SCA 기반 SDR 단말기 구현

김준식 · 박남훈 · 김진업 · 권오준* · 김 남**

한국전자통신연구원 · *동의대학교 · **충북대학교

요 약

본 고에서는 SCA 기반 SDR 단말기 구현을 위해, 통신 애플리케이션 소프트웨어를 지원하는 하부 구조 관점에서 SCA Core Framework의 구조 및 기능을 기술하고, ETRI에서 SCA 2.2규격을 준수하여 개발한 SCARLET 미들웨어 및 SDR 단말 기능 검증용 공용 하드웨어 플랫폼을 소개하고, SCA기반의 WiMAX/HSDPA 통신 응용 컴포넌트 구현을 통한 실험 내용을 소개한다.

Ⅰ. 서 론

음성 서비스 위주의 이동 통신 사용자 요구사항이 다양한 형태의 멀티미디어 데이터를 하나의 단말에서 통신 모드, 네트워크의 종류 및 사용 장소의 구애없이 사용하는 방향으로 변화하고 있다. 이러한 요구 사항의 변화는 통신 사업자 및 단말기 제조업체에게 기존 통신 규격을 포함한 여러 통신 규격의 지원 및 새로운 표준 규격의 신속한 적용 또는 업그레이드에 적용할 새로운 기술 또는 패러다임의 변화를 요구하고 있다[1]. 이를 위해서는, 상이한 복수 개의 표준으로 구성된 핵심 네트워크들과 이들 네트워크 위에 구현된 많은 애플리케이션들로 이루어진 기존 데이터 인프라 구조로의 통합이 선행되어야 한다.

SDR(Software Defined Radio) 기술은 이러한 다양한 무선 통신 환경에 유연하게 대처하기 위하여, 하나의 공통 하드웨어 플랫폼에 사용자가 원하는 응용 소프트웨어로 무선 접속 통신 시스템을 재구성할 수 있는 개방형 신호 처리 기술이다[2],[3]. 또한, 아날로그 변조를 대신할 디지털 변조 기술 및 스마트 안테나의 출현 등의 기술 발전도 SDR기술의 실용화에 일조하고 있다[4].

SDR 시스템의 기술 개발 촉진을 위하여 1996년 MMITS(Modular Multifunction Information Transfer System) Forum이 만들어졌으며, 1998년 SDRF(SDR Forum)으로 개칭되었다. SDRF에서는 SDR 시스템을 기능적 관점에서 정의하는 노력을 통하여, 상위 수준 기능 모델인 SDR 소프트웨어 참조 모델을 제시하였으며, 미국방성 JTRS(Joint Tactical Radio System) 프로젝트에서는 컴포넌트 기반 프레임워크인 SCA(Software Communication Architecture)를 설계하였고, SDRF에 의해 SDR 소프트웨어 구조 표준으로 채택되었다[5].

본 고에서는 SDR기반 통신 애플리케이션 소프트웨어를 지원하는 하부 구조 관점에서 SCA Core Framework의 구조 및 기능을 기술하고, ETRI에서 SCA 2.2 규격을 준수하여 구현한 SCARLET(SCA Reconfigurable middleware of ETRI)미들웨어 및 SDR 단말용 기능 검증용 공용 하드웨어 플랫폼을 소개하고, SCA 기반 WiMAX / HSDPA 통신 응용 컴포넌트 구현을

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음. [2006-S-012-01, SDR 단말용 미들웨어 플랫폼]

통한 실험 내용을 소개한다.

Ⅱ. SCA 규격

JTRS의 JPEO(Joint Program Executive Office)가 개발하여 공표하고 있는 SCA 규격은 다음과 같은 개발 목표를 가지고 있다.

- SCA 규격을 준수하는 응용 소프트웨어의 이식성 제공
- 상용 표준을 활용한 개발 비용 절감
- 재사용 가능한 설계 모듈을 통한 소프트웨어 개발 기간 절감
- 상용화되고 있는 프레임워크 및 구조 상에서의 구축

SCA는 시스템 사양이 아니라, 위와 같은 목표를 달성하기 위하여 시스템 설계 시 준수하여야 할 규칙들을 제공하는 것으로, 이러한 규칙은 특정 구현에 종속되지 않는 일반적인 규칙이다.

SCA 규격의 역할은 통신 시스템상에 존재하는 소프트웨어, 하드웨어 구성 요소에 대한 관리 및 요구사항, 요구 용량들이 적절하게 운용될 수 있도록 하는 공통 하부 구조의 제공이다. 이러한 역할을 수행하기 위하여 Core Framework(CF)라 불리는 인터페이스들의 집합을 정의하고, 이를 시스템에서 운용될 애플리케이션과 하부 하드웨어 사이에 두어서, 애플리케이션은 Core Framework에 정의된 인터페이스에 따라 하부 하드웨어를 이용하는 구조를 가지고 있다.

2-1 SCA에서의 소프트웨어 구조

일반적으로 애플리케이션은 "하나 이상의 모듈을 포함하는 실행 가능한 소프트웨어 프로그램을 의미한다. SCA 규격에서의 애플리케이션은 규격에서 요구하고 있는 Base Application 인터페이스를 구현하고 SAD(Software Assembly Descriptor) 파일을 통해 확인되는 하나 이상의 소프트웨어 모듈들로 구성된 것"으로 제한한다. 애플리케이션이 적재되고 실행되었을 때 이들 소프트웨어 모듈들은 애플리케이션을 구성하는 컴포넌트들을 생성한다. 공통 하드웨어 플랫폼상에서 SCA 애플리케이션의 소프트웨어적인 추가, 제거, 수정을 통해 재구성/재배치 가능한 통신 시스템을 실현한다.

SCA는 [그림 1]과 같은 계층적인 소프트웨어 구조를 가정한다. SCA 소프트웨어 구조는 크게 운영환경(Operating Environment)계층과 운영 환경 계층에서 제공하는 서비스를 이용하여 구현되는 애플리케이션 리소스 계층으로 구분할 수 있다. 운영 환경은 다시 Core Framework, CORBA(Common Object Request Broker Architecture) 미들웨어, AEP(Application Environment Profile)계층과 POSIX(Portable Operating System Interface)기반 운영체제로 이루어져 있다.

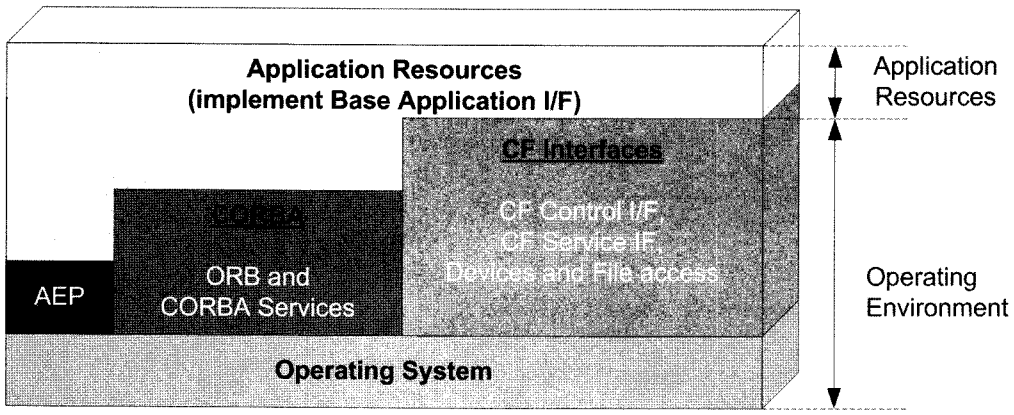
2-1-1 Core Framework[5]~[7]

SCA 소프트웨어 구조에서 Core Framework은 CORBA 상에서 애플리케이션 프로그램에 컴포넌트 기반 컴퓨팅 능력을 제공한다. Core Framework은 애플리케이션 설계자가 사용할 수 있는 하부 소프트웨어/하드웨어 계층에 대한 추상화(abstraction)를 제공하기 위한 개방형 애플리케이션 계층 인터페이스들과 서비스들의 집합이다.

