세션 매니저를 이용한 SyncML 동기화 시스템
(SyncML Data Synchronization System based on Session Manager)

이 병 윤 † 이 길 행 † 조 진 현 †† 류 수 희 ††† 최 훈 ††††
(Byung-Yun Lee) (Gil-Haeng Lee) (Jin-Hyun Cho) (Soo-Hee Ryu) (Hoon Choi)

요 약 PDA, 노트북, 팜탑, 데스크탑 등 다수의 단말기에 분산되어 있는 동일한 데이타들에 대해 가장 최신 값으로 일치시키는 것을 동기화(synchronization)라고 한다. 2000년 12월 노키아, 에릭슨, IBM등에 의해 데이타 동기화에 대한 산업계 표준 프로토콜 규격이 발표되었다. 본 논문에서는 SyncML 프로토콜을 구현하고 PIMS(Personal Information Management System) 서비스 데이타 동기화 기능을 지닌 서버의 개발 연구 내용을 소개한다. 서버 구조 설계와 각 프레임 별 기능, 그리고 각 프레임에서의 SyncML 메시지 처리 절차를 제시하였다. 구현한 서버에 대해 적합성 시험과 상호운용성 시험을 수행하였으며 PIMS 서비스에 대해 복수 개 장치 간에 효과적으로 데이타 동기를 수행함을 확인하였다. 또한 세션매니저라는 모듈에서 여러 세션에 대한 정보를 신속히 액세스하도록 관리함으로써, 우리가 구현한 이전 버전 서버에 비해 우수한 성능을 지님을 확인하였다.

키워드 : 동기화마크업언어, 개인정보관리시스템, 데이타동기화서버, 세션매니저

Abstract Synchronization is the process of making replicated data on multiple devices be consistent, i.e., identical with each other. In Dec. 2000, major handheld computer manufacturers including Nokia, Ericsson, IBM published the SyncML protocol specification. In this paper, we describe the CNU SyncML server for PIMS(Personal Information Management System) service that we developed based on the SyncML specification. The server architecture and capabilities of the frames are presented along with the procedure of message processing by each frame. We put the CNU SyncML server to the conformance tests and interoperability tests to confirm its functionality. The session managing mechanism of the CNU SyncML server showed better performance than our previous implementations.

Key words : SyncML(Synchronization Markup Language), PIMS, Data Synchronization Server Session Manager

1. 서 론

무선 인터넷이 발전함에 따라 이동 데이타 단말기를 통해 인터넷(Internet)을 검색하거나, 정보 제공자(ISP, Internet Service Provider) 서버에 액세스하여 데이타 서비스를 이용하는 것이 보편화되어 가고 있다. 이로 인해, 개인 정보 관리자(PIMS : Personal Information Management System) 소프트웨어를 이용하여 자신의 업무 일정을 관리하거나 전자 메일을 교환하는 것과 같이, 이동 데이타 단말기 이용이 개인적인 용도뿐만 아니라 비즈니스의 새로운 수단으로 발전하고 있다. 또한 무선 이동 통신 환경이 다양해지면서 각 벤더마다 제공하고 있는 이동 단말기의 종류가 다양해지고, 개인 당 보유하고 있는 단말기의 수도 늘어나고 있다.

이러한 다수의 단말기에 분산되어 있는 동일한 데이타들에 대해 가장 최신 값으로 일치시키는 것, 즉 데이

· 조진현은 BK21 충남대학교 정보통신인력양성사업단의 지원을 받았음
· 이 연구는 정보통신부에서 지원하는 대학기초연구지원사업으로 수행
† 비 회 원 : 한국전자통신연구원 네트워크연구소 연구원
bylee@etri.re.kr
ghlee@etri.re.kr
†† 비 회 원 : 마이엔진(주) 연구원
jinhyuncho@hotmail.com
††† 비 회 원 : 이노에이스 연구원
shryoo@innoace.com
†††† 종신회원 : 충남대학교 컴퓨터공학과 교수
hchoi@comeng.chungnam.ac.kr
논문접수 : 2002년 3월 26일
심사완료 : 2002년 8월 8일

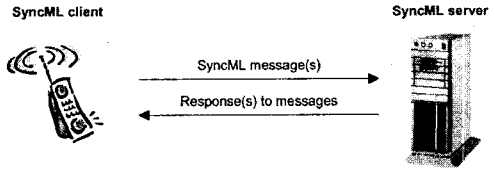


그림 1 SyncML을 이용한 클라이언트와 서버 구조

타 동기화(synchronization)가 이루어져야 하는데, 이를 위해 사용되는 동기화 프로토콜은 현재 PDA(Personal Data Assistant) 제조 업체와 서비스 제공 업체마다 각기 독자적인 방법을 개발하여 사용하고 있다[1,2].

표 1 업체별 동기화 방법

관련 업체	동기화 방법(지원 소프트웨어)
Palm Computing	Hot Sync 사용
Compaq	마이크로소프트사의 ActiveSync 사용
Starfish	StarTAC Mobile Organizer 사용
Synchrologic	다양한 응용에 대한 4가지의 소프트웨어를 지원 • RealSync Server : PDA를 위한 Email과 PIM 응용 지원 • iMobile Data Synchronization : 기업응용을 위한 데이타 동기화 • iMobile Systems Management : 최적화된 이동 단말 시스템 관리 지원 • iMobile File Distribution : 파일이나 웹 사이트 정보의 동기화

