models at application and implementation levels. In particular, such theories must provide means for preserving the data in the implementation, all essential properties of the application software.

4.2 Multi-Processor Environment for Embedded Systems:

In an embedded system the multi-processors are necessary for executing individual tasks to develop an application. The type of programmable multi-processor platform is representative for signal-processing architectures where low power, and support for many features and standards is imperative.

4.3 Task Scheduling in Multi-Processor Environment:

In a Multi-Processor environment a scheduler plays a major role of assigning a task to the processor. In other words scheduler synchronizes the execution of different tasks with the different processors.

4.4 Developing a simulator for task scheduling in Multi - Processor Environment:

Simulators try to model the behaviour of the complete microcontroller in software. Some simulators go even a step further and include the whole system (simulation of peripherals outside of the microcontroller). Simulating external events can become a time-consuming exercise, as you have to manually create "stimulus" files that tell the simulator what external waveforms to expect on which microcontroller pin.

4.5 The Multiprocessor Model

In Directed Acyclic Graph (DAG) scheduling, the target system is assumed to be a network of processing elements (PEs), each of which is composed of a processor and a local memory unit so that the PEs do not share memory and communication relies solely on message-passing. The processors may be heterogeneous or homogeneous. Heterogeneity of processors means the processors having different speeds or processing capabilities. However, we assume every module of a parallel program can be executed on any processor even though the completion times on different processors may be different. The PEs are connected by an interconnection network with a certain topology. The topology may be fully connected or of a particular structure such as a hypercube or mesh.

5. The Dynamic Task scheduling algorithm (DYTAS):

We propose a new dynamic scheduling algorithm based on above scheduler model, DYnamic TAsk Scheduling (DYTAS) algorithm. This strategy makes the whole parallel job finished at the possible earliest time viz. the response time of this parallel task is shortest. According to DYTAS, the tasks in ITQ are scheduled by its dependency. The front task in ITQ is always first scheduled and mapped to a processor by the algorithm. While in the static scheduling algorithm, the tasks are sorted by a certain priority rank, because the data of DAG is known in advance. But, the proposed dynamic scheduling algorithm is different from the static scheduling algorithms, by migrating the task during the runtime.

The algorithm DYTAS is focusing on the processor selection strategy. It mainly lies on how to select a processor on which the tasks are mapped even though, the tasks are scheduled earlier.

When selecting a processor to do the particular task from the PTQ_i, two time-indexes must be considered:

The earliest free time of the processor P_i

The earliest start time of the task v_i on the processor P

In the proposed scheduler model, the parallel tasks in ITQ and the ready tasks are in processor's PTQ. Even though ITQ and DTQ locate at the central scheduler, the processors on which the mapped tasks are actually executed are separated from the scheduler and placed at PTQ_i"s. At the same time, the scheduling process and the executing process are parallel works. So that, the scheduler and the working processors synchronous with each other.

Procedure *DYTAS*

```
1.  dtq[ ] = SORT[Ti , Tj]
2.  l = 0;
3.  while (dtq[ ] is not empty) do
4.    for i = 1 to n
5.      ptqi = dtq [l]
6.      l = l + 1
7.    end for;
8.  end while;
9.  for each processor Pk in processor group do
10.   while (Pk is in running state)
11.       skip and select the next ptqk+1
12.   end while
13.   Pk = ptqk[j]
14.   if (dependent task of ptqk[j] is in ctq)
15.     TASK(ptqk[ ],Pk , j, ctq, cpk, CTj)
16.   else
17.   do
18.     move the pointer to the next ptq
19.      if (dependent task of ptqk[j] is in ctq)
20.       TASK(ptqk[ ],Pk , j, ctq, cpk, CTj)
21.       exit do
22.      endif
23.   while(checking with all ptq's once)
24.   endif
25. end for
26. end DYTAS
```

Procedure for TASK

```
    procedure
27. TASK(ptqk[ ],Pk,j,ctq,cpk,CTj)
28.     do Tj  with Pk
29.     remove Tj  from ptqk
30.     insert Tj  in ctq
31.     cpk  = cpk  + Ctj
32. end TASK
```

Fig 1

Fig 1 lists the algorithm DYTAS. The worst case time complexity for DYTAS is O(t^3), t is the total number of the tasks. The loop 5 – 8 distributes all the tasks in DTQ to PTQ_i „s. The middle loop 11 – 13 tours on every ptq to fetch the task to the processor. The loop 15-25 checks the availability of T_j in CTQ. If T_j is available, proceed with the same processor and if not, proceed selecting the task from the next PTQ"s till fetches the suitable task.

