TMO 프로그램의 실시간 제약 위반을 감시하는 수행시간 모니터의 구현

민 병 준*, 최 재 영*, 김 정 국**
*인천대학교 전자계산학과, ** 한국외국어대학교 컴퓨터공학과

Implementation of a Runtime Monitor Checking Real-time Constraint Violation of TMO Programs

Byoung-Joon Min*, Jae-Young Choi*, Jung-Guk Kim**
*Dept. of Computer Sci., Univ. of Inchon, **Dept. of Computer Eng., Hankuk Univ. of Foreign Studies

요 약

본 논문에서는 실시간 시스템의 시간 제약이 제대로 만족되는가를 시스템 수행 중에 감시하는 수행시간 모니터를 구현하기 위한 환경으로 실시간 객체 모델인 TMO(Time-triggered Message-triggered Object) 모델과 Windows 98/NT 상의 TMO 프로그램 실행 환경인 WTMOS(Windows TMO System)를 이용한다. 모니터의 대상과 모니터하는 조건을 TMO 프로그램에 명시하는 방법이 연구되었고 정의된 모니터 기능을 WTMOS 내부와 TMO 형태의 응용 객체로 분산시켜서 적은 비용으로 모니터 시스템을 구축하는 효과적인 방법이 개발되었다

1. 서론

일반적으로 수행시간 모니터[1](runtime monitor)는 프로그램 디버깅, 시스템의 성능 평가, 또는 시스템의 올바른 실행 상태 확인을 목적으로 사용된다. 이 중에서 시간 제약을 감시하는 수행시간 모니터는 수행 시간 중에 시스템의 시간 관련 정보를 수집하고 그것이 설계 시 기술된 시간 조건을 만족하는가를 확인한다. 위반 사항이 검출되면 모니터는 사용자에게 그 사실을 알리거나 사전에 준비되어 있는 복구 작업을 수행한다.

실시간성 보장을 위한 모니터 관련 연구는 크게 두 가지로 나누어 생각할 수 있다. 하나는 대상 시스템 내에 아무런 모니터 관련 장치를 두지 않고 모니터 하는 경우이다. 즉, 전형적인 입증과 시험(verification and testing) 방법에 따라 대상 시스템을 블랙박스로 보고 입력 대비 출력으로 시험하는 것이다. 이는 형식적 기술 언어(formal specification language)를 이용하여 설계가 시스템 요구사항을 만족한다는 것을 입증하고 이로부터 가능한 시험 시퀀스를 만들어내서 모니터를 이용하여 구현의 오류를 제거하는 것이다. 그러나 이러한 모델은 대부분의 시스템이 동작하는 논리적 상태가 폭발적으로 늘어나는(state space explosion) 문제를 안고 있으며 모델 확인이나 이론 증명 방법으로 설계의 정확도가 입증되었다고 하더라도 구현의 정확도를 확인하는데 제한된 시간 내에 모든 가능한 입력 시퀀스가 시험되었다고 보장하기 힘들다. 첫번째 연구 방향의 한계를 극복하기 위한 두 번째 방법으로 대상 시스템 내에 모니터를 위한 별도의 코드를 삽입하여 수행 시간에 상태를 확인하는 것이다. 이와 같은 모니터는 일반적으로 다음과 같은 목표를 추구한다.

- 최소 방해 작용 : 모니터의 대상이 되는 시스템 내에 적절히 코드를 삽입하여 모니터로 인하여 발생되는 방해 작용을 최소화한다.
- 최소 검출지연 시간 : 위반 사항이 발생하면 가급적 빠른 시간 내에 검출이 완료되도록 한다.
- 최소 비용 : 최소한의 추가 비용(모니터 수행시간, 모니터 코드 삽입 노력)으로 모니터 시스템 구축한다.

