Linux Based Real-Time Feature Supporting Method and Evaluation for Information Devices

YungJoon Jung, Donghyouk Lim, ChaeDeok Lim and Eun-Ser Lee

Abstract Recently, demands of information devices are increasing as we can find so many information devices around us such as smartphone, mobile internet device (MID), tablet. These characteristics of information devices services should support soft real-time based time guaranteed multimedia services and control internet appliances. In this situation, soft real-time supported system should be developed to consider as a total aspect of hardware, kernel, middleware, application. But this paper will describe soft real-time supporting and evaluation methods for information device as an aspect of only kernel.

Keywords Real-time · Linux · Kernel · Information devices

1 Introduction

Information devices are based on the information and services to a variety of control and processing equipment that can provide a means for users such as smartphone, MID, tablet, navigation, and PMP. The demand for these information

Y. Jung · D. Lim · C. Lim
Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea
e-mail: jjing@etri.re.kr

D. Lim
e-mail: befree@etri.re.kr

C. Lim
e-mail: cdlim@etri.re.kr

E.-S. Lee (✉)
Andong National University Computer Engineering, 388 Songcheon-dong, Andong, Gyeongsangbuk-do 760-749, South Korea
e-mail: eslee@andong.ac.kr

James J. (Jong Hyuk) Park et al. (eds.), *Future Information Technology, Application, and Service*, Lecture Notes in Electrical Engineering 164, DOI: 10.1007/978-94-007-4516-2_68,

devices increases recently, and mounted on mobile personal devices around the information can be found very easily. Compared with older devices such information devices richer hardware resources management and to provide users with a variety of services to the kernel, middleware and application software are included are deployed on the platform. These software platforms required various functionality and performance according to characteristics of information devices. In many information appliances, smart phones, MID, Tablet and same kinds of devices are needed real-time multimedia streaming services to control home appliances. These real-time multimedia streaming and appliance control service must be provided in a specified time budget. This means that information devices should be supported in real-time feature. This paper mentioned real-time feature has two characteristics. The first one is hard real-time characteristic. Given the time constraints, or in case system cannot maintain the accuracy of task. A failure of hard real-time system can cause to environmental or human disaster. Hard real-time system is primarily used in aerospace, nuclear, defense systems, etc. The second one is soft real-time characteristic. Various services provided regardless of environmental or human disaster because this soft real-time system characteristic is adapted to information devices such as smartphone, PMP, MID. But, this soft real-time characteristic should be considered performance and responsiveness issues simultaneously. If soft real-time system cannot maintain some time requirement of systems, soft real-time could control and process with exception handling method. As mentioned earlier, the real-time information used for equipment characteristics and environmental and human disasters pose to the soft real-time characteristics, so are supported. In other words, the temporal limits of the system design with performance parts, yet at the same time has the characteristics to be considered. The soft real-time characteristics in order to support real-time information devices that make up each of the kernel, middleware, applications, ranging consideration of all elements is required. In this paper, however limited to the kernel part of information devices to provide a method for real-time characteristic and the evaluation results are presented. In this paper, we propose to use Linux kernel as a software platform for information devices.

2 Related Work

2.1 Real-Time Operating Systems

Real-Time Operating Systems (RTOS) is a dedicated system to execute special functionality such as avionics, nuclear systems and defense systems. Unlike general purpose operating system, this provides a simple and light-weight structure that can support real-time features efficiently. There are many RTOSes, but VxWorks, VRTX and Nucleus are representative commercial RTOSes. In another case, RTLinux [1] is a dedicated operating system that has different underlying

structure. RTLinux provides the real-time services through RT-FIFO which is a real-time communication channel connecting the real-time kernel and linux server systems on the kernel. This OS has the characteristics of RTOS, as well as provides the services that originated from Linux.

2.2 Real-Time Feature Support in Linux

The main interest of this paper is Linux based real-time supports for information devices, such as O(1) scheduler [2, 3], voluntary Kernel Preemption [4], pre-emptible kernel [5, 6] and so on.

Among these methods, O(1) scheduler is employed as main scheduler of Linux kernel since Linux kernel 2.6. In Linux kernel 2.4, the previous scheduler was designed to maximize throughput. Even though the scheduler has one run-queue, it does not matter to be utilized in desktops and server systems. However, the scheduler must compare all tasks in a run-queue to choose a highest-priority task and this run-queue structure makes kernel scheduling not ended in specified time budget. This is not appropriate structure to support real-time features. To resolve this problem, O(1) scheduler adapts multiple run-queue and re-calculates time-slice of task in restricted time budget. Therefore, the scheduler can choose the highest-priority task in designated time and makes the scheduling latency predictable.