표 1에서 살펴본 바와 같이 여러 단말기, 서비스 제공 업체가 독자적인 솔루션을 제공하기 때문에 제품 간 및 서비스 간 호환성이 결여되어 있다. 호환성 없는 동기화 프로토콜이 점점 늘어나면서 데이타 동기화 표준의 필요성이 대두되었고, 이에 2000년 2월 IBM, Lotus, Motorola, Nokia, Palm, Psion, Starfish Software 등의 모바일 관련 업체를 중심으로 SyncML(Synchronization Markup Language) 그룹이 구성되었다[1]. SyncML 그룹은 서로 다른 컴퓨터 플랫폼, 네트웍, 응용 서비스에 이용될 수 있는 데이타 동기 방식의 개방형 표준 인터페이스 개발을 목적으로 하며, 2000년 12월 SyncML 규격 1.0 이어 현재 규격 1.0.1까지 발표하였다. 현재 SyncML 프로토콜은 현재 600개 이상의 스폰서들의 지원 하에 있고, 국내에서도 SyncML 프로토콜을 이용하여 데이타 동기화 솔루션을 개발하고 있는 업체가 20개 이상이 되고 있는 등 연구가 활성화되고 있다[2].

본 논문에서는 SyncML 규격 1.0.1에 기반하여 다수의 단말기로부터 다양한 동기화 요구를 처리하고 서버의 변경사항을 반영할 수 있는 서버의 구조를 설계하고 구현하였다. 본 서버는 SyncML 프로토콜에 따라 데이타 동기화 작업을 처리하면서, 기능의 역할에 따라서 프레임으로 모듈화 하여 응용 서비스에 독립적으로 동기화 처리가 가능하도록 설계하였으며, 새로운 서비스 추가 시 시스템 종속적인 프레임만을 변경하여 처리가 가능하도록 설계하였다. 또한 동기화 처리 시 클라이언트로부터 세션을 유지하고 관리하는 프레임을 정의하였으며, 데이타를 저장하기 위한 인터페이스 프레임을 정의하여 다양한 데이타베이스에 연동이 가능하도록 설계하였다.

이 절 이후의 구성은 다음과 같다. 2장에서는 SyncML 규격에 대한 소개를 하고, 3장에서는 구현된 시스템의 구조에 대해 설명한다. 4장에서는 구현된 서버에 대한 적합성 시험 절차와 시험 내용을 서술한다. 5장에서는 구현된 서버에 대한 성능 테스트를 수행하여 제시된 구조의 효율성을 입증하고 결론을 맺도록 한다.

2. SyncML 규격

데이타 동기화의 목적은 분산되어 있는 여러 단말기 내의 공통된 데이타들을 서로 일치시키도록 하는 데에 있다. SyncML은 임의의 유/무선 네트웍 환경 하에서, 단말기와 서비스의 종류에 구애 받지 않고 데이타 동기화를 수행하는 표준 언어이다[3]. SyncML은 표준 데이타 동기화를 위해 크게 메시지의 데이타 포맷을 정의한 표현(Representation) 프로토콜[4], 동기화 규칙을 정의한 동기(Synchronization) 프로토콜[5], 전송 프로토콜[6]을 정의하고 있다.

SyncML이 메시지를 전송하기 위해 사용하는 전송 프로토콜은 HTTP, WSP(Wireless Session Protocol), OBEX(OBject EXchange Protocol) 등과 같은 기존 프로토콜들인데, SyncML 규격에서는 새로운 프로토콜을 정의하는 것이 아니라 기존 전송 프로토콜과의 바인딩 규칙만을 정의한다. 데이타 표현 프로토콜과 동기 프로토콜이 전송 프로토콜에 독립적이므로 향후 다른 전송 프로토콜과의 바인딩이 가능하다.

2.1 SyncML 데이타 표현 프로토콜

SyncML 데이타 표현 프로토콜(Data Representation Protocol)은 데이타, 메타 데이타, 동기화 명령어 등과 같이 동기화를 수행하기 위해서 필요한 모든 표현 양식을 표시할 수 있는 XML(Extensible Markup Language)의 DTD(Document Type Definition)를 정의하

고 있는 SyncML 메시지에 대한 구조체 규약이다[4,7]. 데이타 표현 프로토콜로 작성된 메시지는 일반 텍스트 형태의 XML 양식이나 압축된 바이너리 형태의 WBXML(Wireless Binary XML) 양식을 따른다.

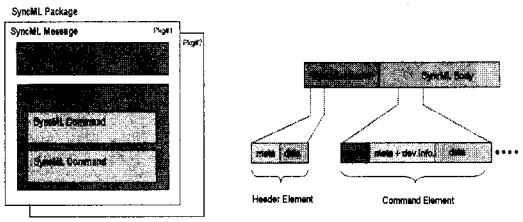


그림 2 SyncML 메시지 구조

SyncML 데이타 포맷은 데이타 동기화 명령을 수행하는 클라이언트와 서버간에 전달되어야 하는 잘 정의된 메시지의 집합으로 정의되며 그림 2에서와 같은 메시지의 구조를 가진다[8]. 해당 메시지는 헤더요소 타입(SyncHdr)과 바디 요소 타입(SyncBody)으로 정의되며, 표 2에서와 같이 5종류의 SyncML 구성요소가 존재한다. 해당 구성요소들은 SyncML 표현 프로토콜을 따른다. 각각의 요소 타입은 특정 목적이나 사용 방법에 따라서 서로 포함관계를 가지기도 하고 제한적인 관계를 가지기도 한다.

