논문 2011-06-49

정보기기들을 위한 리눅스 기반 연성 실시간 커널의 설계 및 평가 방법

(Design and Evaluation Method of Linux Based Soft Real-Time Kernel for Information Devices)

정 영 준, 임 동 혁, 임 채 덕, 최 훈*
(YungJoon Jung, Donghyouk Lim, Chaedeok Lim, Hoon Choi)

Abstract : Recently, demands of information devices are increasing as we can find so many information devices around us such as smartphone, MID(Mobile Internet Device), 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 : Linux, Soft Real-Time, Evaluation Method, Information Devices

1. 서론

정보기기라 함은 스마트폰, MID, 태블릿, 내비게이션, PMP 등의 각종 가전기기를 포함하여 기기 자체에 개인이나 기기의 정보에 바탕을 두어 다양한 제어와 처리 및 서비스를 사용자에게 제공할 수 있는 기기를 의미한다. 최근 이러한 정보기기들의 수요는 가히 폭발적이어서 개인적으로 휴대하거나 거치된 정보기기들을 주변에서 매우 쉽게 찾아볼 수 있다. 이러한 정보기기들에는 예전과 비교하여 더욱 풍부해진 하드웨어 자원을 관리하고 사용자들에게 다양한 서비스들을 제공하기 위해 커널, 미들웨어, 응용들이 포함된 소프트웨어 플랫폼이 탑재되는 경우가 많다. 이러한 소프트웨어 플랫폼은 정보기기의 특성에 따라 다양한 기능과 성능이 요구되는데 많은 정보기기 중 스마트폰, MID, 태블릿 등에서 사용되는 실시간 멀티미디어 스트리밍이나 가전제어 서비스 등에는 시간제한과 관련된 요구가 있을 수 있고 이런 경우에는 실시간성이 지원되어야 한다. 여기서 언급한 실시간성의 특성에는 두 가지가 있는데 첫 번째는 경성 실시간성으로 지원해야하는 실시간적 특성이 시스템의 정해진 시간적인 제약이나 정확성을 유지하지 못하는 경우로 인해 환경적으로나 인적으로 큰 재앙을 불러일으킬 수 있는 것으로 주로 항공, 원전, 국방용 장비 등에서 많이 요구된다. 두 번째는 연성 실시간성으로 앞서 언급했던 스마트폰, PMP, MID와 같은 기기들이 환경적인 큰 재앙과는 상관없이 제공되는 서비스로 서비스에 대한 응답성과 성능적인 이슈가 함께 고려되어 시스템의 정해진 시간적인 제약을 유지하지 못하는 경우에 예외처리를 통해 서비스를 보완해주는 특성을 가지고 있다. 이렇듯 일반적으로 실시간성의 두 가지 특성 중에서 스마트폰, MID, 태블릿에서 사용되어 실시간적 특성이 요구되는 서비스들에게는 환경적으로나 인적인 큰 재앙을 불러일으키는 것과 밀접한 연관성이 없으므로 연성 실시간 특성이 지원되면 된다. 즉, 시스템의 설계 시 대부분의 경우에 시간적인 제한을 가지면서 성능적인 부분도 함께 고려해야하는 특성을 가진다고 볼 수 있

* 교신저자(Corresponding Author)
논문접수 : 2011. 02. 01., 수정일 : 2011. 02. 14.,
채택확정 : 2011. 07. 27.
정영준, 임동혁, 임채덕 : 한국전자통신연구원 임베디드 SW플랫폼 연구팀
최훈 : 충남대학교 컴퓨터공학과

다. 이러한 연성 실시간적 특성을 지원하기 위해서는 각 정보기기를 구성하는 커널, 미들웨어, 응용에 이르기까지 모든 실시간 성능에 영향을 미치는 요소가 고려되어 개발되어야 하지만, 본 논문에서는 커널 부분에 한정하여 정보기기에서 실시간적 특성을 제공하기 위한 방법과 해당 방법에 따라 지원되는 연성 실시간 커널을 평가하는 방법을 제시한다. 본 논문에서 제시하는 정보기기용 소프트웨어 플랫폼 중 커널은 리눅스 커널을 사용한다.