After fetching or assigning the task to the processor, post in the PSW and remove from the $PTQ_{i.}$ This process will repeat all the PTQi"s becomes empty. If any PTQ completes its Task set at the earliest, migrate the suitable task from the available PTQ"s. If suppose, there is no task is suitable to fetch at that cycle, processor has to wait till any one of the dependent task becomes available at CTQ. The waiting time is said to be a processors idle state. The time complexity of the whole DYTAS is O(t3). It is closely related with the length of DTQ and the total number of processors, also related with the precedence relationship among tasks in DAG.

6. The Framework model developed for multiprocessor scheduling

6.1 Need for Framework:

The necessity in designing the Framework is visualize the work carried out by the scheduler i.e the scheduling mechanism can be clearly viewed through the Framework. The block schematic of the framework is as shown in Fig 2.

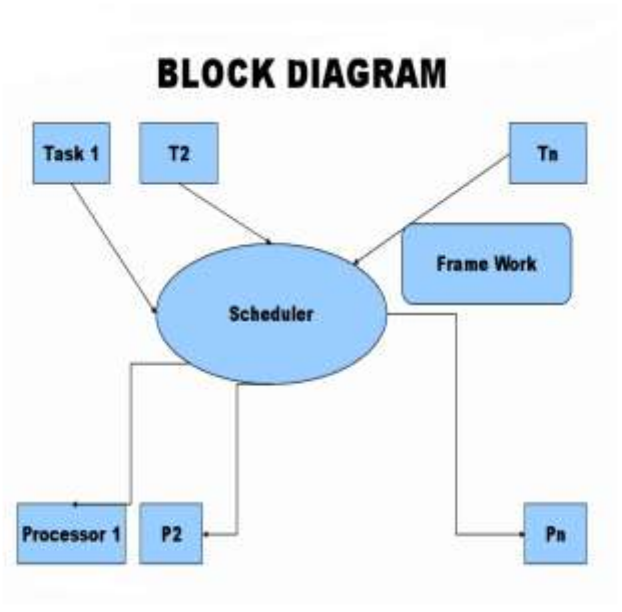
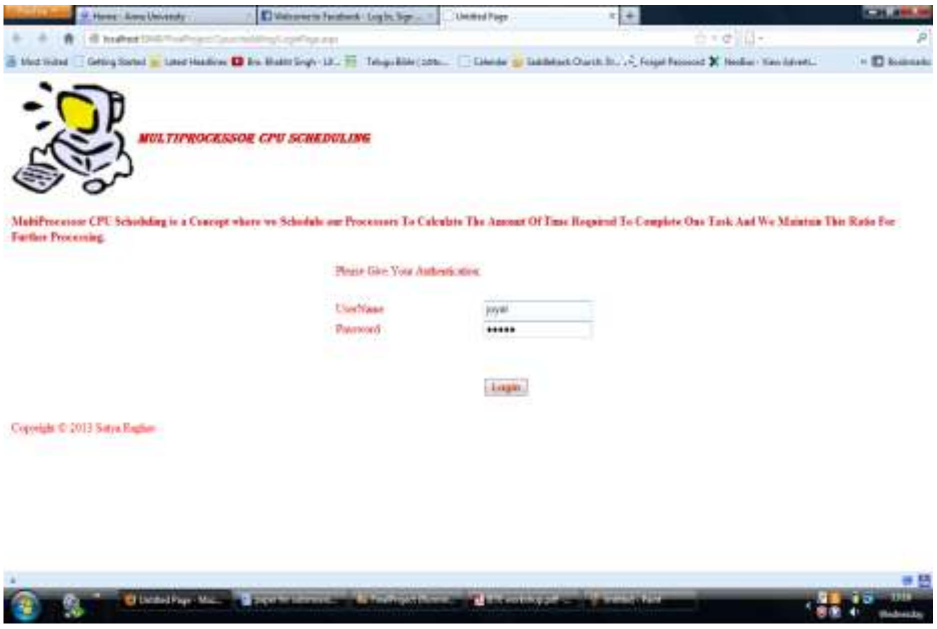