2.2 SyncML 데이타 동기 프로토콜

SyncML 동기 프로토콜은 서버와 클라이언트 간에 데이타를 추가, 삭제, 변경하기 위해서, SyncML 메시지가 교환되는 방법과 실제 동기화 동작 방법 그리고 동기화 타입을 정의한다[5]. 동기화는 클라이언트가 먼저 SyncML 메시지를 서버에게 전송하고, 서버는 서버에 저장되어 있는 데이타와 클라이언트가 전송한 SyncML 메시지 내의 데이타 동기화를 수행하고 응답 메시지를 보내는 일련의 과정을 통해서 이루어진다. SyncML 프로토콜은 클라이언트와 서버간의 데이타 추가, 삭제, 변경, 그리고 상태 정보를 교환하기 위하여 DTD (Document Type Definition)를 만족하는 SyncML 메시지를 교환하는 역할을 하며, 아래와 같은 7가지 시나리오에 따라 동기화를 이룬다.

- Two-way Synchronization

가장 일반적인 경우이며, SyncML 클라이언트와 서버 모두가 장치내의 데이타 변경에 대한 정보 교환을 담당한다.

- Slow Sync

동기화 대상 데이타베이스 항목들 전체에 대해 하나씩 비교하여 동기화를 수행한다. 이 방식은 주로 클라이언트가 자신의 데이타를 손실한 경우, 클라이언트에 의하여 요구되어진다.

- One-way Synchronization from Client Only

SyncML 클라이언트가 모든 자신의 변경 사항을 SyncML 서버로 전송하는 방식이며, 서버는 자신의 변경 사항을 클라이언트로 전송하지 않는다.

- Refresh Sync from Client Only

SyncML 클라이언트가 자신의 모든 데이타베이스 데이타를 SyncML 서버로 전송하는 방식이며, 서버는 클라이언트로부터 전달된 데이타로 자신의 값을 변경한다.

- One-way Synchronization from Server Only

SyncML 클라이언트는 SyncML 서버에서 이루어진 모든 변경을 수신하지만, 클라이언트는 자신의 변경 사항을 서버로 전달하지 않는다.

- Refresh Sync from Server Only

표 2 SyncML 요소타입

공통 요소	정의	다른 SyncML 요소에 의하여 사용되는 요소
	종류	Archive, Chal, Cmd, Final, Lang, LocName, LocURI, MsgID, NoResp, NoResults, RespURI, SessionID, Source, Target
메시지 컨테이너 요소	정의	SyncML 메시지를 위한 기본 컨테이너 요소
	종류	SyncML, SyncHdr, SyncBody
데이타 요소	정의	SyncML 메시지내에서 교환되는 데이타에 대한 컨테이너 요소
	종류	Data, Item, Meta
프로토콜 관리 요소	정의	요구된 명령어에 대한 Status 구조를 위한 요소
	종류	Status
프로토콜 명령어 요소	정의	SyncML Body 내에서 사용되는 명령어 요소
	종류	Add, Alert, Atomic, Copy, Delete, Exec, Get, Map, MapItem, Put, Replace, Results, Search, Sequence, Sync

SyncML 서버가 자신의 모든 데이타베이스 데이타를 SyncML 클라이언트로 전송하는 방식이며, 클라이언트는 서버로부터 전달된 데이타로 자신의 값을 변경한다.

• Server Alerted Sync

상기의 6가지 시나리오는 클라이언트가 서버에 연결하고 동기화를 시작하는 것이지만, 이것은 서버가 클라이언트에게 동기화 시작을 요구하는 방식이다. 클라이언트에게 이 요청을 전달하는 방법은 임의의 방법이 가능하고 규격에서는 별도로 규정하지 않는다.

3. 동기화 시스템 구현

그림 3은 구현된 SyncML 서버의 기본적인 프레임 구조를 블록화 한 것이다. 각 구성 요소를 살펴보면, 서버쪽에서 데이타 변경을 수행하는 주체인 서버 응용프로그램(Server Application)이 있고, 클라이언트 장치로부터 전송된 메시지를 받아서 툴킷으로 전달하기 위한 싱크 어댑터(Sync Adapter)가 있다. SyncML 툴킷(Toolkit)은 전달된 메시지를 디코딩하고, 보낼 메시지를 인코딩하는 기능을 담당한다. Sync Agent는 전달된 메시지를 분석하여 SyncML 프로토콜에 맞도록 메시지를 처리하는 기능을 담당하고, Sync Engine은 Sync Agent로부터 처리된 동기화 메시지를 수신하여 서비스 정책과 시스템에 종속적인 처리를 담당한다. 또한 동기화 처리 중 데이타간 충돌이 발생할 경우 이를 해결해 주는 역할을 담당한다. Session Manager는 클라이언트와 서버간의 동기화를 위한 세션을 유지하고 관리하는 기능을 담당하며, Open DB Interface는 데이타를 저장하기 위한 데이타베이스와의 인터페이스 역할을 담당한다. 각 프레임은 동적 링킹 라이브러리(DLL) 형식으로 구현하였고, 장치와의 통신은 HTTP를 이용하였으며, DLL과의 연결을 위해 JNI(Java Native Interface)를 이용하였다[9,10].