애플리케이션 프로그램 설계자들은 Core Framework에서 제공하는 인터페이스들을 사용하여 하나의 애플리케이션 프로그램을 구성하는 컴포넌트를 프로세싱 노드에 설치, 시작, 정지, 제거 등의 관리 작업을 수행할 수 있다. 각 인터페이스는 특정 기능에 필요한 오퍼레이션과 속성을 제공하며, 오퍼레이션에 필요한 구체적인 파라메터 등의 내용은 프로파일(profile)이라 불리는 XML 형태의 파일 내에 기술된다.

2-1-1-1 Base Application 인터페이스

Base Application 인터페이스는 모든 애플리케이션



[그림 1] SCA 계층 구조[5]

컴포넌트가 구현하여야 하는 인터페이스로 Port, Life-Cycle, Testable Object, Property Set, Port Supplier, Resource와 Resource Factory 인터페이스로 구성된다.

Port 인터페이스는 컴포넌트 포트(port)간의 연결 관리를 위한 오퍼레이션을 제공한다. 애플리케이션 컴포넌트는 Port인터페이스를 상속한 인터페이스를 지정하여 특정 포트 타입을 정의하고 이를 사용하여 데이터 또는 제어 신호를 전달하기 위한 오퍼레이션을 정의할 수 있다. 컴포넌트들간의 포트 연결 정보는 SAD(Software Assembly Descriptor) 파일과 DCD (Device Configuration Descriptor) 파일에 기술한다.

Life Cycle 인터페이스는 생성된 컴포넌트에 따른 데이터 또는 프로세싱 요소(processing elements)를 초기화하거나 해제하는데 필요한 오퍼레이션들을 제공한다.

Testable Object 인터페이스는 컴포넌트의 구현을 시험하는데 필요한 오퍼레이션들을 제공한다. 시험 항목에 대한 내용은 해당 컴포넌트의 SPD(Software Package Descriptor)에서 참조하고 있는 Property Descriptor의 test element에서 기술한다.

Property Set 인터페이스는 컴포넌트의 properties/attributes를 접근하는 configure( ) 오퍼레이션과 query( )오퍼레이션을 제공한다.

Port Supplier인터페이스는 포트를 제공하는 컴포넌트에서 특정 포트에 대한 객체 참조를 얻기 위한 get Port( ) 오퍼레이션을 제공한다.

Resource인터페이스는 Life Cycle, Testable Object, Property Set 및 Port Supplier 인터페이스를 상속하여 애플리케이션 컴포넌트 수준에서의 동작 시작 및 정지를 위한 start( )/stop( ) 오퍼레이션을 제공한다.

Resource Factory 인터페이스는 애플리케이션을 구성하는 리소스를 생성하거나 종료하는데 필요한 오퍼레이션을 제공한다.

2-1-1-2 Framework Control 인터페이스

SCA 규격 용어집에 따르면, 도메인은 "하나의 도메인 관리자 컴포넌트의 제어 하에 있는 하드웨어 디바이스들과 하드웨어 디바이스상에서 이용 가능한 애플리케이션들의 집합"으로 정의된다.

하나의 도메인 내에서의 Framework Control 인터페이스는 도메인 관리와 디바이스 관리를 위한 인터페이스를 제공한다. Framework Control 관련 인터페이스로는 Application, Application Factory, Domain Manager, Device Manager 인터페이스가 있다. 이들 인터페이스는 애플리케이션, 디바이스, 도메인내의 디바이스 관리자들의 등록/해제, 도메인 내에서의 애플리

케이션의 제어와 관련된 오퍼레이션들을 제공한다.

Application 인터페이스는 Resource 인터페이스를 상속하며, 도메인 내의 생성된 애플리케이션 인스턴스의 제어, 구성, 상태와 관련된 오퍼레이션들을 제공한다.

Application Factory 인터페이스는 도메인내의 지정된 타입의 애플리케이션 생성을 요청하는데 필요한 create( ) 오퍼레이션을 제공한다. Create( )오퍼레이션은 SAD 파일과 SPD 파일을 참조하여 애플리케이션 인스턴스를 생성한다. Domain Manager 인터페이스를 구현한 도메인 관리자는 도메인 내에 설치된 SAD별로 각각의 Application Factory 클래스의 인스턴스를 생성한다.

Domain Manager 인터페이스는 시스템 도메인의 제어와 구성을 위한 오퍼레이션을 제공한다. Domain Manager 인터페이스에서 제공되는 오퍼레이션은 크게 HCI(Human Computer Interaction), 등록(registration), CF 관리(CF management)로 구분할 수 있으며, HCI 관련 오퍼레이션은 도메인을 구성하고 도메인의 디바이스, 서비스, 애플리케이션의 역량(capabilities)에 관련된 정보를 접근하고, 유지 관리 기능을 시작하는데 사용한다. 등록 관련 오퍼레이션은 시작 또는 동작 중에 디바이스 관리자, 디바이스 관리자에 등록된 디바이스 및 서비스, 애플리케이션의 등록/등록 해제와 관련된 작업에 사용한다. CF 관리 관련 오퍼레이션은 등록된 디바이스 관리자와 도메인 관리자의 파일 관리자에 대한 인터페이스 접근을 제공한다.

Device Manager 인터페이스는 일련의 디바이스들과 서비스들을 관리하는데 필요한 속성들과 오퍼레이션들을 제공한다. Device Manager 인터페이스의 device Configuration Profile 속성은 디바이스와 서비스가 전개되어 배치될 물리적 디바이스의 위치 및 논리적 명칭에 대한 매핑 정보가 기록된 DCD(Device Configuration Descriptor) 파일에 대한 정보가 기록된다. Device Manager 인터페이스의 fileSys 속성에는 디바이스 관리자와 연관된 파일 시스템에 대한 정보를 유지한다. 또한, Device Manager 인터페이스의 registered Devices 속성에는 해당 디바이스 관리자에 등록되고 관리되는 디바이스들의 리스트가 유지되며, Device Manager 인터페이스의 registered Services 속성에는 해당 디바이스 관리자에 등록되고 관리되는 서비스들의 리스트가 유지된다.

2-1-1-3 Base Device 인터페이스

시스템내의 하드웨어 장치들을 해당 소프트웨어 인터페이스를 통하여 관리, 제어하기 위한 인터페이스로 Device, Loadable Device, Executable Device, Aggregate Device 인터페이스가 있다.

Device 인터페이스는 Resource 인터페이스를 상속하며, 물리적인 하드웨어 디바이스에 대한 소프트웨어 추상화를 제공하기 위한 추가적인 속성과 오퍼레이션을 제공한다.

