SyncML 기반의 자료 동기화 클라이언트 개발
(Development of Data Synchronization Client based on SyncML)

장 대 진 † 박 기 현 †† 주 홍 택 †††
(DaeJin Jang) (KeeHyun Park) (Hong Taek Ju)

요 약 최근 이동컴퓨팅 기술과 하드웨어 기술의 발달로 인해 무선 이동 단말기가 매우 급속하게 보급되고 있다. 무선 이동 단말기 사용자는 시간이나 장소에 구애받지 않고 원하는 정보를 이용하기 위해서는 중앙 통합 서버의 자료와 이동무선통신 단말기의 자료를 일치시키는 작업이 필요하며, 이를 자료 동기화(Data Synchronization)라고 한다. 현재 주요 단말기 제조 회사에서 제공하는 데이타 동기화 솔루션이 존재하지만, 각각의 단말기와 응용 서비스 간의 호환성이 결여되어 있다. 이러한 문제점을 해결하기 위하여 이동무선통신 업체가 주축이 된 OMA(Open Mobile Alliance)에서 SyncML(Synchronization Markup Language) 자료 동기화 방법을 제안하여 공개적인 표준화를 시도하고 있다. 본 논문에서는 SyncML 표준 규격을 준수하면서 PDA와 같은 이동무선 단말기에서 PIMS 데이타를 동기화하는 자료 동기화 클라이언트를 개발하였다.

키워드 : 자료 동기화, 동기화 마크업언어, 이동 컴퓨팅, 개인정보관리시스템

Abstract According to the recent advancement of mobile computing technology and the hardware technology, the mobile device has rapidly come into wide use. The data in the central management server needs to confirm to that in mobile devices in order for the mobile device users to access the needed data regardless of time and place. This is called *Data Synchronization*. Currently, major mobile device manufacturing companies provide the data synchronization solutions, but these solutions are not compatible with other mobile devices/other applied services. In order to solve this problem, OMA(Open Mobile Alliance), mainly composed of mobile telecommunication companies, has suggested using the SyncML(Synchronization Markup Language) data synchronization method. This is an attempt to make a public standard. In this paper, the data synchronization client that synchronizes the PIMS (Personal Information Management System) data in mobile devices like PDA is developed, complying with the SyncML standard requirements.

Key words : Data Synchronization, SyncML, Mobile Computing, PIMS

1. 서 론

이동 컴퓨팅은 PDA, Smart Phone, Mobile Phone 등과 같은 이동무선통신 단말기들을 사용하여 어느 장소에서나 개인 혹은 비즈니스 정보에 대한 접근을 가능하게 해준다. 또한 이동무선통신 환경이 다양해지면서 각 단말기 제조 회사에서 제공하는 이동무선통신 단말기들의 종류가 다양해지고, 한 개인이 소유하고 있는 단말기의 수도 늘어나고 있다. 이러한 무선 네트워크 기반 기술의 발달과 서비스의 다양화로 인해, 이동무선통신 단말기는 일정관리나 전자메일 교환과 같은 개인적인 용도에서 기업 운영이나 업무의 효율성을 향상시키는 비즈니스 영역에 대한 새로운 수단으로 그 역할이 확대되고 있다.

하지만, 이동무선통신 환경이 가지고 있는 네트워크 서비스 범위의 한계로 인하여 각각의 단말기들은 통합 관리서버와 항상 연결 상태를 유지할 수 없으며, 연결 비용에 따른 경제적인 부담도 무시하지 못하는 실정이다[1-4]. 또한, 단말기의 네트워크 서비스에 대한 연결 혹은 비연결 상태 동안 처리 대상이 되는 자료가 갱신될 수 있고, 한 개인이 다수의 단말기를 소지하고 있으므로 동일한 자료에 대해서 일치성(Consistency)을 항

† 학생회원 : 계명대학교 정보통신학부
djjang19@kmu.ac.kr
†† 종신회원 : 계명대학교 정보통신대학 교수
khp@kmu.ac.kr
††† 비 회 원 : 계명대학교 정보통신대학 교수
juht@kmu.ac.kr
논문접수 : 2004년 7월 14일
심사완료 : 2005년 5월 7일

상 보장할 수 없다. 이와 같이, 비연결 상태 동안에 갱신된 자료 항목 및 다수의 단말기들에 분산되어 있는 동일한 자료 항목들에 대해서 최신 값으로 일치시키는 동작을 자료 동기화(Data Synchronization)라고 한다[1,4].

현재 각 단말기 회사에서 제공되는 자료 동기화 방식이 존재하지만, 각각의 제조회사 장치들간의 상호 운용성을 보장할 수 없으므로, 단말기 사용자나 단말기 제조회사, 응용 소프트웨어 개발자 그리고 서비스 제공자 모두에게 많은 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해서 자료 동기화 표준의 필요성이 대두되었고, 2000년 2월 IBM, Lotus, Nokia, Palm, Motorola, Psion, Starfish Software 등의 이동무선통신 관련 업체들을 중심으로 한 연합체인 OMA(Open Mobile Alliance)에서 SyncML(Synchronization Markup Language)을 제안하여 공개적인 표준화를 주도하고 있다[3-6].

본 논문에서는 OMA에서 제안한 공개 표준인 SyncML 기반의 자료 동기화 클라이언트를 설계하고 구현하였다. SyncML 기반의 자료 동기화 클라이언트는 SyncML 규격 1.1을 준수하면서 이동무선통신 환경의 특성 및 상대적으로 처리속도가 낮은 CPU, 제한된 저장 공간에서 구동하는 내장형 소프트웨어의 특성을 고려하여 설계 및 구현되었다. 또한 구현된 동기화 클라이언트의 적합성을 검증하기 위해서 OMA 상호 운용성 시험에[7,8] 합격한 Synthesis 서버와[9] 연동하여 자료 동기화를 수행하였다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구 부분으로서, SyncML에 대한 소개 및 규격, 기존의 SyncML 기반의 자료 동기화 클라이언트 개발 및 연구 사례에 대해서 분석하고, 3절에서는 SynML 기반의 자료 동기화 클라이언트의 설계 및 구현에 대해서 설명한다. 4장에서는 구현된 동기화 클라이언트의 자료 동기화 적합성에 대한 검증 결과를 제시한다. 마지막으로, 5장에서는 결론을 제시하고 향후 연구 방향에 대해서 논의한다.

2. 관련 연구

2.1 SyncML

SyncML은 각 제조회사의 단말기가 가지고 있는 서로 다른 플랫폼, 통신 프로토콜, 자료 형태, 응용 서비스에 이용될 수 있는 자료 동기화 방식과 장치 관리에 대한 개방형 표준 인터페이스 개발을 목적으로 한다. 또한, SyncML은 개방형 표준과 다수의 구현 방식을 따르는 것을 채택함으로써 자료 동기화에 대한 사실상의 표준이 되는 것을 목적으로 한다. 표 1은 각 단말기 제조회사에서 제공하는 동기화 방식에 대해서 보여주고 있다[2,8,10].

표 1 각 단말기 동기화 방식

단말기 제조 업체	단말기 플랫폼	동기화 방식
팜(Palm) PDA	Palm OS	HotSync
휴렛팩커드(HP) PDA	Windows CE	ActiveSync
샤프(Sharp) 자우루스 PDA	Embedded Linux	IntelliSync
노키아(Nokia) 스마트폰	Symbian OS	SyncML

각 단말기 제조사들의 동기화 방식이 다르기 때문에, 관련 사업자 및 제조업체들은 특정한 단말기들만을 사용하거나 모든 동기화 방식들을 지원할 수 있도록 시스템을 내부적으로 다양하게 구성해야 하는 어려움이 발생한다. 그런데, 공개 표준인 SyncML 방식을 사용하게 되면, 이런 어려움이 대부분 해소될 수 있다. 그림 1은 이러한 내용을 그림으로 보여준다.

