제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

SyncML 에서 CLI(Change Log Information)를 이용한 Conflict Resolution

라황균º, 오세만
동국대학교 컴퓨터공학과
e-mail : {amaranth,smoh}@dongguk.edu

Conflict Resolution using CLI in SyncML

Hwang-Gyun Laº, Se-Man Oh
Dept. of Computer Engineering, Dongguk University

요 약

모바일 환경이 일반화 되면서 모바일 디바이스간 또는 모바일 디바이스와 PC 또는 서버와의 동기화가 요구되었다. 모토로라, 에릭슨, 노키아 등의 업체들은 컨소시엄을 구성하여 데이터의 동기화와 다양한 전송 프로토콜과의 연동을 목표로 산업계의 표준인 SyncML(Synchronization Markup Language)을 제정하였다. 그러나, 동기화 과정에서 둘 이상의 클라이언트가 동일한 데이터 동기화를 요청했을 때 데이터 충돌이 일어나게 된다.

본 논문에서는 데이터 동기화 과정에서 발생할 수 있는 다양한 충돌에 대해서 분석하고, 체계적으로 분류하였다. 이를 바탕으로 동기화에 대한 정보를 추적할 수 있는 Change Log Information을 구성하고 운영 원칙을 제정하여, 데이터의 안정성과 일치성을 보장하기 위한 Conflict Resolution을 제안한다.

1. 서론

모바일 환경이 일반화 되면서 모바일 디바이스간 또는 모바일 디바이스와 PC 또는 서버와의 동기화가 요구되었다. 지금의 동기화 방식은 디바이스 제조업체 및 통신 서비스업체 등이 자사의 사용자를 위한 응용 프로그램을 제공하는 방식이었으며, 사용자들의 PIMS(Personal Information Management System)등의 정보에 대한 동기화만이 제한적으로만 가능했다. 이러한 폐쇄적인 방식은 다양한 디바이스, 플랫폼 및 네트워크 프로토콜에서의 동기화를 수행하지 못한다. 이와 같은 문제점을 해결하기 위해서 상호 운용성이 보장되는 동기화에 대한 표준이 요구되었다.

모토로라, 에릭슨, 노키아 등의 업체들은 컨소시엄을 구성하여 데이터의 동기화와 다양한 디바이스 및 전송 프로토콜과의 연동을 목표로 산업계의 표준인 SyncML(Synchronization Markup Language)을 제정하게 되었다. 그러나, 표준은 다수의 디바이스간에 빈번히 일어나는 예측할 수 없는 상황에 대한 데이터의 일치성과 안정성을 보장할 수 없다. 동기화 과정에서 둘 이상의 클라이언트가 동일한 데이터 동기화를 요청했을 때 데이터 충돌이 일어나게 된다. SyncML 에서는 Change Log Information 을 통해 동기화시 추적할 수 있는 유일한 정보를 제공한다. 그러나 Change Log Information 의 운영 원칙과 구현에 대하여 스펙에서는 구체적 기술을 하지 않고 있다.

본 논문에서는 데이터 동기화 과정에서 발생할 수 있는 다양한 충돌에 대해서 분석하고, 체계적으로 분류하였다. 이를 바탕으로 Change Log Information 의 정보를 구성하고 운영 원칙을 제정하여, 데이터의 안정성과 일치성을 보장하기 위한 Conflict Resolution 을 제안한다.

2. 배경 연구

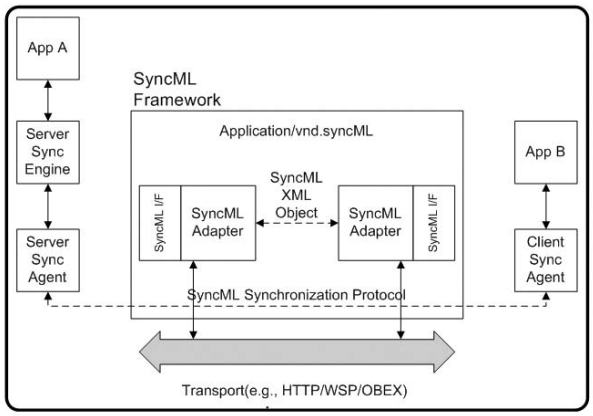
2.1 데이터 동기화

데이터 동기화란 둘 이상의 논리적 장치간 원시 데이터와 목적 데이터간의 특정 데이터를 일치시켜주는 기술이다. 모바일 디바이스 제조사에서 자사의 데이터 동기화를 이용한 응용 프로그램을 사용자에게 제공하고 있다. Win CE 계열의 ActiveSync, 모바일 디바이스

