2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

SyncML 데이터 동기를 위한 데이터베이스 설계 및 구현

*이지연[0], *조진현, *최 훈
*충남대학교 컴퓨터 공학과

e-mail : {eunbi, jhcho, hchoi}@ce.cnu.ac.kr

Design and Implementation of the SyncML Database for Data Synchronization

*JiYeon Lee[0], *JinHyun Cho, *Hoon Choi
*Dept. of Computer Engineering, Chungnam National University

요 약

이동 통신 기술과 인터넷의 비약적인 발전으로 개인이 관리해야 하는 정보량의 증가 및 복수 개 디바이스로의 데이터 분산화로 인해 데이터 동기화를 위한 기술의 요구가 절실해지고 있다. 본 논문에서는 이기종 단말간의 상호 운용성을 보장하면서 데이터 동기화를 위한 표준 프로토콜인 SyncML 과 SyncML 클라이언트 서버 구조를 소개하고, PIMS 데이터 동기화에 필요한 데이터 관리 방안 및 구현 내용에 대해서 기술한다.

1. 서론

이동 통신 기술과 인터넷의 비약적인 발전으로 생산되는 정보의 양은 급속도로 증가하고 있다. 개인이 관리해야 하는 정보도 증가하였을 뿐만 아니라, 데이터가 PDA(Personal Data Assistant)나 휴대용 PC, 이동 단말기 등과 같이 복수 개의 디바이스로 분산되는 경우도 많아졌다. 일반적으로 개인의 일정, 주소록과 같은 정보는 PC, 이동 단말기, PDA 등의 여러 디바이스에서 사용할 수 있으나, 어느 디바이스이든 항상 데이터가 일치하도록 보장하기는 어렵다. 하나의 디바이스에서 주소록 정보가 갱신되는 경우 사용자의 다른 디바이스에 대해 모두 동기화를 하려면, 현재로서는 각 단말 개발자나 서비스 제공자들의 독자적인 동기화 프로토콜을 통해 동기화 작업을 수행할 수는 있으나, 이는 매우 제한적이며, 장비에 종속적이다. 즉, 서로 다른 응용 프로그램이나 이기종 단말간 상호 운용성이 보장되지 않는 문제점이 있다[1].

이런 문제를 해결하고자 2000 년 2 월, 유무선 이동 통신 분야의 몇 개 대표적인 산업체들의 주도로 SyncML (Synchronization Markup Language) 이라는 새로운 업체간 컨소시엄이 결성되었다[1]. SyncML 은 데이터 동기화를 위한 표준 프로토콜을 제시하여 이기종 단말간 상호 운용성을 보장한다.

본 논문의 2 장에서는 SyncML 에 대한 소개와 SyncML 클라이언트 서버간 동작 방식과 SyncML 구조에 대하여 설명하고, 3 장에서 실제 본 연구팀이 구현한 PIMS(Personal Information Management Service) 데이터 동기화에 필요한 데이터 관리 방안 및 구현 내용에 대하여 기술한다.

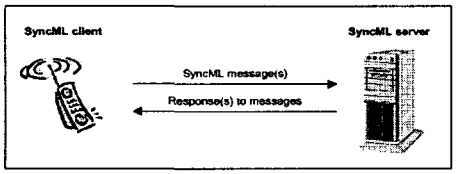
2. SyncML 구조

SyncML 은 서로 다른 디바이스, 어플리케이션 간의 데이터 동기화를 위한 표준 프로토콜이다. 데이터 동기화(Data Synchronization)란 갱신된 데이터를 변경하고 데이터간의 버전 차이를 해결하는 중재 동작을 의미한다[2]. SyncML 은 이기종 간의 상호 운용성을 보장하기 위해 메시지 기술 언어로서 XML (Extensible Markup Language)을 사용하며, 메시지 내의 명령어 및 데이터 동기화를 위한 정보는 XML DTD(Document Type Definition)로 정의되어 있다.

SyncML 클라이언트 디바이스와 서버간 데이터 동기화를 위한 동작 수행과 구조에 대하여 살펴보도록 한다.

2.1 SyncML 클라이언트 서버간 동작

<그림 1>은 SyncML 클라이언트 기능을 수행하는 이동 단말기와 SyncML 서버 기능을 수행하는 서버간의 데이터 동기화를 도시화 한 것이다.