수행시간 모니터와 관련된 최근 연구 결과로 다음과 같은 예를 들 수 있다. [SAN93]에서는 Ada 프로그램 수행을 연속적으로 모니터하는 방법이 개발되었다 ANNA 라는 형식적 기술 언어를 이용하여 Ada 프로그램에 사용자가 주석을 추가하도록 하고 있다. [MOK97]은 RTL(Real-Time Logic)에 기반을 둔 기술 언어로 작성된 시간 제약 위반을 가능한 빠른 시간에 감시하는 방법을 제시하고 있다. 또한, [LEE98]에서는 원시 코드에 자동으로 모니터를 위한 코드를 삽입할 수 있도록 해주는 모니터링 스크립트와 요구사항을 표현하는 프레임워크와 언어를 제시하고 있다.

본 논문에서는 수행시간 모니터를 구현하기 위한 환경으로 실시간 객체 모델인 TMO(Time-triggered Message-triggered Object) 모델[KIM95]과 Windows 98/NT 상의 TMO 프로그램 실행 환경인 WTMOS(Windows TMO System)[KIM98]를 이용한다. TMO 모델을 기반으로 실시간성 관련 사건 중에서 모니터의 대상과 모니터 하는 조건을 묘사하는 방법을 제시한다 수행시간 모니터의 기능을 정의하고 이것을 TMO 모델 실행 환경인 WTMOS 에 맵핑시켜서 최소한의 추가 비용으로 모니터 시스템을 구축하는 방법에 대하여 논한다

2. TMO 프로그램 실행 환경

최근 TMO(Time-triggered Message-triggered Object) 모델이 실시간 시스템의 효율적인 개발 방법으로 소개된 바 있다[KIM95]. TMO 는 일반적은 객체 모델과 비교할 때 다음과 같은 주요특징을 갖는다.

- SpM(spontaneous method) : TMO 에는 기존의 객체 모델에서 볼 수 있는 서비스 메소드 외에 시간에 의해 구동되는(Time-triggered) 메소드(SpM)가 정의되어 있다. 일반 서비스 메소드(SvM)는 클라이언트로부터 서비스 요청 메시지에 의해 구동되는 반면에, SpM 은 실시간 클럭이 설계할 때에 정해진 시간에 이르면 자동적으로 구동된다. 따라서, SpM 이 구동되는 시간은 설계 시 상수 값으로 주어질 수 있다
- BCC(Basic Concurrency Constraint) : BCC 는 기본적으로 메시지에 의해 구동되는 SvM 의 활성화는 SpM 수행과 충동이 일어나지 않는 경우에만 가능하도록 규정하고 있다. 즉, 이 규칙 하에서 SvM 은 SpM 의 수행을 방해할 수 없다. 한 SvM 의 수행시간이 그 SvM 에 의해 접근되는 ODS(Object Data Store)와 같은 부분을 겹근하는 SpM 의 수행시간과 겹치게 되면 SpM 에게 우선권이 주어진다.

클라이언트 메소드(SpM's 또는 SvM's)는 다른 TMO 의 SvM 의 서비스를 요청하게 되는데, 동시성(concurrency)을 최대한 제공하기 위하여 비동기(non-blocking) 호출이 가능하다. 또한, 설계자는 각 SvM 과 SpM 에 의한 출력에 마감시간을 정할 수 있다.

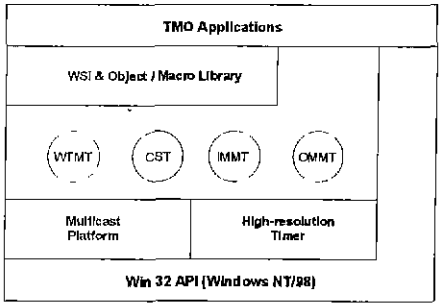
WTMOS[KIM98]는 위와 같은 TMO 의 주요 기능을 C++ 객체로 구현한 실시간 객체 네트워크의 수행 엔진으로서, Win32 API 위에서 개발된 미들웨어 플랫폼이다. WTMOS 가 실시간 객체 TMO 의 수행을 위하여 제공되는 기능은 다음과 같다.

- 최고 1msec 시간단위의 적시 실행
- 마감시간 감시에 의한 예외처리기의 구동
- SpM 및 SvM 의 최악(historic worst-case)수행시간과 마감시간여유(deadline-laxity)에 의한 우선순위 제어

[1] 본 연구는 한국과학재단 지원(과제번호 96-0101-07-01-3)에 의해 수행되었음.

