제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

SyncML 을 이용한 플랫폼 독립적인 동기화 서버의 설계

이민순*, 이병수**
*인천대학교 전자계산과
e-mail : zerone@incheon.ac.kr

Design of Platform Independence Synchronization Server Using SyncML

Min-Soon Lee*, Byoung-Soo Lee**
*Dept. of Computer Engineering, University of Incheon

요 약

다양한 제품의 유무선 단말간의 데이터 동기화 표준인 SyncML 을 이용한 동기화 서버를 설계한다. 이 논문에서는 SyncML 에 대한 개략적인 설명과 SyncML Initiative 에서 제공하는 레퍼런스 툴킷과 SDA 를 살펴보고, 자바에서 플랫폼 의존적인 프로그래밍을 가능하도록 하는 JNI 기술의 유용성과 동기화 서버에서의 사용범위에 대하여 논의하고, 이 논문에서 설계한 플랫폼 독립적인 동기화 서버에 대하여 살펴본다. 플랫폼 독립적인 SyncML 과 Java 프로그래밍 언어를 이용하여 개발된 동기화 서버는 다양한 웹 서버 환경과 다양한 운영체제에서 운용되어질 수 있다.

1. 서론

무선 네트워크와 휴대 단말기 기술이 발달함에 따라서 사용자들은 모바일(Mobile) 컴퓨팅 환경하에 시간과 공간의 제약에서 벗어나 자유롭게 자신의 정보를 접근, 수정, 저장, 전송할 수 있게 되었다. 그러나 대부분의 사용자에게 휴대 단말기는 일시적인 데이터 입출력 및 조작 기기일 뿐이며, 실질적인 데이터 저장소는 유무선 네트워크에 접속되어있는 서버이다. 따라서 휴대용 단말기 사용자는 데이터 저장소와의 데이터 동기화(Synchronization)를 통하여 데이터의 일치성을 확보하여야 한다.

초기 휴대 단말기들은 벤더(Vender)에서 제공하는 응용 프로그램을 PC 나 서버에 설치하고, 휴대 단말기에 내장된 클라이언트 응용 프로그램을 통하여 데이터 동기화 기능을 제공하였다. 그러나, 다른 벤더에서 개발된 단말기와의 데이터 동기화를 수행하는 데에는 한계가 있게 되었고, 동기화 프로토콜의 표준화 필요성이 대두되었다.[1,2]

*본 연구는 한국과학재단 지정 인천대학교 동북아전자물류센터의 지원에 의한 것임.

이러한 요구에 따라, SyncML Initiative 가 결성되게 되었다. 2000 년 12 월 SyncML 1.0 스펙(Specification)이 발표되었으며, 2001 년 6 월 1.0.1 그리고 2002 년 2 월 1.1 스펙이 발표되었다.

SyncML Initiative 의 회원 구성은 스폰서(Sponsors), 프로모터(Promoters)와 서포터(Supporters)로 되어있으며, 국내에서는 10 개의 기관이 서포터로 활동하고 있다.[1]

SyncML Initiative 에서는 SyncML 스펙에 따라 개발된 제품(Product)들에 대하여 호환성 테스트를 실시하고 있으며, 현재 28 개의 서버와 58 개의 클라이언트가 호환성 테스트를 통과하였다.

본 논문의 2 장에서는 SyncML 에 대한 소개와 레퍼런스 툴킷(Reference Toolkit), SDA(SyncML Demo Application)에 대하여 살펴보고, 자바 프로그래밍 언어와 다른 프로그래밍 언어와의 인터페이스를 지원하는 JNI 기술에 대하여 살펴보며, 이 논문에서 설계하는 시스템에서의 이용 범위에 대하여 살펴본다. 3 장에서는 SyncML 과 Java 를 이용한 동기화 서버의 프레임웍을 제시하였다. 4 장에서는 결론을 짓도록 한다.

