소프트 리얼타임 기반의 네트웍 인터럽트 처리에 관한 연구

김정섭° 이대성 김기창
인하대학교 전자계산공학과
inhaboy@super.inha.ac.kr° kchang@inha.ac.kr

A study on advanced network interrupt processing based on soft real-time

Jeong-Seob Kim° Dae-Sung Lee Ki-Chang Kim
Dept. of Computer Science and Engineering, Inha University

요 약

최근 real-time 시스템들이 많이 개발되어 상용화되고 있고, 여러 산업분야에 적용되고 있다. 한편 리눅스 운영체제의 사용이 증가하고 안정화 됨에 따라, 이 시스템에 real-time 기능을 추가하기 위한 노력으로 여러 프로젝트가 진행중에 있다. 이 가운데 시스템 간에 네트웍 패킷을 송수신하고 이를 처리하는 과정에 대해 본 논문은 soft real-time을 적용하고자 한다. 기존 방식은 TOP HALF와 BOTTOM HALF로 인터럽트 처리과정을 나누어 BOTTOM HALF의 크기와 시스템 콜의 사용 빈도에 비례하여 종료시간을 예측할 수 없다. 이에 본 논문에서는 real-time 네트웍 패킷을 좀더 신속히 처리할 수 있도록 네트웍 인터럽트 처리 방안을 소개하고자 한다.

1. 서 론

리얼타임 시스템(Real-time system)이란 기정의된 마감시간(predefined deadlines) 이내에 특정 작업을 처리하는 시스템이다. 리얼타임 시스템은 크게 2가지로 분류된다. 기정의된 마감시간에 대해 절대적인 시간의 정확성(timing correctness)가 보장되어야 하는 시스템을 하드(hard) 리얼타임 시스템이라 하고 좀더 완화된 시간의 정확성을 구현한 시스템을 소프트(soft) 리얼타임 시스템이라고 한다[1].

리얼타임 시스템의 개념은 넓은 영역에 걸쳐 있는데, 본 논문은 그중에서도 소프트 리얼타임 시스템 범주의 네트웍 인터럽트 처리에 초점을 두었다. 기존 상용 real-time 운영체제로는 VxWorks, microC/OS, LynxOS, QNX 등이 있으며, 리눅스 기반의 리얼타임 운영체제는 RT linux 가 사용되고 있다.

리눅스는 범용 운영체제로 설계되었기 때문에 리얼타임을 지원하기에는 많은 제약이 있다. RT linux는 기존 리눅스에 리얼타임 기능을 부여하기 위한 프로젝트로써, 호환성을 위해 기존 리눅스 커널을 그대로 유지하면서, RT Linux 상에서 돌아가는 하나의 태스크(task)로 다룬다. RT Linux는 범용 Linux에 비해 좀더 리얼타임적인 요소를 가미하였으나, 대부분의 서비스는 기존 리눅스에 의존하고 특정 디바이스에 리얼타임 기능을 추가할 경우 디바이스 드라이버를 다시 설계해야 하는 단점이 있다[2].

한편, 범용 리눅스 운영체제상의 인터럽트 서비스는 2단계로 구성되는데, 급히 처리되어야 할 작업인 TOP HALF, 그다지 급하게 처리되지 않아도 되는 BOTTOM HALF로 구성된다. 이는 멀티 디바이스로부터 발생하는 다양한 인터럽트 처리에 있어 효율과 공평성을 위해 설계된 것이다. TOP HALF는 인터럽트 발생시에 곧바로 실행되는 ISR(Interrupt Service Routine)이며, 종종 BOTTOM HALF의 실행을 요구만 하고 종료(exit)된다. 인터럽트의 처리는 BOTTOM HALF와 함께 종료되므로, 인터럽트의 정확한 처리시간을 예측하기 어렵다.

네트웍 인터럽트 서비스는 NIC(Network Interface Card)로부터 메인 메모리로 패킷을 옮기는 단계인 TOP HALF와 TCP/IP 단계인 BOTTOM HALF로 구성된다. 앞서 언급한대로 BOTTOM HALF는 처리시간이 예측될 수 없을 뿐 아니라 코드 자체가 크다는 단점이 있다. 따라서 소프트 리얼타임을 네트웍 서비스에 도입하기 위해서는 크기가 적은 BOTTOM HALF를 제공해야 하고, TOP HALF 처리가 끝나면 BOTTOM HALF의 즉시 실행을 보장해야 한다.

이에 본 논문은 리눅스 운영체제의 네트웍 서비스에 대하여 기존 네트웍 드라이버와 프로토콜을 변경없이 사용하면서, 작은 크기의 BOTTOM HALF, TOP HALF와 BOTTOM HALF의 연속적인 실행을 보장하는 방법을 제안한다.

본 논문의 구성은 2장에서 리얼타임 네트웍 패킷의 처리에 대한 연구를 소개하고, 3장에서는 본 논문에서 제안하는 소프트 리얼타임 기반의 네트웍 인터럽트 처리에 관한 내용을 설명한다. 마지막으로 4장에서는 결론 및 향후 연구과제에 대해 설명하고 끝을 맺는다.