- 윈도우 메시지 및 분산 IPC 메시지 (Hitachi Data Field)에 의한 SvM 구동
- 분산 환경에서의 노드간 클럭 동기화

이러한 기능을 지원하는 TMO 수행환경인 WTMOS 는 (그림 1)에 나타낸 바와 같은 계층 구조를 갖는다.



(그림 1) WTMOS(Windows TMO System) 계층 구조

(1) 제 1 계층 (Win 32 API)

TMO 객체 인스턴스의 SpM 과 SvM 은 윈도우 시스템의 스레드로 맵핑되어 그 수행이 관리된다. WTMOS 는 SpM 이나 SvM 에 대해 직접 컨텍스트 변환을 수행하지 않으며, 메소드들의 시간 조건을 충족시키기 위해 맵핑된 스레드의 우선순위를 제어하는 간접적인 방법을 사용한다. TMO 응용 계층에서는 모든 Win32 API 를 사용할 수 있어서 윈도우 관리, GUI 등을 사용자에게 제공할 수 있다.

(2) 제 2 계층 (Multicast Platform & High Resolution Timer)

1msec 정밀도의 선점(preemption)을 제공하는 Microsoft 'multimedia timer' 의 내부 클럭 핸들러로 제 3 계층의 시스템 태스크들을 주기적으로 구동하여 그들로 하여금 TMO 내의 SpM, SvM 의 적시 호출과 마감시간에 의한 우선순위 조정, 예외처리기 구동과 분산 노드간의 주기적 클럭 동기화, 분산 IPC 메시지 버퍼의 주기적 관리 등의 작업을 수행토록 한다 또한, 멀티캐스트 통신 계층으로 Microsoft 'mail-slot' 서비스를 기반으로 TMO 응용 계층에 분산 노드간의 메시지 교환과 버퍼링을 제공한다

(3) 제 3 계층 (System Task Layer)

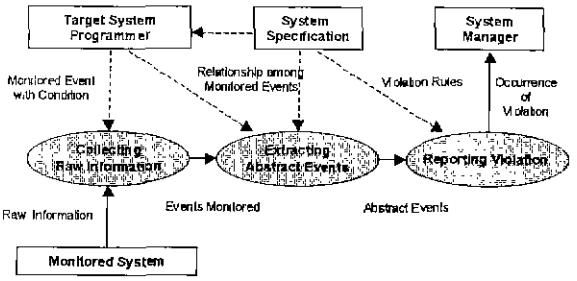
클럭 핸들러에 의해 주기적으로 구동되는 윈도우 시스템의 시간적으로 민감한(time-critical) 스레드 군으로 구성된다. CST(Clock Synchronization Thread)는 분산 환경하에서 각 노드의 클럭을 동기화한다. WTMT(Watchdog & TMO Management Thread)는 주기적으로 구동되어 구동 시간 조건에 의해 SpM 을 구동 시킨다 해당 SpM 을 구동시킨 후 노드 내에서 수행중인 모든 SpM 과 SvM 의 잔여 마감시간, 구동후 수행시간, 현재까지 최장 수행시간 등을 점검하여 우선순위를 제어한다. 마감시간이 경과한 메소드에 대해서는 사용자가 정의한 예외처리기가 구동될 수 있도록 한다. IMMT(Incoming Message Management Thread)와 OMMT(Outgoing Message Management Thread)는 메소드간 메시지 전달을 수행한다.