2. 관련연구

SyncML 은 XML(Extensible Markup Language) 기반의 데이터 동기화 프로토콜 표준이다. 이동 전화기나 PDA 등의 단말기와 같은 유무선 네트워크 환경에서 여러 장치간 데이터 동기화를 위해서 이용되어진다.

클라이언트와 서버간 데이터 동기화를 위한 시스템 구조는 그림 1 과 같다[2].

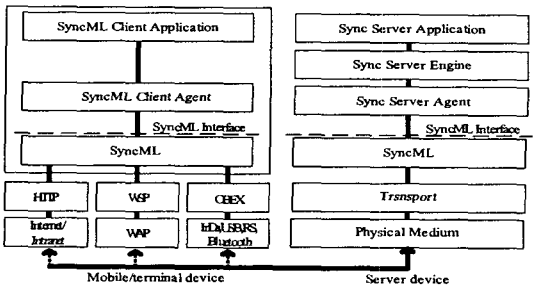

그림 1. SyncML 프로토콜 아키텍처

위 그림 1 에서 클라이언트와 서버 사이에는 물리적인 미디어(Physical Medium) 레이어(Layer), 전송(Transport) 레이어, SyncML 인터페이스 레이어와 에이전트(Agent) 레이어가 공통으로 존재한다. 클라이언트와 서버의 차이점은 서버에 동기화 엔진(Sync Engine)이 존재한다는 것이다. 물론 클라이언트 측에도 동기화 엔진이 존재할 수 있으나, 많은 자원(Resource)을 필요로 하는 동기화 서버의 특징으로 서버에 동기화 엔진이 존재하는 것이 일반적이다.

SyncML 의 전송 레이어에서는 웹(World Wide Web)에서 이용되어지는 HTTP 통신뿐만 아니라, 무선통신의 WSP (*Wireless Session Protocol*), 근거리 통신망에서 이용하는 OBEX(OBject EXchange) 등을 지원할 수 있다.

2.1 레퍼런스 툴킷

SyncML Initiative 에서는 SyncML 동기화 서버와 클라이언트 개발자(Supporters)들을 위하여 레퍼런스 툴킷이라는 공개 소스를 제공하고 있다. 레퍼런스 툴킷은 SyncML.org 사이트에서 개발 관리하였으나, 현재는 공개 프로젝트로 진행되어지고 있다.[5]

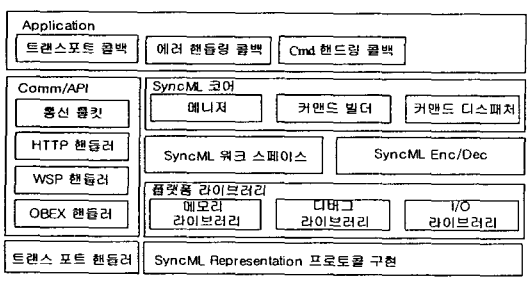

그림 2.레퍼런스 툴킷 아키텍처

그림 2 는 크게 응용프로그램(Application) 레이어와 통신(Communication) 레이어, SyncML 코어(Core) 레이어, 플랫폼 라이브러리(Paltform Library) 레이어로 구성되어 있다. 통신 레이어의 경우 다양한 형태의 통신 방법을 지원할 수 있도록 되어 있으며, HTTP, WSP, OBEX 핸들러의 구현으로 되어있다. SyncML 코어 레이어의 경우 플랫폼 독립적인 SyncML API 를 제공하며, SyncML 메니저, 커맨드 빌더, 커맨드 디스패처로 구성되어 있다. 이러한 코어 모듈 하단에는 플래폼에 의존적인 플랫폼 라이브러리가 존재하게 된다. 이러한 두 개의 레이어는 응용프로그램 레이어의 콜백 함수들을 이용하여 상호 운영되게 된다. 이 논문에서는 설계하는 시스템은 플랫폼 라이브러리 부분을 플렛폼에 의존하지 않도록 구성한다.[5,13]