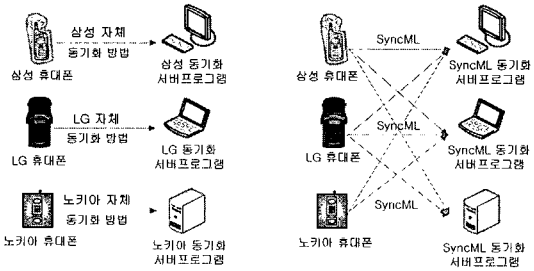

그림 1 SyncML 적용 전과 적용 후

SyncML은 이동무선통신 단말기와 서버간의 원격 동기화를 목표로 설계되었으나, 로컬 동기화와 유선 네트워크로 연결된 장치들 간의 동기화에도 사용될 수 있다. SyncML을 기반으로 하는 클라이언트와 서버간에 동기화가 수행되는 동안, 동기화 대상이 되는 객체는 논리적으로 패키지를 교환한다. 또한, 실제로 각 패키지는 이동무선통신 단말기의 제한된 자원과 무선 통신상의 낮은 대역폭으로 인해 다수의 메시지들로 분리되어 교환된다. 한 번의 완전한 자료 동기화를 위해 클라이언트와 서버 간에 주고받은 모든 패키지를 세션(Session)이라고 하며, 세션은 여러 개의 패키지를 포함하고 하나의 패키지는 여러 메시지를 포함한다. 그림 2는 SyncML을 기반으로 하는 클라이언트와 서버 간의 자료 동기화를 도식화하고 있다[6,11].

SyncML 클라이언트가 먼저 서버에게 갱신된 자료를 포함한 SyncML 메시지를 전송하면, SyncML 서버는 클라이언트가 요청한 동기화 타입에 의해 서버측 자료와 동기화 작업을 수행한 후, 다시 클라이언트에게 작업 결과 및 서버 자신의 변경된 자료도 전송한다. 이러한 수차례의 패키지 교환을 수행함으로써 자료 동기화가 이루어지며, 클라이언트와 서버 간의 해당 자료에 대하

여 일치성을 보장하게 된다[12-14].

2.2 SyncML 규격

SyncML 자료 동기화 규격은 XML 기반의 자료 표현(Data Representation) 프로토콜, SyncML 동기화(Synchronization) 프로토콜 그리고 전송 바인딩(Transport Bindings) 프로토콜로 구성되어 있다[1,4,12,13].

2.2.1 XML 기반의 자료 표현 프로토콜

SyncML 자료 표현 프로토콜은 자료 동기화를 위해 교환되는 SyncML 메시지의 논리적인 구조와 형태를 XML 형식으로 정의하고 있다. 각각의 필드가 어떠한 정보를 담고 있으며 해당 정보가 어떤 의미를 내포하는 것인지에 대한 약속을 정의하고 있다.

SyncML은 메시지의 구조체를 정의하기 위해 세 가지의 규격을 제시하고 있다. 자료 표현 프로토콜은 SyncML의 기본 표현을 기술한 핵심 규격이며, 장치 정보 표현을 위한 SyncML 장치 정보(Device-Information) DTD 규격과 추가적인 동기화 정보와 해석을 표현하기 위한 SyncML 메타 정보(Meta-Information) DTD 규격을 포함하고 있다. 장치 정보에 대한 DTD와 메타 정보 DTD는 자료 표현 프로토콜에 캡슐화해 사용한다. 자표 표현 프로토콜로 작성된 메시지는 일반 텍스트 형식을 가지고 있는 XML 양식이거나 WBXML(Wireless Binary XML) 양식을 가지고 있다[14].

2.2.2 동기화 프로토콜

SyncML 동기화 프로토콜은 SyncML 클라이언트와 서버 간에 이루어지는 자료 표현 프로토콜 규격에 의해 생성된 자료의 추가, 삭제, 갱신과 같은 동기화 명령과 그 밖의 상태 정보에 대한 메시지가 교환되는 방법에 대해서 정의하고 있다. 일반적으로 동기화 처리는 클라

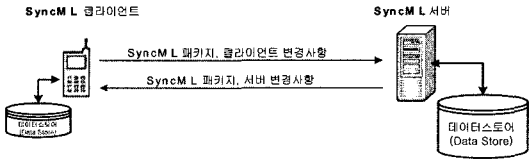

그림 2 SyncML 기반의 클라이언트-서버 간의 자료 동기화

이언트가 먼저 SyncML 메시지를 서버측으로 전송하고, 서버는 클라이언트가 전송한 메시지 내의 자료와 동일한 자신의 데이타스토어에 저장되어 있는 자료 간에 동기화를 수행한 후, 응답 메시지를 보내는 일련의 과정을 통해서 이루어진다. 표 2는 SyncML 동기화 프로토콜의 7가지 모드를 기술하고 있다[15].

2.2.3 전송 바인딩 프로토콜

SyncML이 동기화 메시지를 전송하기 위해서 사용하는 전송 바인딩 프로토콜은 HTTP, WSP, OBEX 등과 같은 프로토콜인데, SyncML 규격에서는 이러한 전송 바인딩 프로토콜에 대해서 새로운 프로토콜을 정의하는 것이 아니라, 기존의 전송 프로토콜과 바인딩 규칙만을 정의한다. 따라서, 자료 표현 프로토콜과 동기화 프로토콜이 전송 프로토콜과 독립적으로 구성되어 있으므로 향후 다른 전송 프로토콜과의 바인딩이 가능하다[1,12].

2.3 사례 연구

2.3.1 Sync4j 동기화 시스템

sync4j.funambo.com에서 소스가 공개된 동기화 시스템이다. SyncML 기반의 Sync4j 동기화 서버는 자바를 이용하여 개발되었고, SyncML 기반의 Sync4j 클라이언트는 자바와 C++를 이용하여 개발되었다. Pocket PC 기반에서 클라이언트가 동작하며, Microsoft Exchange 서버와 연동하여 주소록이나 이벤트와 같은 PIMS 정보를 동기화 할 수 있다. 현재 자료 동기화 기능을 확장하

표 2 SyncML 동기화 모드

동기화 모드	설 명
Two-way Sync	가장 일반적인 동기화 방법으로 SyncML 클라이언트와 SyncML 서버 모두 변경 항목을 교환하여 수행함.
Slow Sync	Anchor 정보 교환에서 오류가 발생하거나 SyncML 클라이언트와 SyncML 서버가 최초로 동기화를 시도하는 경우 클라이언트는 서버로 모든 자료를 전송하고 서버는 이 자료를 자신의 자료와 일대일 비교한 후 그 차이점에 대해서만 클라이언트 측에 동기화를 요청함.
One-way Sync from Client only	SyncML 클라이언트는 자신의 변경 항목에 대해서만 SyncML 서버와 자료 동기화를 수행하고, SyncML 서버의 변경 항목에 대해서는 자료 동기화를 수행하지 않음.
One-way Sync from Server only	SyncML 클라이언트는 SyncML 서버의 변경된 항목에 대해서만 자료 동기화를 수행하고, 자신의 변경된 항목에 대해서는 자료 동기화를 수행하지 않음.
Refresh Sync from Client only	SyncML 클라이언트가 자신의 모든 자료 항목을 SyncML 서버로 전송하는 방식이며, 서버는 클라이언트로부터 전달된 데이타로 자신의 값을 변경.
Refresh Sync from Server only	SyncML 서버가 자신의 모든 자료 항목을 SyncML 클라이언트로 전송하는 방식이며, 클라이언트는 서버로부터 전달된 데이타로 자신의 값을 변경.
Server Alerted Sync	위의 6가지 동기화 모드는 클라이언트가 서버에 동기화를 요청하는 방식임에 비해서, 이 방식은 SyncML 서버가 SyncML 클라이언트 측에 동기화를 요청.