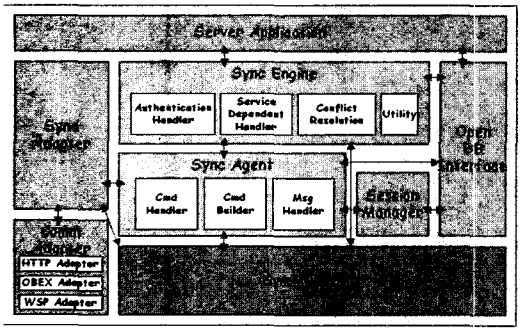


그림 3 동기화 시스템 구조

3.1 Sync Agent

Sync Agent는 응용 서비스에 독립적 이도록 프로토콜 부분만을 구현한 프레임이다. 즉, 서비스의 확장으로 새로운 MIME 타입을 추가로 지원하게 되는 경우, 서비스에 독립적인 처리 부분을 재사용할 수 있게 되어 빠른 서비스 추가가 가능하게 된다. Sync Agent는 동기화를 위해 수신된 메시지를 파서(parser)를 통하여 디코딩하는 기능을 담당하는데, 데이타 처리단위는 추가(Add), 변경(Replace), 삭제(Delete) 등을 지시하는 프로토콜 커맨드 단위로 수행된다. 본 논문에서 제시한 Sync Agent는 클라이언트로부터 받은 요청 메시지에 대해서 툴킷이 커맨드를 파싱하면 파싱된 커맨드를 즉시 처리하고 다음 커맨드를 파싱하는 방식으로 동작한다. Sync Agent는 크게 기능별로 Command Handler, Command Builder, Message Handler 등 3개의 클래스 단위의 모듈로 구성된다.

3.1.1 Command Handler

Command Handler는 수신된 메시지를 처리하는 모듈로서 툴킷이 파싱한 메시지를 커맨드 단위로 처리하기 위해서 Sync Engine이나 Open DB Interface, Session Manager 프레임과의 관계를 설정해주고 동기화 수행을 위해 해당 모듈을 호출하는 역할을 담당한다. 실제적으로 동기화를 수행하는데 있어서의 동작 흐름을 제어하고 있는 것이 Command Handler내에서 이루어진다.

3.1.2 Command Builder

Command Builder에서는 클라이언트 메시지에 대한 응답 메시지를 생성하거나 서버 내에서의 변경 사항이 있을 경우, 클라이언트에게 변경 내용을 반영시키기 위한 커맨드를 생성하도록 툴킷의 커맨드 생성 모듈과 인터페이스하는 역할을 담당한다.

3.1.3 Message Handler

Message Handler는 클라이언트로부터 수신된 메시지를 처리하기 위해 자주 접근되는 정보들을 관리하는 모듈로서, 클라이언트로부터 들어온 메시지를 처리하는데 있어 빈번하게 사용되는 정보들을 전역 구조체로 정의하여 메모리에 두고 리스트로 관리한다. 이 때 전역 구조체에 정의해 두고 사용하는 정보는 현재 세션 내에 유효한 세션 관련 정보와 Add, Replace, Delete 등의 개별 커맨드들을 포함하고 있는 Sync, Atomic, Sequence 등의 커맨드들을 스택으로 관리하고 있는 구조를 포함한다. Message Handler에서 관리하는 정보들은 동기화를 수행하는데 있어 세션 전체에 유효한 정보를 유지하는 것이 아니라, 현재 Request 메시지를 처리

하는 동안만 유효한 정보이다. 메시지가 들어오면 Command Handler의 SyncML 메시지의 헤더를 처리하는 루틴 내에서 Message Handler에 대한 구조체를 초기화 하고, SyncBody의 내용을 커맨드 단위로 메시지를 파싱하면서 생기는 추가적인 정보가 생기면 리스트로 연결하여 관리한다.

3.2 Sync Engine

Sync Engine은 크게 Service Dependent Handler, Conflict Resolution, Authentication Handler, Utility 등 4개의 모듈로 구성된다.

3.2.1 Service Dependent Handler

SyncML은 일정관리, 주소록 이외에도 전자메일, 메시지, 텍스트 등 여러 가지 응용서비스에 적용할 수 있다. 이미 구현된 SyncML 서버에 이러한 새로운 서비스를 추가하려 할 때, SyncML 프로토콜에 관련된 부분까지 수정을 한다면 매우 많은 부분을 새로 구현하거나 변경해야 한다. 이렇게 추가되는 서비스에 독립적인 부분과 서비스에 종속적인 부분을 따로 모듈로 구현함으로써 서비스의 확장성을 유지 할 수 있고, 적은 비용으로 빠르게 서비스를 적용할 수 있도록 설계하였다.

3.2.2 Conflict Resolution

클라이언트 장치로부터 온 데이타가 서버에 있는 데이타와 충돌이 발생할 경우, 이를 해결하는 모듈이다. 충돌에 대한 정책은 서비스의 성격, 사용하는 용도에 따라 많이 좌우가 되는데 이렇게 정책에 따라 결정될 수 있는 부분을 쉽게 변경할 수 있도록 모듈로 구현하였다. 충돌은 장치간 데이타의 시간적 충돌(temporal conflict)과 의미적 충돌(semantic conflict)로 정의하였다[10]. 장치간 데이타의 시간적 충돌은 한 유저에 해당하는 복수개의 장치가 시간적 차이를 두어 서버의 같은 데이타를 변경할 경우 발생하며, 서버는 가장 최근에 변경된 데이타를 채택한다. 데이타의 의미적 충돌은 한 유저에 해당하는 여러 장치들이 하나의 데이타에 다른 명령을 요구하였을 때 발생하게 되며, 중복된 명령에 대한 처리를 담당한다.

3.2.3 Authentication Handler

SyncML 서버는 클라이언트 장치로부터 요구된 메시지가 인증된 사용자에 의한 것인지를 확인하기 위해 인증 절차가 필요하다. 현재 구현된 SyncML 서버가 지원하고 있는 인증 방식은 Basic과 MD5인데, 인증 방식의 정책 또한 서버의 정책사항이 된다. 이를 위해 인증에 관련된 부분을 모듈로 구현하였다.

