실시간 확장 윈도우 운영체제에서의 소프트 타이머 인터럽트 UML-RT 모델의 구현과 성능 분석

최진호° 심재익 임승철
국방과학연구소
{jhchoi93°, jishim, scyim}@add.re.kr

The Implementation and Performance Analysis of Soft Timer Interrupt UML-RT Model on a Windows Platform with Real-Time Extension

Jinho Choi° Jaeik Shim Seungcheol Yim
Agency for Defense Development

요 약

본 논문에서는 UML-RT(Real-Time) 모델에서 태스크를 직접 제어하기 위한 목적으로 일정 시간마다 틱을 발생하는 소프트 타이머 인터럽트 모델을 구현하였으며 생성된 모델 코드의 실행 성능 결과를 제시하였다. 그리고 소프트 타이머 인터럽트 모델 코드의 실시간 실행이 가능하도록 UML-RT 도구의 TargetRTS 라이브러리를 실시간 확장 윈도우 환경에 맞게 수정하였다. 실시간 확장 윈도우 운영체제와 UML-RT 개발 환경에서 설계, 구현한 소프트 타이머 인터럽트 모델의 성능 측정 결과 실시간으로 동작 가능함을 보여주었다.

1. 서 론

최근 UML(Unified Modeling Language)을 실시간 소프트웨어 개발에 적용시키기 위한 연구와 개발이 많이 이루어지고 IBM Rational Rose RealTime, I-Logix Rhapsody, Telelogic TAU Developer 등의 다양한 실시간 모델링 개발 도구들이 출시되고 있다[1][2]. 하지만 현재의 모델링 개발 도구는 실시간 소프트웨어 개발에서 가장 중요한 시간 제약성 문제를 명확히 해결하지 못하고 있으며 자동으로 생성된 실행 바이너리의 실시간 성능이 만족스럽지 못한 경우가 많다. 그리고 캡슐(Capsule) 기반 쓰레딩 매핑 구조를 가지는 UML-RT 모델은 캡슐 완전 실행 조건(Run-To-Completion)으로 인해 즉시 처리해야 할 이벤트가 지연될 가능성이 있다[3].

우리는 이러한 UML-RT 모델의 문제점을 극복하고 강성 실시간(Hard Real-Time) 제약성을 만족하기 위해 모델 내에서 태스크를 직접 제어할 수 있는 실시간 소프트웨어 구조를 설계하고자 한다. 본 논문은 우리의 선행 연구로써 UML-RT 도구인 Rational Rose RealTime(이하 Rose RT)을 이용해서 일정 시간마다 틱(Tick)을 발생하는 소프트 타이머 인터럽트 모델을 설계, 구현하였으며 윈도우 운영체제 환경에서 UML 코드의 실시간성을 분석하기 위해 Rose RT의 TargetRTS(Run Time System) 라이브러리를 실시간 확장 윈도우 환경으로 이식하였다. 그리고 생성된 모델 코드의 실시간 실행 성능을 측정하고 결과를 제시하였다.

본 논문의 구성은 다음과 같다. 2절에서 윈도우 환경하에서 실시간 실행 성능을 확보하기 위한 실시간 확장 윈도우 운영체제 환경에 대해 서술하고 3절에서 Ratioanl Rose RT의 기반 기술인 UML-RT에 대해 개관한다. 4절에서 구현된 타이머 인터럽트 모델의 구성에 대해 설명하고 5절에서는 범용 윈도우와 실시간 확장 윈도우 환경에서의 타이머 인터럽트 실행 모델의 응답 시간 측정 결과를 보이고 마지막으로 6절에서 결론을 맺는다.

2. 실시간 확장 윈도우 운영체제

하드웨어 프로세서의 성능이 매우 빨라졌음에도 불구하고 적은 범위의 쓰레드 우선 순위 수준과 인터럽트 처리시의 우선순위 역전 현상을 가진 범용 윈도우 운영체제의 특성상 소프트웨어의 실시간 실행 성능을 구하기는 어렵다[4]. 따라서 범용 윈도우 운영체제의 실시간 성능을 확장하기 위해 Ardence RTX(Real-Time Extension)를 사용했는데 RTX는 범용 윈도우 운영체제에 RTSS(Real-Time SubSystem)를 이식해서 강성 실시간 환경을 제공하는 시스템 소프트웨어로 정밀도가 높은 클럭(Clock) 서비스와 128개의 쓰레드 우선 순위 수준을 가지며 시간 결정성을 가진 Win32 호환 API 함수들을 제공한다[5][6]. 본 논문에서는 RTX가 이식된 윈도우 운영체제를 실시간 확장 윈도우 운영체제로 명명한다.

