A Real Time Scheduling Method for Embedded Multimedia Applications

Byoungchul Ahn* **Ji-Hoon Kim**
Department of Computer Engineering
Graduate School, Yeungnam University
Gyungsan, Korea

Dong Ha Lee **Sang Heon Lee**
Department of SW Research Team
Daegu Gyeongbuk Institute of Science & Technology
Daegu, Korea

Abstract –*For applications of embedded systems, several processors are scheduled on a processor because very limited applications are used. Recently application areas are broadened and used to be like personal computers. Operating systems are considered one of critical factors to develop embedded systems. Linux is becoming one of popular operating systems for embedded applications, because of open sources and royalty free. Since Linux is designed for general purpose applications, it is not suitable for embedded multimedia systems. For multimedia applications, it is required that the scheduler operate without jitters or jams. This paper presents a method to schedule many processes by guaranteeing quality of services for video and audio. This goal is achieved through the mixed scheduling using critical based EDF and round robin method to utilize the CPU time. Simulation results show that the proposed algorithm decreases the number of violations and schedules many processors effectively for multimedia embedded applications.*

Keywords: Real Time, Linux, Embedded Software, Scheduling.

1 Introduction

Linux operating system is implemented on different hardware platforms and is used for embedded applications. Most embedded systems are used to control or monitor simple devices and they are required to fast response to events. Nowadays embedded systems are required to operate very complex tasks and also used to be personal computers. Many current applications do not need to be fully real-time systems, since they are applied to non-time critical applications. However, it is very sensitive to time for multimedia embedded applications. When they do not deliver audio and video to users at the same time, embedded systems do not provide the quality of services because of still frames and jitters.

Although present embedded operating systems are relatively stable and suitable for special-purpose applications, those are too big to use for limited resources like embedded systems[2]. Since some operating systems which provide very good development tools and assistance are not opened to public as free of charge, Linux comes into the spotlight and is used for embedded applications[3,4,5].

Linux is a general-purpose operating system, and it uses time-sharing scheduling algorithm. Each process uses a slice of operating time. The time-sharing scheduling method on embedded Linux is not suitable to schedule several processes optimally if it schedules video data, audio data, game applications and so on. Each process of embedded systems used for special purposes, has different importance. For example, if there is a LCD graphics system which plays animation software, the animation process must have higher priority than any other processes. If the animation process is not scheduled at pre-assigned time, the system generates jitters or still frames to users.

This paper proposes a scheduling scheme which schedules a critical based EDF algorithm based upon task importance to utilize CPU time instead of time-sharing scheduling algorithm. In section 2, real time scheduling schemes are explained briefly to use for embedded multimedia applications. In section 3, the proposed algorithm is introduced. Section 4 shows the simulation results. The last section concludes with a summary.

2 Formatting instructions

Embedded systems are designed to perform specific applications. To design and implement embedded systems for multimedia applications, the scheduling algorithm is the most critical factor in operating systems. Several scheduling algorithms are discussed briefly.

2.1 Linux Scheduling Algorithm

Linux scheduling scheme is based upon the time-sharing scheduling algorithm[6]. This algorithm shares a CPU time and all processes use a quantum slice of operating time. Each process is given a time quantum slice to run. If it is not completely done by that time interval, a process is suspended and another process is continued. After all other processes have been given a quantum, the first process gets its chance again. The time sharing scheduler uses the time interrupt tick, the context switching between processes is invisible to users.

2.2 Rate Monotonic Algorithm

The rate monotonic scheduling algorithm, introduced by Liu and Layland in 1973, is a static algorithm applied in real-time systems by National Aeronautics and Space Administration and European Space Agency [7]. It assigns static priorities to tasks at the connection setup stage according to their request rates. Subsequently, each task is scheduled with the priority calculated at the beginning, with no further rearrangement of priorities required. The priority corresponds to the importance of a task relative to other tasks. The task with the shortest period gets the highest priority, and the task with the longest period gets the lowest priority. It is an optimal and static, priority-driven preemptive scheduling algorithm for preemptive, periodic tasks[8].

