임베디드 시스템을 위한 동적 업그레이드 프레임워크에 관한 연구

경주현O 이민석
한성대학교 컴퓨터공학과
leaders3@naver.com, minsuk@hansung.ac.kr

Dynamic Software Upgrade Framework for Embedded Systems

JuHyun KyungO Minsuk Lee
Dept. of Computer Engineering, Hansung University

요 약

시스템의 중단 없는 서비스를 요구하는 운영체제 커널과 응용 프로그램은 빈번한 기능 추가와 성능 향상 그리고 버그 수정이 필요하다. 현재 이러한 시스템은 업그레이드를 한다고 해도 프로그램을 종료한 후 수행하거나 시스템을 재시작하는 과정에서 시스템 중단 및 재부팅에 따른 비용이 발생한다. 특히 임베디드 시스템의 경우 운영체제 또는 응용프로그램의 재설치는 쉽지 않다. 이러한 임베디드 시스템에서의 동적 업그레이드는 일반 PC의 경우와는 달리 플랫폼에 종속적인 부분이 상당히 많다. 플랫폼 종속적인 부분들은 차후 동적 업그레이드 기술을 다른 플랫폼으로 이식할 때 상당히 많은 시간과 노력이 필요하다. 본 연구에서는 이러한 문제점을 해결하고자 임베디드 시스템을 위한 동적 업그레이드에 대한 프레임워크를 설계하였다.

1. 서 론

모든 임베디드 시스템에서 소프트웨어의 규모가 커짐에 따라, 이미 현장에서 작동 중인 임베디드 시스템에서 오류의 발생 가능성이 높아지고 있으며, 지속적으로 기능 및 성능개선 요구가 있다. 임베디드 시스템에서 운영체제와 응용프로그램의 업그레이드는 버그수정, 성능향상, 기능추가 등의 이유로 빈번히 발생한다. 하지만 불행하게도 임베디드 시스템에서는 운영체제와 응용프로그램이 플래쉬 메모리 등에 내장되어 설치된 상태에서 제품으로 판매되기 때문에 새로운 버전의 운영체제 또는 응용프로그램의 재설치가 쉽지 않으며 제품 판매 후 업그레이드를 위해서는 많은 시간적 노력과 비용이 필요하다.

현재의 임베디드 소프트웨어 업그레이드 작업은 유무선 네트워크나 메모리 카드를 통해 새로운 버전의 소프트웨어를 대상 임베디드 시스템으로 다운로드하고 시스템을 재부팅시키는 방법으로 업그레이드 작업이 수행된다. 이 경우 시스템의 재부팅으로 인해 작업의 연속성이 끊겨 서비스의 일시 공급중단이라는 문제가 야기됨과 동시에 시간적, 비용적 낭비의 문제가 발생한다.

특히 서비스의 상시성이 요구되며, 사용자가 업그레이드와 관련된 기술적인 절차를 수행하기 어려운 환경에서는 소프트웨어 업그레이드를 최소한의 네트워크 트래픽으로 사용자가 알지 못하는 사이에 시스템 중단 없이, 즉 재부팅 없이 시스템이 서비스를 계속 진행하면서 수행하여야 한다.

또 시스템의 중단이 인명 및 상당한 비용 손실을 일으킬 수 있는 환경인 인공위성, 군용 임베디드 시스템, 보건의료 시스템, 원자력 발전소, 통신 장비, 항공 트래픽 관제시스템, 생산라인 로봇, 등에서 시스템의 중단 없이 소프트웨어 업그레이드를 수행하는 것은 매우 중요하다. 따라서 이를 안전하고 효율적으로 처리할 수 있는 프레임워크 및 도구가 절실히 필요하다

본 연구에서 수행할 임베디드 시스템을 위한 동적 업그레이드 프레임워크에 대한 연구는 일반 PC의 경우와는 달리 플랫폼에 종속적인 부분이 상당히 많다. 이러한 플랫폼 종속적인 부분들은 차후 동적 업그레이드 기술을 다른 플랫폼으로 이식할 때 상당히 많은 시간과 노력이 필요하다. 본 연구에서는 이러한 문제점을 해결하고자 임베디드 시스템을 위한 동적 업그레이드에 대한 프레임워크를 설계 동적 업그레이드의 프레임워크를 설계하였다.