3. UML-RT

UML-RT는 기존 UML에 실시간 시스템에 필요한 개념을 확장한 것으로 가장 중요한 개념은 캡슐과 프로토콜(Protocol), 포트(Port)이다[7]. 캡슐은 active object라고도 하며 병행적으로 수행 가능한 태스크로 정의할 수 있는 가장 기본적인 모델 단위이다. 프로토콜은 캡슐들간의 교환되는 메시지 집합을 말하며 포트는 캡슐의 메시지 송수신을 전담한다. 따라서 각 캡슐 간의 정보 공유는 동일한 프로토콜을 가진 포트들간의 메시지 전달을 통해 이루어진다. 캡슐의 행위는 상태 다이어그램(State Diagram)을 통해 정의되며 클래스 다이어그램(Class Diagram)과 구조 다이어그램(Structure Diagram)을 통해 각 캡슐간의 관계를 정의한다. 이러한 UML-RT 개념을 이용한 실시간 소프트웨어 설계 모델링 도구인 Rose RT는 설계된 UML-RT 모델을

다양한 타겟 플랫폼의 모델 코드로 생성하기 위한 TargetRTS 라이브러리 코드를 개발자가 변경할 수 있도록 제공한다[8]. TargetRTS는 모델 코드 생성시에 필요한 타켓 플랫폼에 의존적인 타이머 서비스와 쓰레드 생성, 동기화 서비스 등이 정의된 라이브러리 코드이다. 우리는 실시간 확장 윈도우 운영체제에 TargetRTS를 이식하였으며 본 논문에서는 그 이식 과정에 대한 설명은 하지 않는다.

4. 소프트 타이머 인터럽트 모델

Rose RT의 C 프레임워크(Framework)를 이용해서 일정 시간마다 타이머 인터럽트를 발생할 수 있는 UML-RT 모델을 설계하였다. 그림 1은 설계, 구현한 타이머 인터럽트 모델의 클래스 다이어그램으로 타이머 인터럽트가 타이머 기능을 가진 TickGenerator 캡슐로 구성됨을 보인다. TickGenerator 캡슐의 구조 다이어그램에는 시간 서비스를 제공하는 타이머 포트를 정의하였고 상태 다이어그램은 일정 시간마다 틱이 발생하도록 설계하였다.

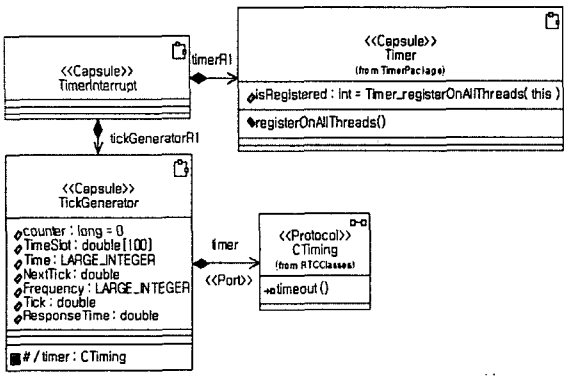

그림 1. 타이머 인터럽트 모델의 클래스 다이어그램

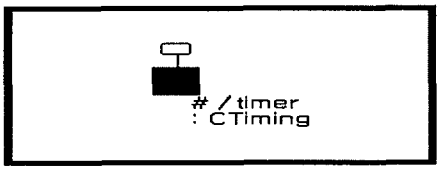

그림 2. TickGenerator 캡슐의 구조 다이어그램

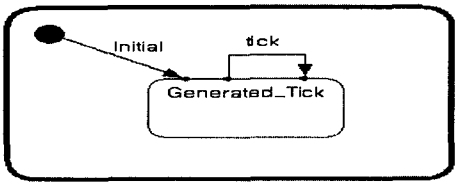

그림 3. TickGenerator 캡슐의 상태 다이어그램

5. 소프트 타이머 인터럽트 모델의 성능 측정

5.1. 실험 방법

실험은 2.0Ghz Pentium 4 프로세서의 윈도우 XP Professional Service Pack 2 운영체제에서 이루어졌으며 범용 윈도우 운영체제 환경과 실시간 확장 윈도우 운영체제 환경에서 소프트 타이머 인터럽트 모델을 실행하였다. 틱 발생 시간 성능을 계산하기 위해 초기 전이(initial transition) 시에 고해상도 타이머(High-Resolution Timer)의 초당 카운터 값을 측정하였고 틱 발생 때마다 고해상도 타이머 카운터 값을 메모리에 저장한 다음 100 번째 틱 발생 시점에서 타이머 인터럽트 모델의 평균 응답 시간을 계산하였다. 응답 시간은 한 틱 발생에서 다음 틱 발생까지의 시간을 의미한다. 1 초, 100 밀리초, 10 밀리초, 1 밀리초 타이머 인터럽트에 대해 응답 시간을 측정하였고 모델 코드 내에서는 쓰레드 우선 순위 변경작업을 하지 않았으며 Rose RT 도구에서 생성된 실행 바이너리 코드를 그대로 사용하였다.