2.2 SDA

그림 2 의 레퍼런스 툴킷은 SyncML 메시지를 만드는 핵심 API 를 제공하지만, 동기화 방식이나 이에 따른 동기화 엔진, DB 연계에 대한 부분을 다루지는 않는다. 따라서 실질적인 응용 프로그램을 개발 하기 위해서는 더 많은 기능 들이 필요하게 되면 SyncML Initiative 에서는 SDA(SyncML Demo Application)라는 것을 통하여 이러한 목적을 이루고 있다. 다음 그림 3 은 SDA 의 서버 시스템 구조를 나타내고 있다.[14]

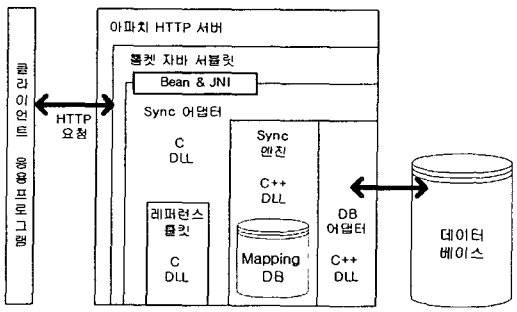

그림 3. SDA 서버 아키텍처

SDA 의 서버 시스템은 아파치 그룹의 아파치 웹 서버와 톰캣 서블릿 엔진을 이용하여 클라이언트 응용프로그램과의 HTTP 통신 메커니즘을 제공한다. SyncML 서버에는 Sync 엔진, Sync 어댑터, 레퍼런스 툴킷, DB 어댑터로 구성되어 있다.

클라이언트에서 생성된 동기화 메시지는 레퍼런스 툴킷의 HTTP 통신 핸들러를 통해 Sync 서버로 전송하며, Sync 서버는 해당 IP 와 포트로 클라이언트의 매시지를 수신한다. 웹서버는 수신한 메시지를 서블릿으로 전송하며, 서블릿은 자바빈(Bean)과 JNI(Java Native Interface)를 이용하여, Sync 어댑터에 메시지를 보낸다. Sync 어댑터는 수신한 메시지를 레퍼런스 툴킷의 SyncML 디코딩(Decoding) 모듈을 통하여 프로그램이 인식할 수 있는 SyncML 메시지로 변환한다. 변환된 메시지는 Sync 엔진에 보내지며, 메시지의 종류에 따라서 데이터베이스 어댑터를 이용하여 데이터베이스

에 데이터를 추가, 삭제, 수정한다. 이러한 데이터 베이스 변경작업의 결과는 서버의 "Mapping DB"에 클라이언트 데이터와 서버데이터와의 매핑 관계를 기록하게 된다.

2.3 JNI

시스템에 의존적인 프로그래밍을 가능하게 하는 자바의 JNI(Java Native Interface) 기술을 사용하는 목적은 다음과 같다. [6,7]

- 자바 프로그래밍 언어를 이용하여 다른 프로그래밍 언어로 개발되어진 모듈을 이용할 때.
- 실행 속도의 향상을 위하여 하드웨어와 운영체제에 특화된 코드를 작성하고자 할 때.
- 자바 언어가 지원하지 않는 시스템의 기능을 다른 프로그래밍 언어로 구현하고자 할 때.

그림 3 에서 자바 빈즈가 JNI 기술을 통하여 SDA 에서 설계된 Sync 어댑터와 인터페이스하게 된다. 그러나 그림 3 에서 사용한 JNI 기술은 앞에서 기술한 이미 개발되어진 모듈과의 인터페이스를 위해 사용되어진 것이다. 즉, 이 논문에서는 그림 3 과 같은 JNI 기술은 사용되지 않는다.

본 시스템에서 설계한 동기화 서버에서는 자바 언어에서 지원하고 있지 않은 WSP 와 OBEX 통신을 지원하기 위하여 JNI 기술을 필요로 할 수 있다.

