SDR을 위한 SCA에서의 소프트웨어 컴포넌트의 실시간 교체 방법

김민준** 류상렬** 김승호**
*경북대학교 컴퓨터공학과 **충남대학교 컴퓨터공학과
mjkim@mmlab.knu.ac.kr, rsr@cwunet.ac.kr, shkim@knu.ac.kr

Real-time Swapping Method of Software Components in SCA for SDR

M.J Kim** S.R Ryu** S.H Kim*
*Dept. of Computer Engineering, Kyungpook National University
**Dept. of Computer Science, Chungwoon University

요 약

SDR (Software Defined Radio) 기술은 세계적인 경쟁 표준이 없는 이동통신 시스템을 하나의 단일을 이용하여 소프트웨어적으로 쉽게 가능하도록 만들어주는 것이다. SDR에서는 소프트웨어 프레임워크의 표준으로 JTRS JPO에서 정의한 SCA (Software Communications Architecture)를 채택하였다. 본 논문에서는 SCA의 구조에 대해선 간단히 살펴보고 소프트웨어 컴포넌트의 실시간 교체 방법이 필요한 부분을 프로그램 사물레이아웃을 통해 제시한다.

1. 서 론

현재의 개인 이동통신 시스템은 세계적으로 다양한 표준이 존재하고 각 지역마다 사용하는 무선 프로토콜과 대역이 달라 국제적인 로밍(roaming)이 완전하게 이루어지기 쉽지 않다. 제도가 국제적인 표준을 제정하고자 했던 IMT-2000 시스템도 유럽 주도의 비동기적 광대역 CDMA 방식인 3GPP 시스템과 북미에서 주도하는 동기적 CDMA 방식인 3GPP2 시스템으로 나뉘어 표준화가 진행되고 있어 사실적인 국제 표준은 구현되기 힘들게 현실이다. 또한 통신 기술의 빠른 발전과 사용자들의 서비스 욕구의 변동이나 기기의 업그레이드가 주로 필요하며 그에 따라 하드웨어 장비를 교체하거나 사용자에게 불편을 초래하지 않도록 하기 위한 기술로서 요구된다.


SCA는 통상적으로 디바이스 제어 프로그램과 응용 프로그램으로 이루어진 내장형 시스템 소프트웨어의 디자인 패턴을 잘 활용한 프레임워크를 제공하고 있다. SCA는 분산 객체 모델의 실질적인 산업 표준인 CORBA (Common Object Request Broker Architecture)[4]을 기반으로 하며 동작에 의한 하드웨어와 다양한 업로드된 소프트웨어와 유연한 통합 환경을 제공한다. SCA 기반의 모바일 플랫폼은 사용하는 모바일 모드에 따라 공통의 개별화 프레임워크를 바탕으로 모바일의 상호 운용성, 소프트웨어의 유연성, 재사용성, 다양한 범위의 주파수 이용성을 제공한다.

본 논문의 구성으로 2장에서는 SCA를 기반으로 제작한 소프트웨어 프레임워크의 구조와 기능에 대해서 살펴보고 3장에서는 문제점으로 파악된 소프트웨어의 제작에 대한 실시간 교체를 접목하는 방법을 살펴보고 4장에서는 프로그램 사물레이아웃에서 도출한 결과를 통해 5장에서 결론을 내리도록 한다.

### 그림 1. SCA 소프트웨어의 기본구조

![그림 1. SCA 소프트웨어의 기본구조]( attachment://image.png)
2. SCA 소프트웨어 프레임워크

2.1 SCA 소프트웨어 기본 구조

SCA 기본 구조는 소프트웨어 컴퓨턴트 간의 상호 연결 및 관리를 위한 상용 프레임워크 (OS, CORBA) 기반의 개방적, 계층적인 분산 처리 환경을 제공한다. SCA 소프트웨어의 기본 구조가 그림 1에 나타난다. 이를 크게 실행 환경 (OS: Operating Environment)와 애플리케이션으로 이루어져 있고, 실행 환경은 RTOS (Real-Time Operating System)와 CORBA로 이루어진 상용 코어 프레임워크 (CF: Core Framework)로 나누어진다.