Fig 2

A Framework has been developed in Microsoft Visual Studio in which three tasks are considered. The tasks implemented here to illustrate the Multiprocessor scheduling mechanism are 1.News Channel 2. Notepad 3.Windows media player

The first task, news channel has many subtasks involved in it i.e to scroll a message while the news are displayed and updating the current news as and when there is a flash news. And the second task is the notepad task which is just used to enter some information on to it and save it. And the final task is the windows media player which runs both the audio and video files which are considered to be the subtasks and dependent of each other. Finally after the tasks are simulated the framework shows the results on the processor time spent on each task after execution. The task considers the logic involved in the DYTAS Algorithm.



MULTIPROCESSOR CPU SCHEDULING

MultiProcessor CPU Scheduling is a Concept where we Schedule our Processors To Calculate The Amount Of Time Required To Complete One Task And We Maintain This Ratio For Further Processing.

Please Give Your Authentication

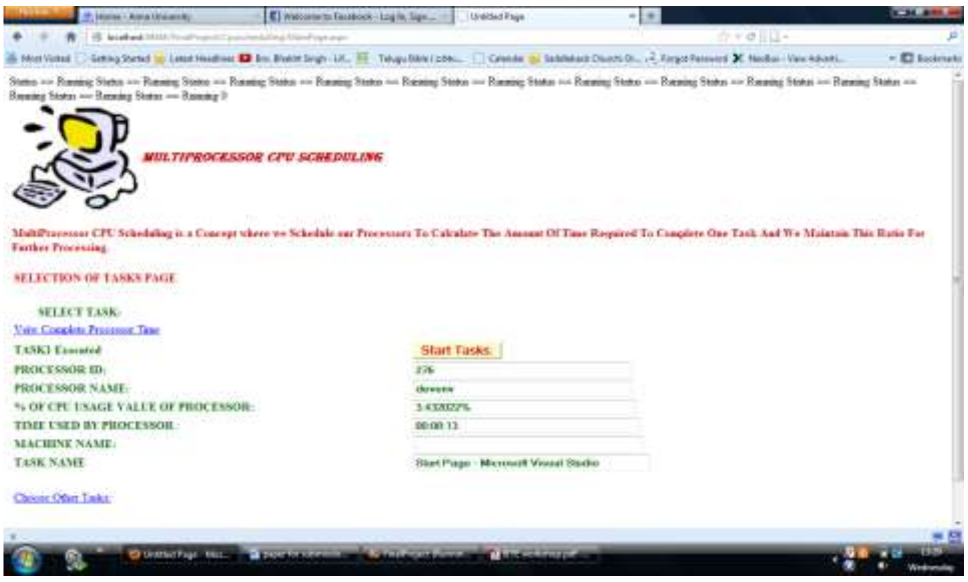
UserName	joyal
Password	•••••

Login

Copyright © 2013 Satya Raghav

Fig 3

The Framework asks the user to enter the username and password before executing the tasks which is shown in Fig 3. Then after pressing the login button, other window pop-ups asking the user to select the task and then after pressing select task the task will start executing and display the parameters of the task as shown in the following Fig 4



Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running Status == Running 3

MULTIPROCESSOR CPU SCHEDULING

MultiProcessor CPU Scheduling is a Concept where we Schedule our Processors To Calculate The Amount Of Time Required To Complete One Task And We Maintain This Ratio For Further Processing.