Device software Profile 속성은 디바이스의 ports, query/configure properties, capacity properties 및 status properties에 대한 정의를 포함하고 있는 SPD파일에 대한 참조를 포함하고 있다. 또한, 상태 관리를 위한 usage State, admin State, operational State 속성을 유지하고 있으며, 디바이스로부터 메모리 등의 특정 용량을 요구하거나 반납하기 위한 allocate Capacity 오퍼레이션과 deallocate Capacity 오퍼레이션을 제공한다.

Loadable Device 인터페이스는 Device 인터페이스를 확장하여 소프트웨어의 적재(loading) 및 적재 해제(unloading)과 관련된 오퍼레이션을 추가한 것이다.

Executable Device 인터페이스는 Loadable Device 인터페이스를 확장하여 디바이스 상에서 소프트웨어 프로세스 또는 쓰레드의 실행을 시작하거나 종료하는 오퍼레이션을 추가한 것이다.

Aggregated Device 인터페이스는 부모 디바이스로부터 자식 디바이스를 추가하거나 제거하는데 필요한 오퍼레이션을 제공하도록 확장한 인터페이스이다.

2-1-1-4 Framework Service 인터페이스

기타 지원 기능 및 서비스를 제공하기 위하여, File, File System, File Manager 인터페이스가 있다. File 인터페이스는 파일을 읽고 쓰는데 필요한 오퍼레이션을 제공한다. File System 인터페이스는 물리적 파일 시스템에 대한 원격 접근을 제공한다. File Manager 인터페이스는 다른 파일 시스템들을 논리적으로 하나의 파일 시스템처럼 다룰 수 있는 기능을 제공한다.

2-1-2 AEP 및 CORBA 미들 웨어 계층

Core Framework 서비스에서 사용하는 분산 컴포넌트 지원에 필요한 명명법(naming and typing), 오류 관리(error management), 서비스 중개 관리(service brokering management) 및 통신 링크 제어(communication link control)등의 하드웨어, 소프트웨어, 운영체제, 네트워크 프로토콜, 프로그래밍 언어의 차이에 따른 이기종 분산 컴퓨팅 관련 순수 미들웨어 기능은 OMG Minimum CORBA[8]를 사용한다.

SCA CF 및 minimum CORBA 미들웨어가 실행되도록 지원하는 하부의 운영체제는 그 위에서 실행되는 애플리케이션 프로그램 또는 컴포넌트들이 무선 통신 프로토콜의 기능을 구현하는 경우, 주어진 종료 시한 내에 실행을 완료하는 실시간 처리 특성을 가지고 있으므로 실시간 운영 체제(Real-Time Operating System: RTOS)이어야 한다. 이를 위해 SCA에서는 IEEE std. 1003.13에 정의된 POSIX Real-time Application Support를 준수하는 운영 체제를 사용하도록 명시하고 있다.

먼저, 애플리케이션 컴포넌트는 CF의 Framework Control 인터페이스를 통하여 제어된다. 또한, 애플리케이션 컴포넌트는 Port 인터페이스를 사용하여 다른 애플리케이션 컴포넌트와 통신하거나 시스템이 제공하는 시스템 컴포넌트(서비스 또는 디바이스)와 통신하게 된다. 애플리케이션 컴포넌트와 Framework Control 또는 Framework Services 인터페이스간의 통신은 CORBA 미들웨어를 통하여 이루어져야 한다. 이러한 통신 제약 사항의 의도는 서비스 또는 디바이스와 같은 시스템 컴포넌트에 대한 API들을 특정 시스템 또는 도메인에 맞게 표준화함으로써, 프레임워크와 함께 애플리케이션과 시스템간의 일관된 통신 메커니즘을 제공하는데 있다.

애플리케이션 컴포넌트는 운영 체제가 제공하는 기능 중 POSIX규격의 부분 집합인 AEP에서 제공하는 기능만을 사용하도록 제한된다.

디바이스와 서비스 같은 시스템 컴포넌트는 하부 시스템에 의존적일 수 있으므로 운영 체제가 제공하는 기능의 사용에 대한 제약 사항은 없으며, Base Device 인터페이스를 통하여 Framework Control 인터페이스에 의해 관리된다.

2-2 도메인 프로파일(Domain Profile)

컴포넌트들을 배치하고 연결하여 애플리케이션이 동적으로 구성될 수 있도록, 시스템 내의 하드웨어 장치 및 소프트웨어 컴포넌트 특성을 기술하는 XML 파일을 통칭하여 도메인 프로파일이라 한다. 도메인 프로파일 내에는 하드웨어 및 소프트웨어 컴포넌트의 인터페이스, 기능적인 면에서의 역량(capabilities), 논리적 위치, 상호 의존성 및 관련 파라메터 등의 내용이 포함된다.

도메인 프로파일은 파일내에 기술되는 내용에 따라, CF에 애플리케이션 프로그램, 애플리케이션 프로그램 구성 컴포넌트 등의 소프트웨어적인 정보를 제공하는 소프트웨어 프로파일, 프로세싱 노드 및 노드의 특성과 같은 하드웨어적인 정보를 제공하는 디바이스 프로파일과 도메인 관리자용 구성 파일로 구분할 수 있다.

2-2-1 소프트웨어 프로파일

소프트웨어 측면에서의 컴포넌트 및 애플리케이션 정보를 기술하는 소프트웨어 프로파일은 다음과 같다.

SAD(Software Assembly Descriptor): 하나의 애플리케이션 프로그램을 구성하는 애플리케이션 컴포넌트들과 이들간의 연결 관계를 기술한다. Application Factory는 SAD 파일을 이용하여 지정된 애플리케이션을 생성한다.

SPD(Software Package Descriptor): 소프트웨어 컴포넌트의 구현들에 대한 정보를 유지한다. 소프트웨어 패키지의 이름, 작성자, property 파일 정보 및 구현 코드의 정보, 하드웨어/소프트웨어 의존성(실행 가능한 플랫폼, 실행 환경, 실행에 필요한 최소 메모리, CPU 요구량) 등의 정보를 기술한다.

SCD(Software Component Descriptor): 특정 SCA 소프트웨어 컴포넌트(Resource, Resource Factory, Device) 구현에서 제공하거나 사용하는 인터페이스에 대한 정보를 포함한다. Device에 대한 SCD인 경우 DPD 파일에 대한 참조를 포함한다.

PRF(Properties Descriptor): 컴포넌트 수준에서 적용 가능한 형상(configuration), 검사(test), 실행(execute) 및 할당(allocation) 관련 properties에 대한 정보를 기술한다.

2-2-2 디바이스 프로파일

하드웨어 측면에서의 구성 정보를 제공하는 디바이스 프로파일은 다음과 같다.

DCD(Device Configuration Descriptor): 특정 디바이스 관리자와 연관된 디바이스에 대한 정보, 도메인 관리자를 찾는 방법, 개별 디바이스에 대한 구성 정보(Log, File System등)을 기술한다.

DPD(Device Package Descriptor): 개별 노드 수준에서의 하드웨어 장치 정보(제작자, 모델 번호, 하드웨어 타입, 일련 번호, 메모리 크기 등)를 기술한다.

