임베디드 시스템을 위한 리눅스내의 RMO설계 및 구현

오지성 박현희° 양승민
삼성전자(주) 숭실대학교
jiseong.oh@samsung.com, darklight@realtime.ssu.ac.kr,yang@computing.ssu.ac.kr

Design and Implementation of RMO for Linux-based Embedded System

Ji-seong Oh Hyun-hee Park° S.M Yang
SAMSUNG Electronic Company , Soonsil University

요 약

내장 실시간 시스템은 고 신뢰성 및 고 가용성이 요구된다. 이를 위해서 반드시 필요한 요소들은 결함을 감지 할 수 있는 온라인 모니터링과 감지된 결함에 대한 결함허용(Fault Tolerance) 기법으로 온라인 모니터링에 관련된 연구로는 dRTO 모델[1]의 RMO(Region Monitoring Object)[2]가 있다. 본 논문에서는 효율적인 온라인 모니터링 프레임워크를 내장 시스템에서 널리 사용되는 운영체제인 리눅스에 제공하기 위하여 리눅스상에서 동작하는 RMO 프레임워크를 적용한 LinuxRMO를 설계하고 구현한다. LinuxRMO가 감시하는 대상은 dRTO모델에서 정의된 RMO의 감시대상인 RTO(Realtime Object)[3][4]가 아닌 리눅스 프로세스들이다. LinuxRMO는 리눅스 프로세스의 변수 데이터들에 대한 실시간 추적, 프로세스들에 대한 주기적인 감시 및 사용자가 정의한 허용시간을 초과한 프로세스들을 감지하는 역할을 한다.

1. 서 론

고 신뢰성을 지니는 내장 시스템을 구축하기 위해서는 시스템 내에서 발생할 수 있는 결함에 대해서 효과적으로 처리할 수 있는 시스템내의 장치가 필요하다. 이러한 장치 또는 기법들을 결함허용(Fault Tolerance)이라 하며, 결함허용 기법을 시스템에 적용하기 위해서 먼저 결함이 발생할 만한 요소들에 대한 모니터링(monitoring)이 존재해야 하고, 모니터링과 결함허용이 시스템 내에서 유기적으로 상호 연관되어야 한다.

모니터링과 결함허용의 상호 연관성을 고려한 고 신뢰성, 고 가용성 시스템을 개발하기 위해서는 시스템의 설계 단계에서부터 각각이 모델링되어야 하며, 이를 기반으로 한 프레임워크가 구축되어야 한다. 따라서 모니터링과 결함허용에 관련된 모델의 존재가 필수적인데, 이런 모델로는 dRTO모델의 RMO가 있다. RMO는 시스템을 구성하는 구성요소들(RTO)에 대한 모니터링을 수행하는 객체이다. 지역(Region)이라 부르는 여러 객체들의 집합을 모니터링 할 수 있으며, 모니터링시 감시된 객체에 결함이 발생되면 그에 대한 적절한 처리를 할 수 있는 인터페이스를 지원한다.

본 논문에서는 RMO기반 모니터링 프레임 워크를 내장 시스템에 적용시키기 위해서 RTO모델의 RMO를 임베디드 시스템의 플랫폼으로 가장 많이 사용되고 있는 리눅스에 적용한 LinuxRMO를 설계하고 구현한다. 이를 위해 리눅스 커널 버전 2.6에 RMO에 대한 관리기능을 추가하며, 사용자는 사용자 정의 RMO를 생성하여 특정 프로세스를 모니터링 할 수 있다. 또한 사용자가 쉽게 적용할 수 있도록 API형태로 제공한다. LinuxRMO는 내장 시스템 환경을 고려하여 최소한의 자원을 사용한다.

2. 관련 연구

2.1 dRTO(dependable Real Time Object)

dRTO 모델은 신뢰도 신뢰도 높은 내장 실시간 시스템의 효율적인 구출을 위하여 객체지향과 실시간성, 그리고 신뢰성의 세가지 기본 개념을 단일 모델에 통합한 것이다.

dRTO모델에서 내장 실시간 시스템은 RTO의 집합으로 구성된다. RTO는 자료와 메소드들로 구성되어 있는데 메소드들은 주어진 시간 명세에 따라 주기적 또는 특정시간에 구동하는 TM(Time Triggered Method), 외부로부터의 호출에 의해 구동되는 MM(Message Triggered Method), 주어진 조건이 만족될 때 구동하는 CM(Condition Triggered Method)로 구성된다. RTO들 사이의 메시지 전달은 일반 객체 모델과 마찬가지로 MM호출을 통해서 이루어진다. CM은 조건이 만족돼 었을때 능동적으로 구동하는 메소드로써, TM과 마찬가지로 다른 메소드들에 의해서는 구동이 되지 않는다.

