컴포넌트 기반 소프트웨어 플랫폼을 위한 컴포넌트 관리자 설계 및 구현

이수원°, 유인선, 유용덕, 최훈
충남대학교 컴퓨터공학과
{swlee°, isyou, yyd7724, hc}@cnu.ac.kr

Design and Implementation of the Component Manager for Component based Software Platform

Su Won Lee°, In Seon Yoo, Yong Duck Yoo, Hoon Choi
Mobile Distributed Computing Lab, Department of Computer Engineering,
Chungnam National University, KOREA

요 약

소형 모바일 디바이스는 일반적으로 배터리로 동작하는 등 전력제한적이며 시스템 메모리 및 자원 제약적이다. 또한 소형 모바일 디바이스에서 실행되는 응용프로그램은 일반적으로 시스템의 일부 기능만을 이용하여 수행된다. 따라서 본 연구에서는 시스템에서 제공하는 서비스를 각각의 컴포넌트로 구현하고 응용프로그램의 요구에 따라 소프트웨어 플랫폼을 동적으로 재구성할 수 있는 소프트웨어 플랫폼을 설계 및 구현하였다. 구현한 플랫폼을 웨어러블 컴퓨터 보드에 포팅하여 정상 동작함을 확인하였다.

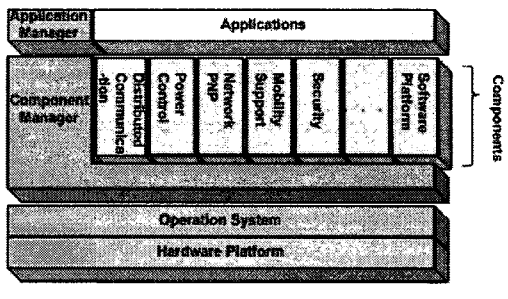
1. 서 론

최근 하드웨어 및 통신 기술이 발전함에 따라 휴대 가능한 소형 임베디드 디바이스(embedded device)로서 웨어러블 컴퓨터가 부각되고 있다. 웨어러블 컴퓨터는 몸에 착용하는 각종 물건에 내장되며, 배터리(battery)로 모든 시스템의 구성 요소가 동작하며, 시스템의 메모리 또한 매우 제한적이다. 따라서 자원제약적이고 전력제한적인 환경에 적합한 소프트웨어 플랫폼에 대한 요구가 급속히 증가하고 있다.

일반적으로 웨어러블 컴퓨터와 같은 소형 임베디드 디바이스에서 실행되는 응용프로그램은 실행 시, 시스템의 모든 기능이 아닌 일부 기능만을 이용하여 수행된다. 따라서 본 연구에서는 시스템에서 제공하는 서비스를 각각의 컴포넌트로 구현하고 응용프로그램의 요구에 따라 소프트웨어 플랫폼을 동적으로 재구성할 수 있는 소프트웨어 플랫폼을 설계 및 구현하였다([그림 1]).

구현한 소프트웨어 플랫폼은 시스템이 제공하는 서비스에 따라 프로세스간 통신 지원을 위한 통신 미들웨어(Communication Middleware) 컴포넌트, 디바이스 프로파일을 생성 및 관리하는 프로파일 관리(Profile Management) 컴포넌트, 단말의 이동성을 지원하는 이동성 지원(Mobility Support) 컴포넌트, 동적 BAN(Body Area Network)을 구성하는 네트워크 PNP(Plug and Play) 컴포넌트 등의 여러 컴포넌트들과 이들 컴포넌트를 플랫폼에 동적으로 재구성할 수 있는 컴포넌트 관리자(Component Manager)로 구성된다.

본 논문에서는 구현한 소프트웨어 플랫폼에서 응응프로그램 및 다른 컴포넌트의 요구에 따라 플랫폼을 동적으로 재구성하는 컴포넌트 관리자에 대하여 기술하며, 한국전자통신연구원이 개발한 WPS(Wearable Personal Station) 보드와 테스트 응용프로그램을 이용하여 컴포넌트 관리자의 기능을 테스트하였다.