(4) 제 4 계층 (WTMOS Service Interface & Object / Macro Library)

WSI(WTMOS Service Interface) 및 Object/Macro Library 는 TMO 프로그래머에게 제공되는 TMO 관련 서비스 인터페이스들로 구성된다 주요 인터페이스로 메소드 관리 인터페이스, 분산 IPC 인터페이스, 윈도우 관리 인터페이스 등이 있다 다음 절에서 설명되는 모니터의 원시정보수집 기능이 이 계층에 구현되었다

3. 수행시간 모니터 구조 설계

수행시간 모니터란 시스템 수행 중에 시스템이 올바르게 동작하는지를 감시할 목적으로 필요한 정보를 입수하고 이 정보를 토대로 현 수행 상태가 시스템 요구 규격을 만족하는가를 판단하는 기능을 말한다 (그림 2)에 나타낸 바와 같이 모니터는 원시정보수집, 추상사건추출, 위반사항 보고 기능들로 구성된다. 그림에서 실선 화살표는 수행시간 중에 정보의 흐름을 나타낸 것이고 점선 화살표는 시스템 설계와 구현 단계에서 요구되는 정보를 표시한 것이다.



(그림 2) 수행시간 모니터 기능

(1) 원시 정보 수집

이 기능은 프로그램이 실행될 때 시스템 내부에서 발생되는 정보를 수집하는 기능이다 발생되는 모든 정보를 자동적으로 저장하는 경우, 짧은 시간동안에 엄청난 분량의 자료가 모이게 되어 메모리 공간과 통신 채널 지원을 소모하게 된다 따라서, 모니터 대상이 되는 프로그램에 프로그래머가 명시할 수 있도록 한다. 실시간 시스템 설계자는 설계 시에 많은 시간 중에서 모니터에 의해 감지되어야 하는 사건과 모니터하는 조건을 명시적으로 나타낼 수 있다. 여기서 시간이란 크게 두 가지로 구분된다. 하나는 특정 코드 블럭(메소드)의 실행이 시작되고 종료되는 것이고 다른 하나는 특정 변수 값이 변경되는 것이다 메소드 실행의 모니터 조건은 모니터할 시간 범위와 모니터하는 빈도(예를 들면, 특정 메소드기 매번 실행될 때 마다, 또는 n 번 실행될 때 한번씩)를 말한다. 메소드 실행 사건을 모니터하기 위한 구문은 다음과 같다.

```
if BOOLEAN_STATEMENT
Monitor(MethodName, StartTime , StopTime, Frequency);
```

여기서. MethodName 은 모니터 대상 메소드이고, StartTime 과 StopTime 은 모니터 시작과 종료 시간을 나타낸다. Frequency 는 모니터 빈도를 말한다

(2) 추상 사건 추출

원시 정보 수집 기능에 의해서 수집된 각 사건은 다음과 같은 형태로 추상 사건 추출 기능에 전달된다

```
EventID(MethodName, ExeStartTime, ExeStopTime, Count)
```

ExeStartTime 과 ExeStopTime 은 MethodName 메소드가 처음 시작부터 Count 번째 실행될 때의 시작 시간과 실행 완료 시간을 말한다.

개별적으로 수집된 각 사건으로부터 시스템의 시간 요구사항이 충족되었는가를 판별하기 위해서는 위와 같은 방법으로 명시된 사건들 간의 선후 관계나 포함 관계가 제공되어야 한다.

추상 사건이란 시간적 관련이 있는 사건들의 집합에서 한 시작사건으로부터 종료사건으로 연결되는 연속된 사건을 말한다. 예를 들어 사건 집합 {E1, E2, E3, E5}이 존재하고 각 사건이 E2-- E1- E3- E5 의 순서로 발생된다면 이것을 시작사건 E2. 종료사건 E5 로 하는 하나의 추상 사건으로 만들 수 있다. 이와 같은 추상사건을 추출하기 위해서는 모니터된 사건들간의 관계 정보가 제공되어야 한다.

(3) 위반사항 보고

위반사항 보고 기능은 시스템 요구 규격에 정해진 각 추상 사건의 허용 한도를 확인하고 이것이 위반되었을 경우 시스템 관리자에게 보고한다. 보고 방법을 사전에 정의할 수 있도록 한다.

4. 모니터 구현 결과

이 절에서는 3 절에서 설명된 모니터 기능 중에서 WTMOS 내에 구현된 원시정보수집 기능의 구현을 중심으로 설명한다. .