3.2.4 Utility

장치들과 서버와의 동기화를 위해서는 메시지 안의 실제 데이타 부분에 대한 디코딩과 인코딩 작업이 필요하다. 서비스가 확장되면서 이러한 데이타의 디코딩, 인코딩 과정도 서비스에 맞게 확장되어야 한다. 그 밖에 여러 가지 서비스에 공통적인 기능을 Utility 모듈에 구현하였다.

3.3 Session Manager

SyncML 서버의 구조가 DLL 기반이므로 클라이언트로부터 요청된 메시지를 기반으로 활성화되어 동작되고 소멸되기 때문에 세션이 종료되면 해당 클라이언트에 대한 상태정보를 유지할 수 없다. 따라서 동기화가 시작되고 해당 동기화가 종료될 때까지 특정 클라이언트에 대한 세션 정보를 관리하는 Session Manager 프로세스를 구현하였다. 이 프로세스에서는 특정 사용자의 장치에 대하여 세션을 등록하고, 해제하고, 해당 장치에 대하여 서버가 처리한 커맨드와 Status 커맨드, Results 커맨드에 대한 처리 내역 등 세션관리에 관한 모든 내용을 관리하는 기능을 가진다.

Session Manager를 DLL로 구현하는 경우는 세션 정보를 데이타베이스나 파일시스템에 저장해야만 한다. 이는 많은 I/O작업 수행으로 인해 서버의 처리 속도에 대한 성능 저하를 초래한다. 본 논문에서는 이러한 성능 저하 문제를 해결하기 위해 Session Manager를 독립적인 프로세스로 구현하였다. Session Manager는 독립적으로 실행되는 프로세스로서 각 세션 정보를 메인 메모리 내에 연결 리스트 구조로 관리하며, RPC(Remote Procedure Call)을 통해서 SyncML 서버와 통신한다[11]. 그림 4는 RPC를 이용한 SyncML 서버를 보여준다.

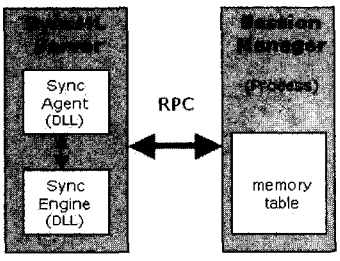


그림 4 RPC를 이용한 SyncML 서버

3.4 개방형 데이타베이스(ODBI) 구조

현재 구현된 서버는 주소록, 일정 관리와 같은 PIMS (Personal Information Management System) 서비스 데이타의 동기화를 제공한다. PIMS용 데이타 동기화 작업 시 필요한 데이타 관리와 데이타 저장소와의 인터페이스를 제공하는 기능을 담당하는 프레임이 Open

표 3 데이타베이스 이름 및 정의 내용

테이블 이름	정의 내용
ADDRESSBOOK, CALENDAR	클라이언트로부터 요구된 PIMS 용 데이타를 저장
MAPINFO 테이블	클라이언트측과 서버측 데이타베이스에서 독립적으로 관리하는 식별자에 대한 인덱스를 매핑
CHANGELOG 테이블	다수의 클라이언트 장치들과 서버와의 동기화 작업 동안에 발생되는 변경 사항에 대하여 효율적으로 처리하기 위하여 변경된 데이타에 대한 변경 정보를 관리
ANCHOR 테이블	SyncML클라이언트와 서버가 마지막으로 동기화 작업을 수행한 시점을 관리
USERINFO 테이블	클라이언트 장치에 대한 세부정보(URI, Credential, Nonce값) 등을 관리
NONCEPERDB 테이블	MimeType별로 인증을 위해 Nonce 정보를 관리

DB Interface이다. 구현된 데이타베이스는 동기화 작업 대상이 되는 컨텐츠(contents)와 데이타 동기화에 필요한 정보를 관리하기 위한 "SYNC" 데이타베이스인데, 다음과 같은 데이타베이스 테이블이 존재한다.

3.5 구현 환경

SyncML 서버는 운영체제는 Windows 2000, 데이타베이스는 오라클 8i를 사용하여 다량의 데이타를 안전하게 관리할 수 있도록 하였다. 개발 언어는 Visual C++ 6.0을 사용하였으며, 각 프레임들은 5개의 DLL로 구성하였다. 장치로부터 HTTP를 이용하여 메시지를 주고받기 위해 아파치 웹 서버 1.3.12를 사용하였고, 아파치 웹 서버가 전달 받은 메시지를 Tomcat 3.0을 이용하여 Java Class파일이 받고, 이를 Sync Agent로 전달하기 위하여 JNI(Java Native Interface)를 이용하였다[9].

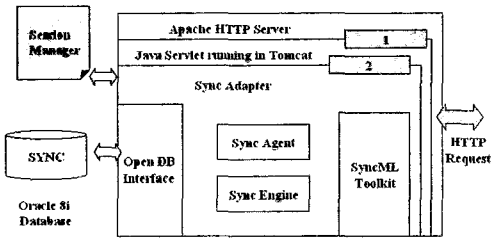


그림 5 SyncML 서버 구현 환경

3.6 구현된 SyncML 서버의 동기화 처리 절차

현재 구현된 SyncML 서버는 다수의 이동단말로부터 전달되는 주소록, 일정 관리와 같은 PIMS 데이타의 동기화가 가능하다. 구현된 서버에서 제공되는 동기화 절차 세부적으로 살펴보면 다음과 같다.