II. 관련 연구

본 절에서는 정보기기를 위한 연성 실시간성 지원 커널과 관련한 기존 연구에 대해 설명하는데 주로 리눅스 기반 임베디드 시스템의 전반적인 실시간적 특성을 제공하는 커널 기술과 관련된 연구를 다룬다.

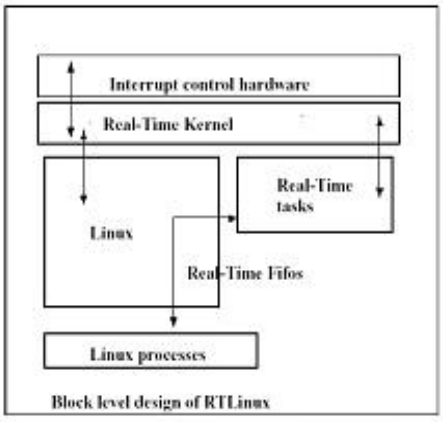

그림 1. RTLinux의 블록 다이어그램
Fig 1. Block Diagram of RTLinux

1. 실시간 운영체제(RTOS)

실시간 운영체제는 항공, 원전, 국방용 등 주로 특정 기능을 수행하는 전용시스템으로 리눅스와 같은 범용 운영체제와는 달리 구조적으로 단순화 및 경량화 되어 응답성이 짧아질 수 있도록 하여 실시간적 특성이 잘 지원될 수 있는 구조를 제공하고 있다. 이에 해당하는 운영체제는 다양하지만 그 중에서도 많이 사용되는 상용 운영체제는 VxWorks, VRTX, Nucleus 등이 있다. 이와는 다소 다른 구조를 가지고 있지만 전용 운영체제로 사용되는 RTLinux[1]가 있는데 주된 특징은 하드웨어위에 실시간 커널을 탑재하고 그 상위에 리눅스 서버를 두어 RT-FIFO라는 실시간 통신 채널을 두어 실시간 통신을 하며 실시간 서비스를 제공한다. 이 RTLinux는 실시간 운영체제의 특성을 가짐과 동시에 기존 리눅스의 서비스를 사용할 수 있는 장점을 가지고 있다. 그림 1은 RTLinux의 구조에 대한 블록 다이어그램이다.

2. 리눅스 기반 기존 실시간성 지원 기법

본 논문의 주 관심 대상인 정보기기를 위한 리눅스 기반의 기존 실시간 특성을 지원하는 방법에는 O(1) 스케줄러[2][3], 자발적 선점 커널(voluntary-preemption kernel)[4], 선점형 커널(preemptible kernel)[5][6] 및 락 브레이크 커널(lock-break kernel) 기법 등이 있다. 이 중에서 O(1) 스케줄러는 리눅스 커널 버전 2.6.x에서 적용되는 기술로서 기존 리눅스 커널 버전 2.4.x까지 처리량(throughput) 위주로 스케줄러가 설계되어 스케줄러에 대기큐(runqueue)를 하나만 두어 서버나 데스크톱에서 활용하기에 큰문제가 되지 않았다. 하지만, 실시간성 지원의 측면에서 태스크가 많은 경우 가장 우선순위가 높은 태스크를 선택하여 스케줄링하기 위해 많은 태스크를 모두 비교해야하기에 일정한 시간 내에 스케줄링이 끝나지 않아 실시간성 지원에 어려움이 있던 구조라 할 수 있었다. 이런 구조를 다중 대기큐와 제한된 시간 내 타임 슬라이스를 재계산하는 기법을 활용하여 항상 고정된 시간 내에 가장 우선순위가 높은 태스크가 스케줄링될 수 있도록 하여 태스크의 수행시간을 예측 가능하도록 함으로써 실시간성을 높일 수 있도록 했다. 자발적 선점 커널은 태스크가 커널 모드로 진입하여 구동되고 있는 중에 더 높은 우선순위를 가지는 태스크가 발생한 경우 기존에 커널 모드 수행을 모두 마치고 발생한 높은 우선순위 태스크에 스케줄링 해줄 수 있었던 것을 커널 내부에 미리 알고 있는 안전한 선점 포인트를 만나면 선점되어 보다 높은 우선순위의 태스크에게 더 빨리 스케줄링해줄 수 있도록 하여 응답지연시간을 짧게 하는데 도움을 줄 수 있도록 하고 있다. 자발적 선점 커널 기법은 현재 구현 이슈로 인해 선점형 커널과 함께 사용되지 못하고 있는 한계가 있다. 선점형 커널은 앞서 설명한 것과 같이 태스크가 사용자 모드에서 커널 모드로 진입하여 구동되고 있을 때 더 높은 우선순위의 태스크가 발생한 경우 기존 리눅스 커널은 수행되고 있는 커널 모드를 모두 수행한 다음 커널 모드를 빠져나올 때 더 높은 우선순위의 태스크에게 스케줄링 해 줄 수 있었으나 락 메커니즘을 수정하여 커널이 전반적으로 재진입(reentrance) 가능한