PRF(Properties Descriptor): 디바이스 패키지에 적용 가능한 구성(configuration), 검사(test), 실행(execute) 및 할당(allocation) 관련 properties에 대한 정보를 기술한다.

2-2-3 도메인 관리자 구성 파일

도메인 관리자는 애플리케이션 프로그램 수준에서의 설치, 제거 및 하드웨어 디바이스의 등록, 해지를 위한 최상위 인터페이스를 제공한다. 도메인 관리자의 실행 위치 및 도메인 관리자에 의해 사용되는 CF 인터페이스 또는 서비스 등의 내용은 DMD(Domain Manager Descriptor)에 기술한다.

Ⅲ. SCA 기반 애플리케이션 개발 절차

SDR용 애플리케이션은 일반적으로 신호 처리, 패킷 입출력, 보안, 라우팅 등의 다양한 기능을 가진 모듈들이 모여서 구성된다. SCA 기반 애플리케이션을 구성하는 컴포넌트들은 각각의 기능 모듈에 해당하는 기능을 수행하는 단위인 컴포넌트로 구현되며, Core Framework에 의해 제어될 수 있도록 Base Application 인터페이스를 구현하여야 한다. 그러나, 기존에 개발된 바이너리 코드만 있는 경우에는 소스코드가 없으므로 컴포넌트화를 할 수 없다. 이때에는 기존 바이너리 코드에 Adaptor코드와 연동하여 다른 컴포넌트와의 통신 및 Core Framework 구조에 호환되도록 구성하여야 한다.

3-1 기능에 따른 모듈화

SCA 기반 애플리케이션 개발은 먼저 해당 애플리케이션의 기능을 정의하고, 각 기능을 분할하여 모듈별로 분리하는 작업이 선행되어야 한다. 이때 모듈화는 재사용성이 높으면서 독립적으로 배포가 가능하도록 하여야 한다.

3-2 외부에 제공할 인터페이스 정의

기능의 모듈별 분리가 끝나면, 각 모듈 기능이 구현될 컴포넌트는 외부에 제공할 인터페이스를 정의하여야 한다. 각 컴포넌트가 제공할 인터페이스는 CORBA IDL(Interface Definition Language)를 사용하여 정

의한다. 컴포넌트가 외부에 제공하여야 할 인터페이스는 SCA Core Framework을 위한 Base Application 인터페이스와 사용자 지정 인터페이스가 있다. IDL로 정의한 인터페이스 명세에 따라, IDL 컴파일러는 클라이언트 stub 코드와 서버용 skeleton 코드를 생성한다. 컴포넌트 내부에서만 사용하는 기능을 구현하는 코드의 경우에는 컴포넌트 외부에서 호출할 경우가 없으므로 인터페이스에 포함시키지 않는다.

3-3 인터페이스 정의에 따른 컴포넌트 구현

컴포넌트 개발자는 IDL로 정의한 Base Application 인터페이스와 사용자 정의 인터페이스에 따라 생성된 stub/skeleton 코드를 사용하여 컴포넌트가 외부에 제공하는 오퍼레이션들과 속성에 대한 구현 코드를 작성한다. SCA 컴포넌트는 컴포넌트간 통신을 위하여 포트 개념을 사용한다. 따라서, 컴포넌트간 통신 기능이 필요한 컴포넌트는 데이터 및 제어 신호의 송수신을 위한 Use Port와 Provide Port를 구현하여야 한다. 데이터 및 제어신호는 소스 컴포넌트의 Use Port에서 대상 컴포넌트의 Provide Port로 전송된다. 컴포넌트간 통신을 위한 포트 구현을 완료한 후, 해당 컴포넌트의 고유 기능인 신호 처리 기능을 구현하여 추가한다.

애플리케이션 수준에서의 시작과 정지 같은 제어 흐름에 따라 애플리케이션 구성 컴포넌트들이 동작하도록 제어하기 위하여 애플리케이션의 대리역할

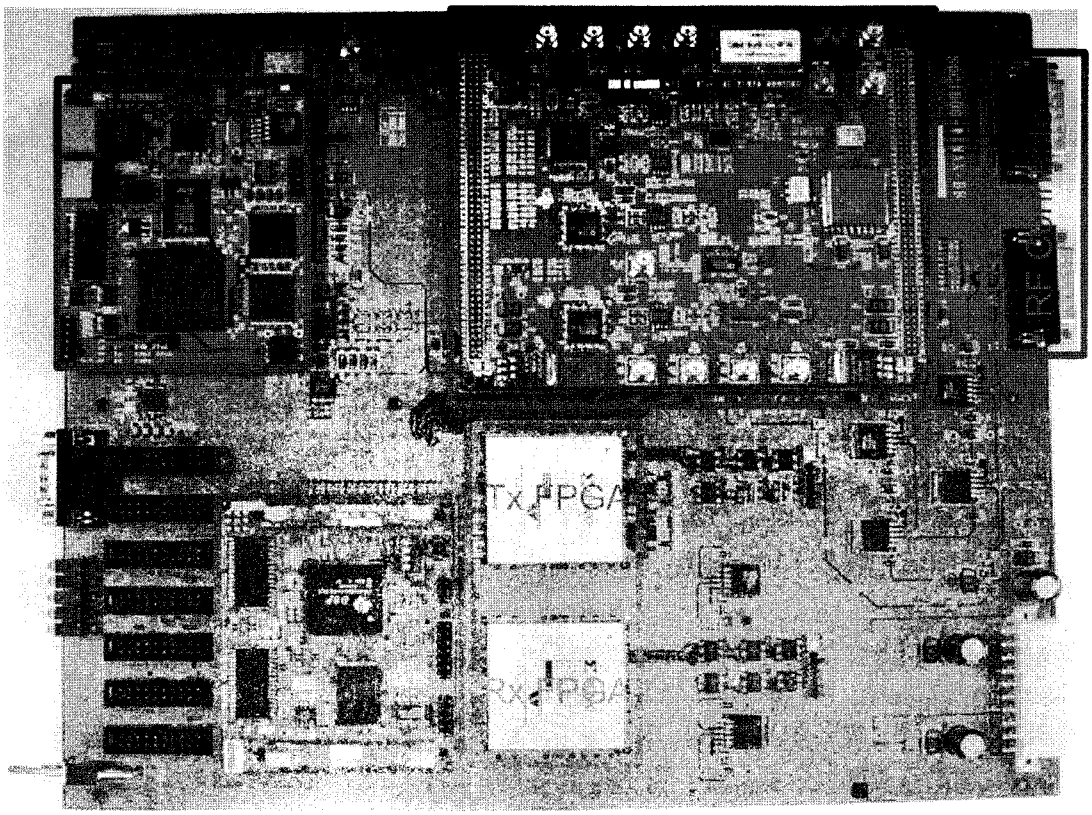


그림 2. 검증용 단말 하드웨어

을 수행하는 assembly controller 컴포넌트가 있어야 한다. Assembly controller 컴포넌트는 Core Framework 로부터의 명령을 애플리케이션을 대리하여 수신 후, 받은 명령에 따라 다른 컴포넌트의 동작을 제어하도록 한다.

3-4 컴포넌트 디스크립터 파일 작성

컴포넌트별로 PRF, SCD, SPD파일을 작성한다. PRF파일에는 컴포넌트의 properties 정보를 기재하고, SCD 파일에는 컴포넌트의 인터페이스 정보 및 컴포넌트 타입 정보를 기재하며, SPD 파일에는 컴포넌트의 SCD 파일 및 PRF파일 정보, 구현별 바이너리 파일의 위치 등의 구현 패키지 정보를 기록한다.