(1) 클라이언트 응용프로그램에서 추가/삭제/변경에 대한 처리가 발생
(2) 클라이언트의 사용자가 양방향 동기화(2-way Sync)를 요구함
(3) 클라이언트의 DBAdapter는 자신의 데이타베이스에서 추가/삭제/변경된 레코드를 가져옴
(4) 클라이언트는 DBAdapter로부터 가져온 데이타에 근거하여 SyncML 메시지를 생성
(5) 연결된 트랜스포트바인딩을 통하여 서버로 전달됨
(6) 서버는 동기화를 시작하기 앞서 인증, 앵커 값 확인과 같은 절차를 수행
(7) 서버의 Command Handler는 수신된 메시지를 분석하고, 새로운 메시지인 경우, Session Manager에 새로운 세션을 생성한 후, 해당 명령에 대한 수행 요구를 Sync Engine의 Service Dependent Handler에게 전달한다.
(8) Service Dependent Handler는 해당 명령을 처리하기 위한 동작을 수행한 후 정의된 Open DB 인터페이스를 통하여 데이타베이스를 변경한다. 이때 발생 가능한 데이타간 충돌(conflict)을 처리한다.
(9) 서버는 해당 사용자의 다른 장치에 대하여도 동기화를 맞추기 위하여 ChangeLog 테이블에 해당 변경 사항을 기록한다.
(10) 서버는 수신된 메시지에 대한 Status 메시지를 생성하여 클라이언트에 반환한다.
(11) 서버의 ChangeLog 테이블 정보로부터 서버의 데이타베이스에 추가/삭제/변경된 레코드를 가져온다.
(12) Map 테이블을 변경하고, ChangeLog 테이블을 반영한다.
(13) 클라이언트로 보낼 SyncML 명령 메시지를 생성한다.
(14) 서버는 클라이언트로 응답메시지를 전송한다.
(15) 클라이언트는 해당 응답메시지를 수신한 후 자신의 데이타베이스를 변경하고, ChangeLog 테이블을 변경한다.
(16) 클라이언트는 서버로부터의 SyncML 명령메시지에

대한 Status를 생성한다. 만일 서버로부터 새롭게 수신된 레코드가 존재하는 경우, 이에 대한 Map 정보를 생성하여 서버로 전달한다.

(17) 서버는 Map 정보를 수신하면, 해당 Map 테이블을 변경하고, ChangeLog 테이블을 반영하고 Status 메시지를 생성하여 클라이언트로 전달한다.

USERINFO 테이블

User ID	Device ID	···
jylee	1	···
jylee	2	···

CHANGELOG 테이블

User ID	Device ID	DB ID	GUID	Action Flag	Origin GUID
jylee	2	A	10052	R	
jylee	2	A	10053	A	
jylee	2	C	10030	C	10025

그림 6 Change Log Information

그림 6에서 보는 바와 같이 USERINFO 테이블에는 사용자 "jylee"가 2개의 장치를 가지고 있음을 나타내며, 현재 장치 2에서 특정 GUID값에 대하여 변경(R), 추가를 요구하였음을 나타낸다. 그림 7은 SyncML 서버측에서 유지하는 GUID와 LUID의 매핑 정보이다. SyncML 서버에서는 클라이언트와 서버간 마지막 동기화가 이루어진 시점을 가리키는 Anchor 정보를 관리하여야 한다[3]. Anchor 정보는 국제 표준 날짜 형식을 준하는 ISO 8601 기본 형식인 UTC(Universal Time, Coordinated)를 따르는 문자열로 구성되며[5], 정상적으로 동기화가 완료된 시점에서는 클라이언트와 서버가 동일한 Anchor 정보를 유지하게 된다. 따라서, 현재 동기화를 작업을 수행한 클라이언트의 장치의 다음 시도에는 클라이언트가 유지하고 있는 Anchor 값과 서버가 유지하고 있는 Anchor 값을 비교하여 동일한 경우는 정상적인 동기화 작업을 수행하고, 다른 경우는 전체 데이타에 대한 동기화 방식(Slow Sync)으로 수행하게 구현하였다.

4. 구현 서버의 검증

구현된 서버가 올바로 동작하는지를 확인하기 위해서 적합성 시험(Conformance Test)과 상호운용성 시험(Interoperability Test)을 수행하였다. 적합성 시험은 SyncML 표준규격을 바탕으로 구현한 시스템을 대상으로 하여 SIC(SyncML Interoperability Committee)에서 제시하는 테스트 절차를 시험 시스템에 적용하여 그 결과를 예측된 결과와 비교 분석하여 프로토콜 구현의 적합성 여부를 판별하는 것이다. 그림 8에서 SyncML 적합성 시험을 위한 절차를 기술하였다[1]. SyncML 적합성 시험을 위해서는 SIC로부터 받은 SICS(SyncML Implementation Conformance Statement)에 기록된 절차대로 12가지 테스트를 수행하였고 그 결과가 모두 SICS와 일치하였다.

Client Device

Client Database:

LUID	Data
11	Car
22	Bike
33	Truck
44	Shoes

Server Device

Server Database:

GUID	Data
1010101	Car
2121212	Bike
3232323	Truck
4343434	Shoes

Server Mapping Database:

GUID	LUID
1010101	11
2121212	22
3232323	33
4343434	44

그림 7 데이타 식별자 매핑 예제

표 4 구현 SyncML 서버를 이용한 테스트 결과

테스트 내용	결과	비 고
① Two-way with Client & Empty Server	○	
② Two-way with Client Add & Server Add	○	
③ Two-way with Client Replace & Server Replace	○	
④ Two-way with Client delete & Server Delete	○	
⑤ Two-way with Client Add(shows empty server sync data)	○	
⑥ Two-way with Server Add(shows empty client sync data)	○	
⑦ Two-way with Server with large amount of data(shows multiple message)	○	
⑧ Two-way with Client with large amount of data(shows multiple message)	○	
⑨ Two-way with Server responding busy	○	
⑩ Two-way with Server not responding		서버는 해당 없음
⑪ Two-way with Communication broken during sync	○	
⑫ Two-way Slow Sync	○	