5.2. 범용 윈도우 운영체제에서의 실험 결과

실시간 기능을 확장하지 않은 범용 윈도우 운영체제 환경에서 타이머 인터럽트 모델을 실행하였다. 표 1은 각 타이머 인터럽트 발생에 대한 평균 응답 시간을 보여준다. 1 초 틱을 제외하고는 설정된 틱에 맞는 응답 시간이 측정되지 않았는데 이것은 범용 윈도우 운영체제 환경의 Rose RT TargetRTS 라이브러리에서 사용된 시간 함수의 시간 측정 정밀도가 낮기 때문이다.

표 1. 타이머 인터럽트의 평균 응답시간 (범용 윈도우)

Tick (ms)	평균 응답시간 (ms)
1000	1000.037088
100	109.378966
10	15.625571
1	15.625404

그림 4는 1 초 틱에서의 측정횟수에 대한 응답 시간 분포도이다. 최대 응답 시간은 1.00011454 초이고 최소 응답 시간은 0.999932114 초로 시간 편차가 크지 않았으나 1 초 미만의 응답 시간을 가진 경우가 전체 측정의 15%를 차지하므로 범용 윈도우 운영체제 환경에서의 소프트 타이머 인터럽트 모델의 실시간 실행은 어렵다.

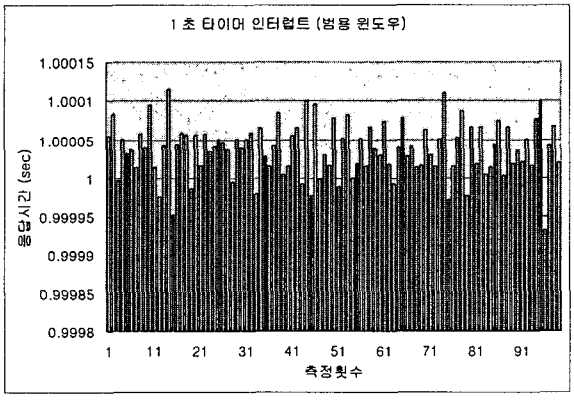

그림 4. 1초 타이머 인터럽트의 응답시간

10 밀리초 틱의 경우는 최대 응답 시간은 17.599723 밀리초이고 최소 응답 시간은 13.667659 밀리초로 측정되었고 설정된 10 밀리초와는 큰 오차를 보여주었다. 실험 결과를 통해 범용 윈도우 운영체제 환경에서는 모델 코드의 실시간 실행이 불가능

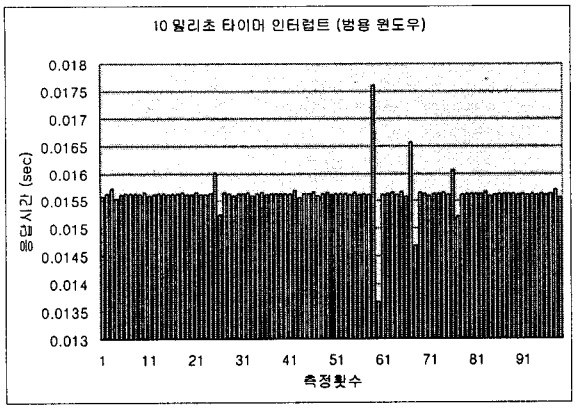

그림 5. 10 밀리초 타이머 인터럽트의 응답시간

함을 알 수 있다.

5.3. 실시간 확장 윈도우 운영체제에서의 실험 결과

실시간 확장 윈도우 운영체제 환경에 Rose RT의 TargetRTS 라이브러리를 이식한 다음 소프트 타이머 인터럽트 모델의 응답 시간을 측정하였다. 표 2는 각 시간 틱 발생에 대한 평균 응답 시간인데 측정 결과는 각 시간 설정에 맞게 소프트 타이머 인터럽트 모델이 동작하였음을 보여준다.