구조를 갖도록 하여 커널 모드를 모두 수행하지 않고도 태스크가 커널 모드의 락 구간 안에 있지만 않으면 선점이 가능하여 더 높은 우선순위의 태스크에게 바로 스케줄링이 가능하도록 지원하는 방법이다. 그림 2는 기존 리눅스 커널의 내부 태스크 스케줄링 구조를 나타내고 있으며, 그림 3은 선점형 커널의 태스크 스케줄링 구조를 그림으로 표현하고 있다.

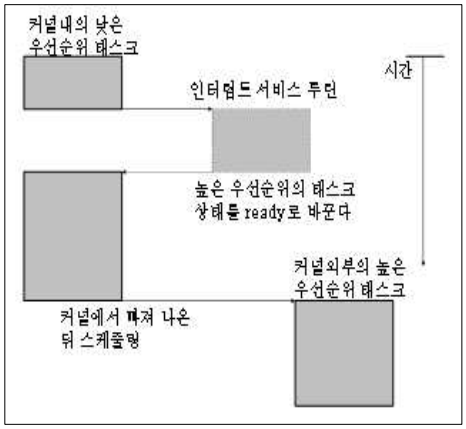

그림 2. 기존 리눅스 태스크 스케줄링 구조
Fig 2. Task Scheduling Structure of Linux

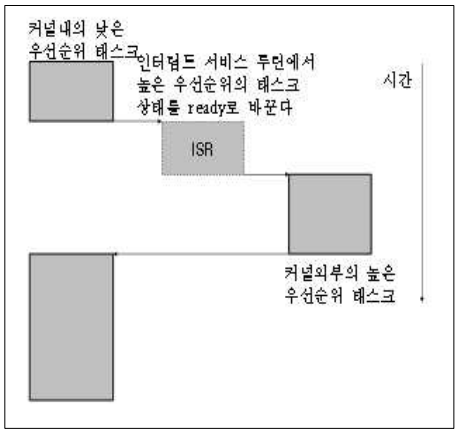

그림 3. 선점형 커널의 태스크 스케줄링 구조
Fig 3. Task Scheduling Structure of Preemptible Kernel

표 1은 선점형 커널을 구성하기 위한 수정된 락의 간략화한 코드를 보여주고 있는데 preempt_lock 이라는 선점형 커널을 위한 새로운 락 함수를 정의하여 기존 스핀락(spin_lock)이 사용된 지역이 락 구간이므로 새로 정의된 락 함수를 이용하여 태스크가 커널 모드로 재진입 가능하도록 만들어 준다.