여 이동무선통신 단말기 관리 기능까지 추가되어 있다[16].

2.3.2 Synthesis 동기화 시스템

SyncML 국제인증(SyncFest)에서 적합성 시험과 상호 운용성 시험에 합격한 상용화 제품의 SyncML 클라이언트와 서버이다. Synthesis 서버는 PHP 기반으로 MS-SQL ODBC를 연동하여 웹을 통하여 PIMS 데이타 관리가 가능하며, Synthesis 클라이언트는 PIMS 정보를 vCard, vCalendar 형식의 XML 파일로 변환하며, Synthesis에 내장된 파서는 각각의 동기화 정보에 대해서 각 필드별로 비교하여 동기화를 수행한다[9].

2.3.3 SyncLE

국내의 Neosteps라는 회사에서 개발한 SyncML 기반의 동기화 시스템이다. SyncML 국제 적합성 시험과 상호 운용성 시험에 합격하였으며, SyncML 스펙 1.1.1을 준수하고 있다. 셀룰러 폰이나 PDA와 같은 이동무선통신 단말기뿐만 아니라 MS-Windows가 설치된 PC 플랫폼 환경에서도 동작하는 동기화 솔루션이다[17].

2.3.4 세션 매니저를 이용한 SyncML 동기화 시스템 (충남대)

충남대에서 개발한 세션매니저를 이용한 SyncML 동기화 시스템은 SyncML 기반의 동기화 서버이다. 본 동기화 시스템은 세션을 유지하고 관리하는 프레임 및 동기화 자료를 저장하기 위한 인터페이스 프레임을 정의하여 구현된 SyncML 기반의 동기화 서버이다[19].

3. 동기화 클라이언트 설계 및 구현

3.1 동기화 클라이언트 설계

본 연구에서 제안된 자료 동기화 클라이언트는 동기화 엔진, 데이타스토어(DataStore: DS) 어댑터, 클라이언트 정보, 동기화 정보, 사용자 인터페이스 부분 그리고 SyncML 툴킷(Tool-Kit) 부분으로 설계하였다. 그림 3은 SyncML 기반의 자료 동기화 클라이언트의 구조를 보여주고 있다.

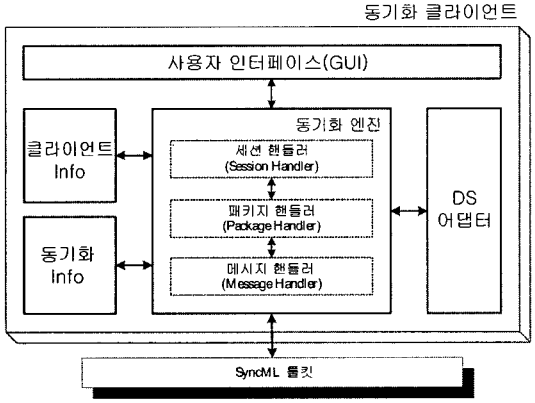

그림 3 자료 동기화 클라이언트 구조

3.1.1 동기화 엔진

본 연구에서 제안한 동기화 클라이언트의 엔진 부분은 동기화 단위에 따라 세션 핸들러(Session Handler), 패키지 핸들러(Package Handler), 메시지 핸들러(Message Handler) 세 부분으로 구성되어 있다. 동기화 엔진 모듈에서 각 핸들러에 대한 기능은 다음과 같다.

• 세션 핸들러

세션 핸들러(Session Handler)는 툴킷에서 연결된 전송 프로토콜 형태에 따라 선택된 해당 프로토콜의 세션에 대한 제어 기능과 동기화 메시지 헤더 부분을 생성하는 기능을 담당한다. 그림 4는 세션 핸들러의 처리 흐름도를 나타낸다.

세션 핸들러는 통신 세션 제어에 관련된 처리는 SyncML 툴킷에서 제공하는 통신 인터페이스를 사용하며, 현재 단말 장치에서 사용 가능한 프로토콜을 선택하고, 선택된 프로토콜에 의해 통신이 가능한 환경으로 초기화한다. 또한, 메시지를 주고받는 과정에서 일어날 수 있는 오류 검사도 함께 한다. 동기화 메시지 헤더

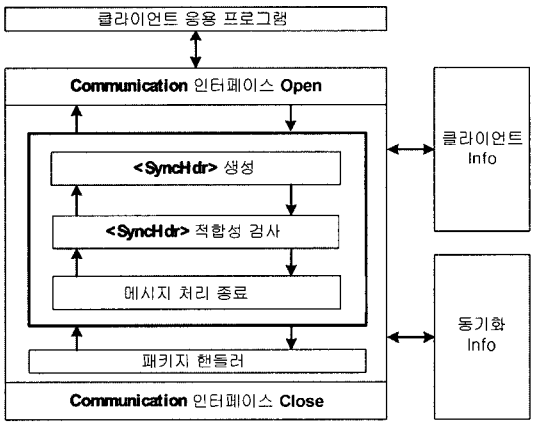

그림 4 세션 핸들러 처리 흐름도

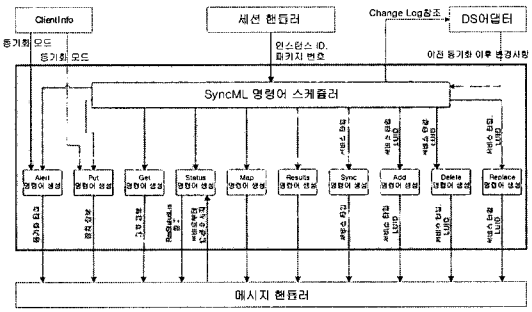

그림 5 패키지 핸들러 처리 흐름도

(SyncHdr) 엘리먼트의 생성과 서버로부터 받은 메시지 내에서 SyncHdr 엘리먼트의 적합성을 검사한 후, 수신된 메시지 전체에 대한 클라이언트의 처리가 완료되면 세션 핸들러에서 통신 세션을 닫는 마지막 처리를 한다.

• 패키지 핸들러

패키지 핸들러(Package Handler)는 SyncML 메시지의 Syncbody 부분에 대한 논리적 구성을 담당한다. 그림 5는 패키지 핸들러의 처리 흐름도를 나타낸다.

패키지 핸들러는 세션 핸들러에서 SyncML 명령어 스케줄러를 호출할 때, 인스턴스 ID와 패키지 번호를 넘겨준다. SyncML 명령어 스케줄러는 패키지 번호를 확인하여 1번일 경우 동기화를 위한 동기화 모드 설정을 대한 메시지 구성을 하고, 3번과 5번일 경우 실제적인 동기화 자료 전송을 위한 메시지 구성을 한다. 패키지 번호가 1번일 경우, 동기화 모드 설정을 위해 클라이언트 정보 모듈에 있는 동기화 모드를 참조하고 Alert 명령어를 생성하기 위해 메시지 핸들러를 호출한다.