3-5 애플리케이션 디스크립터 파일 작성

SCA 애플리케이션은 한 개 이상의 컴포넌트들로 구성되므로, 애플리케이션을 구성하는 컴포넌트들의 결합과 형상 정보를 SAD 파일에 기술하여 제공하여야 한다. SAD에는 애플리케이션에 필요한 컴포넌트들의 SPD 파일 리스트, 컴포넌트간 포트 및 인터페이스 연결 정보, 디바이스 및 서비스로의 연결 정보, 디바이스와 서비스 찾는 방법, 배치에 따른 종속성, 애플리케이션의 assembly controller 역할을 하는 컴포넌트의 지정 등이 포함된다.

3-6 패키징 및 배포

도메인 프로파일 및 도메인 프로파일 내에서 참조하고 있는 모든 파일은 배포 가능한 하나의 패키지로 묶어진다. Jar 또는 tar 형태의 패키징이 주로 사용된다. 배포와 관련된 내용은 구현에 종속적이 부분으로 SCA 규격에서는 다루지 않는다.

Ⅳ. WIMAX/HSDPA 통신 컴포넌트 구현

ETRI SDR 연구팀에서는 2004년부터 2006년까지 SDR 기반의 RBS(Reconfigurable Base Station System) 연구 개발을 통하여 SCA 규격 2.1을 적용한 캐나다 CRC사의 SCARI 제품을 이용하여, 이중 모드 기지국을 구현한 바 있다[7]. 2006년 ETRI 이동미들웨어팀에서는 기지국에서의 SCA 적용 경험을 바탕으로 SCA 규격 2.2를 준수하는 SCA Core Framework을 오픈 소스인 TAO CORBA와 Xerces XML 라이브러리상에서 구현하여 SCARLET이라는 브랜드명으로 자체 개발하였다. SCARLET은 기지국에 비하여 컴퓨팅 리소스가 한정되는 단말기를 목표로 경량화에 주안점을 두고 개발되었다.

4-1 SCARLET 검증용 단말 하드웨어 플랫폼

SCARLET 시스템의 검증용 하드웨어 플랫폼 설계 시 고려한 주요 사항은 다음과 같다.

- 미들웨어 플랫폼 검증에 적합한 구조를 가지도록 한다.
- 지원할 무선 전송 방식은 WIMAX와 HSDPA이다. 이들간의 정합은 시스템 재구성성 향상이 가능하도록 설계 및 구현한다.
- RF 모듈은 SMA 커넥터를 통해서 독립적으로 연결되도록 한다.
- 개발 초기에는 GPP (General Purpose Processor) + FPGA (Field Programmable Array) + DSP(Digital Signal Processor) 구조를 사용한다. GPP는 제어용으로, FPGA와 DSP는 칩 및 심볼 프로세싱용으로 사용된다.

[그림 2]는 검증용 단말 하드웨어 보드의 사진이다. 검증용 단말 하드웨어 플랫폼은 여러 개의 보드들의 조합으로 이루어져 있다. 마더 보드에는 각종 보드와 연결될 수 있는 커넥터와 기저 대역 모뎀의 송신기 및 수신기 기능을 하는 FPGA 2개가 있다. 수신기 FPGA의 확장 예비용으로 DSP 보드가 있다. 메인 마이크로 프로세서가 장착된 프로세스 보드는 전체 보드의 제어를 담당한다. IF 보드에는 IF 모듈과

디지털-아날로그 변환기(ADC, DAC)가 있다. RF 모듈은 독립적으로 구성하여 IF 보드상의 SMA 커넥터로 연결하여 사용할 수 있도록 구성되었다. <표 1>은 검증용 단말 하드웨어 플랫폼을 구성하는 보드별 모듈의 제원을 정리한 것이다.

4-2 검증용 통신 애플리케이션

단말기에서의 SCARLET과 SDR 애플리케이션 동작의 가능성을 확인하기 위하여, 검증용 단말 하드웨어 플랫폼을 제작하고 SCA 기반의 컴포넌트로 WiMAX(Worldwide Interoperability for Microwave Access)[10], HSDPA(High-Speed Downlink Packet Access)[11]용 애플리케이션을 개발하여 시험하였다. 각각의 애플리케이션을 구성하는 컴포넌트들은 SCARLET 시스템에 의해 GPP/DSP/FPGA 디바이스로 배치되어 구동되게 되며, 이러한 정보는 XML 형태의 도메인 프로파일에 기재된다.

WiMAX와 HSDPA 모드의 검증용 애플리케이션 컴포넌트들은 각각 하나의 애플리케이션 단위로 패키징되어 SCARLET 시스템에 의해 관리되고 실행되며, 각각의 애플리케이션은 별도의 SAD 파일에 기술된다. 또한, CORBA의 naming service와 event service를 받기 위해 CORBA와 연동한다. 컴포넌트간 통신은 CORBA의 IDL을 이용한 SCARLET 시스템의 SCA Port 인터페이스를 이용한다.

SDR 단말용 SCA 기반 재구성 가능한 미들웨어 플랫폼 시스템 SCARLET 관리 및 진단 기능을 위하여, GUI(Graphic User Interface) 기반의 클라이언트 프로그램인 Waveform Manager를 MS Windows상에서 개발하였다. 단말 하드웨어 플랫폼과 Waveform Manager는 LAN으로 연결된다. Waveform Manager는 CORBA 기반 메시지 전달을 통해 단말 하드웨어 플랫폼상의 SCA Core Framework에 접근하여 SCA 기반 애플리케이션 동작을 제어하고 관리한다.

4-2-1 WiMAX 애플리케이션

검증을 위하여 구현된 WiMAX 애플리케이션에서 모뎀 기능은 보드상의 FPGA로 구현되었으며, MAC (Medium Access Control) 이상 계층이 GPP 상에서 SCARLET의 제어를 받아 구동되도록 설계되었다. WiMAX

<표 1> SCARLET 검증용 단말 하드웨어 플랫폼 제원

모듈	특징
RF	RF는 메인보드와 별도로 독립 구현 Port를 통해서 AFC, PLL, LNA, TPC, AGC등 RF 파라미터 제어
IF	전형적인 IQ 송수신 구조 (ADC, DAC 포함) Modulation(U2793B), Demodulation(AD8348), ADC(AD6644, 14 bit), DAC(AD9777, 16 bit) IF 주파수: 70 MHz, PLL: 140 MHz, 10 MHz 기준 클럭
프로세서 보드	XC2VP30 (Virtex-II Pro FPGA with dual PowerPC 405 cores), 200 MHz 지원 128 Mbyte Flash, 128MB SDRAM, Montavista Linux RTOS, DMA logic, HCS
FPGA (Modem)	Xilinx FPGA로서 Tx, Rx 사용 Tx: XC2V4000 (RS Encoder Convolution Encoder Interleaver IQ mapping IFFT Add CP) Rx: XC2V6000 (FFT Ch comp Error corrector Demapper FEC Decoder<de-int,viterbi, RS decode>)
DSP	Receiver를 위한 Reserved DSP Board TMS320C6414 (TI DSP) 사용, JTAG포트

용 애플리케이션은 기개발된 소스 및 바이너리 파일을 adaptor 패턴으로 SCA 컴포넌트화하고 추가적인 assembly controller 기능을 위한 컴포넌트만 새로 개발하였다.