표 1. 선점 락의 정의
Table 1. Definition of Preemption Lock

```
// 선점 가능 구간
preempt_lock();
spin_lock();
/* 선점락(preemption lock) 구간,
   선점 불가능 구간 */
spin_unlock();
preempt_unlock();
// 선점 가능 구간
```

락 브레이크 커널 기법은 커널 내 필요이상으로 루프를 돌며 락을 잡고 있는 코드들을 찾아내어 응답성을 개선할 수 있도록 한 커널 기법으로 선점형 커널 기법과 함께 쓰일 경우 보다 개선된 응답성을 기대할 수 있으나, 커널 전반적인 부분에서 활용되는 방법은 아니며 주로 파일시스템과 같은 특정 코드 영역에서 개선할 수 있는 부분을 찾아 실시간성을 지원하는 것으로 기능의 적용이 제한적이라 할 수 있다.

Ⅲ. 연성 실시간 커널의 설계 요구사항

정보기기에 사용되는 일반적인 리눅스 커널을 사용하여 실시간성을 지원하기 위해 고려해야할 사항은 다음과 같다.

1. 리눅스의 기본 자료구조를 유지해야한다.

리눅스의 기본적인 자료구조를 변형하지 않고, 리눅스의 기본적인 틀을 유지하여야 하며, 필요한 자료구조는 추가하여 구현하도록 한다.

2. 구현된 후에도 POSIX 표준 API(Application Programming Interface)를 따를 수 있어야한다 [7].

구현된 후 사용자가 커널을 사용함에 있어 표준 리눅스 커널과 비교하여 투명성(transparency)을 제공하며, 표준 리눅스 커널이 지원하는 각종 POSIX API를 그대로 이용할 수 있도록 하여야 한다.

3. 커널의 환경설정을 이용하여 선택 및 선택 가능하지 않도록 할 수 있어야 한다.

임베디드 리눅스의 특징인 사용자의 편의에 따라 커널 기능으로 선택 가능하도록 커널의 빌드 환경설정 시 선택사항으로 제공하여야 한다.

IV. 연성 실시간 커널의 설계 및 구현

정보기기를 위한 리눅스 기반 커널에 실시간적 특성을 제공하기 위한 방법은 II절에서 설명했던 기존 O(1) 스케줄러, 선점형 커널 및 락 브레이크 커널 기법을 함께 적용하고 다음의 두 가지 중요한 실시간성 지원 구조를 커널에 부가하여 전반적인 실시간 성능이 높아질 수 있도록 지원한다.

1. 인터럽트의 스레드(thread)화

기존 리눅스 커널 내의 직렬화되어 있는 인터럽트 처리 구조를 우선순위에 기반한 인터럽트 처리 구조로 바꾸기 위해 인터럽트를 처리하는 인터럽트 서비스 루틴 자체를 스레드화 한다. 이에 대한 개념적인 설명은 그림 4와 그림 5에서 설명한다.