본 논문의 구성은 다음과 같다. 2장에서는 연구 배경으로 임베디드 시스템에서의 동적 업그레이드에 대한 연구 목적을 기술하였고, 3장에서는 우리가 제안한 임베디드 시스템을 위한 동적 업그레이드 프레임워크(dynamic upgrade framework)를 설명한다. 4장에서는 요약과 향후 연구방향에 대해 설명한다.

2. 연구 목적

2.1 연구 배경

2.1.1. 동적 업그레이드 기법

시스템의 중단 없이 동적 업그레이드 시스템을 구성하기 위한 기법은 함수단위로 명령어 중간에 분기 명령어를 삽입한 후 코드 패치(code patch)로 분기하는 방식을 사용한다. 그림 1은 이러한 동적 업그레이드 기법 중 분기 명령어 대신 트랩 기반 동적 업그레이드를 나타낸다. 업그레이드 전 함수의 코드에 트랩 명령어를 삽입 후 트랩 핸들러에서 업그레이드 함수로 분기하는 방법으로 동적 업그레이드를 수행한다.

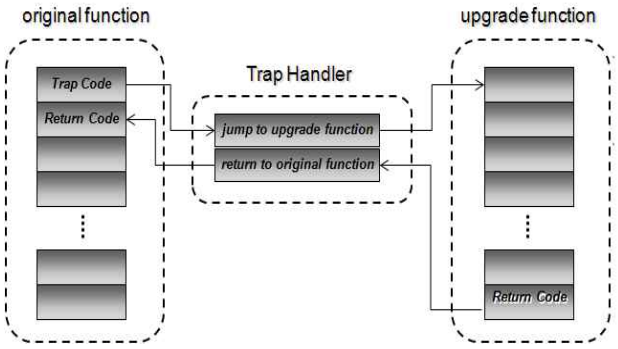

그림 1 트랩 기반 동적 업그레이드 기법

2.2. 기존 연구

2.2.1. 정적 소프트웨어 업그레이드

2차 저장소에 있는 정적인 실행 이미지를 수정하여 업그레이드를 수행하는 방식으로 직접 실행 이미지를 분석하여 이미지에 점프 코드 삽입하는 방식[1]과 기존 실행 이미지와 업그레이드 할 실행 이미지의 차이점만 수정하여 업그레이드를 하는 방식[2]이 있다. 이러한 방법은 동적 업그레이드가 아니라 정적 업그레이드 즉 시스템을 중단 한 후 재시작 해야 하므로 문제가 있다.

2.2.2. 동적 응용프로그램 업그레이드

Hicks[3]가 제안한 응용프로그램의 동적 업그레이드에 대한 연구는 컴파일 단계에서 코드를 업그레이드 가능하게 수정한 후 배포하는 방식으로 구성되어 있다. 이 연구는 유저 영역에서의 업그레이드를 고려한 것이므로 실제 커널 함수를 직접 수정해야 하는 경우에는 적합하지 않다. 또한 컴파일 단계에서 업그레이드 가능하게 코드를 수정하므로 컴파일러를 수정해야 하고 함수 포인터의 사용에 따르는 메모리 낭비, 오버헤드로 인해 실제 임베디드 운영체제에서 적용하기 힘들다.

2.2.3. 동적 업그레이드를 위한 새로운 운영체제 개발

이 방법은 운영체제를 설계단계부터 확장성을 고려하여 제작한 방법이다[6]. 대표적인 연구들로는 K42[4], SPIN[5]등의 연구가 있다. 이러한 연구는 기존에 검증되어온 운영체제를 사용하는 것이 아니라 새로운 운영체제를 제안함으로써 실제 사용하기는 힘들다[6][7].

2.2.4. 기존 커널에 확장성을 부여하는 연구

새로운 운영체제를 제안하는 문제점을 해결하기 위하여 기존 커널에 대해 확장성을 부여하는 연구이다. 대표적인 연구들로는 KernInst[8], DynAMOS[7], 트랩 기반 업그레이드[6]와 가상머신을 활용한 방법[9]이 있다. 이러한 연구는 리눅스와 같은 범용 운영체제를 대상으로 동적 업그레이드를 수행하는 서버 환경에서의 동적 업그레이드에 관한 연구이므로 임베디드 시스템에 적용하기는 한계가 있다. 또한 2차 저장소의 이미지를 수정하는 기술인 FOTA (Firmware Over The Air), OMA-DM (Open Mobile Alliance for Device Management) 등의 주변 기술과 연계가 힘들다.