2.2 SCA의 Core Framework

SCA 소프트웨어 구조에서 가장 핵심이 되는 부분이 바로 Core Framework 부분이다. CF는 그 기능과 역할에 따라서 크게 4 부분으로 구성한다.

* 기본 응용 인터페이스
* 프레임워크 제어 인터페이스
* 프레임워크 서비스 인터페이스
* 도메인 프로파일

기본 응용 인터페이스는 소프트웨어 애플리케이션에서 사용될 수 있는 기반이 되는 인터페이스이다. 예를 들어 CORBA의 ORB를 이용한 통신을 사용할 수 있는 포트를 연결하거나 해제하는 등의 오피레이션들이 정의되어 있는 부분이다. 이 인터페이스를 이용하여 각 컴퓨턴트는 기본적인 생성/삭제, 포트 연결 설정/해제 작업 등을 수행할 수 있게 된다.

프레임워크 제어 인터페이스는 CF 내에서 가장 중요한 도메인 구성요소와 논리 장치들을 제어하는 인터페이스이다. 하나의 도메인 내에서 각종 도메인 구성요소인 애플리케이션, 장치, 장치 관리자, 비트 채널 등들을 관리하거나 논리 장치를 생성하고 실행시키며 해제하는 등의 주요한 오피레이션을 제공하는 부분이다. 프레임워크 제어 인터페이스는 기본 응용 인터페이스를 상속하여 필요한 기능을 확장하는 형식으로 존재한다.

프레임워크 서비스 인터페이스는 코어/Non-코어 애플리케이션에 사용되는 서비스를 제공하는 인터페이스로서 지역 파일 시스템뿐만 아니라 원격 파일 시스템도 파일 관리를 통해서 지역 파일 시스템처럼 접근이 가능하고, 또 파일 시스템에 존재하는 파일을 읽고 쓰는 등의 오피레이션을 제공한다.

도메인 프로파일은 SCA 시스템 도메인을 구성하는 모든 요소들의 속성, 능력, 상호 관계 등을 모두 나타내는 파일들의 집합을 일컫는다. 도메인 프로파일 파일들은 XML (eXtensible Markup Language)로 구성되고, DTD (Document Type Definitions) 포맷을 따른다 [5].

3. 컴퓨턴트의 실시간 교체 방법

SCA 소프트웨어 프레임워크에서 애플리케이션을 플랫폼 상에 놓까하기 위해서 도메인 프로파일 중 SAD (Software Assembly Descriptor)를 참조한다. SAD는 하나의 애플리케이션을 이루는 소프트웨어 컴퓨턴트들의 배치와 연결 정보를 기술한다. 시스템에 응용프로그램을 인스톨하는 것은 하나의 SAD 파일을 도메인 영역에 인스톨하는 것으로 볼 수 있다. SAD는 해당 컴퓨턴트에 대한 배치 정보를 알기 위해 도메인 프로파일 중 SPD 파일의 인스턴스에 대한 하나 이상의 참조를 가진다.

SPD (Software Package Descriptor)는 특정 소프트웨어 컴퓨턴트에 대한 구현 정보를 기술하는 파일이다. 이를 제거, 생성/파일, 구현 코드정보 그리고 하드웨어와 소프트웨어의 의존관계와 같은 소프트웨어 폭어지에 대한 일반적인 정보를 포함한다. SCD (Software Component Descriptor)의 인스턴스에 대하여 참조를 가질 수 있다. SCD는 특정 컴퓨턴트 (Resource, Resource-Factory, Device)에 대한 CORBA 인터페이스 정보를 기술한다. 컴퓨턴트가 상속/지원하는 인터페이스와 지원하는 포트도 포함한다.