의 DataSync2000 이 그 대표적 예이다. 기존의 데이터 동기화 방식은 네트워크 환경과 OS 및 응용프로그램에 종속적이었다. 그러므로 사용자의 개인 정보의 저장용으로 이용되거나 제한적인 상업적 용도로 이용되었다.

2.2 SyncML 의 개요

SyncML 은 XML(eXtensible Markup Language)을 기반으로 한 데이터 동기화의 표준이다. 사용자들은 디바이스와 네트워크 프로토콜에 상관없이 자신의 데이터를 동기화 할 수 있다. [그림 1]은 SyncML 프레임워크와 데이터의 흐름을 보여주고 있다.



[그림 1] SyncML Framework

프레임워크는 SyncML 표현 프로토콜과 개념적인 SyncML Adapter, SyncML Interface 로 이루어져 있다. “App A” 와 “App B” 는 데이터 동기화를 제공하는 서비스들이다. 서비스와 디바이스는 HTTP, WSP, OBEX 와 같은 네트워크 트랜스포트를 통해 연결되어 있다. Sync 엔진은 데이터 동기화 프로토콜을 구현하고 응용 서비스가 이를 이용한다. 또한 데이터 충돌에 대한 해결을 엔진에서 한다. Sync 서버 에이전트는 네트워크로 접근하는 Sync 엔진을 관리하고 클라이언트 어플리케이션과 데이터 동기화를 통신한다. Sync 서버 에이전트의 이러한 기능들은 SyncML I/F 를 호출함으로써 수행된다. SyncML I/F 는 SyncML 어댑터에 대한 API 이고 SyncML 어댑터는 SyncML 형식을 갖춘 객체들을 전달하기 위하여 이용하는 프로세스이다. SyncML 클라이언트 에이전트는 SyncML I/F 를 통해 App B 가 네트워크와 SyncML 어댑터에 접근할 수 있도록 한다.

2.3 Change Log Information

SyncML Sync 프로토콜은 동기화 과정에서 변경된 부분에 대한 로그를 관리한다. 또한, 동기화 과정 중에 발생한 변경된 정보를 추적할 수 있는 유일한 정보를 제공한다. Sync 프로토콜 스펙은 Change Log Information 이 관리되는 원칙에 대해서 기술하지 않으며 운영원칙과 구현은 개발자의 몫이다.

SyncML Sync 프로토콜에서는 데이터 동기화가 제대로 수행되었는지를 판단하기 위한 정보를 제공한다.

Sync Anchor 는 서버와 클라이언트의 Anchor 값을 비교하여 동기화를 수행할 수 있는 정보를 제공한다. Last Sync Anchor 는 가장 최근에 동기화를 성공한 시간을 의미하며, Next Sync Anchor 는 지금 현재 동기화가 수행되고 있는 시간을 의미한다. 마지막 동기화 시점을 나타내는 정보는 앵커 정보와 일대일로 매칭해 SyncML 엔진 내부에 저장해야 하고, 마지막 동기화 시점을 나타내는 정보는 Change Log Information 에서 사용하는 방법과 매칭돼야 한다.

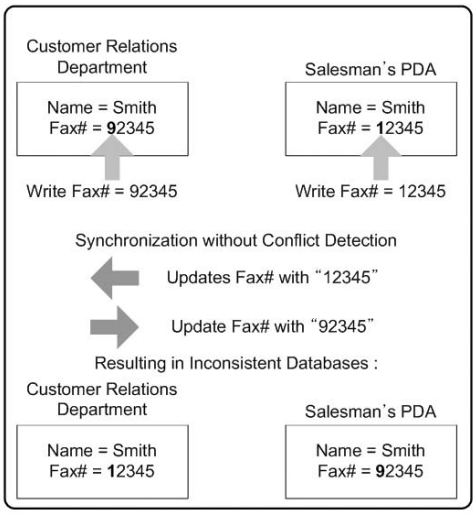
3. Conflict Resolution

3.1 데이터 충돌

둘 이상의 클라이언트가 동일한 데이터 동기화를 요청했을 때의 문제가 데이터의 충돌이다.
기존의 동기화에서는 다수의 디바이스를 고려하지 않았기 때문에 충돌이 일어날 수 없었다. 단지 동일한 데이터가 존재할 경우 덮어쓰기, 삭제 등의 기능이 제공되었다.