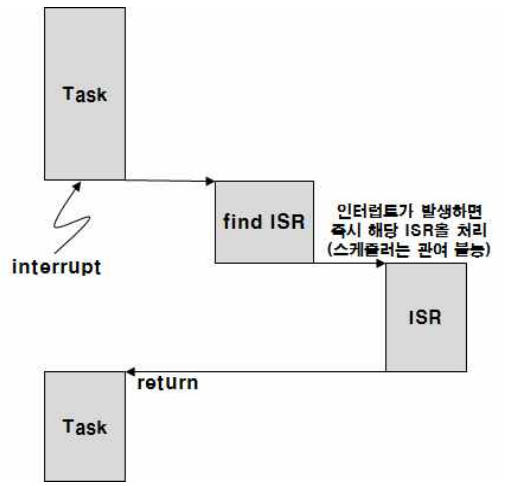

그림 4. 일반 리눅스 커널의 직렬화된 인터럽트 처리 구조

Fig 4. Serialized interrupt processing structure of standard linux

그림 4는 기존의 인터럽트 처리 과정을 설명하고 있는 데 인터럽트가 발생하면 그 즉시 인터럽트 서비스 루틴을 수행하도록 하여 스케줄러가 중간에 개입할 여지가 없는 것이 특징이다. 이렇게 하면 처리량에는 이득이 있지만 인터럽트를 처리하는 도중에 우선순위가 높은 실시간 태스크를 구동시키기 위한 인터럽트가 발생하는 경우에는 처리할 수 없는 단점이 생긴다. 따라서 이를 해결하기 위해 그림 5와 같이 각각의 인터럽트를 모두 스레드화하여 인터럽트가 발생했을 경우에도 해당 인터럽트를 처리할 수 있는 스레드를 수행하여 인터럽트를 처리하도록 한다. 이렇게 하면 인터럽트 처리 자체도 스케줄러가 관여할 수 있어서 스레드화된 인터럽트 스레드의 우선순위에 따라 선점이 가능해져 우선순위가 높은 태스크들에게 보다 짧은 시간 내에 스케줄링 해줄 수 있는 장점이 있다.

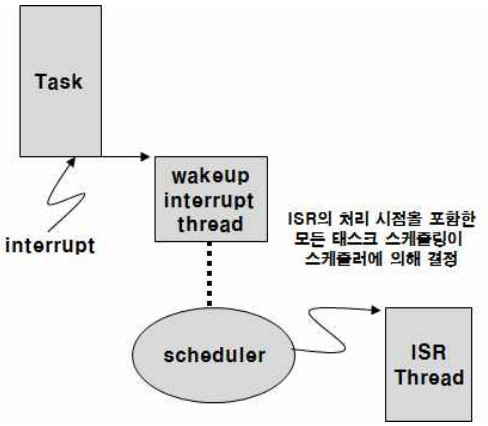

그림 5. 인터럽트 서비스 루틴을 스레드화한 인터럽트 처리 구조

Fig 5. Threaded interrupt processing structure

기존 리눅스 커널의 인터럽트는 전반부 처리(Top Half)와 하반부 처리(Bottom Half)로 나뉘어져 있는데, 구현 방법은 전반부 처리인 경우에는 각각의 인터럽트 핸들러에 대해 “IRQ #"의 이름을 붙여 스레드로 만들어 우선순위에 따라 처리가 가능하도록 해준다. 또한, 하반부 처리인 경우에는 기존에는 softirqd 라는 데몬을 이용하여 순차적으로 모든 하반부 처리를 하도록 했으나 처리의 효율성을 위해 softirqd를 softirq-timer, net-rx, net-tx, scsi로 나누어 각각에 대해 우선순위에 따른 인터럽트 처리가 가능하도록 지원한다.

2. 뮤텍스 락(Mutex Lock) 적용

데이터의 불일치성(inconsistency)를 방지해주는 락 메커니즘에는 busy waiting을 하는 스핀락과 락 구간 내에서 blocking이 가능하도록 지원하는 뮤텍

스 락의 두 가지 종류가 있다. 기존 스핀락을 사용하는 경우에는 락 구간 안에서 선점이 불가능한 구조였으나, 뮤텍스락은 락 구간 내에서도 선점이 가능하도록 지원하여 보다 시스템의 응답성을 높여 실시간적 특성을 효과적으로 지원할 수 있다. 이와 관련하여 그림 6에서 표현하고 있다.