![그림 2. 애플리케이션의 구성 예](image)

이와 같은 기술 구조를 바탕으로 프레임워크 제어 인터페이스의 ApplicationFactory는 SAD를 이용하여 도메인 영역에 애플리케이션을 인스톨할 수 있다. 그림 2와 같은 구조를 갖는 애플리케이션을 인스톨하였다고 가정하자. SAD에는 어셈블러 컨트롤러와 컴퓨턴트 1~3의 SPD를 참조하고 있다. 그리고 각 컴퓨턴트간의 연결 포트를 지정하여 어셈블러 컨트롤러에서 컴퓨턴트들로 조절할 수 있고, 컴퓨턴트들 간에는 통신 포트가 설정되어 서로간에 통신이 가능하다.

그림 2와 같은 애플리케이션에서 구성하는 소프트웨어 컴퓨턴트 2에서 문제가 발생하였을 경우를 가정하자. 현

SCA 소프트웨어 프레임워크가 현재 문제가 발생한 컴퓨터를 교체할 수 있는 오폐리에이션을 제공하지 않을 경우를 교체하기 위해서는 우선 해당 오폐리에이션을 인스톨 해야 한다. 다음으로 교체할 컴퓨터에 대한 SAD를 새로운 컴퓨터에 대한 SDP에 교체한 후 SAD에도 변경된 컴퓨터의 SDP에 대한 정보를 변경해 주어야 한다. 그런 후에 오폐리에이션을 인스톨하는 과정을 거쳐야 교체 작업을 완료할 수 있게 된다.

그림 3. 컴퓨터 제거 오폐리에이션

이러한 복잡한 과정을 거쳐야 하는 문제를 해결하기 위해서 본 논문에서는 오폐리에이션을 배포된 컴퓨터의 오폐리에이션을 인스털 장착 후에 오폐리에이션을 인스톨 해야 한다. 추가된 오폐리에이션의 작업 절차는 그림 3과 같다.

앞의 그림 2에서 살펴본 오폐리에이션에서 컴퓨터의 2가 문제가 발생한 경우를 다시 한번 가정하자. 새로운 컴퓨터에 경린 오폐리에이션을 이용하여 문제가 발생한 컴퓨터의 SAD를 SAD를 교체한다. 앞의 SAD 정보를 바탕으로 컴퓨터의 오폐리에이션의 컴퓨터의 오폐리에이션을 삭제하고 오폐리에이션의 오폐리에이션의 컴퓨터의 오폐리에이션에 오폐리에이션을 제거하는 것으로 제거 오폐리에이션 작업을 완료된다.

4. 실험 결과

프로그램 시뮬레이션을 위해 Fedora Core 3 Linux에서 작성하였고 Virginia Tech의 MPRG의 OSSIE를 기반으로하여 SCA v3.0의 초기 구현을 진행하였다. 프레임워크 제어 인터페이스에 추가된 컴퓨터 제거 오폐리에이션은 교체해야 할 컴퓨터를 제거한다. 그 후에 변경된 컴퓨터를 포함하는 새로운 오폐리에이션을 인스톨하여 교체 작업을 그림 4와 같이 진행하였다.

그림 4. 컴퓨터 제거 방법

시뮬레이션 결과, 교체를 결정한 컴퓨터가 컴퓨터 제거 오폐리에이션을 통하여 도메인 영역과 네이밍 서비스 목록에서 제거되었고, 새로운 오폐리에이션을 인스톨하였을 때, 처음과 같은 연결을 유지하는 것을 확인하였다.

5. 결론

다양한 이동 통신 환경 시스템 통합을 고려할 때 SDR는 좋은 선택이 될 수 있다. 이 SDR 솔루션에서 개발한 소프트웨어 프레임워크에 접속한 SDR의 개발을 위해 SCA를 제택하였다. SDR이 실제 환경에서 사용되기 위해서는 SCA에 필요한 컴퓨터 제거 방법을 논문에서 제안하였다. 실험은 교체 방법을 제택하면 SCA는 지금보다 더욱 유연한 구조를 가질 수 있었다.

제안한 방향이 기존 오폐리에이션에서의 연결 정보 확인을 위해 SAD를 조사한다. 이 때 확인한 연결 정보를 바탕으로 새로운 오폐리에이션을 설치하는 것이 아니라 필요한 부분, 즉 교체되어야 할 컴퓨터와 컴퓨터 교체할 수 있는 방법이 연구되어야 할 것이다.

참고 문헌