2.2 RMO(Region Monitor Object)

RMO는 감시지역(Region)에 소속된 실시간 객체들의 상태를 감시 및 분석함으로써 실시간 객체들의 비정상적인 행위를 감지한 후, 그 증상을 진단하여 결함을 찾아 알맞은 복구 및 재배치 방법을 결정하고 수행하는 실시간 객체이다.

2.3 리눅스 커널 2.6

리눅스 커널 2.6에서 제공하는 워크 큐[5]는 워크 큐 구조체를 이용하여 등록되어 있는 함수의 수행을 keventd와 같은 커널 스레드가 수행해 준다. 그리고 seqlock(sequence lock)[6]는 리눅스 커널 2.6에서 지원하는 새로운 락 메커니즘으로, 작고 단순하고 순차적으로 공유자원에 접근할 수 있도록 빠르게 사용되는 곳에서 이용할 수 있는 락이다. 커널 타이머에서 이러한 락을 사용하여 RMO의 데드라인, 프로세스의 데드라인과 주기적인 감시 시간의 값에 락을 걸어 사용한다.

3. Linux RMO의 설계 및 구현

내장 실시간 시스템내의 LinuxRMO를 구현하기 위하여 고려할 모니터링에 관련된 요소들은 (1)프로세스 내의 변수의 온라인 추적 및 감시, (2)프로세스의 이상 유무 주기적 감시 및 데

드라인 타임의 설정, (3)RMO 자체의 신뢰성 보장, (4)감시 하는 프로세스의 성능 보장, (5)내장 실시간 시스템을 위한 최적의 메모리 사용, (6)시스템에 독립적인 높은 이식성들이다.

3.1 LinuxRMO의 구조

LinuxRMO는 그림 1은 같은 구조로 되어있다.

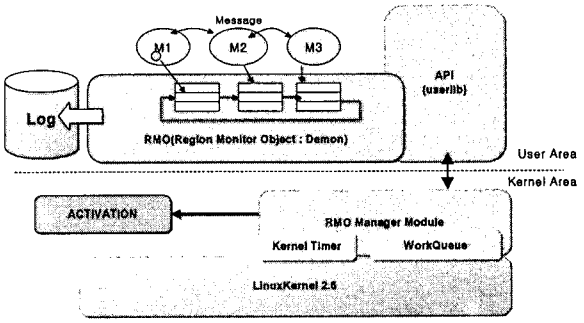

그림 1 LinuxRMO의 구조

리눅스 커널 2.6상에서 CM의 ACTIVATION을 위한 RMOManager 가 Kernel Module상태로 구성되어 있으며 각각의 프로세스를 관리할 수 있는 RMO가 데몬의 형태로 구성되어 각각의 프로세스를 추적할 수 있도록 되어 있고, RMO를 구동 혹은 RMO내의 관리 테이블에 등록 및 해제를 할 수 있는 API가 라이브러리 형태로 되어있다. 또한 TM을 지원하기 위하여 커널타이머를 이용하였다. LinuxRMO는 외부의 추가적인 디바이스가 필요없이 자체적으로 동작하도록 설계하여 동작 중에도 타겟 시스템 내에서 확인할 수 있고, 일반사용자들도 손쉽게 임베디드 시스템상태를 확인하여 결함이 있을 시에 알 수 있도록 구성하였다.

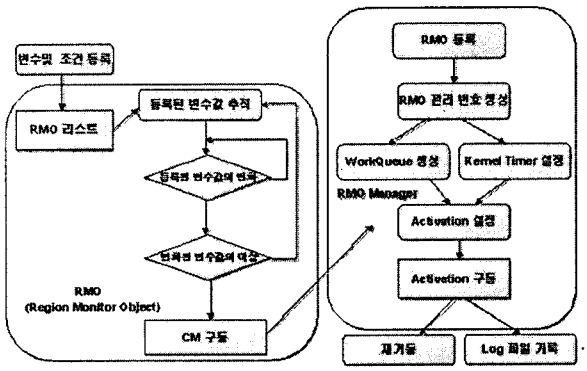

그림 2 LinuxRMO Flow Chart

그림 2는 LinuxRMO의 실행흐름을 보여준다. RMO 데몬은 RMO 테이블 내에 등록된 변수 값의 변화가 있을시 CM의 조건으로 되어있는 Condition과 비교를 하여서 이상이 있다면 RMO내의 CM을 구동을 시킨다. 또한 추적하고 있는 프로세스의 동작여부를 주기적으로 체크하도록 TM을 설정해놓았으며 일정시간 이상이 되고나서도 계속적으로 프로세스가 존재한다면 좀비상태 혹은 프로세스 블록 상태로 가정하고 프로세스 혹은 시스템이 재부팅 되도록 구현한다.