3. 동기화 서버의 설계

본 장에서는 자바 언어에 기반한 플랫폼 독립적인 SyncML 서버 시스템을 설계하며, 각 구성 요소들에 대하여 설명한다. 본 시스템은 그림 2 에서 나오는 " SynML 코어" 의 구성요소들 중 플랫폼 독립적이지 못한 요소들인 " 플랫폼 라이브러리" 를 자바 언어로 대체하며, 본 시스템은 그림 3 의 서블릿에서 SyncML 어댑터에 접근하기 위해, 이용하는 자바의 JNI 기술을 이용하지 않는다. 그림 4 는 플랫폼 독립적인 레퍼런스 툴킷 아키텍처를 나타내고 있다.

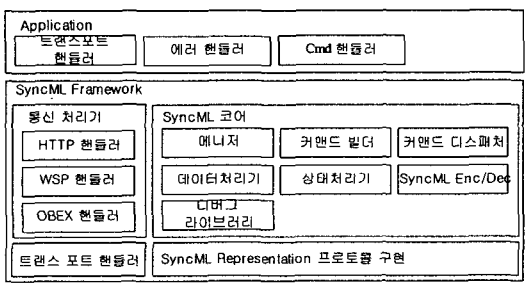

그림 4.플랫폼 독립적인 레퍼런스 툴킷 아키텍처

그림 4 는 본 시스템에서 설계한 플랫폼 독립적인 SyncML 동기화 서버의 핵심 아키텍처로 " 통신 처리기" 와 " SyncML 코어" 로 구성되어져 있다. SyncML 코어는 그림 2 의 " SyncML 코어" 블록과 " 플랫폼 라이브러리" 블록을 통합한 형태이다.

그림 2 의 레퍼런스 툴킷의 경우 구조적 프로그래밍 언어인 C 언어로 개발되어있어, 시스템에 의존적인 메모리 관리와, 디버그 라이브러리, I/O 라이브러리를 이용할 수 밖에 없었다. 본 시스템은 레퍼런스 툴킷 전체를 객체지향 언어인 자바 언어로 개발 가능하도록 설계하였다.

SyncML 코어의 데이터처리기는 SyncML 의 Data 나 Item 엘러먼트에 포함된 동기화 데이터를 처리하기 위한 지원 모듈이며, 디버그 라이브러리는 JUnit 을 이용하여 플랫폼에 의존하던 기존의 디버그 라이브러리를 대체하였다.[8]

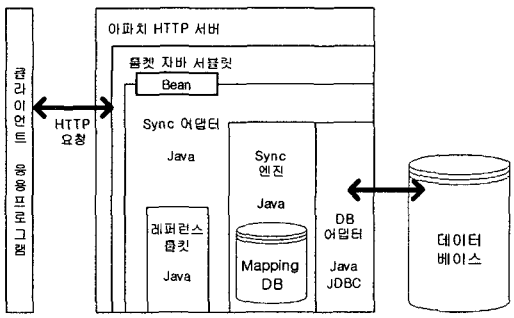

그림 5. 플랫폼 독립적인 동기화 서버 아키텍처

위의 그림 5 는 그림 3 의 시스템 의존적인 부분들을 모두 제거한 후의 SyncML 서버 아키텍처를 나타내고 있다. SDA 에서는 Sync 어댑터가 C 언어로 개발되어져 있어 서블릿과 Sync 어댑터 간의 통신시 자바가 아닌 다른 언어로 개발된 모듈을 이용하기 위하여 JNI 를 이용하여야 했다. 그러나, Sync 어댑터가 자바로 개발되어짐으로써 이러한 시스템 의존적인 부분을 제거할 수 있다. 클라이언트와 웹서버 및 동기화 서버간의 통신시에 HTTP 통신을 이용하고 있다. HTTP 통신은 자바 언어에서 기본적으로 지원하므로 JNI 기술을 필요로 하지 않는다. 그러나 WSP 나 OBEX 를 이용하기 위해서는 이들 프로토콜을 지원하는 라이브러리를 JNI 를 이용하여 인터페이스 하거나, 자바 프로그래밍 언어로 직접 이들 프로토콜을 개발해야 하는 부담을 지어야 한다. 이는 자바언어에서 직접 이들 프로토콜을 지원하기 전까지는 해결하기 쉽지 않은 문제이다. 호환성 테스트를 통과한 28 개의 서버 제품들은 모두 HTTP 통신을 지원하고 있으며, WSP 를 지원하는 제품은 Openwave Systems, Inc 외에는 몇 개 되지 않는다. OBEX 프로토콜을 지원하는 제품은 현재 없다.[3]