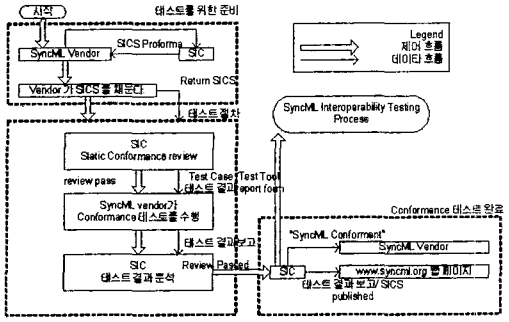


그림 8 SyncML 적합성 시험 절차

SyncML 상호운용성 시험(Interoperability Test)은 SyncML Conformance를 획득한 시스템에 한해서 SyncFest 라는 SyncML 회원사들 간의 회의에서 실시된다. 본 연구팀은 SyncFest에는 참가하지 않았으며, 대신 국내에서 최초로 SyncFest에서 상호운용성 시험에 통과한 ㈜네오스텝스의 클라이언트를 서버와 연동하여 시험을 수행하였으며, 적합성 시험에 적용한 12가지 시험 절차를 적용하여 연동 테스트를 수행함으로써 상호운용성을 검증하였다.

5. 성능 테스트

구현한 CNU SyncML Server의 성능 테스트를 위하여 다음 표 5의 조건 하에서 측정 작업을 수행하였다. 본 연구팀은 SyncML서버 구조와 세션 정보 관리 방식을 기준으로 하여 각각 CNU SyncML Server 2.0, CNU SyncML Server 3.0, CNU SyncML Server 3.1의 세 가지 버전을 구현하였다. CNU SyncML Server 2.0은 Session Manager를 포함한 모든 프레임이 동적 링킹, 로딩 기반이고 세션 정보를 데이타베이스에서 관리한다. CNU SyncML Server 3.0은 Session Manager가 독립적인 프로세스 구조이며 세션 정보를 데이타베이스에서 저장 및 관리하지만, 데이타베이스와의 연결을 한 세션 당 한 번만 수행함으로써 세션 서비스 시간을 줄이도록 구현한 것이다. 마지막으로, CNU SyncML Server 3.1은 Session Manager가 독립적인 프로세스구조이며 세션 정보를 메모리에서 관리하는 방식이다. 세가지 버전에 대하여 모두 동일 환경하에서 테스트를 수행하고 동기화 작업의 평균 소요시간을 측정하였다. 이때, 요청 및 응답 메시지의 네트워크 전송 소요 시간은 고려되지 않았다. 테스트 메시지 안에는 ADD, REPLACE, DELETE커맨드들과, 해당 디바이스의 최초 접근 시 수행되는 PUT, GET 커맨드, 기타 커맨드 등을 포함하였다.

표 5 성능 테스트를 위한 파라미터

파라미터	파라미터 값
사용자 수	10명
디바이스 수	사용자당 2개 이상
동기화 종류	Two-way Sync Slow Sync One-way Sync from Client One-way Sync from Server Refresh Sync from Client only Refresh Sync from Server only
동기화 요청 횟수	500번

그림 9는 CNU SyncML Server 의 성능 테스트 결과를 그래프로 나타낸 것이다. CNU SyncML Server 3.1은 Session Manager가 세션 정보를 메모리에서 관리하므로, 데이타베이스에 대한 연결 생성, 데이타 접근, 연결 종료 과정 등이 생략되어 빠른 속도로 정보를 처리할 수 있어 기존의 Server 2.0에 비해서는 평균 44%, CNU SyncML Server 3.0에 대해서는 평균 31%의 처리 속도가 향상되는 결과를 얻을 수 있었다.

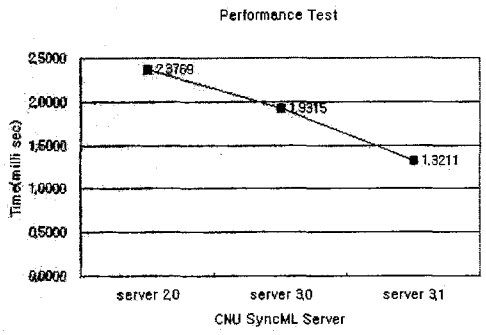


그림 9 CNU SyncML Server 성능 테스트

SyncML Server에서 처리하는 데이타는 크게 동기화의 직접적인 대상이 되는 데이타와, 동기화 작업 수행을 위해 서버가 유지해야 하는 세션 정보들로 나눌 수 있다. 가령, 주소록과 같은 동기화의 대상이 되는 데이타는 당연히 안정적인 저장을 위하여 데이타베이스에서 관리하도록 하는 것이 유리하지만, 세션 유지에 관련된 정보는 한 클라이언트가 동기화 요청 시부터 종료까지의 세션 동안에만 필요하므로, 빠른 수행을 위해서는 오히려 데이타베이스에서 관리하는 것보다 메모리에서 관리하는 구조가 효율적이라고 볼 수 있다. CNU Sync

ML Server 3.1가 바로 Session Manager를 통해 세션 정보를 메모리에서 관리하는 구조이며, 실험 결과를 통하여 처리 속도가 향상 되어짐을 입증하였다.