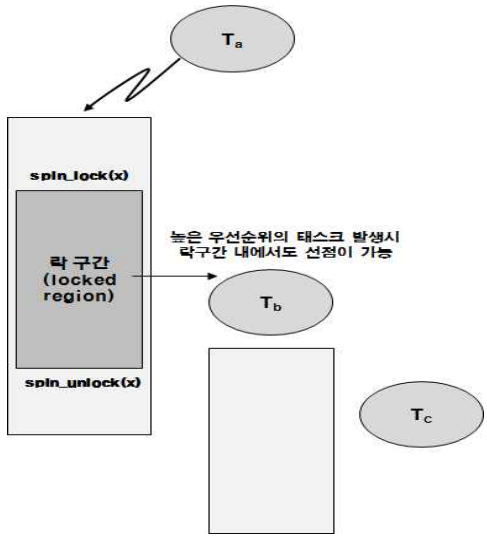

그림 6. 뮤텍스락 기반 락메커니즘
Fig 6. Mutex based Lock mechanism

그림 6에서 보듯이 태스크 Ta가 spin_lock(x) 구간에 들어가서 수행중일지라도 뮤텍스 락을 적용하여 태스크 Tb가 선점하여 보다 높은 우선순위의 태스크가 짧은 지연시간 내에 스케줄링이 가능할 수 있도록 지원한다. 이때 spin_lock(x)는 실제로 뮤텍스 락과 같은 역할을 하지만 리눅스 커널 내의 자료구조 변경을 최소화하기 위해 함수명을 그대로 사용하도록 했다.

V. 평가 방법

설계 및 구현된 리눅스 기반 연성 실시간 커널에 대한 실시간성 지원 정도를 측정하기 위한 평가 기법이 요구되는데 리눅스 커뮤니티에서 사용되는 방법보다 정밀성이 높은 측정 방법을 사용한다. 실시간성 측정을 위해 주기적인 하드웨어 인터럽트를 발생시킨 후 측정해야할 이벤트(인터럽트)의 발생 시각부터 실제 깨워서 수행해야할 태스크의 수행시작 시각까지의 시간을 측정하여 실질적인 선점지연시간(preemption latency)으로 활용하는데, 이는 실시간 응답성을 측정하여 표시하기에 의미 있는 지표라 할 수 있다. 이 때, 사용하는 주기적인 하드웨어 인터럽트는 RTC(Real-Time Clock)을 사용하는데 0~2048Hz의 주기를 미리 설정하여 사용할 수 있다. 이러한 선점지연시간의 측정 결과를 누적하여 최대값을 최대(최악) 선점지연시간으로 활용할 수 있으며, 실시간성 지원의 특성상 측정한 최대 선점지연시간 내에서 실시간성 지원이 보장된다고 의미할 수 있다. 지연시간의 측정은 시스템 내에 지원하는 성능 카운터(performance counter)를 사용하여 측정값의 고정밀성을 지원하게 한다. 일반적으로 성능 카운터를 사용하는 방법은 cpu architecture마다 다르게 되는데 x86인 경우에는 rdtsc(read time stamp counter) 명령어를 사용하고, arm11 이상에서는 mcr(move cp from reg)를 사용한다. 그림 7은 연성 실시간 커널의 실시간성 측정을 위해 실제 측정하려는 태스크 선점지연시간을 보여주고 있다. 실시간성 지원 특정 방법은 인터럽트가 발생할 때부터 실제 구동되어야할 사용자 모드 태스크가 구동될 때까지의 지연시간을 측정하고 이를 선점지연시간으로 사용한다. 여기에서 그림 7에서 보는 것과 같이 선점지연시간은 하드웨어 인터럽트 지연시간, 인터럽트 핸들러 수행 지연시간, 스케줄러 도달 지연시간, 스케줄링 지연시간 및 모드 변경 지연시간을 포함하는데 실제 측정은 하드웨어 인터럽트 지연시간은 구현상의 어려움과 측정구간의 값 대비 하드웨어 인터럽트 지연시간이 차지하는 비중이 매우 낮음으로 인해 측정 시 배제하고 인터럽트 핸들러 수행 지연시간에서부터 성능 카운터를 이용하여 측정하는데 이 또한 의미 있는 측정 결과라 할 수 있다.

VI. 결 과

앞에서 언급했던 대로 정보기기들을 위해 연성 실시간 성능을 지원하고자 O(1) 스케줄러, 선점형 커널, 락 브레이크 기법, 인터럽트 스레드, 뮤텍스 기반 락 메커니즘 등의 기법을 한꺼번에 적용한 커널을 V절에서 설명한 평가 방법으로 실시간 성능 측정 비교를 수행했다. 사용한 타겟보드는 arm11