모니터의 원시정보수집기능은 2 절에서 설명된 WTMOS 의 제 4 계층인 WSI & Object / Macro Library 내에 구현되었다. 이 안에서 각 메소드를 제어하는데 필요한 정보와 실시간 클럭에 접근할 수 있다. 추상 사건 추출 기능은 각 로컬 노드의 지역 모니터 관리자 역할을 하게 되며 하나의 TMO 응용 객체로 구현되었다. 마지막으로 위반사항 보고 기능은 전체 분산 시스템을 관리하는 전역 모니터 관리자의 역할을 담당하게 된다.

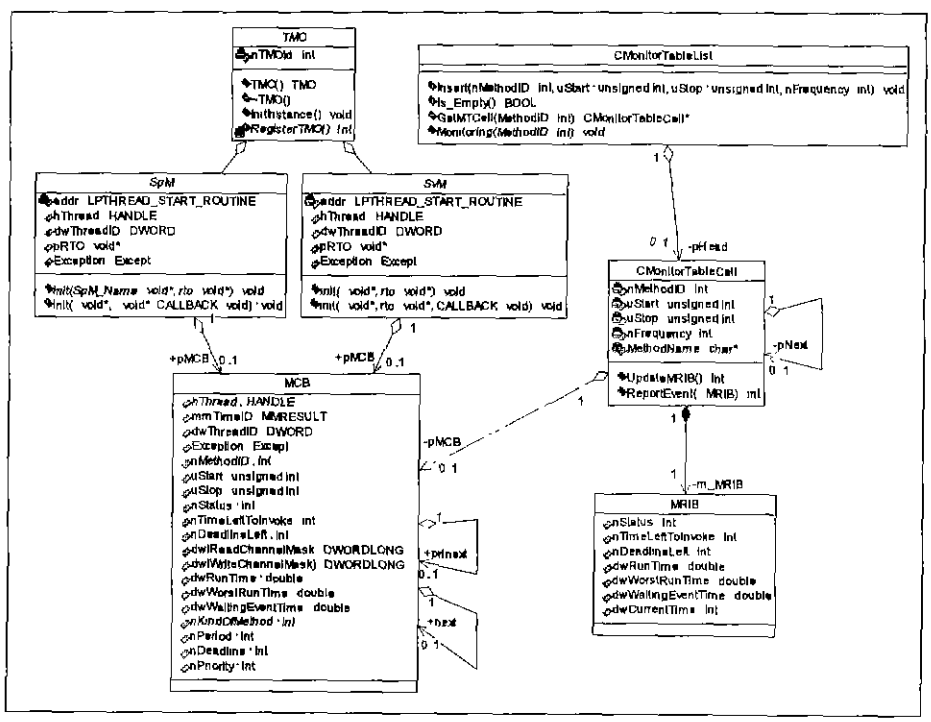
(그림 3)은 정보수집기능이 구현된 관련 클래스들의 관계를 나타낸 것으로 Rational 사의 Rose UML(Unified Modeling Language)도구를 이용하여 도출한 클래스 다이어그램의 일부이다 하나의 TMO 는 여러 SpM 과 SvM 으로 구성될 수 있고 *SpM(SpMName), SvM(SvMName)*과 같은 매크로를 통해 선언된다. 각 메소드는 수행에 필요한 정보(예를 들면, 메소드 상태, 마감시간, 최악 수행시간, 우선순위 등)를 MCB(Method Control Block)에 저장하고 각 MCB 와는 포인터로 연결된다 TMO 프로그램 구현 시 Monitor()라는 매크로를 통해 프로그래머가 모니터하기를 원하는 메소드와 모니터 조건을 명시해줌으로써 WTMOS 자체적으로 정보를 수집할 수 있도록 하였다. 모니터 조건은 모니터하기를 원하는 *TMO* 메소드와 모니터 시작시간과 종료시간, 빈도이다 *Monitor()* 매크로를 선언해 주면 CMonitorTableCell 이라는 클래스 내에 모니터할 메소드의 이름과 WTMOS 내에서 사용되는 메소드 ID, 모니터 조건이 주어진다. CMonitorTableList 는 모니터해야 할 전체 메소드의 목록을 갖고 있다. 하나의 메소드의 실행 상태를 모니터하려면 CMonitorTableCell 의 인스턴스를 만들고 연결리스트로 관리한다. 원시 정보는 *MCB* 를 참조하여 수집하는데 MCB 의 정보가 아주 짧은 시간 간격으로 계속적으로 변하기 때문에 *CMonitorTableCell* 에 명시된 모니터 조건과 일치할 경우에만 정보를 수집하여 MRIB(Monitored Raw Information Block)에 저장한다 Monitor() 매크로를 통해 모니터하기를 원하는 TMO 메소드를 선언해주면 CMonitorTableList 클래스 내의 Insert() 함수를 통해 해당 메소드를 포함하는 CMonitorTableCell 을 추가한다 WTMOS 는 *CMonitorTableList* 에 등록된 *CMonitorTableCell* 의 *MethodID* 를 보고 모니터할 메소드를 식별한다 만약 모니터할 대상이 없다면 CMonitorTableList 는 NULL 이다 이때 Is_Empty() 함수는 TRUE 를 반환할 것이다. GetMTCell() 함수를 통해 모니터할 메소드에 대한 모니터 조건과 모니터 결과인 MRIB 를 얻을 수 있다. Monitoring() 함수는 모니터하고자 하는 메소드를 실제로 모니터하는 것이다. 모니터 기능은 CMonitorTableCell 의 멤버 함수인 UpdateMRIB()에 구현되어 있다. *UpdateMRIB()* 메소드는 *CMonitorTableCell* 내에 있는 모니터 조건을 검사하여 조건과 일치하면 MRIB 정보를 갱신해 준다. 만약, 조건에서 벗어난다면 모니터하지 않으므로 MRIB 값은 변하지 않는다. MRIB 를 갱신한 후에는 그 정보를 추상사건 추출 기능이 구현된 모니터 관리자에게 보고한다 이를 수행하는 함수가 ReportEvent() 이다.