WiMAX용 애플리케이션을 구성하는 컴포넌트는 총 4개로 구성하였다. WiMAX 응용 컴포넌트는 assembly controller 블록인 WACB(WiMAX Assembly Controller Block), Physical 계층 Adapter 블록인 WPAB (WiMAX Physical-layer Adaptor Block), MAC 계층 Adapter 블록인 WMAB(WiMAX MAC-layer Adaptor Block)와 TE(Terminal Equipment) 및 I/O Resource에 대한 Adapter 블록인 WIAB(WiMAX I/O Adaptor Block)으로 나뉜다.

Assembly controller 컴포넌트에 해당하는 WACB는 SCARLET 도메인(Domain)상에 설치된 WiMAX 애플리케이션 컴포넌트인 WPAB, WMAB, WIAB 의 시작, 중지 및 property 제어 기능을 수행하고, 다른 컴포넌트들이 자신에게 접근할 수 있도록 하기 위해 naming service에 자신을 등록(bind)한다.

WPAB는 WiMAX 의 물리계층 프로토콜과의 연결을 담당하는 커널 드라이버 SCA 인터페이스 기능을 담당한다. WPAB는 커널 드라이버를 SCA의 규격에 부합하는 응용 컴포넌트로 바꾸어 SCARLET 환경하에서 원활히 동작할 수 있도록 하여 준다. 상위의 SCA 컴포넌트인 WACB의 지시를 받아 응용 컴포넌트에 대한 시작, 정지 및 property 제어 기능을 수행한다.

WMAB는 WiMAX의 MAC 프로토콜의 SCA 인터페이스 기능을 담당한다. WMAB는 MAC 계층 소프트웨어를 SCA의 규격에 부합하는 응용 컴포넌트로 바꾸어 SCARLET 환경하에서 원활히 동작할 수 있도록 하여 준다. 상위의 SCA 컴포넌트인 WACB의 지시를 받아 응용 컴포넌트에 대한 시작, 정지 및 property 제어 기능을 수행한다.

WIAB는 WiMAX의 I/O기능과의 연결을 담당하는 커널 드라이버의 SCA 인터페이스 기능을 담당한다. WIAB는 커널 드라이버를 SCA의 규격에 부합하는 응용 컴포넌트로 바꾸어 SCARLET 환경하에서 원활히 동작할 수 있도록 하여 준다. 상위의 SCA 컴포넌트인 WACB의 지시를 받아 커널 드라이버에 대한 시작, 정지 및 property 제어 기능을 수행한다.

4-2-2 HSDPA 애플리케이션

HSDPA용 애플리케이션을 구성하는 컴포넌트는 모두 9개로 구성하였다. HSDPA용 애플리케이션 컴포넌트는 기존 소스를 기능별로 재분할하여 직접 SCA 컴포넌트화하였다. 또한, 컴포넌트 설계 방식에 따른 성능 변화를 측정하기 위해서, HGMMB, HPDCPB, HRLCB, HRRCB, HSMB 를 하나의 컴포넌트로 통합 설계하여 총 5개의 컴포넌트로도 구현하여 성능을 비교하였다.

HACB(HSDPA Assembly Controller Block)는 assembly controller 기능을 하는 컴포넌트로 SCARLET 도메인(Domain)상에 설치된 HSDPA 애플리케이션을 구성하는 컴포넌트들인 HPHYB, HMACB, HRLCB, HRRCB, HPDCPB, HGMMB, HSMB, HTEAB Resource들의 start/stop/property 제어 기능을 수행하고, 다른 component들이 자신에게 접근할 수 있도록 하기 위해 CORBA의 naming service에 자신을 등록한다.

HTEAB(HSDPA TE Adapter Resource Block)는 UE (User Equipment)와 TE 사이의 정합을 담당하는 역할을 수행한다. SCA Port 통신을 하는 HSMB, HPDCPB 프로토콜 블록의 CORBA IDL 형식의 데이터를 Non-SCA 환경인 TE로 전달해 주는 역할을 수행한다.

HSMB(HSDPA SM Resource Block)은 SM(Session Management) 프로토콜의 기능에 해당하는 컴포넌트로 패킷 교환 호(packet switched call)를 설정하고 해제하는 역할을 수행한다.

HGMMB(HSDPA GMM Resource Block)은 GMM (GPRS Mobility Management) 프로토콜 기능에 해당

하는 컴포넌트로 패킷 교환 서비스(packet switched service)를 위한 등록 및 인증 기능을 수행한다.

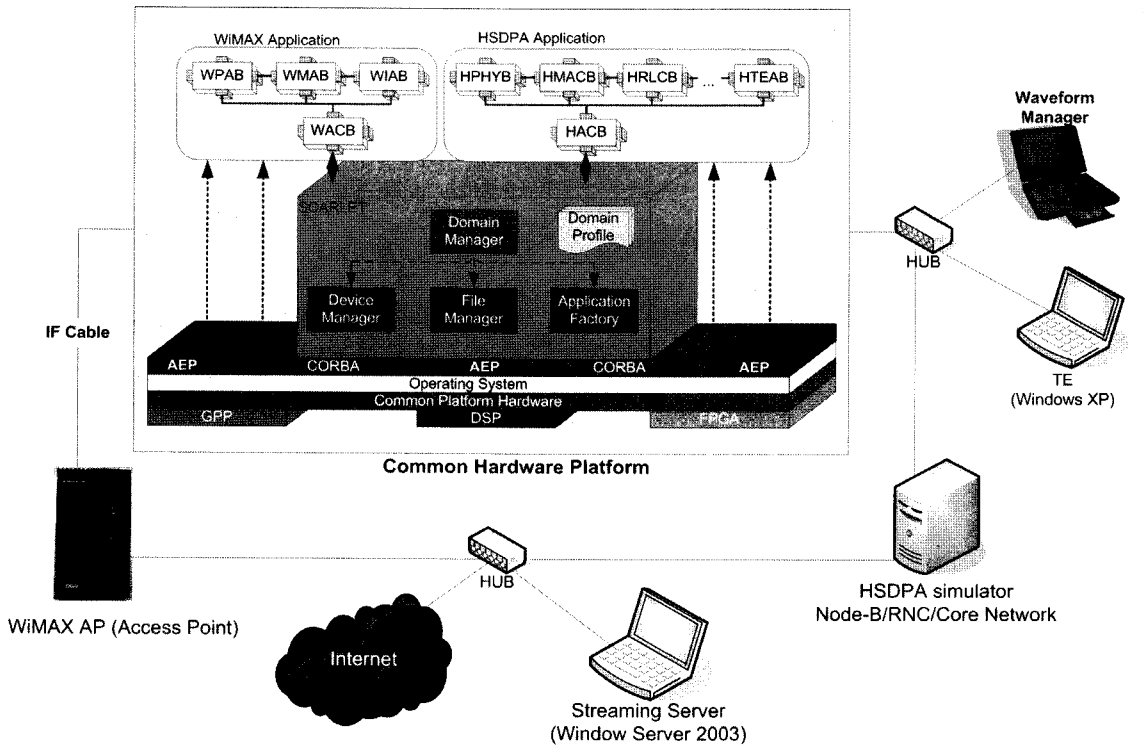
HRRCB(HSDPA RRC Resource Block)은 RRC(Radio Resource Control) 프로토콜 기능에 해당하는 컴포넌트로 UE와 UTRAN(UMTS Terrestrial Radio Access Network) 사이에서 radio resource를 사용하기 위한 필요한 모든 파라메터를 전달하고 각각의 entity들을 제어하는 역할을 수행한다.