Preemptible kernel is a method that makes the most of kernel code reentrant. In the previous kernel, the kernel can switch to higher priority task, only when the task finishes its job in kernel mode and leaves the kernel mode. By modifying the lock mechanism, a task can be pre-emptible except when the task is in the preemption lock region and the kernel can switch to the task which has higher priority immediately. The internal scheduling mechanism is shown in Fig. 1 and the mechanism of preemptible kernel are shown in Fig. 2. Table 1 is the pseudo code of modified lock. Previous lock region which is enclosed by original spin_lock is preemptible lock region. Therefore, newly defined lock function, preempt_lock is added to beginning and end of lock regions as shown in Table 1. In result, other parts of the kernel are reentrant except for preemptible lock region and the kernel is changed into preemptible.

3 Real-Time Feature Supporting Method

3.1 Prioritized Interrupt Thread

The existing Linux kernel interrupt handling has been serialized in the priority-based interrupt handling structure to change the structure of the interrupt service routine to handle interrupts itself is threaded. There is explanation of this concept in Fig. 3 and 4.

Figure 3 illustrates the existing process is interrupted. When an interrupt occurs, the interrupt service routine to perform immediately, the scheduler cannot

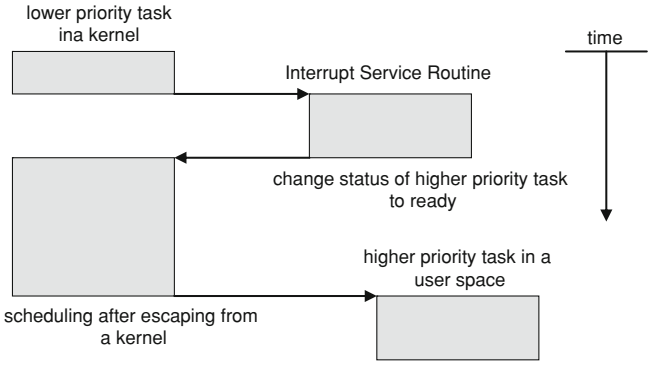

Fig. 1 Task scheduling structure of Linux

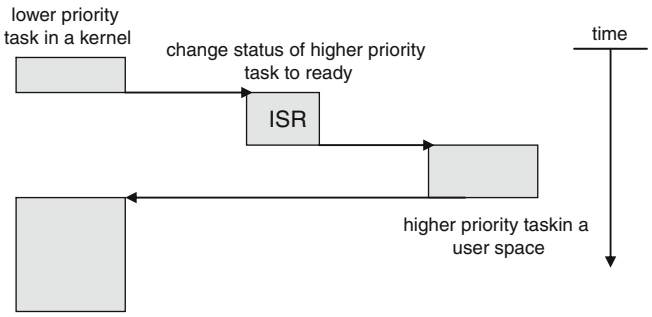

Fig. 2 Task scheduling structure of preemptible kernel

Table 1 Definition of preemption lock

```
//Preemption Enable Region
Preempt_lock();
Spin_lock()
/* Preemption Locked Region */
Spin_unlock()
Preempt_unlock();
//Preemption Enable Region
```

intervene. From the perspective of throughput, these interrupt handling method has the gain definitely. However, during interrupt processing for high-priority interrupt occurs, real-time tasks, if we could not be processed immediately. So to solve this problem using the same method as is shown in Fig. 4. The kernel creates a thread for each of the interrupt when an interrupt occurs, the interrupt by performing the process thread to handle interrupts. In this way, the scheduler is also able to intervene in the interrupt handling interrupts according to the priority of the thread

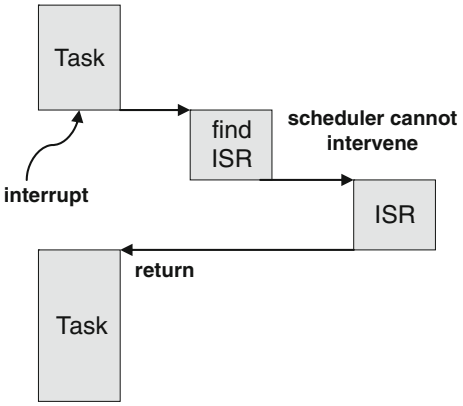

Fig. 3 Serialized interrupt processing structure of standard Linux

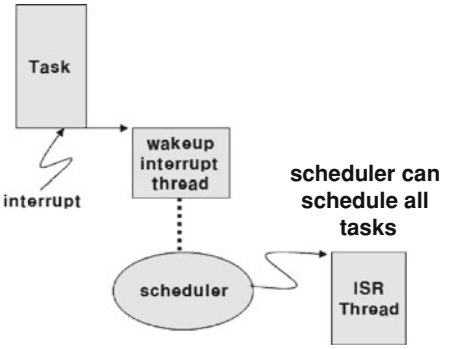

Fig. 4 Threaded interrupt processing structure

can be preempted. Therefore, the high priority task in less time than those who can has the advantage of scheduling.

The existing Linux kernel interrupt is divided into top half and bottom half. If the top half, how to apply to each interrupt handler "IRQ #" named after that to create a thread can be processed according to the priority should be. In addition, if the lower half of treatment, using the existing daemon called the softirqd sequential processing of all was the bottom half. Finally, the effectiveness of treatment for the softirqd to softirq-timer, net-rx, net-tx, scsi into three priorities to enable interrupt processing is based on.