2. 관련 연구

본 논문은 리눅스 기반의 리얼타임 네트웍 서비스를 제안하므로, 관련 연구로써 기존 리눅스에서 네트웍 인터럽트의 발생과 그 처리 방법에 대해서 커널 2.4.2 버전의 코드를 조사하였다. 다음 그림은 2단계에 걸쳐 2.4.2 버전의 커널이 수행하는 인터럽트 처리과정을 보여준다.

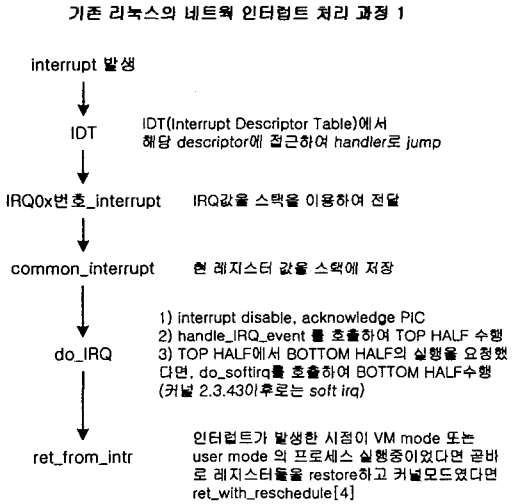

그림 2.1 하드웨어 인터럽트 발생 및 처리과정

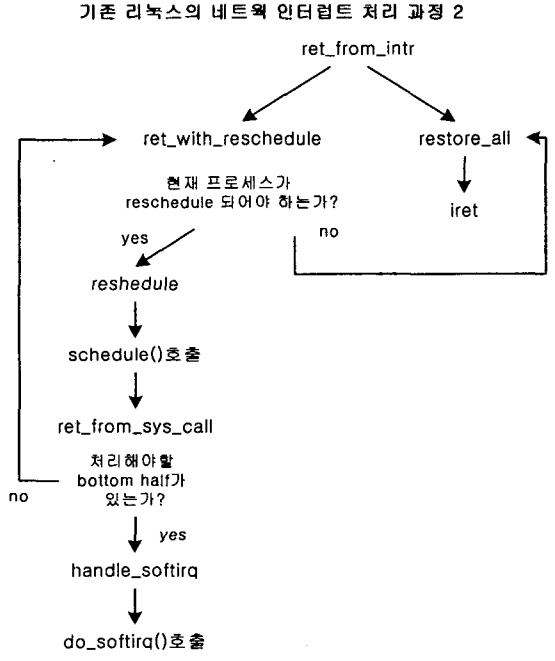

그림 2.2 스케줄링과정

3. 소프트 리얼타임 기반의 네트웍 인터럽트 처리

3.1 리얼타임 패킷의 구조

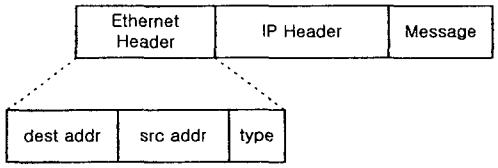

그림 3.1 리얼타임 패킷의 구조

수신된 프레임이 리얼타임 패킷 인지 아닌지를 구분하기 위해 이더넷 헤더(ethernet header)의 type 필드를 이용한다. 이 필드는 이더넷 헤더 다음에 붙는 데이터의 타입을 나타낸다. 이 필드는 802.2/802.3 프레임에서 프레임 길이를 나타내는데 최대 프레임 길이가 1500 임을 감안하여 최대 프레임 길이의 범위를 벗어난 값에서 기존 이더넷 프레임 타입과 겹치지 않는 값을 선택하여 리얼타임 패킷 타입을 지정하였다[3].

TOP HALF에서 패킷을 처리하기 위해서는 기존 BOTTOM HALF의 IP 계층을 TOP HALF에 추가해야 한다. 그러나 IP 계층의 기능을 전부 지원하기 위해서는 코드 사이즈가 커지고, TOP HALF에서 리얼타임 프로세스에 대한 스케줄링까지 해야하므로, 신속한 처리가 어려워 진다. 본 논문에서는 IP 계층의 기능중 최소한의 IP checksum과 reassemble 기능만을 추가하고 라우팅과 IP option 처리 기능은 제거하여 간단하고 코드 사이즈가 작은 IP 계층을 구현한다.

TOP HALF는 Message 필드를 분석하여 리얼타임 프로세스를 스케줄한다. 리얼타임 프로세스는 제한된 기능만을 수행하는 명령 수준의 간단한 어플리케이션이다. Message는 미리 지정된 명령어 테이블에 대한 인덱스이다.

3.2 리얼타임 프로세스 스케줄링

real-time process table

msg ID 0	real-time process 1
msg ID 0	real-time process 2
msg ID 0	real-time process 3
...	...
msg ID 0	real-time process N

그림 3.2 리얼타임 프로세스 테이블

본 논문이 제안하는 시스템은 이원화된 스케줄러를 갖는다. 리얼타임 패킷을 받으면 TOP HALF에서 메시지를 분석하고, 해당 명령을 수행하기위해 리얼타임 프로세스를 리얼타임 프로세스 테이블에서 해당 메시지에 해당하는 리얼타임 프로세스를 찾아서 실행한다. 기존 리눅스의 스케줄러는 enqueue된 프로세스를 다음 4가지

이벤트가 발생될 때 마다 실행된다.