[그림 1] 컴포넌트 기반 소프트웨어 플랫폼

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 개발한

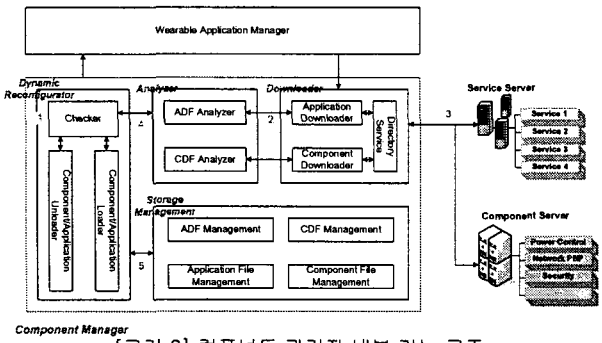
본 논문은 국방과학연구소의 대학 기초 과제인 "실시간 운영체제 인터페이스용 미들웨어 연구"의 수행 결과임.

컴포넌트 관리자의 내부 구조 및 동작 과정을 기술하고, 3장에서는 컴포넌트 관리자에 의한 컴포넌트의 삽입/삭제 과정을 기술한다. 4장에서는 컴포넌트 관리자를 통한 플랫폼 재구성 기능을 테스트하였고, 5장에서는 결론 및 향후 연구 방향에 대해 제시한다.

2. 컴포넌트 관리자

2.1 컴포넌트 관리자 내부 기능 구조

컴포넌트 관리자의 내부 기능 구조 및 실행은 [그림 2]와 같다.



[그림 2] 컴포넌트 관리자 내부 기능 구조

•***동적 재구성기(Dynamic Reconfigurator)***: 동적 재구성기 모듈(module)은 검사기(Checker)와 컴포넌트 로더(Component Loader), 컴포넌트 언로더(Component Unloader) 모듈로 구성된다.

- 검사기: 응용프로그램이나 컴포넌트의 설치 여부를 검사한다. 예를 들면, 플랫폼에 컴포넌트가 설치되어 있지 않으면 다운로더 모듈을 이용해서 컴포넌트를 외부 컴포넌트 서버로부터 다운로드한다. 컴포넌트가 설치되어 있고 메모리에 로드되어 있지 않을 경우, 분석기(Analyzer) 모듈이 컴포넌트의 설치 정보를 분석한 후 정보를 컴포넌트 로더에 전달한다.
- 컴포넌트 로더: 전달된 컴포넌트 정보를 이용하여 해당 컴포넌트를 메모리에 로드한다.
- 컴포넌트 언로더: 응용프로그램의 실행 종료 및 삭제로 인해 플랫폼에서 더 이상 사용하지 않는 컴포넌트가 있을 때, 해당 컴포넌트를 메모리에서 해제 또는 삭제한다.

•***분석기(Analyzer)***: 분석기 모듈은 응용프로그램 및 컴포넌트 정보의 명세를 분석하여 응용프로그램 및 컴포넌트의 다른 컴포넌트에 대한 참조 여부를 검사하거나 컴포넌트의 버전을 관리한다.

•***다운로더(Downloader)***: 다운로더 모듈은 시스템에 설치되어 있지 않은 응용프로그램이나 컴포넌트를 유/무선 네트워크를 이용해서 다운로드하는 모듈이다. 디렉토리 서비스(Directory Service)를 이용하여 인터넷 상에서 해당하는 응용프로그램 및 컴포넌트를 찾아 다운로드한다(② ~ ③).

•***저장 관리자(Storage Manager)***: 저장 관리자 모듈은 응용프로그램 및 컴포넌트들을 설치할 때, 기존에 설치되어 있던 응용프로그램 및 컴포넌트에 대한 백업(backup) 기능을 제공한다(⑤).