6. 결 론

본 논문에서는 데이타 동기화를 위해서 표준 데이타 동기화 기술의 필요성과 함께, 표준 동기화 기술인 SyncML에 대해 소개 하였다. 또한 개발한 SyncML 서버의 전체 프레임웍을 제시하고 SyncML 서버내에서 구현된 5개의 프레임 기능과 구현 내용을 제시하였다. 본 논문에서 제시한 SyncML 서버 프레임 구조는 각기 기능별로 모듈화하여 라이브러리화 하여 사용함으로써, 확장성과 이식성이 높아 불필요한 코드의 낭비를 줄일 수 있다. 특히 Sync Agent는 서비스에 종속적인 Sync Engine과는 달리 응용 서비스에 종속적이지 않으면서 데이타 동기화를 수행할 수 있도록 설계된 프레임이므로, 개발자가 새로운 서비스를 추가한다거나 업그레이드가 한다고 하더라도 전체 모듈을 수정하는 오버헤드를 없애는 장점을 지닌다. 또한 Sync Engine 프레임을 두어 서비스에 종속적인 부분들에 대한 반영을 쉽고, 빠르게 대처할 수 있도록 설계하였으며, 메모리 기반의 Session Manager 프레임을 두어 클라이언트가 요구한 명령들에 대한 이력을 관리하고, 서버가 응답하고 전송한 데이타 요소에 대한 일관적인 관리를 가능하게 하여, 불필요한 데이타의 교환을 줄이고, 데이타에 대한 일치성을 높였다.

향후 과제로는 현재 SyncML 서버에서 제시하는 보안 메카니즘에 더 나은 안정성을 제공하기 위해 해당 메시지를 해독할 수 있는 키를 이용하는 등의 방식을 추가적으로 제안하고 아울러 사용자 인증뿐 아니라 메시지 자체에 대한 보안을 보장할 수 있도록 보안 알고리즘을 강화하는 방안을 고려할 수 있겠다. 더 나아가 현재 윈도우 기반의 C++로 개발되어있는 현재 서버를 향후 플랫폼에 독립적이고 안정적인 자바 기반의 개발 환경으로의 개선을 생각해 볼 수 있다.

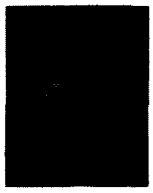
감사문

본 서버 구현에는 충남대학교 컴퓨터공학과 석사과정 이종필, 이지연, 김창회도 참여하여 기여하였음.

참 고 문 헌

[1] SyncML Initiative, http://www.syncml.org
[2] SyncML Initiative, Building an Industry-Wide Mobile Data Synchronization Protocol, SyncML White Paper, Mar. 20, 2000.
[3] SyncML Initiative, SyncML Architecture Version 0.2, May 10, 2000.
[4] SyncML Initiative, SyncML Representation Protocol, version 1.0.1, June 15, 2001.
[5] SyncML Initiative, SyncML Synchronization Protocol, version 1.0.1, June 15, 2001.
[6] SyncML Initiative, SyncML HTTP Binding, version 1.0.1, June 15, 2001.
[7] Extensible Markup Language (XML) 1.0 Second Edition), http://www.w3.org/TR/REC-xml
[8] 하인숙,조재혁,양지현, "SyncML 레퍼런스 툴킷 그 내부를 보자", 마이크로 소프트웨어 2001년 7월, pp.324-336, July 2001.
[9] SyncML Initiative, SDA2 Specification Version 0.2, Aug. 21, 2000.
[10] 조진현,류수희,이지연,최 훈, "멀티플 장치간 데이타 충돌 해결 방식 연구", Technical Document, 충남대학교 컴퓨터 공학과, Jan, 2002.
[11] JiYeon Lee,ChangHoe Kim,Hoon Choi, "Implementation of the Session Manager for a Stateful Server," IEEE TENCON, Beijing, China, Oct. 29, 2002.

이 병 윤
1990년 2월 홍익대학교 전자계산학과 학사. 1992년 2월 서강대학교 전자계산학과 석사. 1992년 2월 ~ 현재 한국전자통신연구원 네트워크연구소 선임연구원 2000년 12월 충남대학교 컴퓨터공학과 박사 수료. 관심분야는 모바일 컴퓨팅, 분산 시스템, 통신망관리시스템



이 길 행
1984년 2월 전남대학교 전자계산학과 학사. 1986년 2월 한국과학기술원 전자계산학과 석사. 1996년 2월 한국과학기술원 전자계산학과 박사. 1986년 2월 ~ 현재 한국전자통신연구원 네트워크연구소 책임연구원. 관심분야는 통신망관리시스템 로드밸런싱과 분산처리, 음성인식, 실시간데이타베이스관리시스템

조 진 현
2000년 2월. 충남대학교 컴퓨터공학과 학사. 2002년 2월 충남대학교 컴퓨터공학과 석사. 2002년 2월 ~ 현재 마이엔진㈜ 전임연구원. 관심분야는 모바일 컴퓨팅 분산 시스템

류 수 희
2000년 2월 충남대학교 컴퓨터공학과 학사. 2002년 2월 충남대학교 컴퓨터공학과 석사. 2002년 2월 ~ 현재 이노에이스㈜ 연구원. 관심분야는 모바일 컴퓨팅 분산 시스템

최 훈
1983년 2월 서울대학교 컴퓨터공학과 학사. 1990년 12월 Duke University 전산학과 석사. 1993년 5월 Duke University 전산학과 박사. 1983년 ~ 1996년 한국전자통신연구원 광대역통신망연구부 근무. 1996년 ~ 현재 충남대학교 컴퓨터공학과 부교수. 2000년 미국 NIST(National Institute of Standards and Technologies) 객원연구원. 관심분야는 모바일 컴퓨팅, 분산 시스템, Fault-tolerant 시스템