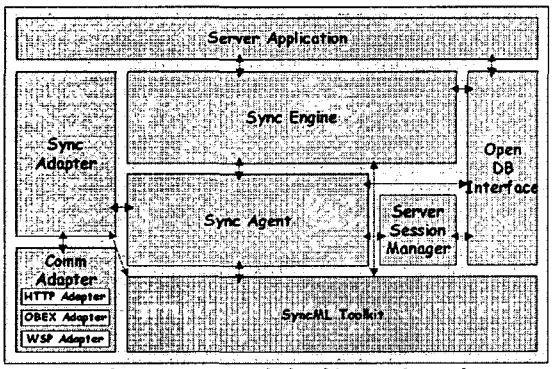
<그림 1> SyncML 클라이언트 서버간 동기 예제

SyncML 클라이언트가 먼저 서버에게 갱신된 데이터를 포함한 메시지를 전송하면, SyncML 서버는 클라이언트가 요청한 동기화 타입에 의해 서버측 데이터와 동기화 작업을 수행한 후 다시 클라이언트에게 작업 결과를 송신한다[3][4].

2.2 SyncML 서버 구조

SyncML 서버는 SyncML 클라이언트로부터 수신한 메시지에 대한 처리와 서버측 변경 사항에 대한 메시지 송신 기능을 포함하며, 필요한 경우 작업 결과를 SyncML 클라이언트에게 송신하는 기능을 갖는다.

<그림 2>은 본 연구팀이 구현한 SyncML 서버의 프레임 구조이다.



<그림 2> SyncML 서버 기능 모듈 구성

SyncML 서버는 크게 Open DB Interface, SyncML Toolkit, Sync Engine, Sync Agent, Session Manager, Server Application 기능을 담당하는 모듈로 나뉘어진다. 서버 어플리케이션을 제외한 5 개의 모듈은 DLL(Dynamic Link Library) 형태로 구성되었으며, 각 역할은 다음과 같다.

- Server Application 은 서버측에서 발생하는 변경 사항 처리를 위한 주체가 되며 독립적인 프로그램으로 동작한다.
- SyncML Adapter 는 클라이언트 디바이스로부터 전송되어지는 메시지를 받아 SyncML Toolkit 으로 전달하는 역할을 담당한다.
- SyncML Toolkit 은 클라이언트 디바이스로부터 전달되어진 메시지를 디코딩하고, 서버측에서 보내질 메시지를 인코딩하는 기능을 담당한다.
- Sync Agent 는 SyncML 프로토콜에 기준하여 클라이언트 디바이스로부터 전달된 메시지를 처리한다.
- Sync Engine 은 Sync Agent 에서 메시지 처리 작업에 서비스 정책에 종속적인 문제, 데이터간 충돌 문제 발생 시 해결을 위한 역할을 담당한다.
- Session Manager 는 클라이언트와 동기화를 위해 세션을 유지하고 관리하는 기능을 한다.
- Open DB Interface 는 SyncML 클라이언트 서버간 데이터 동기작업에 필요한 데이터 관리 및 데이터 저장소와의 인터페이스를 제공한다.

SyncML 서버의 개발 환경은 다음과 같다.

- 개발 OS(Operating System): Windows 2000
- DBMS(Database Management System) : Oracle8i
- 개발 Language : Visual C++
- Web Server : Apache Web Server 1.3.12

3. PIMS 데이터 동기를 위한 정보 관리 방안 및 구현

현재 SyncML 스펙 1.0.1 에서는 주소록, 일정 관리와 같은 PIMS 데이터의 동기화가 가능하다. PIMS 용 데이터 동기화 작업 시 필요한 데이터 관리와 데이터 저장소와의 인터페이스를 제공하는 기능을 담당하는 모듈이 Open DB Interface 이다.

구현된 SyncML 서버의 데이터베이스 스키마와 SyncML 클라이언트 서버간 데이터 동기화를 위해 사용하는 주요 데이터베이스 테이블과 역할에 대해서 살펴보도록 한다.