온도제어를 위한 실시간 시스템 응용 사례를 구현하여 실험한 결과 모든 SpM 메소드의 실행이 설계 시에 주어진 시간 제약을 만족하는 것을 확인할 수 있었다.

5. 결론

시스템 수행시간 중에 실시간성을 감시하는 모니터의 대상과 모니터 하는 조건을 명시하는 방법과 이를 효과적으로 구현한 예를 제시하였다 구현 환경으로 실시간 객체 모델인 TMO 와 Windows 98/NT 상의 TMO 프로그램 실행 환경인 WTMOS 를 이용하였다 본 논문에서 정의된 수행시간 모니터의 기능을 WTMOS 와 TMO 응용 객체에 맵핑시켜서 최소한의 추가 비용으로 모니터 시스템을 구축하였다.

향후에는 이 모니터의 성능을 보다 체계적으로 평가하기 위하여 수행 중에 인위적으로 결함을 삽입하는 방법과 최대 검출 지연시간에 대한 분석 연구를 진행할 것이다.



(그림 3) 모니터 클래스 다이어그램

참고문헌

[MOK97] A.K. Mok and G. Liu, "Efficient Run-Time Monitoring of *Timing Constraints*", Proc. Real-Time Technology and Applications, 1997, 6.

[LEE98] I. Lee, et al, "A Monitoring and Checking Framework for Run-time Correctness Assurance", Proc. Korea-US Technical Conf. on Strategic Technologies. 1998, 10.

[SCH95] B.A. Schroeder, "On-line Monitoring: A Tutorial", IEEE Computer, 1995, 6.

[SAN93] S. Sankar and M. Mandal. "Concurrent Runtime Monitoring of Formally Specified Programs", IEEE Computer, 1993 3.

[KIM98] J-G Kim, J P Hong, B-J Min, M H Kim, "Modeling of Multimedia Services using the TMO Model", Journal of Computer Systems Science and Engineering, 1998, 5.

[KIM95] K. Kim, et al, "A *Timeliness-Guaranteed* Kernal Model : DREAM Kernal and Implementation Techniques", *RTCSA 1995, 10.*