SyncML 메시지 스케줄러는 데이타스토어 어댑터에게 이전 동기화 처리 이후 변경된 내용이 있는지 확인한 후, 추가를 위한 Add 명령어, 삭제를 위한 Delete 명령어 및 내용의 갱신을 위한 Replace 명령어를 생성한다. 각 명령어를 생성하기 위해 데이타스토어 어댑터에게 변경된 내용에 대한 LUID(Local Unique ID)를 찾아낸 후, 메시지 핸들러를 호출한다. 변경된 내용이 없을 때까지 이러한 절차 과정을 반복하면서 SyncML 문서를 완성한다.

• 메시지 핸들러

메시지 핸들러(Message Handler)는 자료 동기화를 위해 필요한 XML 명령어를 실제로 생성해 주는 기능을 담당하고 있으며, 서버로부터 수신된 SyncML 메시지를 받아 처리 및 분석하는 역할을 담당한다. 그림 6은 메시지 핸들러의 처리 흐름도를 나타낸다.

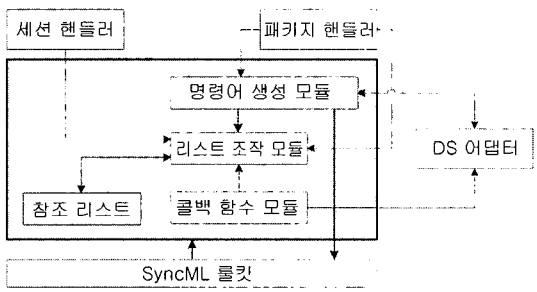

그림 6 메시지 핸들러 처리 흐름도

메시지 핸들러는 내부적으로 명령어 생성 모듈, 리스트 조작 모듈, 콜백 함수 모듈, 그리고 참조 리스트로 구성되어 있다. 명령어 생성 모듈은 패키지 핸들러가 SyncBody의 명령어를 생성하려 할 때 호출되는 함수들의 집합이다. 리스트 조작 모듈은 SyncBody를 참조 리스트에 저장하고, 서버로부터 수신된 명령어를 처리하기 위해 호출된 콜백 함수 모듈들이 다음 패키지를 만들기 위해 호출하는 경우 사용된다. 또한, 콜백 함수 모듈은 SyncBody에 포함된 모든 명령어들에 대하여 적합한 처리를 해주는 함수들의 집합이다. 마지막으로, 참조 리스트는 SyncML 동기화에 필요한 내용들을 리스트 형태로 만들어서 저장한 자료 저장소이다.

3.1.2 데이타스토어 어댑터

데이타스토어 어댑터는 동기화 엔진 모듈의 각 핸들러와 자료 사이를 연결하는 인터페이스 역할을 하며, 여러 가지 서비스 형식에 따라 각각의 데이타스토어 어댑터가 있으며, 외부로부터 실제 자료 저장장치에 접근하여 자료를 삽입, 삭제, 갱신할 수 있는 방법을 제공한다. 또한, 자료 동기화를 위해 이전에 변경되었던 사항에 대한 로그(Log)를 기반으로 새로 변경된 사항들을 검사하여 동기화 할 자료를 결정해주는 기능을 담당한다. 그림 7은 데이타스토어 어댑터의 구조를 나타낸다.

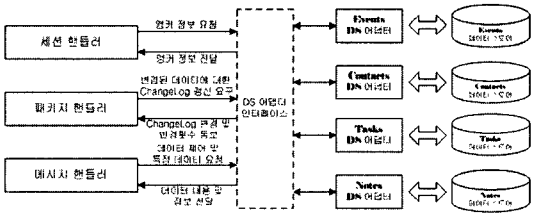

그림 7 서비스 형태에 따른 데이타스토어 어댑터 구조

3.1.3 클라이언트 정보 모듈 및 동기화 정보 모듈

클라이언트 정보 모듈은 동기화 대상이 되는 목적지 주소 및 PIMS(Personal Information Management System) 정보 형태에 대해서 기술하고 있으며, 클라이언트 장치의 정보와 동기화 대상인 클라이언트 주소에 대한 정보도 포함하고 있다. 동기화 정보 모듈은 동기화 종류에 대한 정보 및 이전의 동기화 수행과 현재 수행하려 하는 동기화 간의 상호 일치성을 검사하기 위한 앵커(Anchor) 정보를 담고 있다. 이와 같은 두 개의 정보 모듈은 XML 파일 형태로 존재하며, 동기화 엔진은 두 개의 정보 모듈을 참조하여 동기화를 수행한다.

3.1.4 사용자 인터페이스 및 SyncML 툴킷

사용자가 동기화에 필요한 정보를 조작하고 동기화 명령을 요청하는 사용자 인터페이스 부분은 실제 동기화 명령을 요청하는 사용자와 동기화 엔진 간의 인터페이스를 제공한다. 그리고, SyncML 툴킷은 서버로부터

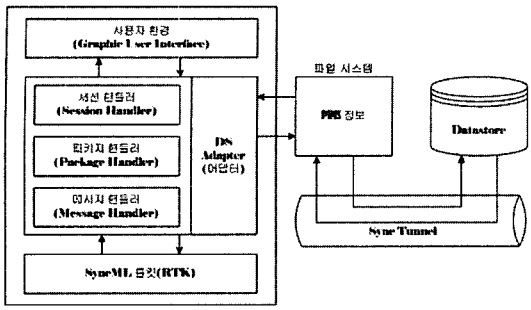

그림 8 동기화 클라이언트 플랫폼 구조

수신된 메시지를 디코딩하고, 클라이언트에서 전송할 메시지를 인코딩하는 기능을 담당하며 전송 프로토콜간의 실제 메시지의 송신과 수신을 담당한다.

3.2 동기화 클라이언트 구현

본 연구에서 제안된 SyncML 기반의 동기화 클라이언트는 리눅스(커널 버전 2.6.0) 운영체제를 기반으로 SyncML 툴킷 4.3 및 C-언어를 이용하여 구현하였다. 그림 8은 구현된 동기화 클라이언트의 플랫폼 구조를 보여준다.

구현된 동기화 클라이언트는 동기화 세션에 관여하는 세션 핸들러와 생성할 메시지를 판단하는 패키지 핸들러, 실제 메시지를 생성하는 메시지 핸들러 및 다양한 형태의 서비스 자료들과 동기화 엔진 사이의 인터페이스 역할을 담당하는 데이타스토어 어댑터로 구성되어 있다.

사용자 인터페이스를 통하여 사용자가 동기화 명령을 요청하면 동기화 엔진이 작동하게 된다. 세션 핸들러는 해당 프로토콜에 대한 통신 세션을 툴킷에서 제공하는 인터페이스를 사용하여 동기화 세션을 열고, 동기화 세션에 필요한 정보들을 초기화하게 된다. 서버와 동기화 할 준비가 완료되면 패키지 핸들러에서 생성할 메시지를 구분하고, 메시지 핸들러를 통하여 SyncML 메시지 생성을 완성한다. SyncML 메시지가 생성되는 과정에서 서버와 동기화 할 자료가 있다면 데이타스토어 어댑터를 통하여 메시지 내에 포함되어야 한다. 동기화 세션 과정 중에서 서버로부터 SyncML 메시지를 받게 되면 세션 핸들러를 거쳐서 메시지 핸들러에서 정의된 콜백(Call Back) 함수들이 호출된다. 각 함수들은 SyncML 메시지들이 원하는 동작들을 수행되도록 한다.