SELECTION OF TASKS PAGE

SELECT TASK:

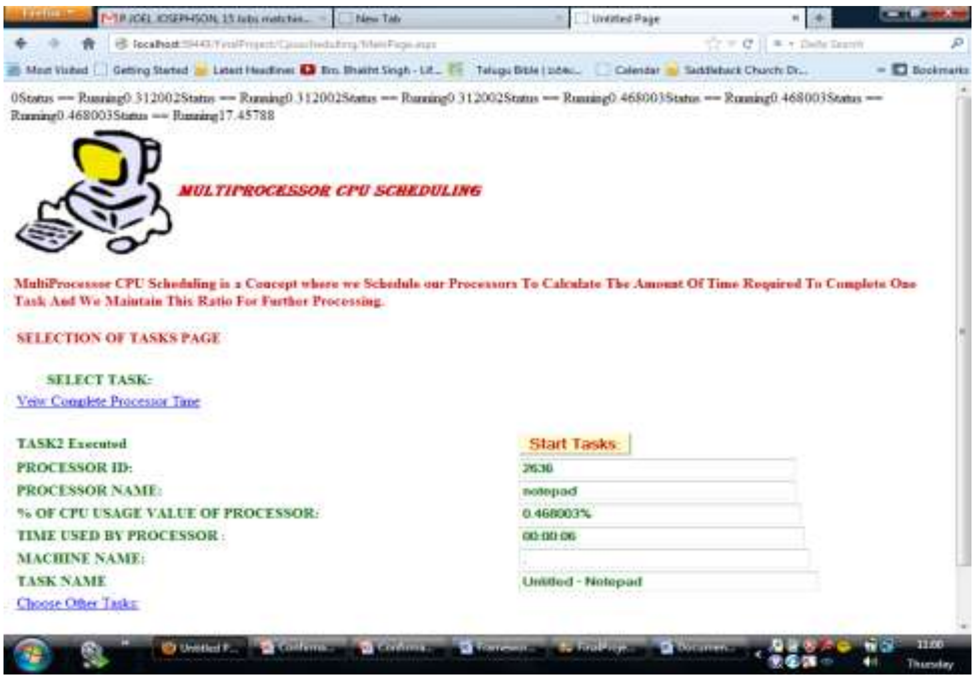
View Complete Processor Time

TASK1 Executed	Start Tasks
PROCESSOR ID:	276
PROCESSOR NAME:	devenv
% OF CPU USAGE VALUE OF PROCESSOR:	3.432022%
TIME USED BY PROCESSOR:	00:00:13
MACHINE NAME:	
TASK NAME	Start Page - Microsoft Visual Studio

Choose Other Tasks

Fig 4

Similarly the Scheduler executes the task 2 and task 3 and is displayed as shown in figures Fig 5 and Fig 6.


0Status --- Running0.312002Status --- Running0.312002Status --- Running0.312002Status --- Running0.468003Status --- Running0.468003Status --- Running0.468003Status --- Running17.45788

MULTIPROCESSOR CPU SCHEDULING

MultiProcessor CPU Scheduling is a Concept where we Schedule our Processors To Calculate The Amount Of Time Required To Complete One Task And We Maintain This Ratio For Further Processing.

SELECTION OF TASKS PAGE

SELECT TASK:

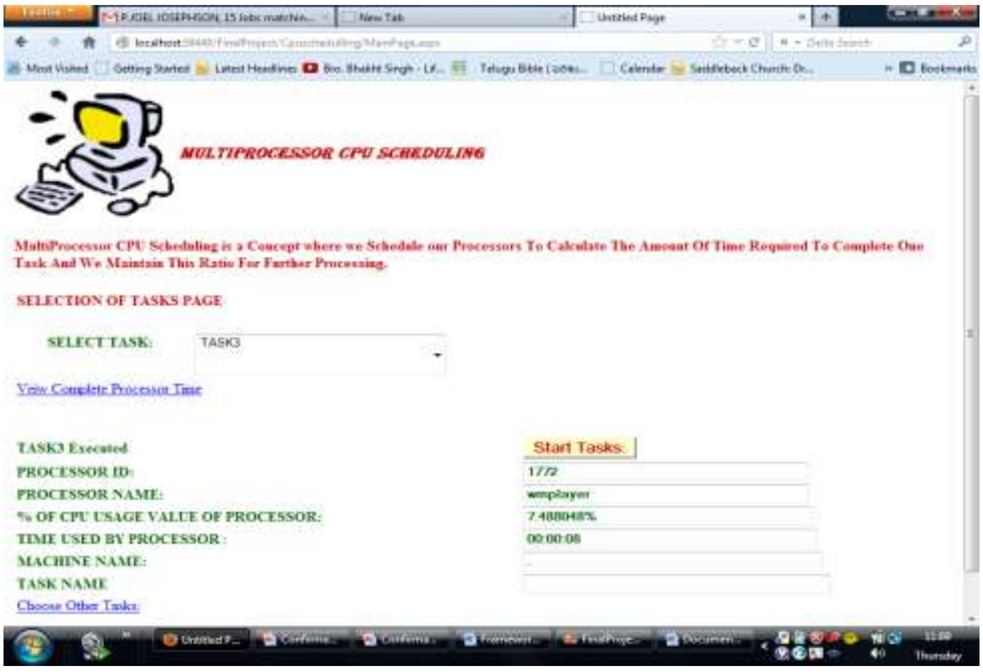
Veiw Complete Processor Time

TASK2 Executed	Start Tasks
PROCESSOR ID:	2636
PROCESSOR NAME:	notepad
% OF CPU USAGE VALUE OF PROCESSOR:	0.468003%
TIME USED BY PROCESSOR :	00:00:06
MACHINE NAME:	.
TASK NAME	Untitled - Notepad

Choose Other Tasks:

Fig 5

The figure below shows the status of the task which is in running state and the processor utilization on executing the task.


MULTIPROCESSOR CPU SCHEDULING

MultiProcessor CPU Scheduling is a Concept where we Schedule our Processors To Calculate The Amount Of Time Required To Complete One Task And We Maintain This Ratio For Further Processing.

SELECTION OF TASKS PAGE

SELECT TASK: TASK3

Veiw Complete Processor Time

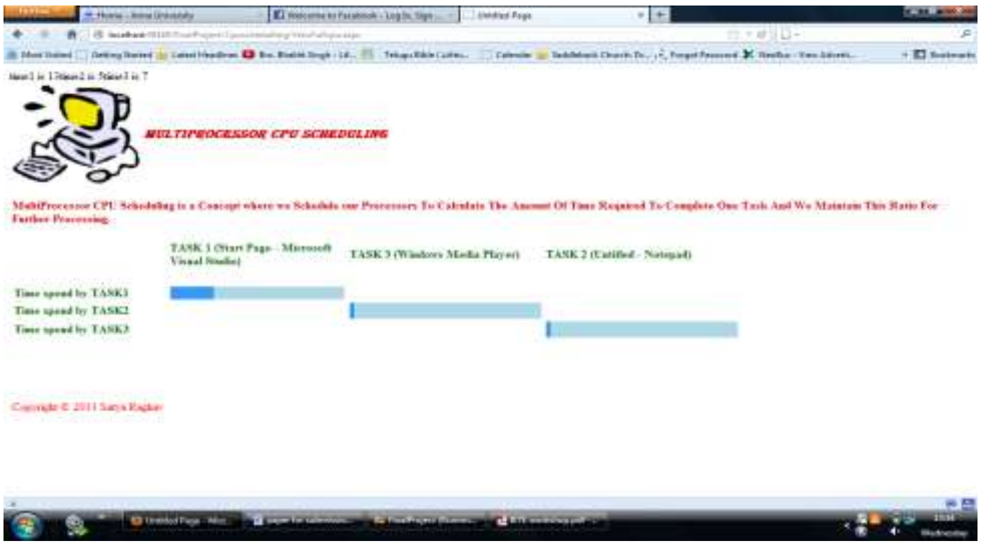
TASK3 Executed	Start Tasks
PROCESSOR ID:	1772
PROCESSOR NAME:	wmplayer
% OF CPU USAGE VALUE OF PROCESSOR:	7.488048%
TIME USED BY PROCESSOR :	00:00:08
MACHINE NAME:	.
TASK NAME	