- 시스템 콜 처리후
- 예외(exception) 처리후
- BOTTOM HALF(soft irq) 처리 후
- 커널이 새로운 프로세스를 선택하여 구동하기 위해 reschedule()을 호출할 경우

따라서 이들 이벤트가 발생할 경우 다음과 같은 흐름(flow)가 추가된다.

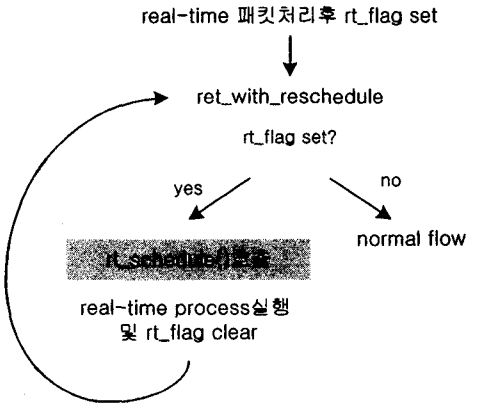

그림 3.3 리얼타임 패킷을 처리하기위해
그림 2.2의 흐름에 추가될 흐름(flow)

TOP HALF에서 리얼타임 패킷을 처리하고, 메시지를 추출한다음, real-time flag인 rt_flag를 set시킨다. 그리고 나서 ret_with_reschedule에서 rt_flag값이 set 되어 있다면, rt_schedule()을 호출하고, 아닌경우 기존 동작을 수행한다. 다음 코드는 커널 2.4.2에서 linux/arch/i386/kernel/entry.S 의 코드에 추가될 부분을 음영처리 하였다.

```
ret_with_reschedule:
        [redacted]
        cmpl $0,need_resched(%ebx)
        jne reschedule
        cmpl $0,sigpending(%ebx)
        jne signal_return
restore_all:
        RESTORE_ALL

[redacted]
```

리얼타임 스케줄러는 리얼타임 프로세스 테이블에서 추출한 리얼타임 프로세스를 처리하고, 기존 리눅스의 scheduling path로 진행한다.

기존 RT linux의 스케줄링방식은 RT linux 커널의 자료구조 일부를 기존 리눅스 시스템과 공유하는 방식임에 비해 본 논문에서는 스케줄링과 관련된 자료구조를 기존 리눅스 시스템에 포함시키기 때문에 좀더 효율적인 처리를 기대할 수 있다.

3.3 리얼타임 프로세스

리얼타임을 지원하기 위해서는 운영체제 뿐 아니라 어플리케이션의 지원도 필요하다. 어플리케이션에서 불필요한 시스템 콜과 시스템 자원을 요청할 경우, 지연시간이 증가된다. TOP HALF 내에서는 인터럽트를 disable 시키므로, 중첩된 리얼타임 인터럽트는 무시하게 된다. 따라서 리얼타임 프로세스는 동작이 단순하고 시스템 자원의 요청과 시스템 콜의 사용을 최대한 낮춰야 한다. 본 논문에서 제안하는 리얼타임은 기존 리눅스 운영체제 기반하에 설계되었기 때문에 시스템 콜과 자원 관리 기능은 기존 리눅스에 의존한다.

4. 결론 및 향후 연구 과제

본 논문은 특정 네트웍 패킷을 받아서 패킷을 분석하고 프로세스를 스케줄하여 실행할 수 있는 소프트 리얼타임 패킷 처리 방식을 제안하였다. 이 기능은 긴급 메시지를 보내어 특정 작업을 네트웍을 통해 처리해야 할 경우에 적용될 수 있다. 그러나 시스템 콜처럼 블록되거나 코드 크기가 큰 작업은 리얼타임 처리에 불리하므로 제한된 경우에 적용된다. 따라서 모든 네트웍 패킷에 대해 소프트 리얼타임을 적용하기 위해서는 크기가 작은 TCP/IP의 구현과 블록되지 않는 시스템 콜을 제작해야 할 것이다.

참고문헌

[1] F.Panzieri R.Davoli, " Real Time Systems" Technical Report UBLCS-93-22, 1993
[2] 이호, " Real Time과 리눅스, 그리고 RT-Linux" http://linuxkernel.net/doc/flyduck/rtlinux/rtlinux.html, 1999
[3] J. Postel, " A Standard for the Transmission of IP Datagrams over IEEE 802 Networks" , RFC 1042, 1988
[4]http://kelp.or.kr/korweblog/upload/12/20020316012454/LinuxKernel34_1.doc
[5] W. Richard Stevens, " TCP/IP Illustrated, Volumel The Protocols" Addison-Wesley, 1994
[6] Daniel P. Bovet Marco Cesati, " Understanding the Linux Kernel" O' Reilly, 2001