RMOManager에서는 RMO의 등록 여부를 확인하고 RMO가 등록이 된다면 각 RMO를 관리할 수 있도록 관리 번호를 생성해 준다. 그리고 등록된 RMO를 위한 WorkQueue를 생성을 해주고, CM이 구동이 되면 RMOManager내에 정의 되어 있는 ACTIVATION을 수행하도록 하여 Log 파일에 기록을 하게 된다. 그리고 TM을 위해 커널타이머를 이용한 TM의 마감 및 주기를 관리할 수 있도록 모듈로 구성이 되어있다.

3.2 RMO

RMO의 기본적인 기능은 실시간 오브젝트의 실시간 오브젝트로 RMO 오브젝트상에서 각각의 감시지역에 있는 각각의 객체들을 감시하도록 설계되었지만 일반적인 리눅스의 프로세스를 감시하기 위해서는 MM역할은 일반적인 IPC(Inter Process Communication)으로 이용할 수 있고 CM, TM의 기능에 초점을 두고 어떠한 상태가 되면 동작이 되고 각 프로세스들의 마감시간 및 주기적인 감시를 위한 시간의 구현의 초점을 맞춘다. RMO는 데몬 형태로 제공되어져서 프로세스를 실시간적으로 추적할 수 있도록 구성된다.

그림 3은 임의의 프로세스 P내부의 변수 traced를 RMO에서 모니터링 하는 예를 보여준다.

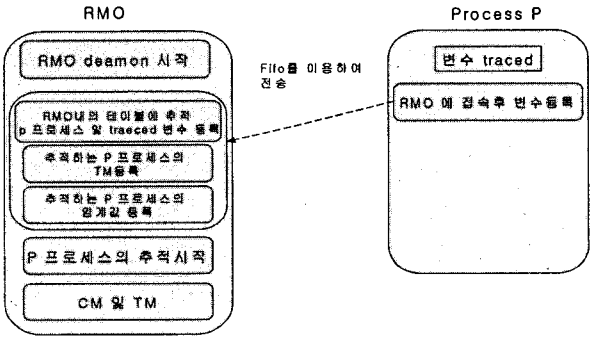

그림 3 RMO 동작 예

RMO 데몬을 시작하면 RMO 내부의 변수 테이블이 생성이 되고 테이블 내에 추적하고자 하는 프로세스 및 변수가 있는지 확인하고 없으면 아무런 일도 하지 않는다. 사용자가 추적하고자 하는 프로세스 P, 변수 traced 및 임계값을 등록을 하면 fifo를 이용하여 RMO내에 전송을 한다. RMO내에서는 이를 받아서 테이블 내에 등록하고 각각 CM내에는 임계값, TM에서는 P 프로세스의 주기적 감시 및 데드라인 타임이 설정이 된다. 이상조건이 발생한다면 이에 맞는 ACTIVATION을 구동한다.

3.2.1 RMO의 CM 및 TM의 처리

CM은 전체 6단계를 거쳐 처리 된다. (1)처음에 감시할 조건을 등록한다. (2)이후 계속 적으로 리스트 내에 등록되어 있는 변수들을 추적한다. (3)각각의 변수 이름과 주소에 맞는 조건들을 확인(cond_comp)한다. (4)조건을 벗어나는 경우 각 오류의 종류에 따른 메시지로 ACTIVATION함수를 호출한다. (5)RMO_Manager내의 WorkQueue에 등록한 후 (6)로그화일에 원인을 기록 후 각 조건에 따른 동작 수행한다.

반면, TM은 4단계 과정으로 이루어진다. (1)처음에 임의의 P 프로세스를 감시할 시간을 등록(주기 및 데드라인)한다. (2) 주기적으로 동작중인 프로세스가 동작중인가를 검사한다. (3) 데드라인 타임이 지났을 경우에도 프로세스가 동작하는 경우

좀비 및 프로세스 블록 상태로 판단한다. (4)로그화일에 기록하고 프로세스나 시스템을 재 기동한다.