3. 컴포넌트 관리

3.1 컴포넌트 정보

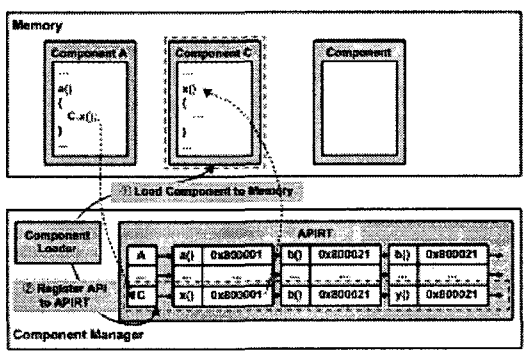
컴포넌트를 관리하기 위해서는 컴포넌트 정보를 명세화하는 것이 필수적이다. 컴포넌트의 정보는 CDF(Component Description File)에 명시되어 있다[1]. 메모리에 로드될 컴포넌트가 다른 컴포넌트를 참조하게 될 경우 의존성이 있다고 정의하며, 의존성을 갖는 컴포넌트는 CDF에서 require 필드에 명시하고 컴포넌트를 로드할 때, require에 있는 정보를 이용하여 의존성 관계에 있는 컴포넌트를 같이 메모리에 로드한다.

<표 1> CDF 형식

항목 및 값	비고
Name = Socket Component	컴포넌트 이름
Version = 1.0	컴포넌트 버전 정보
Require = none	의존성 정보
functions=DC_socket;DC_getI nterfaceList; …	컴포넌트에 정의되어 있는 API의 이름으로 구성

3.2 컴포넌트 설치 및 갱신

컴포넌트 관리자가 컴포넌트를 메모리에 로드할 때 컴포넌트 로더는 컴포넌트를 메모리에 로드하고(①), 컴포넌트 API들의 주소를 이용하여 APIRT(API Reference Table)를 구성한다(②).



[그림 3] 컴포넌트 로더 기능 구조

APIRT는 응용프로그램 및 컴포넌트 간의 컴포넌트 서비스 API에 대한 참조를 제공한다. [그림 3]에서 APIRT는 서비스 컴포넌트에 따라 각 컴포넌트에 속한 API들의 이름과 주소를 별도의 링크드 리스트(linked list)로 관리한다. 플랫폼에 설치된 모든 컴포넌트의 API를 단일 링크드 리스트로 관리하면, 메모리에 로드된 컴포넌트 및 API가 많을 경우, API에 대한 검색 및 삭제 과정에서 오버헤드가 많이 발생한다. 따라서 추가, 삭제의 단위로 이루어지는 컴포넌트 별로 관리하는 것이 효과적이다. 임의의 컴포넌트가 삽입/삭제/갱신될 때, APIRT은 갱신되며, 메시지 큐

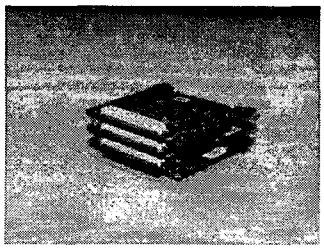
(message queue) 방식을 이용하여 갱신된 정보를 응용프로그램 및 다른 컴포넌트에게 전달한다.

3.3 컴포넌트 삭제

자원 제약적 환경에서 메모리 관리는 중요하다. 따라서 컴포넌트 관리자는 플랫폼에 설치된 컴포넌트가 더 이상 응용프로그램이나 다른 컴포넌트에서 사용되지 않는다면 메모리에서 해당 컴포넌트를 제거하여 메모리를 확보한다. 예를 들면 컴포넌트가 로딩될 때 충분한 메모리가 없을 경우, 현재 시스템에서 사용되지 않는 컴포넌트를 검색하여 메모리에서 제거한 후, 새로운 컴포넌트를 메모리에 로딩한다. 컴포넌트 삭제 시, APIRT은 설치된 컴포넌트들의 참조 카운터를 관리하고 있으므로 다른 컴포넌트 및 응용프로그램에서의 참조 여부를 확인할 수 있다.

4. 실험 및 실험결과

구현한 컴포넌트 관리자의 실행 테스트는 한국전자통신연구원이 개발한 WPS 보드를 이용하여 테스트하였다. 사용한 WPS의 운영체제는 Linux-2.4.20이고, 하드웨어 사양은 CPU: APM926EJ-S, SDRAM: 64MB, flash memory: 32MB이며, 네트워크 디바이스 USB WLAN을 이용하였다.