표 2. 타이머 인터럽트의 평균 응답시간 (실시간 확장 윈도우)

Tick (ms)	평균 응답시간 (ms)
1000	1000.039168
100	100.041886
10	10.037899
1	1.034187

그림 6은 1 초 틱의 응답 시간으로 1 초 미만의 응답 시간은 전혀 발생되지 않았으며 최대 응답 시간은 1.000055314 초이고 최소 응답 시간은 1.000031568 초로 시간 편차가 크지 않았다. 10 밀리초 틱 모델을 실행할 경우에는 최대 응답 시간은 10.049043 밀리초이고 최소 응답 시간은 10.031163 밀리초로

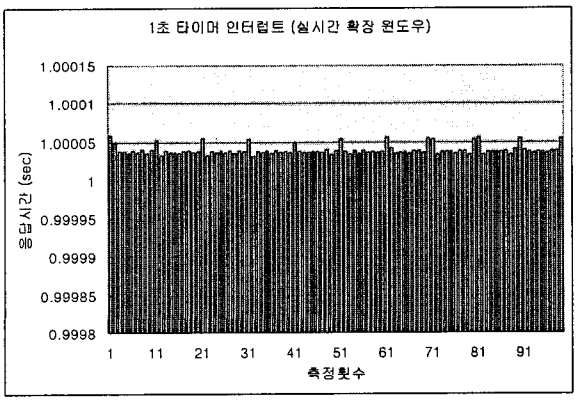

그림 6. 1 초 타이머 인터럽트의 응답시간

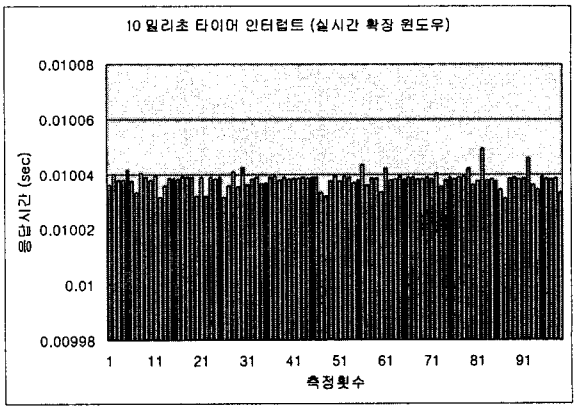

그림 7. 10 밀리초 타이머 인터럽트의 응답시간

측정되었다. 1 밀리초 미만의 시간 오차는 Rose RT TargetRTS 라이브러리에서 밀리초 단위로 요구 시간을 처리하기 때문에 발생하는 문제이다. 따라서 실시간 확장 윈도우 운영체제 환경에서는 소프트 타이머 인터럽트 모델 코드의 실시간 실행이 가능함을 확인할 수 있다.

6. 결론

본 논문에서는 모델 내에서 태스크를 직접 제어하기 위한 선행 연구로 소프트 타이머 인터럽트 모델을 설계하였으며 윈도우 운영체제 환경에서 UML-RT 모델 코드를 실시간으로 실행하기 위한 방안을 제시하고 타이머 인터럽트 모델의 실행 바이너리를 이용해서 응답 시간을 측정하였다. 실험 결과는 실시간 확장 윈도우 운영체제 환경에서 모델 코드를 생성할 경우에 윈도우 환경에서도 모델의 실시간 실행 분석이 가능함을 보여준다.

향후 과제로 소프트 타이머 인터럽트 모델의 기능을 확장한 실시간 태스크 제어 연구와 실제 임베디드 타겟 환경에서 모델의 실시간 실행 연구가 이루어질 것이다.

7. 참고 문헌

[1] J. Masse, S. Kim, S. Hong. Tool Set Implementation for Scenario-based Multithreading of UML-RT Models and Experimental Validation, In Proc. IEEE Real-Time/Embedded Technology and Applications Symposium, pp. 70-77, 2003.

[2] L. A. J. Dohmen, L. J. Somers, Experience and Lessons Learned Using UML-RT to Develop Embedded Printer Software, In Proc. PROFES 2002, LNCS 2559, Springer-Verlag, pp. 475-484, 2003.

[3] M. Saksena, P. Freedman, P. Rodziewicz. Guidelines for Automated Implementation of Executable Object Oriented Models for Real-Time Embedded Control System, In Proc. IEEE Real-Time Systems Symposium, pp. 240-251, Dec. 1997.

[4] Venturcom. Hard Real-Time with Venturcom RTX on Microsoft Windows XP and Windows XP Embedded, 2003.

[5] Ardence. RTX 6.1 SDK Documentation, 2005.

[6] B. Carpenter, M. Roman, N. Vasilatos and M. Zimmerman. The RTX Real-Time Subsystem for Windows NT. In Proc. the USENIX Windows NT Workshop, 1997.

[7] B. Selic, J. Rumbaugh. Using UML for Modeling Complex Real-Time Systems. White paper, IBM, 1998.

[8] IBM, Adapting Rational Rose RealTime for Target Environments version 2003.06.00, 2003.