서버로부터 받은 SyncML 메시지 내에 저장되거나 변경될 자료에 대한 정보가 있다면 데이타스토어 어댑터를 통하여 자료 변경을 수행한다. 동기화 처리 대상의 자료는 단순한 파일 시스템으로 제어를 하며, 현재 구현되어 있는 파일 시스템에는 PIMS 정보가 저장되어 있다. 동기화 프로그램이 동기화를 진행하는 동안에 PIMS 정보에 접근하면 동기화 오류가 발생하므로, 이를 방지하기 위하여 동기화 터널을 통하여 파일에 접근하게 하였다.

3.2.1 세션 핸들러(Session Handler)

구현된 동기화 클라이언트의 세션 핸들러 모듈은 두 가지 기능을 담당하고 있다. SyncML 메시지를 주고받는 통신 세션을 제어하는 기능과 SyncML 메시지 중에서 SyncHdr 엘리먼트에 관련된 처리를 한다. 통신 세션 제어에 관련된 일은 SyncML 툴킷에서 제공하는 통신 인터페이스를 사용한다. 세션 핸들러에서는 현재 단말장치에서 사용 가능한 프로토콜을 선택하고, 선택된 프로토콜에 따라서 통신이 가능한 환경으로 초기화하며 메시지를 주고받는 과정에서 일어날 수 있는 오류 검사도 함께 한다. 또한 SyncHdr 엘리먼트의 생성과 서버로부터 받은 메시지 내에서 SyncHdr 엘리먼트의 적합성을 검사한 후, 수신된 메시지 전체에 대한 클라이언트의 처리가 완료되면 세션 핸들러에서 통신 세션을 닫는 마지막 처리를 한다. 그림 9는 세션 핸들러 모듈의 주요 API의 처리 순서를 보여주고 있다.

다음은 동기화 클라이언트에 구현된 세션 핸들러 모듈의 주요 API에 대한 기능과 각 API들의 처리 흐름도에 대해서 보여준다.

• *kmu_smlStartSession* 함수 : SyncML 메시지 중에서 SyncHdr 부분을 생성한다. 메시지 단위의 유일한 값인 msgID를 증가시키고, SyncML 문서 각각에 부여하는 변수인 pkgNum 값도 증가시킨다. SyncHdr 부분을 모두 생성하게 되면 보낸 메시지에 대한 정보를 저장하며, 생성된 자료들을 SyncML 문서로 생성하기 위

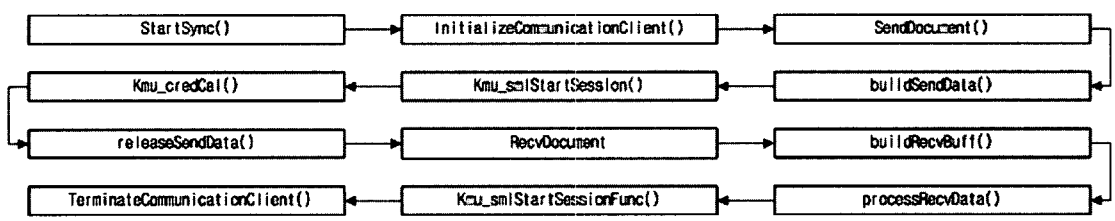

그림 9 세션 핸들러 모듈의 주요 API 처리 순서

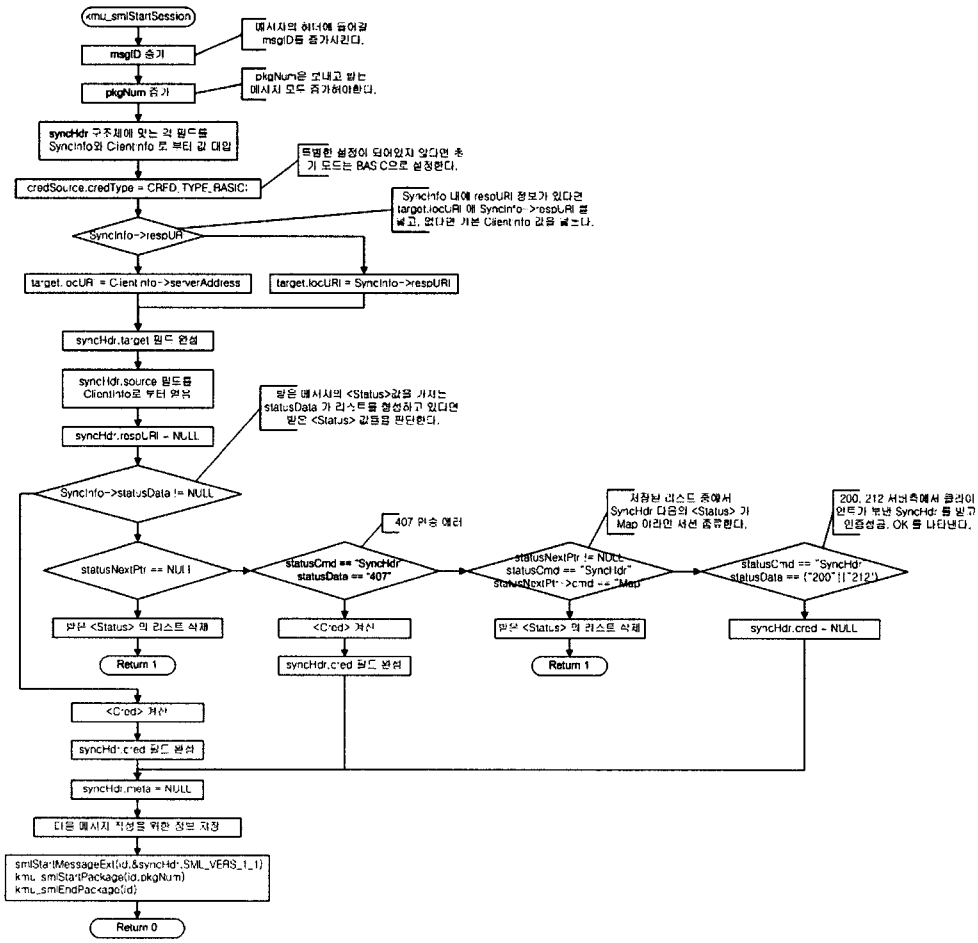

그림 10 kmu_smlStartSession()

하여 smlStartMessageExt 함수를 호출한다. 그리고 SyncBody 부분 생성을 위한 함수인 kmu_sml-StartPackge를 호출하고, 전체 메시지 생성을 마무리하기 위하여 kmu_smlEndPackage 함수를 호출한다. 그림 10은 kmu_smlStartSession 함수의 처리 흐름도를 보여주고 있다.

• *kmu_smlStartSessionFunc* 함수 : 서버로부터 수신한 메시지에서 SyncHdr 부분의 자료들이 올바르게 구성되었는지 검증한다. 수신된 메시지를 <Status> 생성에 사용하기 위하여 저장하고, 메시지에서 flags, respURI와 같이 다음에 보낼 메시지 생성에 영향을 주는 값들은 SyncInfo에 저장한다. 그림 11은 kmu_sml-StartSession 함수의 처리 흐름도를 보여주고 있다.

• *kmu_smlEndSessionFunc* 함수 : 서버로부터 받은 메시지를 모두 처리하고 나머지 정리 작업을 한다. pkgNum 변수는 보내고 받는 문서 각각을 구분하는 임

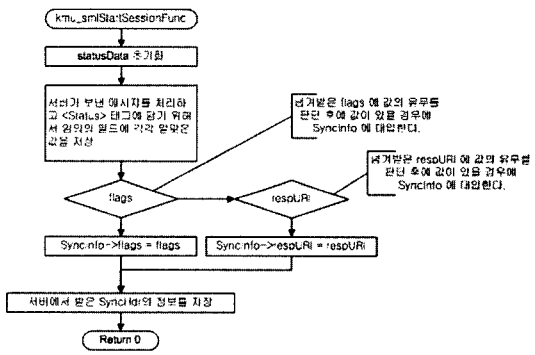

그림 11 kmu_smlStartSessionFunc()

의의 고유 값이다. 그러므로 서버로부터 문서를 받은 후에는 증가시킨다.