HPDCPB(HSDPA PDCP Resource Block)은 PDCP (Packet Data Convergence) 프로토콜 기능에 해당하는 컴포넌트로 RRC에 의하여 생성된 트래픽 경로를 통하여 IP 패킷 헤더를 압축하여 전송하는 기능을 수행한다.

HRLCB(HSDPA RLC Resource Block)은 RLC(Radio Link Control) 프로토콜 기능에 해당하는 컴포넌트로 상위 계층과 하위 계층인 MAC사이에 있는 계층으로서 송/수신단 간에 신뢰성 있는 데이터 전송을 목적으로 한다.

HMACB(HSDPA MAC Resource Block)은 MAC (Medium Access Control) 프로토콜 기능을 수행하는 컴포넌트이다. MAC은 상위의 logical channel을 통해 내려오는 데이터를 하위의 transport channel로 전송시키거나 하위의 transport channel로 올라온 데이터를 상위의 logical channel을 통해 상위로 전송하는 기능 수행 및 MAC 프로토콜을 MAC-d, MAC-c, MAC-hs entity로 구분하여 각기 다른 transport channel을 담당한다. MAC-hs 프로토콜은 HS-DSCH를 통해 전송되는 데이터를 핸들링하고, HSDSCH를 위해 할당된 physical resources를 관리한다.

HPHYB(HSDPA PHY Resource Block)은 PHY 프



[그림 3] SCARLET상에서의 단말 애플리케이션 동작 실험 환경

로토콜 기능을 수행하는 컴포넌트이다. 즉, MAC에서 transport channel을 통해 내려오는 데이터를 physical channel로 전송시키거나 physical channel로 수신된 데이터를 transport channel을 통해 MAC에게 전송하는 기능을 수행한다.

4-3 공용 단말 플랫폼상에서의 동작 실험

동작 실험은 [그림 3]과 같은 환경에서 실시하였다. HSDPA의 경우 물리 계층은 시뮬레이터를 통하여 실험하였으며, Node-B, Core Network 측도 시뮬레이터를 사용하였다. WiMAX의 경우, 무선 구간에 RF 대신에 IF Cable을 사용하였으며 WiMAX AP는 실제 하드웨어를 사용하였다. SCARLET 미들웨어 플랫폼은 WiMAX/HSDPA 단말 공용 플랫폼에 실장되어 수행된다. 또한, LAN으로 연결된 원격 Waveform Manager가 단말 공용 플랫폼에 있는 SCARLET 미들웨어와 연동하여 사용자가 HSDPA 서비스를 원할 경우, HSDPA 응용 컴포넌트를 단말 공용 플랫폼에 실장한 후 HSDPA 서비스를 제공하며, 사용자가 WiMAX 서비스를 원할 경우 WiMAX 응용 컴포넌트를 단말 공용 플랫폼에 실장한 후 WiMAX 서비스를 제공하게 된다.

실제 서비스를 위한 웹브라우져나 동영상 플레이어와 같은 애플리케이션은 TE에서 실행되며, Waveform Manager에 의해 단말 공용 플랫폼에 실장된 해당 모드의 응용 컴포넌트와 연동하여 각 모드별 발신호 처리 절차를 거친 후 인터넷 서비스 및 스트리밍 서버(streaming server)를 이용한 동영상 서비스 시험을 수행하였다.

실험 결과에 따르면, WiMAX와 HSDPA 두 가지의 서비스를 기반으로, WiMAX는 컴포넌트와 작업 이전에 1.5 Mbps 정도의 전송율에서 컴포넌트화 작업 이후에 1 Mbps~1.5 Mbps의 전송율을 보여 주었다. SCA Port는 사용하지 않았고, adapter 방식으로 컴포넌트화 하였다. HSDPA는 5 컴포넌트의 경우 120 kbps~160 kbps의 전송율이 측정되었으며, 제어 명령은 SCA Port를 활용하였고, 트래픽은 UDP 소켓을 이용하였다. 9 컴포넌트 구성의 경우는 110 kbps 정도의 전송율이 나왔고, 제어 명령은 SCA Port, 트래픽은 UDP 소켓을 이용하였다. 그리고 9 컴포넌트 구성에서 제어 명령과 트래픽을 모두 SCA Port를 활용한 경우는 11~22 kbps 정도의 속도가 측정되었다. 상대적으로 HSDPA가 저속의 서비스를 보여준 것은 프로세서 성능이 부족한 것과, WiMAX에 비해서 상대적으로 많은 프로토콜 스택 및 컴포넌트 개수로 인한 통신 오버헤드에 의한 지연으로 생각된다. 특히, SCA Port의 사용 여부에 따라 성능이 많은 영향을 받음을 볼 수 있었다. 성능 면에서는 적은 수의 컴포넌트가 유리하지만, 기능에 따른 유지 관리 측면에서는 다수의 컴포넌트로의 분할이 유리하였다. 따라서 성능을 고려한, 합리적인 수준에서의 컴포넌트화가 필요함을 확인할 수 있었다.

재구성 시간(단말기가 새로운 모드로 전환하기 위해 통신모드 애플리케이션을 재구성 가능한 하드웨어상에 설치하고 시작될 때까지 소요되는 시간)은 26초에서 27초 정도가 소요되었다. 단말용 공용 하드웨어 플랫폼에서 사용된 GPP 보드상의 프로세서 속도가 200 MHz이었으나, 운영 중에는 150 MHz로 구동하였고, 메모리가 128 MB인 점을 고려한다면 어느 정도 만족스러운 성능이라고 분석된다.

GPP 보드에서의 적은 메모리 크기로 인해, 초기 시작 시 적재되는 다수의 공유 라이브러리 로딩시 로딩 위치를 계산하는데 많은 시간이 소요되는 것으로 분석되었으며, 이를 해결하기 위하여 여러 개의 공유 라이브러리를 하나의 통합 공유 라이브러리화 하거나, prelink 유틸리티를 이용하여, 공유 라이브러리의 로딩 위치를 미리 계산한 후 계산된 위치에 올리는 방식을 사용하여 초기 로딩 시간을 어느 정도 줄일 수 있었다. 그러나, 근본적으로 제한된 메모리 크기로 인하여 운영체제 및 CORBA 서비스 프로세스

가 적재되고 남은 여유 메모리로는 SCA 애플리케이션을 원활히 실행하기에는 부족하였다.

단말 하드웨어 플랫폼의 프로세서 속도 및 메모리 증설을 통하여 재구성 시간을 합리적인 시간범위 내로 단축하려는 노력과 더불어, 단말기는 기지국에 비하여 컴퓨팅 리소스가 제한적일 수 밖에 없으므로, SCARLET이 사용하는 CORBA 미들웨어, XML 파서 등의 추가적인 경량화, 최적화 개발이 필요한 것으로 판단된다[12].