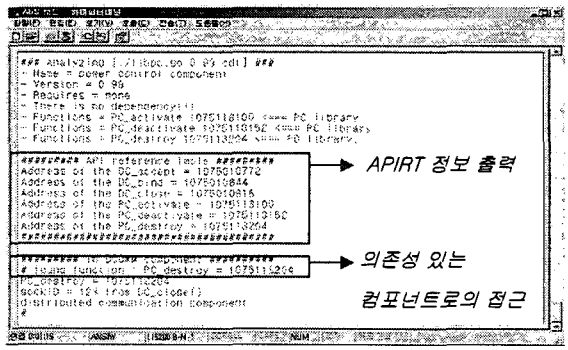


[그림 4] WPS 보드

실험은 응용프로그램의 요구에 따라 소프트웨어 플랫폼이 동적으로 재구성되는 것을 확인하기 위하여, 통신 서비스를 지원하는 통신 미들웨어 컴포넌트(사이즈: 40KB)의 API를 이용하는 채팅프로그램을 개발하여 컴포넌트가 설치되어 있지 않는 환경에서 실험하였다. 채팅프로그램이 실행될 때, 필요한 통신 서비스 컴포넌트를 메모리에 로드하여 서버/클라이언트간 채팅이 정상적으로 동작하였고, 응용프로그램 실행 중에 통신 컴포넌트를 메모리에서 삭제한 후 통신을 하려고 하면 컴포넌트가 다시 메모리에 로드되어 계속적으로 통신 서비스를 이용할 수 있었다.

<표 2> 실험 결과

내 용	테스트 결과
컴포넌트 설치/삭제에 따른 플랫폼 이미지 사이즈	설치 전: 43KB
	설치 후: 83KB
	삭제 후: 43KB
프로그램 재시동 여부	프로그램 재시동 없이 컴포넌트 설치/삭제 가능



[그림 5] APIRT 및 컴포넌트간 참조

또한 APIRT을 이용한 컴포넌트 간의 참조 여부 확인 및 의존관계에 있는 컴포넌트의 동적인 추가는[그림 5]와 같이 메모리에 로드된 컴포넌트들의 API 참조 정보와 의존성 정보를 확인함으로써 확인할 수 있었다.

5. 결 론

본 논문에서 응용프로그램 및 컴포넌트의 요구에 따라 소프트웨어 플랫폼을 동적으로 재구성하는 컴포넌트 관리자를 설계 및 구현하였다. 구현된 컴포넌트 관리자는 컴포넌트 단위로 플랫폼을 동적 재구성하므로 리소스가 절약되며 플랫폼의 확장 및 축소가 자유롭다. 또한 컴포넌트 관리자의 용량이 작기 때문에 경량의 디바이스 환경에서 적합하다. 현재 컴포넌트 관리자는 컴포넌트의 버전 체크 및 플랫폼의 동적 재구성 기능을 제공한다.

향후에는 응용프로그램 및 컴포넌트의 구동 중에 예기치 못한 오류가 발생할 경우, 플랫폼을 자가 치료할 수 있는 기능이 추가되어야 할 것이다. 또한 네트워크에서 응용프로그램 및 컴포넌트를 다운로드할 때 디렉터리 서비스 기능을 추가하여 다운로드에 발생되는 제약을 줄여야 할 것이다.

6. 참고문헌

[1] 박충범, 유용덕, 최훈, " 임베디드시스템용 소프트웨어플랫폼을 위한 동적 재구성 관리자 설계" , 한국컴퓨터종합학술대회 2005 논문집, vol.32 , pp. 667-669, 2005. 7.

[2] 유용덕, 김연수, 최훈, " 동적 Upgrade 기능을 구비한 WIPI 개발환경 설계" , 한국 통신학회 추계 종합 학술발표회 논문초록집, vol. 28 p.343, 2003. 12.

[3] 임형택, 장준, 최훈, " 무선 인터넷 플랫폼에서의 API 관리자 설계 및 구현" , 한국 정보처리학회 2004 추계 학술발표 논문집, 제 11권 2호 pp. 1763-1766, 2004. 11.

[4] 김연수, 강민철, 유용덕, 최훈, " 무선인터넷 플랫폼에서 다중 응용프로그램 수행을 위한 스케줄러 설계" , 한국정보처리학회 2004 추계학술발표논문집, 제 11권 2호 pp. 1759-1762, 2004. 11.