Choose Other Tasks:

Fig 6

7. The simulated results of the Framework

A chart is being displayed which shows the processors utilization in executing the tasks which can be observed in the figure shown below. Fig 7



time1 is 17time2 is 5time3 is 7

MULTIPROCESSOR CPU SCHEDULING

MultiProcessor CPU Scheduling is a Concept where we Schedule our Processors To Calculate The Amount Of Time Required To Complete One Task And We Maintain This Ratio For Further Processing

TASK 1 (Start Page - Microsoft Visual Studio) TASK 3 (Windows Media Player) TASK 2 (Untitled - Notepad)

Time spend by TASK1
Time spend by TASK2
Time spend by TASK3

Copyright © 2011 Satya Raghav

Fig 7

8. Conclusion

This paper presents a Framework for Real time Heterogeneous Multiprocessor systems which illustrates the functionality of scheduling. The Framework has been tested and verified.

REFERENCES

[1] Anderson, J.H., Srinivasan, A., 2000. Early-release fair scheduling. In: *Proceedings of Euromicro Conference on Real-Time Systems*, pp. 35–43.

[2] Anderson, J.H., Bud, V., Devi, U., 2005. An EDF-based scheduling algorithm for multiprocessor soft real-time systems. In: *Proceedings of Euromicro Conference on Real-Time Systems*, pp. 199–208.

[3] Andersson, B., Bletsas, K., 2008. Sporadic multiprocessor scheduling with few preemptions. In: *Proceedings of Euromicro Conference on Real-Time Systems*, pp. 243–252.

[4] Andersson, B., Tovar, E., 2006. Multiprocessor scheduling with few preemptions. In: *Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 322–334.

[5] Andersson, B., Baruah, S., Johnsson, J., 2001. Static-priority scheduling on multiprocessors. In: *Proceedings of IEEE Real-Time Systems Symposium*, pp. 193–202.

[6] Andersson, B., Bletsas, K., Baruah, S., 2008. Scheduling arbitrary-deadline sporadic task systems on multiprocessor. In: *Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications,* pp. 197–206.

[7] B. Sprunt, L. Sha and J. Lehoczky, "Aperiodic task scheduling for hard real-time systems," *Journal of Real-Time Systems*, Vol. 1,1989, pp. 27-60.

[8] Baker, T.P., 2005. Comparison of empirical success rates of global vs. paritioned fixedpriority and edf scheduling for hand real time, Tech. Rep. *Technical Report* TR- 050601, Dept. of Computer Science, Florida State University, Tallahasee.

[9] Baker, T.P., Cirinei, M., 2006. A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks. In: *Proceedings of IEEE Real-Time Systems Symposium*, pp. 178–190.

[10] Baker, T.P., Cirinei, M., Bertogna, M., 2008. EDZL scheduling analysis. Real-Time Systems 40, 264–289.

[11]Baruah, S., Mok, A., Rosier, L., 1990. Preemptively scheduling hard-real-time sporadic tasks on one processor. In: *Proceedings of IEEE Real-Time Systems Symposium*, pp. 182–190.

[12]Baruah, S., Cohen, N.K., Plaxton, C.G., Varvel, D.A., 1996. Proportionate progress: *a notion of fairness in resource allocation. Algorithmica 15 (6)*, 600–625.

[13]Baruah, S., Bonifaci, V., Marchetti-Spaccamela, A., Stiller, S., 2009. Implementation of a speedup-optimal global EDF schedulability test. In: *Proceedings of Euromicro Conference on Real-Time Systems*, pp. 259–268.

[14]Bertogna, M., Cirinei, M., 2007. Response-time analysis for globally scheduled symmetric multiprocessor platforms. In: *Proceedings of IEEE Real-Time Systems Symposium*, pp. 149–160.