모바일 환경에서는 변경된 데이터만의 수정, 삭제, 업데이트 등의 기능을 수행해야 하며 잘못된 정보의 변경사항에 대하여 검출할 수 있어야 한다..

그림[2]는 서버와 클라이언트가 충돌에 대한 검출 없이 동기화를 했을 경우에 발생하는 문제점을 보여준다.



[그림 2] Synchronization without conflict detection

PDA 는 고객 팩스번호를 12345 로 수정하여 정보를 가지고 있다. 서버는 이메일 또는 전화 등으로 변경 요청을 받아 서버의 데이터를 92345 로 수정하였다. 충돌에 대한 검출이 없다면 PDA 와 서버는 각각 수정된 정보를 변경하게 되며 동기화가 이루어지지 않게 된다. 이러한 문제를 write-write conflict 라고 한다.

3.2 SyncML 에서 Conflict Resolution

충돌 해결은 두 디바이스 중에 동기화를 요청 받는 쪽을 서버로 가정한다. 일반적으로 충돌 해결은 서버의 엔진에서 이루어지지만 클라이언트에서의 충돌 해결도 가능하다. SyncML 프로토콜에서는 충돌 해결 방법에 대해서는 명시하지 않고 있으며 단지 SyncML 표현 프로토콜이 충돌 해결의 정책에 대한 코드를 제공한다.

SyncML 에서는 7 가지의 동기화 타입을 정의한다.

Two-way sync 는 클라이언트와 서버가 변경된 데이터에 대한 정보를 교환하는 일반적인 방식으로 클라이언트가 먼저 변경된 정보를 보낸다.

Slow-sync 는 Field-by-field 로 모든 아이템들이 서로 비교하여 클라이언트는 데이터를 Server 로 전송한다.

One-way sync from client only 는 클라이언트의 변경 정보를 서버로 전송하지만 서버는 변경 정보를 클라이언트로 전송하지 않는다.

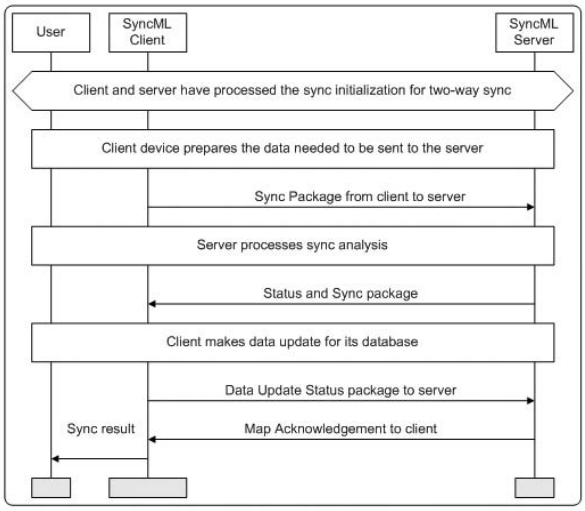
Refresh sync from client only 는 클라이언트가 모든 데이터를 서버로 전송하고 서버는 클라이언트로부터 전송 받은 데이터로 치환하는 방식이다.

One-way sync from server only 는 클라이언트가 서버로부터 모든 변경 정보를 획득하지만 자신의 변경 정보는 서버로 전송하지 않는다.

Refresh sync From server only 는 서버가 모든 데이터를 클라이언트로 전송하고 클라이언트는 서버로부터 전송 받은 데이터로 치환하는 방식이다.

Server Alerted 는 서버가 클라이언트에게 지정된 타입의 동기화를 시작하라고 통보하는 방식이다.