2.3. 연구의 요구 사항

임베디드 시스템 안에 내장되어 있는 운영체제 또는 그 위에 동작하는 응용프로그램을 동적으로 업그레이드하기 위해서는, 업그레이드와 관련된 모든 시스템, 개발 도구에 대하여 다음과 같은 요구 사항들을 만족하여야 한다.

첫째, 플랫폼 독립적인 동적 업그레이드 프레임워크를 제공해야 하며 운영체제 전체에 대한 수정 없이 적용할 수 있는 동적 업그레이드 시스템이 되어야 한다. 또한 업그레이드 과정에서의 보안 문제 해결을 위한 다양한 기술, FOTA, OMA-DM등 주변 기술과 연계될 수 있는 확장성을 가져야 한다.
둘째, 사용자가 쉽게 접근할 수 있도록 동적 업그레이드 서비스를 제공해야 한다. 즉 동적 업그레이드가 가능한 소프트웨어를 구현하는 경우에도 개발 프로세스 상의 변화가 거의 없도록 하여야 한다.
셋째, 동적 업그레이드를 수행하기 전에 현재 업그레이드가 가능한 상태인지 체크를 할 수 있어야 하고 만약 잘못된 업그레이드를 통해 시스템에 문제가 발생할 가능성이 있을 경우 업그레이드 수행하기 전의 시스템 상태로 복원할 수 있어야 하는 유연성을 가져야 한다.
넷째, 어떠한 업그레이드를 수행하더라도 업그레이드된 코드로 인한 시스템 오버헤드를 줄여야 한다.
다섯째, 동적 업그레이드를 위한 고려한 개발을 위해, 또, 소프트웨어 형상과 동적 업그레이드를 위한 이미지의 관계를 효과적으로 관리하기 위하여, 기존의 통합 개발 환경에 동적 소프트웨어 업그레이드 프레임워크가 잘 융합되어야 한다.

3. 동적 업그레이드 프레임워크 설계

그림 2은 동적 업그레이드 프레임워크를 나타낸다.

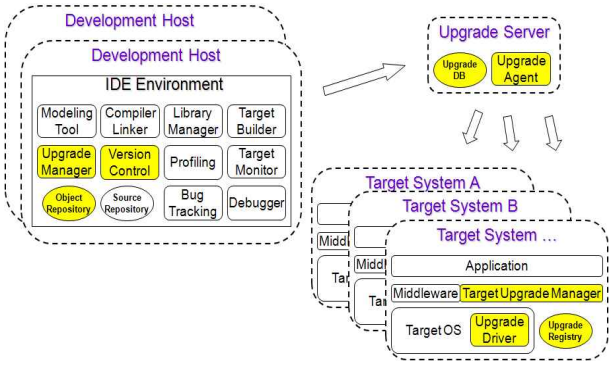

그림 2 동적 업그레이드 프레임워크

• 개발 호스트 (Development Host)

동적 업그레이드 시스템을 구성하기 위한 개발 호스트는 기존 통합 개발 환경에 추가로 업그레이드 관리자, 버전 관리자, 오브젝트 저장소를 가지고 있다. 개발 호스트는 빠른 업그레이드 수행을 위해 최소화된 업그레이드 이미지를 생성하는 일을 하며 생성된 업그레이드 이미지를 업그레이드 서버에 전송하는 일을 한다.

• 업그레이드 서버 (Upgrade Server)

업그레이드 서버는 업그레이드 데이터베이스와 업그레이드 에이전트를 가지고 있다. 개발 호스트에서 받은 업그레이드 이미지를 버전 별로 관리하고 저장하는 역할을 하며 타겟 시스템과 통신을 하여 업그레이드 이미지를 암호화한 후 전송하는 일을 한다.

• 타겟 시스템 (Target System)

타겟 시스템은 타겟 업그레이드 관리자와 업그레이드 드라이버 그리고 업그레이드 저장소를 가지고 있다. 실제 업그레이드를 수행하는 역할을 하며 업그레이드 서버로부터 받은 업그레이드 이미지를 복호화하여 동적으로 업그레이드를 수행하는 일을 한다.

3.1. 개발 호스트

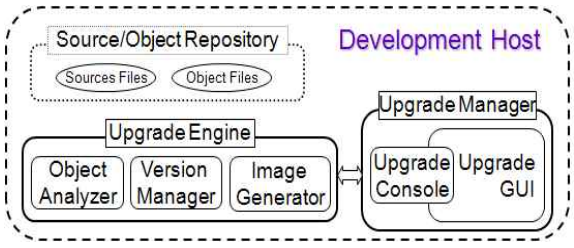

그림 3 개발 호스트 모듈 구성도

• 업그레이드 관리자 (Upgrade Manager)

업그레이드 GUI (Graphical User Interface) 모듈은 사용자로부터 업그레이드 명령을 입력으로 받아 명령어에 따라 업그레이드 콘솔(upgrade console)에 전달하게 되고 업그레이드 콘솔은 입력 받은 명령어에 따라 업그레이드 엔진에게 보고하는 역할을 한다.