3.2 Mutex Lock Adaptation

There are two types of lock mechanism. First, the spin locks to be a busy waiting in critical section. The second, mutex lock is capable of blocking in critical section. If you are using a spin lock existing lock structure was impossible to preemption in section. Mutex lock within a critical section can be preempted by

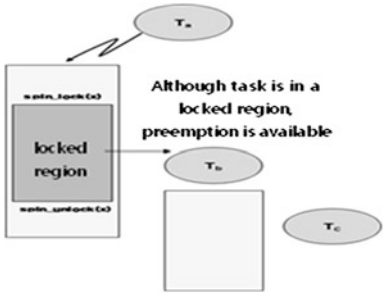

Fig. 5 Mutex based lock mechanism

helping improve the responsiveness of the system can support real-time characteristic. This is shown in Fig. 5.

As shown in Fig. 5, the task Ta spin_lock (x) section went into effect during execution of the task, even mutex lock Tb preempted by the higher priority task scheduling within a short delay should be able to help. The spin_lock (x) should serve as a real mutex lock. However, changes within the Linux kernel data structures are used as function names in order to minimize.

4 Evaluation Result

As mentioned earlier, information devices, soft real-time response to support of the O(1) scheduler, preemptive kernel, prioritized interrupt threads and mutex lock mechanism applied techniques such as real-time kernel based on a comparison of performance measurement were performed. Using target system is arm11 based smdk6410 board. CPU is a 533 MHz on the memory 256Mbyte. The kernel version 2.6.21.5, and to give the same stress was used to measure stress hackbench 20 is 1 h. Figure 6 to the measuring task, from the time an interrupt occurs, the corresponding time of preemption until the task is performed by measuring the time delay was measured real-time performance. Figure 6 measurement of the x-axis represents the time and, y-axis indicates the preemption latency unit is msec. The shorter the preemption latency time from the graph in real time higher preemption responsive means. The opposite in the case of real-time response performance can be lower. As shown in the figure in Fig. 6b, the kernel supports real-time characteristic of the task is much shorter compared to the normal Linux kernel preemption latency in Fig. 6a can be seen. Measured values can be seen in Table 2.

5 Conclusion and Future Work

Time limitations of the information required services for the information device increasing demands on the system too real-time characteristic is the growing trend. To this end, the Linux kernel support for real-time characteristic is essential,

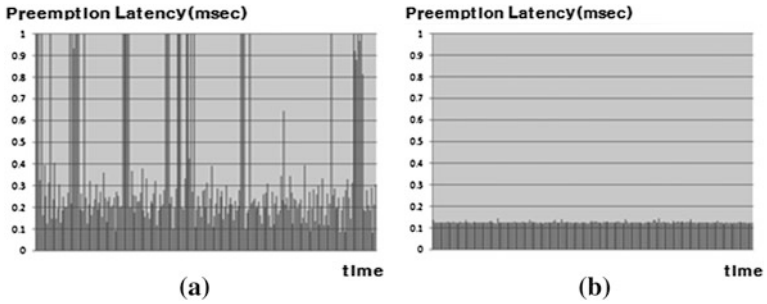

Fig. 6 **a** Preemption latency measurement graph of standard Linux based information device **b** Preemption latency measurement graph of soft real-time feature supporting kernel based information device

Table 2 Summary of preemption latency

	Average preemption latency (ms)	Minimum preemption latency (ms)	Maximum preemption latency (ms)
Vanilla Linux	0.742	0.022	7.633
Real-time feature supporting Linux	0.125	0.055	0.145

as mentioned earlier, O(1) scheduler, preemptible kernel, interrupt threads and mutex lock mechanism-based real-time support functions such as information devices using a real-time showed that the performance can be improved. This despite the fact that real-time performance improves POSIX API is preserved, while providing transparency for applications that enhance real-time response results were found to be derived. Planning future research that exists between the responsiveness and throughput trade-off studies the relationship that is optimized for the purpose of the system kernel technology that can provide ongoing research will be performed.

References

1. Yodaiken, V.: The RTLinux Manifesto. In: Proceedings of the 5th Linux Expo, Raleigh, NC (1999)
2. O(1) Scheduler: http://people.redhat.com/mingo/O(1)-scheduler/
3. Aas, J.: Understanding the Linux 2.6.8.1 CPU Scheduler, 2005 Silicon Graphics, Inc. (SGI), 17 Feb 2005
4. Linux kerneltrap: Voluntary Kernel Preemption, http://kerneltrap.org/node/3440, Kerneltrap, 10 July, 2004
5. Love, R.: Preemptible Kernel Patch, http://www.kernel.org/pub/linux/kernel/people/rml/preempt-kernel/
6. ELJOnline: Real-Time and Linux, Part 2: The Preemptible Kernel by Linux Devices, 1 Mar, 2002