• *InitializeCommunicationClient* 함수 : 서버와 통신하기 위하여 초기화 작업을 한다. 선택된 프로토콜에

필요한 메타 정보 생성하고 통신 세션을 열기 위해서 xptOpenCommunication 함수를 호출한다.

• *SendDocument* 함수 : 서버로 문서를 전송하는 기능을 수행하며, buildSendData 함수를 호출하여 전송할 메시지를 생성한다.

• *buildSendData* 함수 : 메시지를 생성하기 위한 작업공간 할당 및 콜백 함수들을 정의하며, 작성될 메시지의 형식과 작업공간의 이름, 크기를 정의한다.

• *releaseSendData* 함수 : 메시지가 모두 전송될 때까지 보호하던 메시지가 저장된 영역의 보호를 해제하고, smlTerminateInstance 함수를 호출하여 메시지 인스턴스를 종료한다.

• *RecvDocument* 함수 : buildRecvBuff 함수를 호출하여 서버로부터 받을 메시지가 저장될 공간을 마련하고 보호한다.

• *buildRecvBuff* 함수 :. 메시지를 수신하기 위한 작업공간 할당 및 콜백 함수들을 정의하며, 작성될 메시지의 형식과 작업공간의 이름, 크기를 정의한다. 작업 공간 보호를 위하여 smlLockWriteBuff 함수를 호출한다.

• *processRecvData* 함수 : RecvDocument 함수로부터 수신한 서버의 메시지를 처리하며, 수신한 메시지를 담고 있는 작업 공간의 보호를 해제한다.

• *TerminateCommunicationClient* 함수 : 현재의 통신 세션을 닫고 사용된 프로토콜을 해제한다.

• *StartSync* 함수 : 자료 동기화 처리과정에서 제일 먼저 수행된다. 통신 세션을 구별하기 위한 값을 가지는 XptServiceID_t 형의 변수를 초기화한다. 또한, 세션에서 사용되는 임시 자료들이 저장될 SyncInfo를 초기화하며, 출력 함수와 최대 작업 크기를 정의한다. 동기화 세션의 시작을 위해서 smlInit 함수를 호출한다.

3.2.2 패키지 핸들러(Package Handler)

동기화 클라이언트 내의 패키지 핸들러는 세션 핸들러에서 SyncML 명령어 스케줄러를 호출할 때, 인스턴스 ID와 패키지 번호를 넘겨준다. 클라이언트 정보 모듈을 참조하여 동기화 모드가 Slow Sync로 설정되어 있을 경우, 장치 정보를 포함하고 있는 Put 명령어를 생성하기 위해 메시지 핸들러를 호출한다. 또한, 서버의 장치 정보를 참조하기 위해 서버 측의 장치 정보를 요청하는 Get 명령어를 생성하기 위해 메시지 핸들러를 호출하고 SyncML 문서를 완성한다. RespStatusList 중에 Add 명령어가 있을 경우 서버로부터 받은 자료와 일치하는 클라이언트의 파일을 알려주기 위한 Map 명령어 그리고 Get과 Search 명령어가 있을 경우, 그 결과 값을 가지고 있는 Results 명령어를 생성하기 위해 메시지 핸들러를 호출한다.

다음은 동기화 클라이언트에 구현된 패키지 핸들러 모듈의 kmu_smlStartPackage 함수와 kmu_smlEndPackage 함수에 대해서 설명한다.

• *kmu_smlStartPakage* 함수 : 패키지 번호에 따라 SyncML 스펙에 명시된 동기화 모드 설정를 설정하고 이전 동기화 이후 변경된 내용을 참조하여 SyncML 문서의 Body 부분을 생성한다.

• *kmu_smlEndPackage* 함수 : 현재 메시지의 마지막을 나타내고 메시지가 완성되었음을 알려준다. 최대 메모리 사이즈, 최대 오브젝트 사이즈 등의 제약으로 인해 현재 메시지로 패키지가 완성되지 않을 경우 Final 명령어를 추가하지 않음으로서 패키지가 완성되지 않았음을 알려주게 된다.

3.2.3 메시지 핸들러(Message Handler)

구현된 동기화 클라이언트 내의 메시지 핸들러는 네 부분으로 구성되어 있으며, 메시지 핸들러의 동작 메커니즘은 다음과 같다. 첫째, 명령어 생성 모듈은 패키지 핸들러가 SyncBody의 명령어를 생성하려 할 때 호출된 생성 모듈들은 리스트 조작 모듈들을 사용하고, 클라이언트의 변경사항이 있다면 데이타스토어 어댑터를 참고하여 완성된 명령어를 툴킷에 전달한 후 전송에 필요한 준비를 한다. 둘째, 리스트 조작 모듈은 세션 핸들러가 만들어낸 SyncML 문서인 SyncBody를 참조 리스트에 저장하거나, 명령어 생성 모듈이 리스트를 추가, 삭제 및 검색이 필요한 경우, 혹은 서버가 보내온 명령어를 처리하기 위해 호출된 콜백 함수 모듈들이 다음 패키지를 만들기 위해 호출하는 기능을 수행한다. 셋째, 콜백 함수 모듈은 서버가 보낸 SyncML 문서를 툴킷으로부터 받아서 리스트 조작이 필요할 경우에는 리스트 조작 모듈을 호출하고, 클라이언트가 가진 자료의 추가, 삭제 및 변경이 필요할 경우에는 데이타스토어 어댑터를 호출한다. 마지막으로 참조 리스트는 SyncML 동기화에 필요한 내용들을 리스트 형태로 저장하는 기능을 수행한다.

다음은 동기화 클라이언트에 구현된 패키지 핸들러 모듈의 kmu_smlStartPackage 함수와 kmu_smlEndPackage 함수에 대해서 설명한다.

• *kmu_smlAdd[Delete, Get, Map, Put, Replace, Results, SyncStart, SyncEnd, Alert, Status]Cmd* 함수 : SyncML 동기화 관련 명령어(Add, Delete, Replace 등등..)를 작성하고, 현재 SyncML Body를 각각 처리하는 함수 smlAdd[Delete, Get, Map...]Cmd()를 호출한다.

• *kmu_smlAdd[Delete, Get, Map, Put, Replace, Results, SyncStart, SyncEnd, Alert, Status]CmdFunc* 함수 : 수신된 SyncML 메시지 중에서 관련된 동기화 명령어에 대해서 각각 호출되어 어플리케이션에서 수행

된다.

• *tsGetHeadNodeFromSentCmdList* 함수 : SentCmdList 에 자료가 있는지 알아보기 위한 기능을 수행한다.

• *tsGetHeadNodeFromRespStatusList* 함수 : Status를 작성해야 할 명령어가 있는지에 대한 여부를 알기 위한 기능을 수행한다.

3.2.4 데이타스토어 어댑터

데이타스토어 어댑터는 세션 핸들러에서 동기화 모드 결정을 위해 앵커정보를 요청하면 데이타스토어 어댑터는 파일에서 앵커 정보를 참조하여 세션 핸들러에게 전달한다. 패키지 핸들러에서 동기화 되어야 할 자료 결정을 위해 ChangeLog 갱신을 요청하면, 서비스 타입에 해당하는 데이타스토어 어댑터를 호출하여 ChangeLog와 데이타스토어의 비교를 수행한다. 또한 메시지 핸들러에서 동기화 할 자료를 서버에 보내기 위해 해당 자료 내용을 요청하면 그에 해당하는 자료 정보를 전달하고, 반대로 서버에서 보낸 동기화 자료를 해당 데이타스토어에 반영한다.