3.1 SyncML 서버 데이터베이스 스키마

본 구현에서는 크게 두개의 데이터베이스를 갖는다. 하나는 동기화 작업 대상이 되는 컨텐츠(Contents)와 데이터 동기화에 필요한 정보를 관리하기 위한 "SYNC" 데이터베이스이고, 다음은 서버측에서 한 세션 내에 계속 유지해야 하는 정보를 관리하기 위한 "SYNCSM" 데이터베이스이다.

"SYNC" 데이터베이스에는 다음과 같은 데이터베이스 테이블이 존재한다. 우선, PIM 용 데이터 저장을 위해 ADDRESSBOOK, CALENDAR 테이블을 갖는다. 같은 항목에 대한 클라이언트측과 서버측 데이터베이스에서 각각 관리하는 ID 가 서로 다를 수 있으므로

ID 맵핑을 시켜주기 위해 MAPTABLE 을 갖는다.

SyncML 프로토콜은 클라이언트 디바이스와 서버와의 동기화 작업 동안에 발생되는 내용에 대한 기록을 하여, 복수개의 다른 디바이스들의 접근이 있을 때 동기화가 가능하도록 할 것을 요구한다. 구현에서는 CHANGELOG 테이블에 정보를 관리한다. 또한, SyncML 클라이언트와 서버가 마지막으로 동기화 작업을 수행한 시점을 알기 위한 Anchor 정보를 ANCHOR 테이블에서 관리한다. 클라이언트 디바이스에 대한 URI, Credential, Nonce 값 등을 USERINFO 테이블에서 관리하며, 각 MimeType 별로 인증을 위해 NONCEPERDB 테이블에서 Nonce 정보를 관리한다.

"SYNCSM" 데이터베이스는 서버측에서 한 세션 내에 계속 유지해야 하는 정보를 관리한다. 한 세션이라 함은 클라이언트 디바이스가 처음 동기화 요청을 시작하여 클라이언트측 변경사항과 서버측 변경사항에 대한 동기화 작업이 모두 끝나는 시점까지를 일컫는다. HTTP (HyperText Transfer Protocol) 전송 프로토콜[2][5]을 이용하므로, 초기 연결에서 설정된 정보들을 세션의 마지막까지 유지해야 할 필요성이 있으므로, 이는 Session Manager 에서 기능을 수행하며 실제적인 데이터는 "SYNCSM" 데이터베이스에 저장,관리한다.

"SYNCSM" 데이터베이스에는 다음과 같은 테이블을 두어 세션 정보를 관리한다. 세션 내에 유지해야 하는 세션 ID, 인증 종류에 대한 정보를 유지하는 SESSIONINFO 테이블, 클라이언트로부터 전송되어진 SyncML 메시지 내의 커맨드 정보들을 관리하는 CMDINFO 테이블, 처리 결과 값에 대한 정보를 유지하는 STSINFO 테이블, 그리고 RESULTINFO 테이블이 있다.

3.2 데이터 동기에 필요한 정보 관리

SyncML 클라이언트 서버간 데이터 동기화 작업을 수행함에 있어서 중요한 정보가 되는 Change Log Information, Map 테이블 정보와 Anchor 정보 관리 방안에 대해서 살펴보도록 한다.

클라이언트 디바이스와 서버와의 동기화 작업 동안에 발생되는 정보 변경 기록을 Change Log Information 이라 하며, 복수개의 다른 디바이스에서 동일한 정보를 변경하는 상황에서는 동기화 작업을 위해 Change Log Information 을 반드시 유지해야 한다[3]. SyncML 프로토콜 스펙에서는 Change Log Information 에 대한 구조체나 운영 원칙에 대한 구체적인 사항을 기술하지 않는다. 이는 서버 구현자가 결정할 사항이다.

<그림 3>은 본 구현에서 적용한 Change Log Information 을 기록을 위한 테이블들이다. DB ID 의 필드의 내용은 변경 대상이 Address Book 데이터인 경우는 'A', Calendar 인 경우는 'C'로 나타낸다. GUID(Global Unique Identifier)[3] 필드는 각각의 데이터 항목을 가리키는 서버측 고유 번호이며, Origin GUID 필드는 <COPY> 커맨드 실행 시 원본 GUID 를 지시한다. Action Flag 필드는 변경 작업 내용을 뜻하며 <ADD>는 'A', <REPLACE>는 'R', <DELETE>는 'HD', COPY 는 'C'로 표현한다.

