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

모바일 인터넷 동기화 미들웨어(MoIM-Sync) 시스템의 설계 및 구현

서영호, 이강우, 박남식, 손승범, 함호상
한국전자통신연구원 컴퓨터• 소프트웨어 연구소
전자거래연구부 이동분산처리연구팀
e-mail : { yhsuh, kwlee, nspark, ssb63113, hsham }@etri.re.kr

Design and Implementation of a Mobile Internet Middleware for Data Synchronization

Young-Ho Suh, Kang-Woo Lee, Nam-Sik Park, Seung-Bum Song and Ho-Sang Ham
Dept. of Electronic Commerce, Computer&Software Technology Lab.,
ETRI(Electronics and Telecommunications Research Institute)

요 약

근래 들어 무선 인터넷이 가능한 고성능의 휴대형 단말들이 널리 보급되어 감에 따라, 모바일 기업 응용에 대한 요구가 증가하고 있다. 모바일 기업 응용에서는 모바일 클라이언트와 기업 서버 간의 데이터 동기화가 필수적이다. 왜냐하면, 모바일 클라이언트는 그 특성상 기업 서버에 항상 접속해 있을 수 없기 때문이다. 하지만 이러한 모바일 기업 응용을 작성하기 위해서는 확장성, 이형성, 자원제약, 보안등과 같은 여러 기술적인 문제들을 해결해야만 한다. 따라서 본 논문에서는 이러한 기술적인 문제들 뿐만 아니라 데이터 동기화 부분을 처리해 줌으로써, 응용 개발자들에게 오직 데이터 동기화를 위한 추상화된 인터페이스만을 제공해주는 데이터 동기화 미들웨어 시스템인 MoIM-Sync 시스템의 설계 및 구현에 관해 기술한다. 우리 시스템은 구현 언어로 Java 를, 동기화 프로토콜로 표준 동기화 프로토콜인 SyncML 을 사용함으로써 이형성 문제를 극복하였으며, 3 계층 구조를 통해 확장성 및 기존 동기화 시스템/서버 시스템들과의 연동 문제를 해결하였다.

1. 서론

모바일 인터넷의 사용자가 급속히 증가함에 따라, 주로 개인 정보 관리(Personal Information Management : PIM)에 사용되던 Pocket PC, PDA, Sub-Notebook 등과 같은 휴대형 단말들을 기업의 IT 시스템 범주에 포함시키는 형태의 새로운 기업 응용들에 요구가 날로 증가되고 있다. 모바일 기업 응용에서는 휴대형 단말들이 무선 네트워크를 통해 기업 서버에 직접 접속하는 모바일 클라이언트로서 동작하게 된다. 이때 모바일 클라이언트와 기업 서버간의 데이터 동기화가 필수적이다. 왜냐하면, 모바일 클라이언트는 그 특성상 기업 서버에 항상 접속해 있을 수 없기 때문이다. 즉, 모바일 클라이언트는 기업 서버에 접속하여 필요한 데이터를 가져온 후에, 기업 서버와 연결을 끊고 단절된 상태에서 데이터를 갱신하게 된다. 따라서, 모바일 클라이언트가 기업 서버에 재접속 되었을 때 클라이언트 응용에 의해 모든 갱신들이 기업 서버에 전달되고, 또한 기업 서버에서 이루어진 모든 갱신들이 클라이언트로 보내져 양자간에 데이터를 동기화해야 한다.

이와 같은 모바일 기업 응용이 갖는 많은 기술적인 문제들 중에 주요한 것들은 다음과 같다[1].

- 휴대형 단말의 급속한 보급으로 인해 다수의 모바일 클라이언트들이 존재하므로 서버 구조에 있어 확장성이 요구되며, 트래픽의 폭발적인 증가에도 견뎌내야 한다.
- 다양한 단말기와, 네트워크 프로토콜, 그리고 기업 서버 시스템들이 시스템 내에 공존하게 되므로 이들간의 상호운용이 필수적이다.
- 휴대형 단말기는 메모리나, 계산 능력, 전원 배터리 등에 제약을 갖는다.