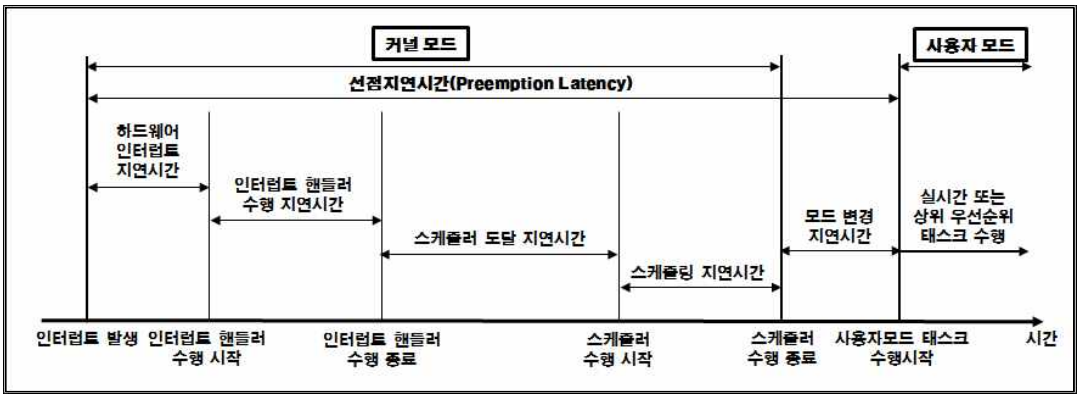

그림 7. 실시간성 측정 구간

Fig 7. real-time performance measurement period

기반의 smdk6410 보드이며, cpu는 533MHz에 메모리는 256Mbyte이다. 사용한 커널 버전은 2.6.21.5이고, 동일한 스트레스를 주기위해 스트레스는 hackbench 20을 사용했으며 측정시간은 1시간이다. 그림 8와 그림 9을 통해 측정하는 것은 태스크가 인터럽트가 발생했을 때부터 해당 태스크가 수행될 때까지의 시간인 선점지연시간을 측정하여 실시간 성능 측정을 했다. 그림 8와 그림 9의 x축은 측정시간을 의미하고, y축은 선점지연시간을 나타내는 것으로 단위는 msec이다. 그래프에서 선점지연시간이 짧으면 짧을수록 실시간 응답성능이 높은 것을 의미하고 그 반대의 경우에는 실시간 응답성능이 낮다고 할 수 있으며, 그림에서 보이는 것과 같이 그림 9에 있는 실시간 특성 지원 커널이 일반 리눅스 커널에 비해 훨씬 짧은 태스크 선점지연시간을 보이는 것을 알 수 있다.

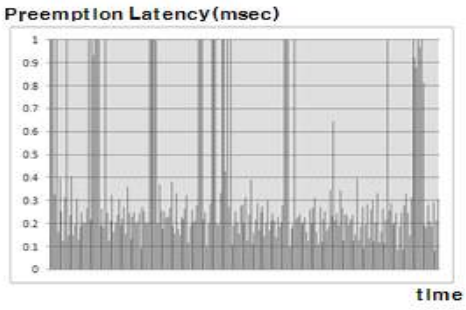

그림 8. 기존 리눅스 커널 기반 정보기기의 선점지연시간 측정 그래프

Fig. 8. Preemption Latency Measurement Graph of standard linux based information device

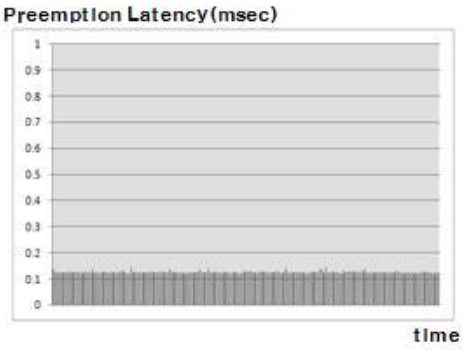

그림 9. 실시간 특성 지원 커널의 선점지연시간 측정 그래프

Fig. 9. Preemption Latency Measurement Graph of soft real-time feature supporting kernel based information device

표 2는 측정된 선점지연시간에 대한 정리 및 비교표이다.