[15]Bertogna, M., Cirinei, M., Lipari, G., 2005. Improved schedulability analysis of EDF on multiprocessor platforms. In: *Proceedings of Euromicro Conference on Real- Time Systems*, pp. 209–218.

[16]Bertogna, M., Cirinei, M., Lipari, G., 2009. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. *IEEE Transactions on Parallel and Distributed Systems* 20, 553–566.

17] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, Vol. 20, No. 1, January, 1973, pp. 46-61.

[18] Cheng, B.-C., Stoyenko, A., Marlowe, T., Baruah, S., 1997. LSTF: A new scheduling policy for complex real-time tasks in multiple processor systems. *Automatica* 33 (5), 921–926.

[19] Cho, S., Lee, S.-K., Ahn, S., Lin, K.-J., 2002.Efficient real-time scheduling algorithms for multiprocessor systems. *IEICE Trans. on Communications* E85–B (12), 2859–2867.

[20] Cho, H., Ravindran, B., Jensen, E.D., 2006. An optimal real-time scheduling algorithm for multiprocessors. In: Proceedings of *IEEE Real-Time Systems Symposium*, pp. 101–110.

[21]Davis, R.I., Burns, A., 2011. FPZL schedulability analysis. In: *Proceedings of IEEE Real- Time Technology and Applications Symposium,* pp. 245–256.

[22]Easwaran, A., Shin, I., Lee, I., 2008. Towards optimal multiprocessor scheduling for arbitrary deadline tasks. In: *Proceedings of the Work-in-Progress Session of IEEE Real-Time Systems Symposium*, pp. 1–4. [23]Easwaran, A., Shin, I., Lee, I., 2009. Optimal virtual cluster-based multiprocessor scheduling. *Real-Time Systems* 43 (1), 25–59.

[24]Funaoka, K., Kato, S., Yamasaki, N., 2008. Work-conserving optimal real-time scheduling on multiprocessors. In: *Proceedings of Euromicro Conference on Real-Time Systems*, pp. 13–22.

[25] G. Manimaran and C. Siva Ram Murthy, "An efficient dynamic scheduling algorithm for multiprocessor real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 3, 1998, pp. 312-319.

[26] G. Fohler and K. Ramamritham, "Static scheduling of pipelined periodic tasks in distributed real-time systems," *Proceedings of the Ninth Euro Micro Workshop on Real-Time Systems, Toledo*, Spain, June, 1997, pp. 128-135.

[27]G. J. Joyce Mary and Dr.D.I.Amalarethinam "A new DAG based dynamic Task Scheduling Algorithm (DYTAS) for Multiprocessor Systems" International *Journal of Computer Applications*, April 2011, Vol 19, no 8 pp 24-28.

[28] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez-Mejfa, "Optimal reward based scheduling for periodic real-time tasks," *Proceedings of the IEEE Real-Time Systems Symposium*, Phoenix, Arizona, December, 1999, pp. 79-89.

[29] H. Kaneko, J. Stankovic, S. Sen, and K. Ramamritham, "Integrated scheduling of multimedia and hard real-time tasks," *Proceedings of the Seventeenth IEEE Real-Time Systems Symposium*, Washington DC, December, 1996, pp. 206-217.

[30] H. Chetto, M. Silly, and T. Bouchentouf, "Dynamic scheduling of real-time tasks under precedence constraints," *Journal of Real-Time Systems*, Vol. 2, No. 3, 1990, pp. 181-194.

[31] I. Bate and A. Burns, "An approach to task attribute assignment for uniprocessor systems," *Proceedings of the Eleventh Euro Micro Conference on Real-Time Systems*, York, England, June, 1999, pp.46-53.

[32]Kopetz, H., 2008. On the design of distributed time-triggered embedded systems. *Journal of Computing Science and Engineering* 2 (4), 340–356.

[33] K. Jeffay, D. Stanat, and C. Martel, "On non-preemptive periodic and sporadic tasks," *Proceedings of the Twelfth IEEE Real-Time Systems Symposium*, San Antonio, Texas, December, 1991, pp. 129-139.