2.3 Least Laxity First Algorithm

Least laxity first algorithm(LLF) assigns priority bases upon the slack time of a task. The laxity time is temporal difference between the deadline, the remaining processing time and the run time. LLF always schedules first an available task with the smallest laxity. The laxity of a task indicates how much the task will be scheduled without being delayed. LLF is a dynamic scheduling algorithm and optimal to use a exclusive resource. LLF is commonly used in embedded systems. Since the run time is not defined, laxity changes continuously. The advantage of allowing high utilization is accompanied by a high computational effort at schedule time and poor overload performance.

2.4 Earliest Deadline First Algorithm

The earliest deadline first (EDF) algorithm is the best-known algorithm for real-time processing. At any arrival of a new task, EDF immediately calculates a new order. It preempts the running task and schedules a new process according to its deadline. The interrupted task is rescheduled later. EDF schedules not only periodic tasks, but also tasks with arbitrary requests, deadlines, and service execution times. However, EDF cannot guarantee its performance under overload scheduling condition[12].

EDF is an optimal and dynamic algorithm. A dynamic algorithm schedules every instance of each incoming task according to its specific demands. It may reschedule periodic tasks in each period. For a dynamic algorithm like EDF, the upper bound of processor utilization is 100 percent. If a set of tasks can be scheduled by any priority assignment, EDF is optimal scheduling algorithm. To schedule the continuous multimedia data by EDF on a single processor, task priorities are likely to be rearranged frequently. If EDF has already assigned the priority for a new task, the scheduler must rearrange the priorities of other tasks until the required priority is free. In worst case, the priorities of all tasks have to be rearranged, which may cause considerable overhead to the processor.[12].

3 Proposed Algorithm

In Section 2, four algorithms are introduced briefly and discussed their advantages and disadvantages. Those are not suitable to schedule multimedia data for embedded systems. The scheduling scheme of the proposed algorithm is based upon the CPU utilization factor. If the utilization factor is zero, CPU is idle. If the utilization factor is one, CPU is loaded in full and there is no time quantum to use other tasks.

If the utilization factor is less than the threshold value, the scheduler uses EDF algorithm. If the utilization factor is equal to or higher than the threshold value, the scheduler selects the most critical process first. The threshold value is determined by applications of embedded systems

Because EDF only considers the deadline of process and does not select critical processes, critical processes may not be scheduled at a given time. It is very important to schedule the most critical process first. And the next less critical processes or general processes must be scheduled. This method guarantees CPU time for critical processes properly. Figure 1 shows a simple flowchart of the proposed algorithm.

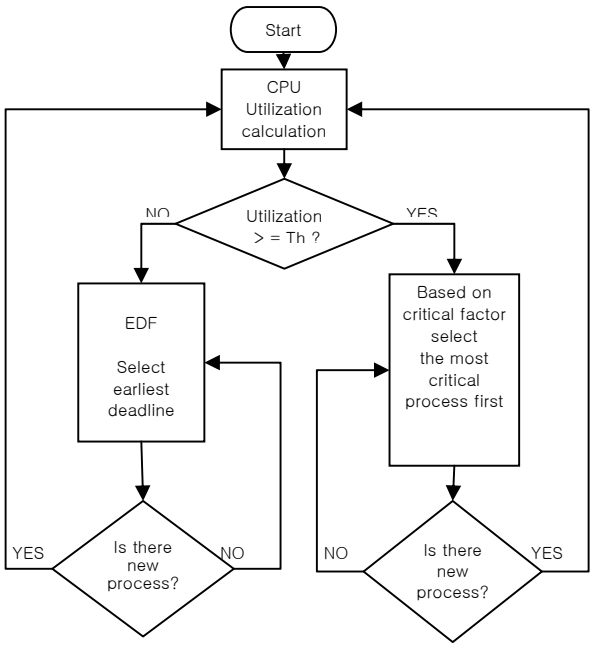

Figure 1. Proposed scheduling algorithm

4 Simulation

The proposed algorithm is simulated and verified its performance. CHEDDAR simulator is used to evaluate the performance of the algorithm[13]. For simulation, the number of deadline violation is measured while the number of processes is increased. Processes for simulation are categorized to three types, which are regular processes, audio and video processes and mixed priority processes. Also the number of context switching is measured. The threshold value is 0.9 for this experiment.

4.1 Regular Processes

Five algorithms are used to compare deadline violations. The priorities of 10 processes among 24 processes are higher than the other processes.