4. 성능 평가

4.1 성능평가 결과 및 분석

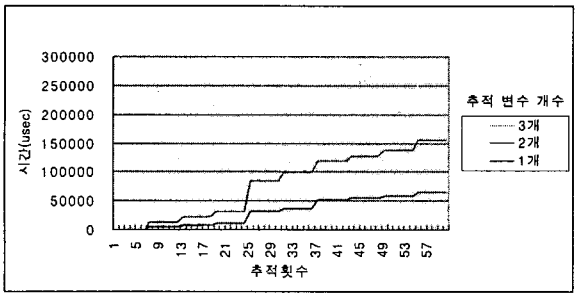

그림 4 프로세스내 변수 추적 시간

그림 4는 변수의 개수에 따른 시간측정의 결과를 그래프로 나타 낸 것으로써 추적하는 프로세스의 수가 늘어가고 측정하는 횟수가 증가함에 따라 시간도 보다 더 많이 증가 하는 것을 볼 수 있다[7].

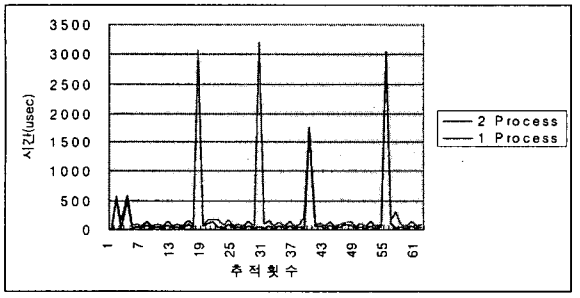

그림 5 프로세스내 시스템 콜 및 시그널 추적시간

그림 5는 프로세스의 개수에 따른 시스템 콜 추적[8] 및 시그널 함수 추적시간 측정의 결과를 그래프로 나타낸 것으로써 프로세스가 fork()를 호출하여 하나의 프로세스를 복제하는 경우, 불규칙적인 시간이 크게 증가 되는 것을 볼 수 있는데,이것은 자식 프로세스의 시스템 콜시 부모프로세스가 대기하는 시간 때문이다.

5. 결론 및 향후 연구 과제

임베디드 시스템의 개발에 앞서 전체 시스템에 대한 분석을 통하여 시스템의 결함이 발생 했을 때 이를 탐지하고 처리할 수 있는 시스템의 모델링이 고려되어야 하나 단순한 기능 구현에 초점을 두고 개발이 되고 있다. 이러한 시스템들은 고 신뢰도를 요구하는 시스템에 적용 시 많은 비용과 손실을 초래 할 수 있다. 본 논문에서는 이를 보완하기 위해서 제안된 dRTO모델의 RMO를 이용, 리눅스에 적용시켜 결함에 대비한 시스템에 자체적인 모니터링 툴인 LinuxRMO를 설계 및 구현 하였다.

LinuxRMO는 시스템에서 발생할 수 있는 결함을 파악하도록 설계되었으며, 유저 프로세스형태로 따로 시스템을 수정할 필요 없이 제거가 가능하도록 구현하였다. 아키텍처에 상관없는 리눅스상에서 동작이 가능하기 때문에 보다 쉬운 적용성과 이식성을 가진다. 성능평가에서 나온 결과는 이러한 적용성과 이식성이 뛰어남을 보여주고 있다.

향후과제로는 LinuxRMO의 구현상에서 커널자체를 수정해야 하는 부분에 대해서 커널을 수정하지 않고 LinuxRMO를 적용할 수 있는 방법이 연구되어야한다. 또한 RMOManager와 API의 최적화를 통해 구현상의 성능을 향상시켜야 할 것이며, 모니터링 기능뿐이 아니라 결함허용이 가능하도록 ACTIVATION 부분에 복구 기능을 추가 하여 결함이 발생 시 자체적으로 진단 후 복구가 되도록 하는 연구가 이루어져야 한다.

참고문헌

[1] 이신, 손혁수, 양승민, "실시간 객체 모델 dRTO (Real-Time Object Model dRTO)", 한국정보과학회 논문지: 소프트웨어 및 응용, 제27권, 제3호, pp.300-312, 2000. 3. (정통부 대학기초 지원).

[2] H.T. LIM and S.M Yang. "A Framework to Model Dependable Real-Time Systems based on Real-Time Object Model" to appear in RTCSA'00, December 2000.

[3] K.H(Kane) "Kim Object Structes for Real-Time Systems and Simulators", 0018-9162 IEEE ,1997

[4] K.H.Kim "Falut-Tolerant Real-Time Object" Communications of ACM 1997.Jan

[5] Jonathan corbet,rubini & Greg "Linux Device Drivers 3rd", oreilly, 2004

[6] Linux Journal "http://www.linuxjournal.com"

[7] Erik(J.A.k.) Mouw Delft University of Techonlogy "Linux kernel Procfs Guide"

[8] Volkmar Sieh "Fault-Injector using UNIX ptrace Interface"