[34]Lee, S.K., 1994.On-line multiprocessor scheduling algorithms for real-time tasks. In: *IEEE Region 10's Ninth Annual International Conference*, pp. 607–611. [35]Lee, J., [35]Easwaran, A., Shin, I., 2010. LLF schedulability analysis on multiprocessor platforms. In: *Proceedings of IEEE Real-Time Systems Symposium*, pp. 25–36.

[36]Lee, J., Easwaran, A., Shin, I., Lee, I., 2010. Multiprocessor real-time scheduling considering concurrency and urgency. *ACM SIGBED Review* 7 (1).

[37]Lee, J., Easwaran, A., Shin, I., 2011. Maximizing contention-free executions in multiprocessor scheduling. In: *Proceedings of IEEE Real-Time Technology and Applications Symposium*, pp. 235–244.

[38]Leung, J.Y.-T., 1989. A new algorithm for scheduling periodic, real-time tasks. *Algorithmica* 4, 209–219.

[39]Levin, G., Funk, S., Sadowski, C., Pye, I., Brandt, S., 2010. DP-FAIR: A simple model for understanding optimal multiprocessor scheduling. In: *Proceedings of Euromicro Conference on Real-Time Systems*, pp. 3–13.

[40]Liu, C., Layland, J., 1973. Scheduling algorithms for multi-programming in a hard realtime environment. *Journal of the ACM* 20 (1), 46–61.

[41] M. Harbour, M. Klein, and J. Lehoczky, "Timing analysis for fixed priority scheduling of hard real-time systems," *IEEE Transactions on Software Engineering*, Vol. 20, No. 1, January,1994, pp. 13-28.

[42] M. Spuri and G. Buttazzo, "Efficient aperiodic service under earliest deadline scheduling," *Proceedings of the IEEE Real-TimeSystems Symposium*, Pisa, Italy, 1994, pp. 2-11.

[43] M. Natale and J. Stankovic, "Dynamic end-to-end guarantees in distributed real-time systems," *Proceedings of the Fifteenth IEEE Real-Time Systems Symposium*, San Juan, Puerto Rico, 1994, pp. 216-227.

[44] Mok, A., 1983. "Fundamental design problems of distributed systems for the hardrealtime environment", *Ph.D. thesis*, Massachusetts Institute of Technology. [45]Park, M., Han, S., Kim, H., Cho, S., Cho, Y., 2005. Comparison of deadline-based scheduling algorithms for periodic real-time tasks on multiprocessor. *IEICE Transaction on Information and Systems* E88-D, 658–661.

[46]Srinivasan, A., Baruah, S., 2002. Deadline-based scheduling of periodic task systems on multiprocessors. *Information Processing Letters* 84 (2), 93–98.

[47]Stavrinides, G.L., Karatza, H.D., 2011. Scheduling multiple task graphs in heterogeneous distributed real-time systems by exploiting schedule holes with bin packing techniques. *Simulation Modelling Practice and Theory* 19 (1), 540–552.

[48] T. Abdelzaher and K. Shin, "Combined task and message scheduling in distributed real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 11, November 1999, pp. 1179-1191.

BIOGRAPHY

P .Joel Josephson is Assistant Professor in Electronics and Communication Engineering Department in Narasimha Reddy Engineering College. He received B.Tech.(Electronics and Communication Engineering) degree and M.Tech. (Embedded Systems) degree in 2005 and 2007 from Jawaharlal Nehru Technological University Hyderabad. He is currently pursuing Ph.D. programme at Anna University Chennai, in the area of scheduling in Multi-processor environment.

Dr. Matta J Chandra Prasad did B.E in Electronics & Communication Engineering, M.Tech in Electronics & Instrumentation Engineering and Ph.D in Electronics & Communication Engineering from reputed universities. He has total experience of 20 years .He published 15 papers in National and International journals and conferences .He is Life Member of ISTE. He guided projects of UG 32 and PG 12 and sponsored project 1(UGC) .He is having 3 Years International Experience.