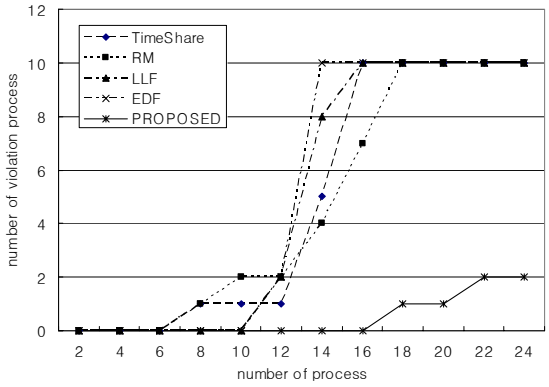

Figure 2. Scheduling comparison for regular processes

The simulation results show in Figure 2. Both RM algorithm and time sharing algorithm show that deadline violations start when the 7th process is scheduled. These two algorithms do not schedule properly although the utilization factor is smaller than 1. EDF algorithm and LLF algorithm show that deadline violations start when the 11th process is scheduled. The violation reason is that the utilization factor is 1. The proposed algorithm schedules up to the 16th process without violations. From the 17th process, it shows the deadline violation because the utilization factor is one. The violation number of the proposed algorithm is increased slowly compared to LLF and EDF algorithms when the utilization factor is 1.

Figure 3 shows the number of context switching. At the beginning stage, the difference of number of context switching is about 10% but the difference is grows up to 60% as the number of processes is increased. The proposed algorithm and EDF algorithm show very low context switching compare to the other algorithms. The reason is that proposed algorithm uses the EDF scheduling method when utilization factor is smaller than the threshold value. When the utilization factor is equal to or greater than the threshold value and the proposed algorithm schedules the higher process first. Therefore the number of context switching is smaller than those of the other algorithms.

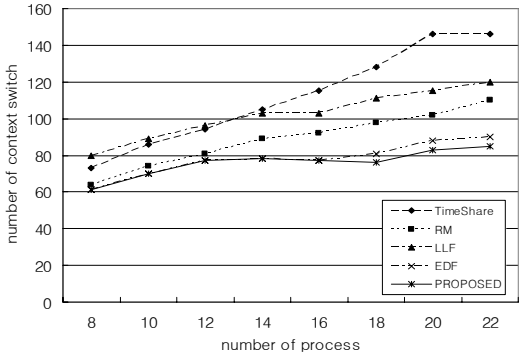

Figure 3. Number of context switching

4.2 Audio and Video Processes

To apply this algorithm to the video multimedia application, numbers of audio and video processes are increased and compared the performance of algorithms.

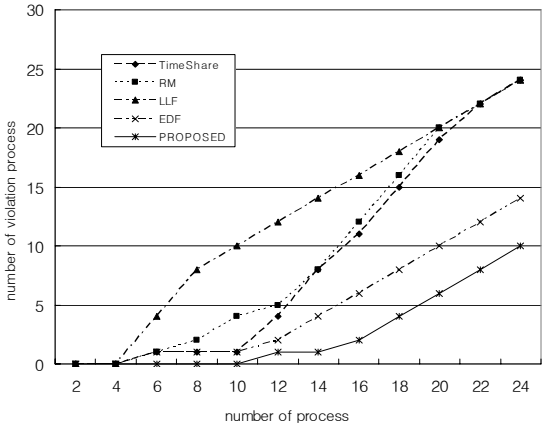

Figure 4. Scheduling comparison of video processes

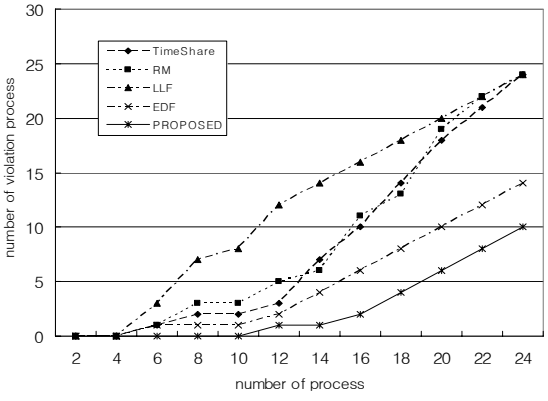

Figure 5. Scheduling comparison of audio processes

From Figure 4 and Figure 5, the proposed algorithm shows much better scheduling performance. The proposed algorithm can schedule up to 10 processes but EDF algorithm shows 4 processes. After the proposed algorithm meets the first violation, the increase rate of violations is very slow. For simulation the time quantum requirement is decided to satisfy QoS.