• 업그레이드 엔진 (Upgrade Engine)

업그레이드 엔진은 실제 동적 업그레이드 이미지를 생성하는 모듈이며 오브젝트 분석기(object analyzer)와 버전 관리자(version manager) 이미지 생성기(image generator)를 가지고 있다. 오브젝트 분석기는 업그레이드 이미지를 생성할 때 필요한 심볼 정보를 얻기 위해서 필요하다. 이미지 생성기는 버전 관리자로부터 버전 정보를 요청하고 버전에 따르는 동적 업그레이드 이미지를 생성한다.

• 소스/오브젝트 저장소(Source/Object Repository)

소스/오브젝트 저장소는 업그레이드를 수행하기 위한 정보를 얻기 위해 존재한다. 각 오브젝트에 존재하는 심볼 정보 등을 활용하여 업그레이드 이미지를 작성할 때 필요한 심볼 주소 등 기타 필요한 정보를 제공한다.

3.2. 업그레이드 서버

그림 4 업그레이드 서버 모듈 구성도

• 업그레이드 데이터베이스 (Upgrade Database)

업그레이드 데이터베이스에는 플랫폼 정보와 동적 업그레이드 이미지를 가지고 있다. 업그레이드 에이전트의 요청에 따라 해당 업그레이드 이미지를 반환하는 역할을 한다.

• 업그레이드 에이전트 (Upgrade Agent)

업그레이드 에이전트는 실제 업그레이드 이미지를 전송하는 전송 관리자(Delivery Manager)와 보안 관리자(Security Manager)를 가지고 있다. 전송 관리자는 타겟 시스템에 요청 받거나 직접 업그레이드 명령을 내리는 역할을 하며 유무선 환경에서 업그레이드 이미지를 전송하는 역할을 한다. 보안 관리자는 업그레이드 이미지를 암호화하는 역할을 한다.

3.3. 타겟 시스템

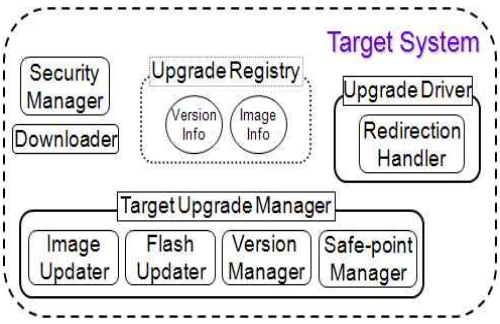

그림 5 타겟 시스템 모듈 구성도

• 보안 관리자 (Security Manager)

보안 관리자는 암호화된 업그레이드 이미지를 복호화 하는 역할과 타겟 시스템을 인증하는 역할을 한다.

• 다운로더 (Downloader)

다운로더는 업그레이드 서버의 에이전트로부터 업그레이드 이미지를 받은 후 보안 관리자로부터 복호화를 수행한 다음 타겟 업그레이드 관리자로 전송하는 역할을 한다.

• 업그레이드 등록소 (Upgrade Registry)

업그레이드 등록소는 업그레이드 이미지의 버전과 이미지 정보를 저장하는 모듈이다. 차후 잘못된 업그레이드가 발생할 가능성이 있으면 다시 예전 버전으로 복구하는 일과 업그레이드 할 때 필요한 정보들을 보관하는 일을 한다.

• 타겟 업그레이드 관리자(Target Upgrade Manager)

실제 타겟에서 업그레이드를 수행 하는 모듈이다. 구성 모듈로서는 이미지 업데이터 (image updater), 플래쉬 업데이터 (flash updater), 버전 관리자 (version manager), 세이프 포인트 관리자 (safe-point manager)로 구성되어 있다. 이미지 업데이터 모듈은 현재 수행 중인 프로그램을 동적으로 업그레이드 수행하는 일을 하고 플래쉬 메모리 업데이터 모듈은 시스템의 재시작 후에도 업그레이드가 지속적으로 유지하기 위하여 플래쉬 이미지를 바꾸는 역할을 한다. 버전 관리자는 올바른 업그레이드를 위하여 버전을 관리하는 모듈이고 마지막으로 세이프 포인트 관리자는 업그레이드가 현재 시스템의 상태를 파악하여 안전한 업그레이드를 수행하기 위한 모든 일을 관리한다.