V. 결 론

본 고에서는 무선 데이터 통신 서비스인 HSDPA와 WiMAX 프로토콜 규격을 만족하는 통신 애플리케이션 컴포넌트를 설계 및 구현하였다. 범용 프로세서 기반의 실험용 단말 하드웨어 상에서, SDR 기술의 핵심 중의 하나라고 할 수 있는 단말 미들웨어 플랫폼과 함께 통신 애플리케이션 재구성 기능에 대한 구현을 통해 상용화의 가능성을 검증하였다. 종래의 ASIC 기반의 무선 통신 단말은 규격이 개정되거나 새로운 규격이 출현하는 경우 새로운 칩을 출시하기까지 많은 시간과 비용이 요구되는 어려움이 있었으나, 본 SDR 단말 플랫폼에서는 통신 애플리케이션 소프트웨어의 변경만으로 언급한 문제가 해결되는 장점을 가진다.

본 고에서는 SDR 단말 소프트웨어를 구성하고, 미들웨어 및 통신 애플리케이션의 컴포넌트화 구현과 관련하여 다음과 같이 접근하였다.

SCA에서 제공하는 규격을 근거로 구현된 미들웨어 플랫폼 상에서 SDR 단말용 통신 애플리케이션 컴포넌트를 설계 및 구현하기 위해, 우선 현존하는 다양한 통신 서비스들을 분석하여 이들 중 현실성, 활용 가능성 그리고 개발 용이성 등을 고려하여, 양방향 통신 서비스 중 WiMAX와 HSDPA의 조합을 선정하였다. 선정된 서비스들을 수행하기 위하여 기존의 프로토콜들을 SCA wrapper와 SCA adapter를 사용하여 SCA의 기준에 맞도록 설계 및 변경하였고, 이들과 미들웨어 간의 연동을 위하여 assembly controller를 설계 및 구현하였다. 또한, 두 가지 모드의 통신 애플리케이션 컴포넌트들이 공용으로 수행될 수 있는 실험용 SDR 단말용 하드웨어 플랫폼도 구현하였다. 개발된 통신 애플리케이션 컴포넌트들은 실험용 하드웨어 플랫폼과 미들웨어 상에서 원활히 수행됨을 확인하였으며, Core Framework의 지시에 따라 재구성 동작이 원활히 수행됨을 확인함으로써 상용화 가능성을 검증하였다.

참 고 문 헌

[1] Jeffrey H. Reed, *Software Radio: A Modern Approach to Radio Engineering*, Prentice Hall Ptr., 2002.

[2] 김지연, 김진업, "차세대 이동 통신시스템을 위한 SDR 기술," *IT Standard Weekly*, 2002년 5월.

[3] 김덕배, 김진업, "SDR 기지국 구조/기술," 대한전자공학회학회지 제33권 2호, pp. 29-39, 2006년 2월.

[4] Walter Tuttlebee et al., *Software Defined Radio: Origins, Drivers and International Perspectives*, John Wiley & Sons, 2002.

[5] JTRS, Software Communication Architecture Specification (Ver. 2.2.2), May 2006.

[6] 김세화, 유종훈, 홍성수, "SDR(Software Defined Radio)의 소프트웨어 구조," 대한전자공학회학회지, 33(2), pp. 17-28, 2006년 2월.

[7] 김홍숙, 오상철, 박남훈, 김진업, SDR용 미들웨어 구조: SCA 코어 프레임워크, 전자통신 동향분석, 한국전자통신연구원, 21(3), pp. 71-78.

[8] OMG, Minimum CORBA Specification Version 1.0, Aug. 2002.

[9] ANSI/IEEE, POSIX 1003.13: Standardized Application Environment Profile - POSIX Realtime Application Support (AEP), IEEE Std. 1003.13, 1998.

[10] Loutfi Nuaymi, *WiMAX*, John Wiley & Sons, Jan. 2007.

[11] Harri Holma, Antti Toskala, *HSDPA/HSUPA for UMTS*, John Wiley & Sons, Jun. 2006.

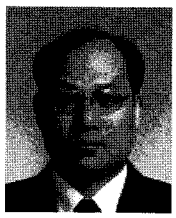
[12] 김홍숙, 김준식, 오상철, 박남훈, 김진업, "SCA 기반 SDR 단말기용 통신 컴포넌트 구현", 한국통신학회지, 24(7), pp. 47-58, 2007년 7월.

≡ 필자소개 ≡

김 준 식

1990년: 서강대학교 전자계산학과 (공학사)
1992년: 서강대학교 전자계산학과 (공학석사)
2007년: 충북대학교 정보통신공학과 (공학박사)
1992년~1998년: 효성컴퓨터㈜ 전자통신연구소 책임연구원
1999년~현재: 한국전자통신연구원 이동통신연구단 이동미들웨어연구팀 선임연구원
[주 관심분야] SDR, WiBro, 이동통신프로토콜, 이동통신단말기

박 남 훈



1983년: 전남대학교 계산통계학과 (공학사)
1987년: 중앙대학교 컴퓨터공학과 (공학석사)
1999년: 충남대학교 컴퓨터과학과 (공학박사)
1995년: 정보통신기술사 (P.E)
2002년 2월~2003년 2월: 한국무선인터넷포럼 모바일표준플랫폼 분과위원장
1988년~현재: 한국전자통신연구원 이동통신연구단 이동미들웨어연구팀 팀장
[주 관심분야] 컴퓨터 네트워크, 광대역 통신망/신호망, ATM 기술, 이동통신망, 차세대무선인터넷, 모바일컴퓨팅기술, 이동통신 단말 기술, SDR 및 Cognitive Radio기술

김 진 업

1985년: 고려대학교 전자공학과 (공학사)
1987년: 한국과학기술원 전기 및 전자공학과 (공학석사)
1996년: 한국과학기술원 전기 및 전자공학과 (공학박사)
1987년~현재: 한국전자통신연구원 이동통신연구단 이동컨버전스연구그룹 그룹장
[주 관심분야] SDR기술, Cognitive Radio기술, 이동통신 시스템, Data compression, 채널 코딩

권 오 준

1986년: 경북대학교 전자공학과 (공학사)
1992년: 충남대학교 전산학과(이학석사)
1998년: 포항공대 전자계산학과 (공학박사)
1986년~2002년: 한국전자통신연구원 선임연구원
2000년~현재: 동의대학교 컴퓨터소프트웨어공학과 부교수
2007년 7월~현재: 한국전자통신연구원 초빙연구원
[주 관심분야] 컴퓨터네트워크, 정보보호, 무선 인터넷, 패턴인식, 인공신경망

김 남

1981년: 연세대학교 전자공학과 (공학사)
1983년: 연세대학교 전자공학과 (공학석사)
1988년: 연세대학교 전자공학과 (공학박사)
1992년~1993년: 미국 Standford University 방문교수
2001년~2001년: 미국 California Technology Institute(Caltech) 방문교수
1999년~2000년: 충북대학교 컴퓨터정보통신연구소장
1989년~현재: 충북대학교 전기전자컴퓨터공학부 교수
1992년~현재: 충북대학교 컴퓨터정보통신연구소
[주 관심분야] 디지털 이동통신, 무선시스템, 전자파해석, EMI/EMC, Diffractive Optics, WDM Optical Filter & DEMUX Optical Memory, Holography Application