- 무선 데이터 통신은 안정적이지 못하고, 비용이 비싸다. 또한 단절 연산을 지원해야 한다.
- 무선 네트워크상에서의 기업 정보가 유출될 수 있으므로 보안이 유지되어야 한다.

이러한 다양한 문제들로 인해 응용 개발자들은 모바일 기업 응용을 작성하는 데 어려움을 겪게 된다. 따라서, 위와 같은 기술적인 문제들 뿐만 아니라 데이터 동기화 프로토콜까지 처리해 줌으로써, 응용 개발자들에게는 데이터 동기화를 위한 추상화된 인터페이스만을 제공할 수 있는 미들웨어가 요구된다.

본 논문에서는 SyncML[2]이라는 표준화된 데이터 동기화 프로토콜을 지원하는 데이터 동기화 미들웨어인 MoIM-Sync(Mobile Internet Middleware for data Synchronization)의 설계 및 구현을 기술한다. MoIM-Sync 는 확장성 및 기존의 동기화 시스템/서버 시스템과의 유연한 연동을 위해 3 계층 구조로 설계되었으며, 자바를 기반으로 구현되었다. 본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 살펴보고 , 3 장에서는 3 계층 구조에 대한 설계 근거를 논한다. 4 장에서는 MoIM-Sync 시스템의 설계 및 구현에 대해 기술하고, 5 장에 결론을 맺는다.

2. 관련 연구

Bayou[3] 프로젝트는 tuple store 들 사이의 복제와 동기화를 다루었다. 하지만, 주로 협업 응용을 목적으로 하였기 때문에 모바일 기업 응용에 적용하기엔 너무 복잡하고 비대하다. DataX[4] 시스템은 주로 클라이언트 데이터 부분 발췌, 렌더링에 중점을 둔 시스템이다. MDSS[1] 시스템은 MNCRS[5] 표준 사양을 기반으로 구현된 모바일 가상 기업 정보 접근 프레임워크이다. 이 시스템은 XML 을 이용한 상위 수준 개방 동기화 프로토콜을 도입하여 다양한 전송 프로토콜을 수용할 수 있도록 하였다. 또한 궁극적인 일관성(eventual consistency)[6]만을 보장하는 완화된 정확성 모델(relaxed correctness model)을 사용하였다.

한편 다양한 상용 데이터 동기화 시스템들이 제공되고 있다. Palm Computing 의 HotSync[7], Microsoft 의 ActiveSync[8]는 PIM 데이터의 데스크탑과의 동기화를 제품이다. Advance System 의 ASL Connect[9], AvantGo 의 AvantGo.com[10], Puma 의 Intellisync[11], Synchrologic 의 RealSync[12], Oralcle 의 Oracle Lite 8i[13], Sybase 의 SQL Anywhere[14]등은 휴대형 단말에서 기업 정보를 접근할 수 있는 동기화 기능을 제공한다. 그러나 이들 상용 제품들은 다른 제품들과 상호운용이 되지 않는다.

3. 설계 근거

이번 장에서는 MoIM-Sync 구조의 가장 큰 특징인 3 계층 구조에 대한 설계 근거에 대해 논한다. 그림 1 에서 보는 바와 같이, 본 구조의 가장 큰 특징은 3 계층 구조라는 점이다. 이는 확장성 측면이나 기존 동기화 시스템과의 연동 측면에서 볼 때, 기존의 2 계층 구조만으로는 유연성에 한계가 있기 때문이다. 따라서