USERINFO 테이블은 SyncML 서버 측에 등록된 사용자의 모든 디바이스에 대한 목록을 가지고 있다.

USERINFO 테이블

User ID	Device ID	...
jylee	1	...
jylee	2	...

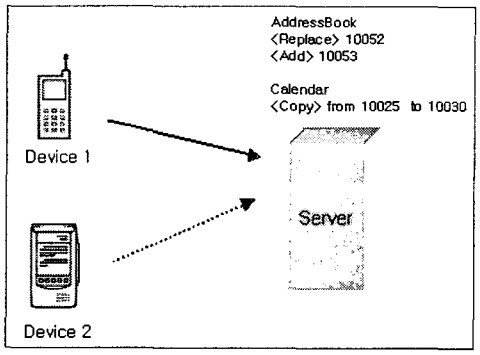
CHANGELOG 테이블

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 3> Change Log Information

예제로 Change Log Information 기록되는 과정을 보면 다음과 같다. 현재 사용자의 ID 가 "jylee"이고, 디바이스 ID 가 1 인 SyncML 클라이언트 디바이스가 데이터 동기화를 위해 SyncML 서버로 접근한 상황을 설정하고, 다음과 같은 변경 작업이 이루어졌다고 가정하자. ADDRESSBOOK 의 GUID 10052 번 데이터에 대한 <REPLACE>와 GUID 10053 번 데이터에 대한 <ADD> , 그리고 CALENDAR 의 GUID 10030 번 데이터가 GUID 10025 번 데이터를 <COPY> 하였다.

<그림 4>는 예제의 상황을 도시화 한 내용이다.



<그림 4> 복수 개의 디바이스에 대한 동기화 예제

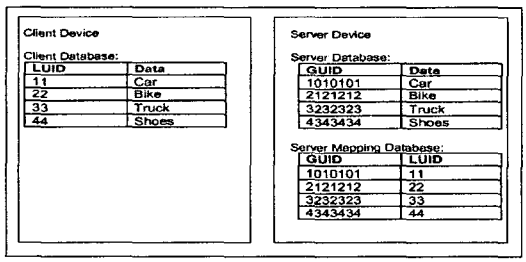
현재 연결된 "jylee"의 1 번 디바이스에 대해서는 변경작업이 이루어진 상태이며, 복수개의 디바이스 데이터 동기를 위해 Change Log 테이블에 정보를 기록해야 한다. 따라서, "jylee" 사용자의 다른 디바이스가 존재하는지를 알기 위해 USERINFO 테이블을 참조한다. "jylee" 사용자가 디바이스 1 번 이외에 디바이스 2 번을 가지고 있으므로, CHANGE LOG 테이블에 디바이스 1 번이 행한 변경 작업에 대해서 똑같이 디바이스 2 번이 할 수 있도록 작업 내용을 기록한다. 즉, ADDRESSBOOK 테이블의 GUID 10052 번 데이터의 <REPLACE>, 10053 번 데이터의 <ADD>, CALENDAR 테이블의 GUID 10025 번 데이터를 <COPY>하여 새로 생긴 10030 번 데이터가 있음을 기록한다. 이로써, 다

음 "jylee"사용자의 디바이스 2 번이 서버와 연결할 때 Change Log Information 을 참조하여 동기화 작업을 수행하게 된다.

SyncML 클라이언트와 서버간의 데이터 동기화 수행을 위해 관리해야 하는 또 하나의 중요한 정보는 Map 테이블 정보이다.

클라이언트측의 데이터 식별자를 LUID (Locally Unique Identifier) 라 하고, 서버측 데이터 식별자를 GUID 라 한다. 일반적으로 SyncML 클라이언트는 서버에 비해 메모리나 시스템 성능이 낮으므로, 데이터 식별자의 길이가 서버와 다를 수 있다. 따라서, 같은 데이터를 가리키기 위해서는 서버측에 클라이언트측에서 사용하는 LUID 와 서버측에서 사용하는 GUID 를 맵핑시켜주는 테이블을 둔다[3]. 동기화 작업 수행시에 서버측 Map 테이블을 참조하여 LUID 에 해당하는 GUID 를 얻어 데이터에 대한 동기화 작업을 수행할 수 있다. 서버측에서 새로운 데이터가 생성되는 경우도 Map 테이블에 GUID 와 LUID 의 맵핑 정보를 유지해야 한다. 이 때, LUID 값은 아직 부여되지 않은 상태이므로 다음에 클라이언트로부터 MAP 커맨드에 의해 전송되어지는 LUID 값으로 맵핑 테이블의 정보를 변경해야 한다.