SyncML 의 7 가지 동기화 타입을 서버와 클라이언트로 나누어 충돌 가능성이 있는 부분을 분석하고 재분류한다. 이를 바탕으로 공통적인 충돌 해결 방안에 대한 정책을 세운다.



[그림 3] 대표적인 동기화 시나리오 예

맵 테이블은 동기화 과정에서 서버 내에 생성되고 관리된다. 최초로 동기화 요청 패키지를 보내는 쪽을 클라이언트, 최초 동기화 요구를 받는 쪽을 서버라고 부른다. 클라이언트의 데이터 식별자를 LUID 라고 부르고 서버의 데이터 식별자를 GUID 라고 부른다. 동기화 명령을 실행하면 서버 측은 맵 테이블을 통해 LUID 와 매핑되는 GUID 를 얻어 해당 GUID 의 데이터를 변경한다. 만약 LUID 와 GUID 가 동일하다면 맵 테이블이 없어도 된다.

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 Table :

GUID	LUID
1010101	11
2121212	22
3232323	33
4343434	44

[그림 4] ID Mapping of Data Items

4. CLI 를 이용한 Conflict Resolution

4.1 CLI 구성 및 운영원칙

SyncML 에서 Change Log Information 을 이용한 기존 방법의 문제점은 Table 의 특정 필드를 Change Log Information 의 정보로 이용이 가능하기 때문에 Table 의 구조변경이 필요하다는 단점이 있었다.또한 Table 의 레코드 삭제할 경우 Change Log Information 도 삭제되는 문제가 발생했다.

이러한 문제점을 개선하기 위한 방법으로 Table 과 Change Log Information 을 분리하여 구현하여 필요한 정보만을 추출하여 Change Log Information 을 작성하는 방식이 추천되기도 한다.

동기화 타입에 따라 서버 우선, 클라이언트 우선일 때와 시간에 따른 최근 데이터에 우선순위를 두는 해결정책을 둔다. 또한 추출한 정보에 가중치를 부여하여 중요도가 높은 데이터의 손실을 막는다. 다수의 모바일 디바이스간의 데이터 동기화시 일어나는 충돌에 대한 운영원칙을 기술한다.

- 서버 우선 운영원칙
- 클라이언트 우선 운영원칙
- 시간에 따른 운영원칙
- 의미적 가중치에 따른 운영원칙

디바이스의 제약사항이었던 메모리와 속도에 대한 제한이 거의 없어지면서 맵 테이블의 운영에 대한 필요성이 적어지기 때문에 ID 를 동일하게 운영하여 충돌 해결에 있어 ID 를 활용한다.

Change Log Information 의 테이블 구성에는 사용자의 ID 인 UID, 디바이스의 ID 인 DID, LUID 와 GUID 를 통합한 TID 와 변경시간을 기록할 수 있는 CHT 로 구성한다. 마지막으로 동기화를 요청한쪽에서 우선순위를 두어 다른 조건에 상관없이 데이터를 삽입 또는 변경이 가능하도록 우선순위에 대한 ID 인 PID 를 제공한다.

UID	DID	TID	DATA	
1	0001	10101	Car	
1	0002	20202	Bike	
2	0003	30303	Truck	
2	0002	40404	Shoes	

UID	ACTION	CHT	PID
1	A	20040215T121212	0
1	R	20040215T144212	0
2	D	20040216T090515	0
2	R	20040216T153010	1

[그림 5] 추출된 Change Log Information

4.2 CLI 를 이용한 Conflict Resolution

서버의 데이터가 우선일 경우 현재 존재하는 데이터의 PID 를 확인하여 보내는 데이터에 우선순위가 있다면 해당 ACTION(A(Add), R(Replace), D(Delete)…)에 따라 데이터를 변경한다. PID 는 다른 ID 보다 우선하여 자료의 우선순위를 부여하는 ID 이다. PID 가 0 이라면 변동시간(CHT)이 최종 변경된 시간보다 늦다면 서버의 데이터를 변경하지 않고 클라이언트에 해당 데이터가 이미 동기화되어 있다는 메시지를 보낸다.