4. 결론

다양한 단말기들과 통신 방법을 이용한 모바일 컴퓨팅이 더욱 일반화 될 것이다. 사용자는 더욱 다양한 데이터를 비 동기적으로 생성해 낼 것이며, 이를 동기

화 하길 원할 것이다. 따라서, 사용자의 이러한 다양한 요구에 부응하기 위한 서버 시스템의 확장성과 플랫폼 독립성을 필요로 하게 된다. 본 시스템은 자바 언어의 플랫폼 독립성을 최대한 이용하여 설계 되어 졌다. 따라서 이러한 변화에 능동적으로 대처할 수 있다.

현재는 HTTP 통신을 제외한 OBEX 나 WSP 관련 클래스가 JDK(Java Development Kit)에 포함되지 않은 관계로, 본 서버에서는 HTTP 프로토콜만을 지원하고 있다. 자바 표준화 협의체인 JCP(Java Community Process)에서 USB 및 Bluetooth 관련 부분을 논의하고 있으니, WSP 및 OBEX 도 논의 될 것으로 본다.[9] 향후 연구 과제로는 제안한 플랫폼 독립적인 동기화 서버 시스템을 개발하고, 동기화 클라이언트와의 운영을 검증하는 것이 필요하다.

참고문헌

[1] SyncML Initiative, Introduction(http://www.syncml.org)

[2] SyncML Initiative, Building an Industry-Wide Mobile Data Synchronization Protocol, SyncML White Paper, Mar 30, 2000

[3] SyncML Initiative, Interoperability (http://www.syncml.org /interop-over.html)

[4] SyncML Initiative, Sync Protocol Version 1.1 2002.02.15

[5]Sourceforge.net(http://www.sourceforge.net/projects/syncml-ctoolkit/)

[6] Beth Stearns, "Java Native Interface" (http://java.sun.com /docs/books/tutorial/native1.1)

[7] Dov Bulka. "Java PERFORMANCE and SCALABILITY Volume 1 Server-Side Programming Techniques", Addison Wesley, pp 161, 2000.06

[8] JUnit.org(http://www.junit.org)

[9] JCP.org (http:/www.jcp.org/jsr/all/)

[10] 조진현, 최훈, 김경용. "SyncML 서버 데이터 동기 엔진의 설계 및 구현", 한국정보과학회, Vol.28 No.02 pp.0580 – 0582, 2001.10

[11] 이종필, 최훈, 윤대균. "SyncML 적합성 및 상호 연동성 시험 연구", 한국정보과학회, Vol.28 No.02 pp.0724 – 0726, 2001.10

[12] 류수희, 최훈, 류시원. "SyncML 프로토콜을 이용한 데이터 동기화 서버 Agent 설계 및 구현", 한국정보처리학회, 제 8 권 제 2 호, Vol.08 No.02 pp.1347 – 1350, 2001.10

[13] 하인숙, 조재혁, 양지현, "SyncML 레퍼런스 툴킷 그 내부를 보자", 마이크로 소프트웨어 2001 년 7 월, pp.324-336, July 2001

[14] 하인숙, 조재혁, 양지현, "모든 동기화 프로그램의 조상 SDA 분석" 마이크로 소프트웨어 2001 년 8 월 pp.309-313,Aug 2001