• 업그레이드 드라이버 (Upgrade Driver)

업그레이드 드라이버 모듈은 MMU(Memory Management Unit)가 동작하는 CPU일 경우 커널 영역에 존재하며 운영체제 커널 함수를 수정할 때 커널 영역으로 진입하기 위하여 필요하다. 업그레이드 관리자는 동적 업그레이드 대상이 커널 함수일 경우 업그레이드 드라이버를 통해 커널 함수의 업그레이드를 수행한다. 또한 응용프로그램을 동적으로 업그레이드 할 때도 커널 자원을 사용하기 때문에 업그레이드 드라이버는 필요하다. 업그레이드 드라이버의 리다이렉션 핸들러(redirection handler)는 이전 버전의 함수에서 업그레이드된 함수의 실제 주소로 분기할 때 사용되는 모듈이다.

4. 결론 및 향후 연구 방향

모든 임베디드 시스템에서 소프트웨어의 규모가 커짐에 따라, 이미 현장에서 작동 중인 임베디드 시스템에서 오류의 발생 가능성이 높아지고 있으며, 지속적으로 기능 및 성능 개선 요구가 있다. 특히 인공위성, 군용 임베디드 시스템, 보건의료 시스템, 원자력 발전소, 통신 장비, 항공 트래픽 관제, 생산 라인 상의 로봇 등 시스템의 중단이 인명 및 상당한 비용 손실을 일으킬 수 있는 환경에서 시스템의 중단 없이 소프트웨어 업그레이드를 수행하는 것은 중요하다. 따라서 이를 기술적으로 안전하고 효율적으로 처리할 수 있는 동적 업그레이드 프레임워크에 대한 연구는 매우 중요하다.

우리는 이러한 동적 업그레이드의 프레임워크를 설계하였으며, 이클립스와 같은 통합 개발 환경에 개발 호스트 측 소프트웨어 모듈을 통합하고, 플랫폼 독립모듈과 종속모듈을 구현하여 프로토타입 시스템에 구현 중에 있다. 이 모든 것을 임베디드화 된 범용 운영체제와 상용 실시간 운영체제 센서네트워크에 사용되는 운영체제를 대상으로 우리가 만든 프레임워크를 적용하여 임베디드 시스템에서 사용되는 모든 운영체제에 적용할 수 있는 플랫폼 독립적인 동적 업그레이드 프레임워크를 구성해 나갈 것이다.

참고문헌

[1] Srivastava, A., and Eustace, A. ATOM: A System for Building Customized Program Analysis Tools. In ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation (PLDI) (June 1994), ACM SIGPLAN.

[2] J. Koshy and R. Pandey. Remote Incremental Linking for Energy-Efficient Reprogramming of Sensor Networks. In Proceedings. of the European Workshop on Sensor Networks, pages 354-365, Istanbul, Turkey, January 2005.

[3] Iulian Neamtiu, Michael W. Hicks, Gareth Stoyle, Manuel Oriol. Practical dynamic software updating for C. In Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI), June 2006

[4] Andrew Baumann et al., Module Hot-Swapping for Dynamic Update and Reconfiguration in K42, LCA 2005

[5] B. Bershad et al., Extensibility, Safety and Performance in the SPIN Operating System, In Proceedings of the 15th ACM SOSP, pp. 267-284, 1995

[6] 박현찬, 김세원, 유혁, 리눅스 환경에서의 함수 단위 동적 커널 업데이트 시스템의 설계와 구현 한국정보과학회 2007 한국컴퓨터종합학술대회 논문집 제34권 제1호(A), pp. 244 ~ 245 (2pages), 2007,

[7] Kristis Makris and Kyung Dong Ryu, Dynamic and Adaptive Updates of Non-Quiescent Subsystems in Commodity Operating SystemKernels. EuroSys 2007, March 2007

[8] Ariel Tamches and Barton P.Miller, Fine-grained dynamic instrumentation of commodity operating system kernels, PhD thesis, University of Wisconsin, 2001

[9] Haibo Chen, Rong Chen, Fengzhe Zhang, Binyu Zang and Pen-Chung Yew, Live. updating operating systems using virtualization. In Proc. VEE, pages 35-44, Ottawa, Canada, June 2006