4.3 Mixed Priority Process

To compare the scheduling performance, processes are equally mixed with three different priorities, which are low priority, middle priority and high priority, are set. After shceduling the mixed priority processes, a new regualr process is added one by one their scheduling is and monitored. Figure 6 and Figure 7 show the violation number of EDF algorithm and the proposed algorithm. When many processes with different priority are scheduled, the proposed algorithm schedules more stable than the EDF algorithm.

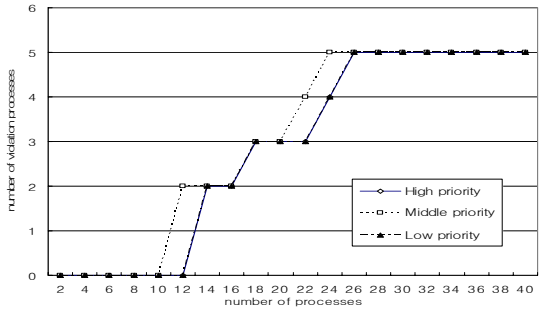

Figure 6. EDF scheduling for mixed priority processes

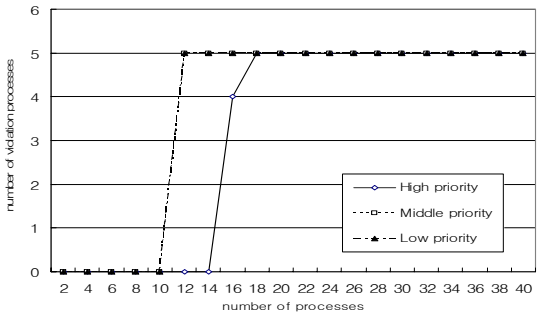

Figure 7. Proposed scheme scheduling for mixed priority processes

5 Conclusion

Linux assigns CPU time to each process fairly but real-time scheduling algorithms select the next process according to its schedule policy. EDF shows the good scheduling performance by simulation results. Since EDF only considers deadline, it is not adequate for multimedia embedded systems. For multimedia applications, the proposed algorithm reduces the number of violation by implementing the CPU utilization factor.

Also the context switching number is reduced up to 60% compare to the time sharing scheme and up to 10% compare to the EDF algorithm. This paper proposes a method by modified EDF based on critical factor and CPU utilization factor. The proposed algorithm shows low deadline violations and stable although number of processes is increased.

Please address any questions related on this paper to, Email: b.ahn@yu.ac.kr.

6 References

[1] M. Beck et al. Linux Kernel Internals, 2nd Ed, Addison-Wesley, 1998.

[2] Sokolsky O., "Resource Modeling for Embedded System Design," IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous System, PP. 99-103, May. 2004

[3] Young-Hwan Park, Embedded System & Embedded Linux, Scitec media, 2002

[4] Yoon-Mi Park, Embedded Linux System Design and Implementation, SCI 2003 autumn , Apr. 2003

[5] Shahid H. Bokhari, "The Linux Operation System," IEEE computer, Vol. 28, No. 8, 1995

[6] Diniel P. Bovet, Marco Cesati, Understanding the Linux Kernel, 2nd Edition, O' REILLY, September, 2003.

[7] C. L. Liu, J. W. Layland, "Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment," Journal of the ACM, 20-91, Jan, 1973.

[8] Ralf Steinmetz, "Analyzing the Multimedia Operating System," IEEE Multimedia, Spring, 1995.

[9] P. Goyal, X. Guo, H. M. Vin, "A hierarchical CPU scheduler for multimedia operating system," Proc. of the Symposium on Operating System Design and Implementation, October, 1996.

[10] J. Y. T Leung, M. L. Merrill, "A Note on Preemptive Scheduling of Periodic Real-Time Tasks," Information Processing Letters, PP.115, Nov, 1980.

[11] C. M. Krishna, Kang G. Shin, "Real Time System," The McGraw-Hill Companies, Inc, 1995.

[12] R. Steinmetz, "Analyzing the multimedia operating system," IEEE Multimedia, pp. 68, Spring, 1995.

[13] The cheddar project. "http://beru.univ-brest.fr/ ~singhoff/ cheddar".