표 2. 선점지연시간 정리 및 비교표

Table 2. Summary of Preemption Latency

	평균 선점지연 시간 (msec)	최소 선점지연 시간 (msec)	최대 선점지연 시간 (msec)
일반 리눅스 커널	0.742	0.022	7.633
실시간 성능 지원 커널	0.125	0.055	0.145

VI. 결론 및 향후계획

시간적인 제한이 서비스에 영향을 미칠 수 있어서 실시간적 특성이 요구되는 응용의 경우 기존 리눅스 커널에 실시간적 특성을 지원해야할 필요성이 있는데 앞서 언급했던 대로 O(1) 스케줄러, 선점형 커널, 락 브레이크, 인터럽트 스레드화 및 뮤텍스 기반 락 메커니즘 등의 실시간 지원 기능들을 이용하여 정보기기의 경우 실시간 성능 향상이 가능했음을 알 수 있었다. 이렇게 실시간 성능이 향상되는 가운데 POSIX API는 그대로 유지되어 사용자들에게 투명성(transparency)를 제공하여 응용 사용의 불편함은 없지만 실시간 응답성을 향상시키는 결과를 도출할 수 있음을 알 수 있었다. 향후계획은 응답성과 처리량 사이에 존재하는 trade-off 관계로 인해 실시간성이 향상된 만큼 처리량은 감소하게 되는데 이를 시스템의 용도에 맞도록 튜닝하여 최적화된 커널 기술들을 제공하는 기법에 대한 연구가 계속되어야 할 것이다.

참 고 문 헌

[1] V. Yodaiken, “The RTLinux Manifesto” paper, 2000.

[2] O(1) Scheduler, http://people.redhat.com/mingo/O(1)-scheduler/

[3] J. Aas, “Understanding the Linux 2.6.8.1 CPU Scheduler, 2005 Silicon Graphics, Inc., (SGI) Feb. 17, 2005

[4] Linux: “Voluntary Kernel Preemption “http://kerneltrap.org/node/3440,July10,2004,Kerneltrap

[5] R. Love. “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, Mar. 1, 2002

[7] Bill O. Gallmeister “POSIX.4.: Programming for the Real World” O’Reilly & Associates, Inc.

저 자 소 개

정 영 준 (YungJoon Jung)

1997년 : 한국외대 물리학과 학사.
1999년 : 한국외대 컴퓨터공학과 석사.
1999년 : ITS전문벤처 (주)오성아이엔씨 연구원
2001년 : 한국전자통신연구원 연구원
현재, 한국전자통신연구원 선임연구원.
관심분야 : 임베디드 소프트웨어, 실시간 시스템, 운영체제, 분산 컴퓨팅.
Email : jjing@etri.re.kr

임 동 혁 (Donghyouk Lim)

2003년 : KAIST 전산학 학사.
2005년 : KAIST 전산학 석사.
2005년 : 한국전자통신연구원 연구원
현재, 한국전자통신연구원 연구원
관심분야 : 임베디드소프트웨어, 운영체제, 실시간 시스템, 저전력 시스템.
Email : befree@etri.re.kr

임 채 덕 (Chaedeok Lim)

1988년 : 전남대학교 전산학 학사.
2000년 : 충남대학교 전산학 석사.
2005년 : 충남대학교 전산학 박사.

현재, 한국전자통신연구원 임베디드 SW플랫폼 연구팀장.
관심분야 : 임베디드소프트웨어, 분산실시간컴퓨팅.
Email : cdlim@etri.re.kr

최 훈 (Hoon Choi)

1983년 : 서울대학교 컴퓨터공학 학사.
1990년 : Duke University Computer Science 석사.
1993년 : Duke University Computer Science 박사.

1983년: 한국전자통신연구원 연구원
1996년: 충남대학교 컴퓨터공학과 교수
현재, 충남대학교 컴퓨터공학과 교수
관심분야 : 분산이동시스템, 운영체제, 웨어러블 컴퓨팅
Email : hc@cnu.ac.kr