<그림 5> SyncML 서버측에서 유지하는 GUID 와 LUID 의 맵핑 정보 예제이다.



<그림 5> 데이터 식별자 맵핑 예제

SyncML 클라이언트와 서버측 데이터베이스 테이블에 존재하는 데이터 항목에 대해서 서버측 맵핑 테이블에서 GUID 와 LUID 의 맵핑 정보를 유지하고 있다.

구현에서는 여러 사용자의 복수 개 디바이스에 대한 경우를 고려하여 맵핑 정보 유지를 위한 MAPTABLE 에 GUID, LUID 이외에 User ID, Device ID, DB ID 필드를 추가하였다.

SyncML 서버에서는 클라이언트와 서버간 마지막 동기화가 이루어진 시점을 가리키는 Anchor 정보를 관리하도록 한다[3]. 구현에서는 SyncML 클라이언트와 서버간의 동기화 작업이 모두 수행된 후에 서버에서 클라이언트에게 마지막 SyncML 메시지를 보내면서 Anchor 정보를 갱신하였다. Anchor 정보는 국제 표준 날짜 형식을 준하는 ISO 8601 기본 형식인 UTC 를 따르는 문자열로 구성되며[2], 정상적으로 동기화가 완료된 시점에서는 클라이언트와 서버가 동일한 Anchor 정보를 유지하게 된다. 따라서, 현재 동기화를 작업을 수행한 클라이언트의 디바이스의 다음 시도에는 클라이언트가 유지하고 있는 Anchor 값과 서버가 유지하고 있는 Anchor 값을 비교하여 동일한 경우는 정상적인 동기화 작업을 수행하고, 다른 경우는 전체 데이터에 대한 동기화 방식으로 수행하게 하였다.

Anchor 정보는 <ALERT> 커맨드의 <META> 엘리먼트 안에 들어가게 된다[4][7].

4. 결론

유무선 이동 통신의 보편화로 개인이 소유하게 되는 단말의 개수도 증가하였다. 여러 디바이스로 분산되어 존재하는 동일한 데이터에 대한 동기화 방법을 각 단말 회사 또는 서비스 회사에서 제공하지만, 상호 운용성이 보장되지 않는 문제점이 존재한다. 이런 문제에 대한 해결 방안으로 등장한 동기화 작업을 위한 표준 프로토콜인 SyncML 에 대하여 소개하고, SyncML 클라이언트 서버 구조, PIMS 데이터 동기화 작업을 위해 필요한 데이터의 효율적인 관리 방안 및 구현 내용에 대해 살펴 보았다.

추후 연구 과제로 데이터베이스의 최적화 및 성능 향상을 위한 튜닝작업이 필요하며, 실제 사용자들이 많이 사용하는 PIMS 어플리케이션을 연동함으로써 사용자에게 편리한 인터페이스를 제공하도록 한다. 또한, 다양한 PIMS 데이터 동기를 위한 상위 버전에 대해서도 지원이 필요하다.

참고문헌

[1] SyncML Initiative, http://www.syncml.org
[2] SyncML Initiative, SyncML Sync Protocol version 1.0.1, 2001. 6.15
[3] SyncML Initiative, SyncML Representation Protocol version 1.0.1, 2001. 6.15
[4] SyncML Initiative, SyncML Architecture, version 0.2, 2001. 5.10
[5] SyncML Initiative, SyncML HTTP Binding, version 1.0.1, 2001. 6.15
[6] SyncML Initiative, SyncML Meta-Information DTD, version 1.0.1, 2001. 6.15
[7] 하인숙, 조재혁, 양지현, "SyncML 레퍼런스 툴킷 그 내부를 보자", 마이크로 소프트웨어, pp.324-336, 2001. 7