구현된 각각의 데이타스토어 어댑터는 내부적으로 데이타 변경 체크 모듈, 데이타 제어 모듈 및 앵커 관리 모듈로 구성되어있다. 자료변경 체크 모듈은 이전 동기화 이후 클라이언트에 변경된 자료를 검사하기 위해 ChangeLog 파일을 리스트로 만들어 실제 자료 객체와 비교 검사하며, 변경된 사항을 ChangeLog에 반영하고, 변경된 사항의 개수를 패키지 핸들러에게 알려준다. 데이타 제어 모듈은 외부로부터 데이타스토어에 접근하여 자료를 추가, 삭제 및 갱신을 할 수 있는 기능 제공하며, 주로 메시지 핸들러의 요청에 의해 동작한다. 앵커 관리 모듈은 서버와의 동기화 모드를 결정하기 위해 이전 동기화 시간과 현재 동기화 시간으로 이루어진 앵커 정보를 세션 핸들러에게 전달하며, 동기화가 끝날 때 현재 동기화 시점을 앵커파일로 저장한다.

4. 동기화 클라이언트 검증

4.1 동기화 적합성 시험

SyncML 기반의 동기화 클라이언트는 SyncML 규격 1.1을 준수하면서 PDA와 같은 이동무선통신 단말기에서 동작한다. 표 3은 구현한 동기화 클라이언트 코드 크기 및 모듈 개수를 보여준다.

구현된 동기화 클라이언트의 정상적인 동기화 수행 여부를 검증하기 위해서 OMA 상호 운용성 시험에 합격한 Synthesis 자료 동기화 서버와 연동하여 시험하였다. Synthesis 서버에는 기본적으로 PIMS 정보가 저장되어 있으므로, 동기화 검증에 사용된 동기화 대상 자료로서 PIMS 정보를 사용하였다. 동기화 클라이언트에서 생성된 PIMS 정보는 HTTP 프로토콜을 통해서 SyncML 메시지 형태로 Synthesis 서버로 전송되고, 전송된 PIMS 정보는 Syntheis 엔진으로 전달된다. Synthesis 서버는 도착된 PIMS 정보를 데이타스토어

표 3 ThinkSync 클라이언트 코드 크기 및 모듈 개수

구 성 요 소	코드크기	모듈개수
사용자 Application(User Interface)	11 Kbytes	6
세션 핸들러(Session Handler)	14 Kbytes	12
패키지 핸들러(Package Handler)	11 Kbytes	2
메시지 핸들러(Message Handler)	65 Kbytes	26
데이타스토어 어댑터 (Datastore Adapter)	7 Kbytes	13

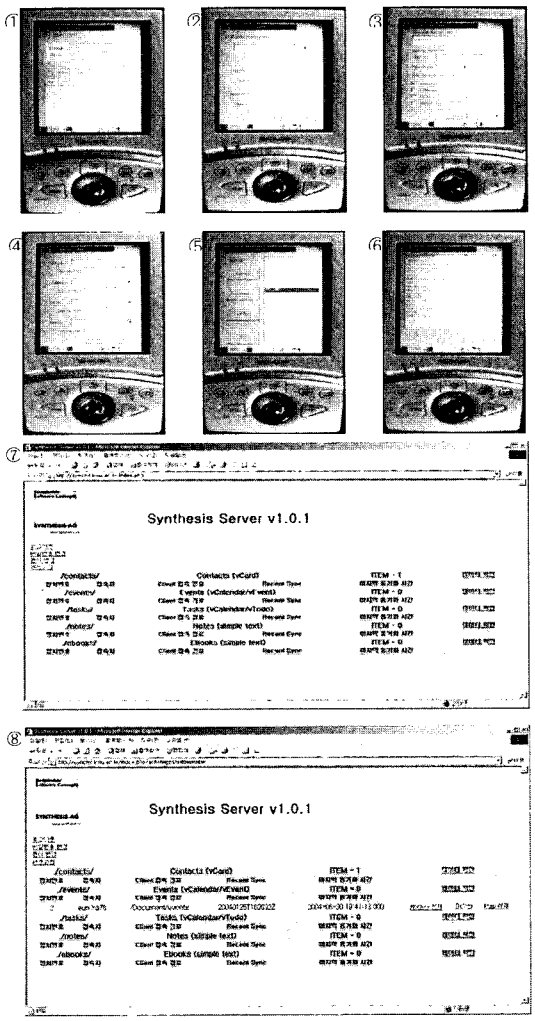

그림 12 동기화 클라이언트와 Synthesis 서버간의 PIMS 자료 동기화

객체로 변환하여 처리한 후, 자신의 변경사항 혹은 처리 결과에 대한 응답을 SyncML 메시지로 작성하여 동기화 클라이언트로 전송한다. 동기화 클라이언트에서 동기화 수행 가능한 서비스의 형태는 주소록(Contacts), 일정관리(Events), 작업관리(Tasks), 메모(Notes) 등과 같은 PIMS 자료이다.

동기화 클라이언트는 PC 기반의 터미널 환경에서 개발한 후, 자우루스(Zaurus) PDA에 포팅(Porting)하였으며, GUI 인터페이스를 위해서 qt 라이브러리를 이용하였다. 그림 12는 자우루스 PDA에서 GUI 방식으로 동기화를 수행하는 결과를 보여준다[20].

그림 12에서 ①번 및 ⑦번 화면은 동기화 수행 전의 동기화 클라이언트와 Synthesis 서버의 초기화 화면을 각각 보여준다. ②번 및 ③번 화면은 동기화 클라이언트에서 사용자, 서비스 형태 및 동기화 대상 서버에 대한 프로파일 정보를 설정하고, ④번 화면은 동기화 모드를 설정하는 것이다. ⑤번 화면은 현재 동기화 클라이언트가 동기화 대상이 되는 자료들에 대한 파일 정보를 나열하고 있으며, ⑥번 화면은 선택된 동기화 자료에 대한 PIMS 정보를 담고 있는 화면이다. ②번에서 ⑥번까지의 과정을 통해서 클라이언트 정보와 동기화 정보를 작성한 후, ①번 화면을 통해서 동기화 수행 명령을 입력하게 된다. 동기화 클라이언트와 Synthesis 서버간의 PIMS 자료에 대한 정상적인 동기화 처리가 수행된 후, ⑧번 화면의 Synthesis 서버 화면을 통해서 동기화 수행된 PIMS 객체 정보를 보여준다.

4.2 SCTS(SyncML Conformance Test Suite) 상호 운용성 검증 결과

구현된 동기화 클라이언트가 올바로 동작하는지를 확인하기 위해서 적합성 시험 (Conformance Test)과 상호운용성 시험 (Interoperability Test)을 수행하였다. 적합성 시험은 SyncML 표준규격을 바탕으로 SIC(SyncML Interoperability Committee)에서 제시하는 SCTS 툴킷을 이용하여 테스트 절차를 시험 시스템에 적용하여 그 결과를 비교 분석하여 프로토콜 구현의 적합성 여부를 판별하는 것이다. 표 4에서 SyncML 적합성 시험에 대한 검증 결과를 기술하였다.