두 개 이상의 클라이언트가 동일한 데이터에 대한 ACTION 을 보낸다면 서버는 PID 의 우선순위를 확인한 후에 변경시간을 비교하여 최종 변경시간의 클라이언트의 정보를 ACTION 에 따라 처리한다. 이때 동일한 시간에 동일한 데이터의 대한 접근은 DB 가 사전에 검증하여 변경이 불가능하다.

클라이언트의 데이터가 우선인 경우에 클라이언트가 데이터를 서버로 보내면 서버는 우선순위(PID)와 변경시간(CHT)을 비교하여 서버의 데이터를 모두 ACTION 에 따라 처리한다.

동기화의 정책에 따라 일정시간 데이터를 저장하여 삭제된 데이터의 변경요청이 일어날 경우에는 삭제한 클라이언트에 메시지를 보낸다. 클라이언트의 해당 UID, DID 를 확인하여 복구메시지를 보내고 해당 데이터를 변경한다.

5. 결론 및 향후 연구

동기화의 표준에 대한 7 가지의 동기화 타입과 동기화 과정에서 발생될 수 있는 충돌에 대한 해결책을 스펙을 중심으로 연구하였다. 충돌의 해결을 스펙에서는 제시하지 않고 개발자에게 일임하고 있기 때문에 충돌 해결 방식에 대한 일반화된 또는 표준화된 방법이 존재하지 않는다. 모바일 디바이스가 일반화되고 보편화되면서 데이터의 동기화 및 다수의 장치에 대한 클라이언트의 데이터 보존을 위한 표준화된 데이터 충돌의 방법이 요구된다.

본 논문에서는 데이터 동기화 과정에서 발생할 수 있는 다양한 충돌에 대해서 분석하였다. 충돌에 대하여 서버 우선, 클라이언트 우선 정책에 의해 동기화가 이루어 질 때 발생되는 예측 가능한 충돌에 대해서 좀 더 체계적으로 분류하였다. 이를 바탕으로 동기화에 대한 정보를 추적할 수 있는 Change Log Information 을 구성하고 데이터의 자료에 대한 우선순위를 부여하는 방식을 도입하였으며 데이터의 시간에 따른 비교를 하였다. 다양한 디바이스에서 발생할 수 있는 충돌 가능성에 대한 운영 원칙을 제정하여, 데이터의 안정성과 일치성을 보장하기 위한 Conflict Resolution 을 제안하였다.

향후 공개되지 않은 다양한 충돌 해결 정책들에 대한 연구가 필요하고 충돌 해결 기법에 대한 기업체, 연구소, 개인들의 연구에 대한 공개와 함께 의견수렴이 있어야 할 것이다. 폐쇄적인 동기화의 단점을 극복하기 위해 표준이 제정되었으나 세부 구현에 있어 표준에 맞는 다양한 부분의 표준화 또는 일반화된 방식이 제안되고 논의가 이루어져야 한다. 본 논문에서 제안된 방법은 기존의 정책과 충돌의 분석을 통해 일반화를 시도하고 동기화시의 충돌 해결의 한 방법을 제안하였다. 실제 다양한 테스트 환경을 통해서 통해 단점을 보완해야 할 것이다.

참고 문헌

[1] Martyn Mallick, Mobile and Wireless Design Essentials, WILEY, 2003.

[2] Uwe Hansmann, Riku Mettala, Apratim Purakayastha, and Peter Thompson, SyncML: Synchronizing and Managing Your Mobile Data, Prentice Hall, 2002.

[3] OMA, DataSync 1.1.1, http://www.openmobilealliance.org/syncml/

[4] OMA, ,DevMan1.1.1, http://www.openmobilealliance.org/syncml/

[5] Mark Birbeck, Profession XML, Wrox, 2001

[6] 이상윤, “동기화 표준 SyncML 의 표준화 동향”, 정보통신연구 진흥원, 주간 기술 동향 1031 호 2002.01.22

[7] 조진현, 최 훈, 김경용, ”SyncML 서버 데이터 동기 엔진의 설계 및 구현”, 한국정보과학회 가을학술발표논문집, 제 28 권 제 2 호, pp. 580-582, 2001.10./

[8] 하인숙, 조재혁, 양지현, “SyncML 레퍼런스 툴킷 그 내부를 보자”, 마이크로 소프트웨어 2001 년 7 월, pp.324-336, 2001.7