본 연구에서 제안된 동기화 클라이언트를 SCTS 툴킷을 이용하여 Synthesis 서버와 연동하여 시험을 수행하였으며, 적합성 항목에 해당하는 10개의 그룹 및 총 22개의 항목에 대해서 상호 운용성을 검증하였다.

5. 결 론

이동무선통신 컴퓨팅 환경의 발달과 이동무선통신 단말기 보급의 확대로 인해 다양한 플랫폼 기반의 단말기들이 사용되고 있으나, 각각의 서로 다른 자료 동기화 방식을 사용하기 때문에 여러 가지 불편함과 오버헤드가 발생하고 있다. 따라서, 자료 동기화 기술의 표준화에 대한 중요성이 부각되었고, OMA에서는 자료 동기화의 표준으로서 SyncML을 제안하였다.

본 논문에서는 SyncML 방식의 자료 동기화 클라이언트를 설계 및 구현하였다. 동기화 클라이언트의 전체적인 구조 및 각 데이타 조작 모듈 구조도를 제시하였고, 데이타 조작 모듈 부분의 처리 흐름을 중점적으로 기술하였다. 동기화 클라이언트는 동기화 자료 단위별로 데이타 조작 모듈을 분류함으로써, 다중의 자료 동기화 요청에 대해서 원활하게 동기화 명령을 수행하도록 설계되었다. 구현된 동기화 클라이언트는 현재 자우루스 PDA에서 동작하고 있으며, OMA 인증을 획득한 Synthesis 자료 동기화 서버와의 연동을 통하여 SyncML 표준 규격 준수를 검증하였다.

향후 과제로는 동기화 클라이언트를 (PIMS 정보 이외의) 다른 자료에 대해서도 자료 동기화 검증을 수행하는 것이 필요하다. 또한, 동기화 클라이언트 코드를 세밀히 점검하여 불필요한 코드의 제거, 메모리 사용에 대한 최적화 및 실행 코드를 최소화함으로써, PC에 비

표 4 SCTS 상호 운용성 검증 결과

Data Synchronization Client Conformance Test Group	
요구사항	지원여부
Support of the Basic authentication scheme	O
Support of the MD5 Digest authentication scheme	O
Respond with Results for a Get on device information	O
Sending Alerts for all databases	O
Sending of valid Sync Tags	O
To check if the TestObject sends Alerts for Multiple databases	O
Matching of Sync Anchors	O
Correct handling of Add	O
Sending of valid Add	O
Handling of Replace (SCTS item)	O
Sending of valid Replace	O
Handling of Replace (client item)	O
Sending of valid Replace (client item)	O
Handling of Delete (client item)	O
Sending of valid Delete (client item)	O
Sync verification	O
Handling of Add with multiple items	O
Handling of Replace with multiple items	O
Handling of Delete with multiple items	O
Respond for Delete - Multiple Status	O
Handling of multiple messages	O
Handling of NumberOfChanges	O

해 상대적으로 처리 환경이 열악한 이동무선통신 단말기에서 좀 더 빠른 응답시간과 효율적인 자원 사용이 가능하도록 개선할 필요가 있다. 더 나아가 현재 개발된 동기화 클라이언트로 인한 축적된 기술을 바탕으로 SyncML 기반의 자료 동기화 서버 개발을 생각해 볼 수 있다.

참 고 문 헌

[1] Uwe Hansmann, Riku Mettala, Apratim Purakayastha, Peter Thompson, SyncML Synchoronizing and Managing Your Mobile Data, pp. 21-34, PRENTICE HALL PTR, New Jersey, 2003.

[2] S. Agarwal, D. Starobinski, A. Trachtenberg, "On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices," Network IEEE, Vol. 16, Issue 4, pp. 22-28, 2002.

[3] DaeJin Jang, Hong Taek Ju, KeeHyun Park, B.H.Ha, M.C.Lee, Sung-Chae Bae, "Design of ThinkSync DM based on SyncML Device Management," The 3rd APIS, pp. 569-574, 2004.

[4] Byung-Yun Lee, Tae-Wan Kim, Dae-Woong Kim, Hoon Choi, "Data Synchronization Protocol in Mobile Computing Environment Using SyncML," The 5th IEEE International Conference, pp. 133-137, July 2002.

[5] 장대진, 주홍택, 박기현, "SyncML DM 기반의 이동무선통신 단말기 관리 시스템 설계", KNOM Review 제 6권 2호, pp. 7-12, 2003.

[6] Ligang Ren, Junde Song, "Data Synchronization in the Mobile Internet," The 7th IEEE International Conference, pp. 95-98, September 2002.

[7] SyncML Initiative, SyncML Interoperability Testing Process, 2001.

[8] SyncML Initiative, SyncML Implementation Conformance Statement, 2002.

[9] Synthesis AG, http://www.synthesis.ch/, Zürich Switzerland, 2003.

[10] David Starobinski, Ari Trachtenberg, Sachin Agarwal, "Efficient PDA Synchronization," IEEE Transaction on Mobile Computing, Vol. 2, Issue 1, pp. 40-51, 2003.

[11] Ari Trachtenberg, David Starobinski, Sachin Agarwal, "Fast PDA Synchroni-zation Using Characteristic Polynomial Interpolation," IEEE INFOCOM 2002, Vol. 3, pp. 1510-1519, June 2002.

[12] Maria Butrico, Norman Cohen, John Givler, Ajay Mohindra, Apratim Purakayastha, Dennis G. Shea, "Enterprise Data Access from Mobile Computers: An End-to-end Story," The 10th IEEE International Workshop on Research Issue in Data Engineering, pp. 9-16, February 2000.

[13] SyncML Initiative, SyncML White Paper version 1.0, 2000.

[14] SyncML Initiative, SyncML Representation Protocol Specification version 1.1, 2002.

[15] SyncML Initiative, SyncML Synchronization Protocol Specification version 1.1, 2002.

[16] Sync4j, http://sync4j.sourceforge.net/web/theproject.html

[17] SyncLE, http://neosteps.com/

[18] 이지연, 김연수, 최 훈, "SyncML 클라이언트 설계 및 구현", 제 18회 한국정보처리학회 추계학술발표대회, 논문집 제 9 권 제 2 호, 2002년 11월.

[19] 세션매니저를 이용한 SyncML 동기화 시스템 설계 및 구현, http://strauss.comeng.cnu.ac.kr/research/sync/paper/07_SyncML_SyncMLServer.pdf

[20] Mattbias Kalle Dalbeimer, Programming with Qt, O'REILLY, 2000.

장 대 진

1998년 2월 계명대학교 정보통신학부 컴퓨터공학 학사. 2001년 8월 계명대학교 정보통신학부 컴퓨터공학 석사. 2003년 12월~현재 계명대학교 정보통신학부 컴퓨터공학 박사과정 재학. 관심분야는 Mobile Data Synchronization, Wireless System

박 기 현

1979년 2월 경북대학교 전자계산학 학사 1981년 2월 한국과학기술원 전자계산학 석사. 1990년 8월 미국 Vanderbilt 대학교 전자계산학 박사. 1981년 3월~현재 계명대학교, 정보통신대학, 교수. 관심분야는 Parallel Processing System, 모바일 소프트웨어, 임베디드 소프트웨어

주 홍 택

1989년 8월 한국과학기술원 전산학 학사 1991년 8월 포항공과대학교 컴퓨터공학 석사. 2002년 2월 포항공과대학교, 컴퓨터공학 박사. 1991년 9월~1997년 2월 대우통신, 종합연구소, 선임연구원. 2002년 9월~현재 계명대학교, 정보통신학부, 조교수. 관심분야는 Web-based Network Management, Network Monitoring