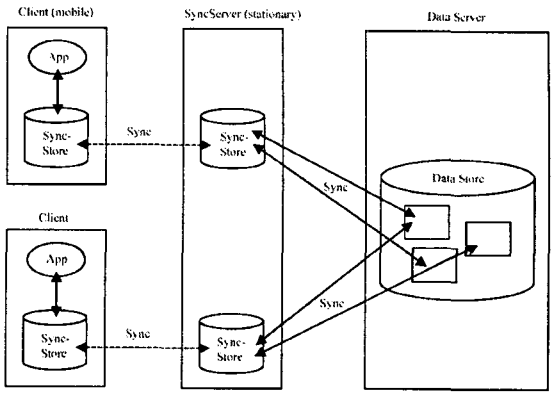

그림 1 MoIM-Sync 3 계층 구조

중간에 SyncServer 계층을 Data Server 와 분리시킴으로써 3 계층 구조가 갖는 유연성을 통해, MoIM-Sync 가 갖는 다양한 요구사항들을 유연하게 대처할 수 있도록 하였다. 구체적인 도입 배경은 다음과 같다.

3.1 확장성

클라이언트 응용 프로그램에서 사용되는 자바 객체들의 각 필드는 일반적으로 Data Server 의 DBMS 상에서 여러 테이블의 레코드들에 대한 join 연산을 통해 이루어지는 경우가 보통이다. 따라서 클라이언트와 서버 간의 데이터 동기화는 서버 측에서 복잡한 레코드의 역추출 작업을 필요로 한다. 이로 인하여 동시에 다수의 클라이언트들이 서버와 동기화를 요청할 경우 서버측에는 부하가 크게 증가하고, 결과적으로 확장성을 저해하는 병목으로 작용하게 된다. 그러므로 본 구조에서는 SyncServer 계층에 모든 클라이언트 SyncStore 에 대한 mirror store 를 두고, 필요한 레코드 역추출 작업을 수행하도록 함으로써 서버측에 집중되는 부하를 분산시키고자 하였다.

3.2 적절한 처리 시간

클라이언트가 직접 서버와 데이터 동기화를 하는 경우는 서버에 부하가 집중되어 처리시간이 크게 증가하여 무선 통신의 특성상 오류가 발생할 확률이 증가할 뿐만 아니라 비싼 경비를 지불해야 한다. 서버가 과부하 상태가 아닌 경우에도 전술한 바와 같이 서버 측에서 발생하는 복잡한 레코드 역추출 과정으로 인해, 전체 동기화 과정을 완료하는 데 소요되는 시간은 모바일 클라이언트 입장에서 볼 때 문제가 될 수도 있다. 따라서 본 구조에서는 모바일 클라이언트와 mirror store 간의 단순 중복 형태의 동기화를 제공함으로써, 응용에 따라서는 다양한 동기화 방식에 의해 매우 효율적인 처리 시간을 보장하고자 하였다.

3.3 기존 동기화 시스템과의 연동

모바일 기업 응용은 보통 다양한 모바일 클라이언트 플랫폼, 기업 서버 플랫폼 그리고 데이터 전송/동기화 프로토콜들로 구성된다. 따라서, 본 구조에서는 중간 계층인 SyncServer 계층에 클라이언트 어댑터,

서버 어댑터를 각각 두어 일종의 교량(bridge)역할을 함으로써 기존의 동기화 시스템과 유연하게 연동될 수 있도록 하고자 하였다.

4. 시스템 설계 및 구현

이번 장에서는 MoIM-Sync 시스템의 설계 및 구현에 대해 논한다. 3 장에서 살펴본 바와 같이 본 구조는 3 계층 구조를 갖는다.

4.1 가정

- 클라이언트들간의 동기화는 지원하지 않는다.
- 클라이언트와 서버와의 동기화 시 동기화 요청은 클라이언트만이 할 수 있다.
- 모든 클라이언트 측 SyncStore 에 대해 각각에 상응하는 mirror store 가 SyncServer 계층에 하나씩 존재한다.
- SyncServer 계층과 DataServer 계층간의 동기화는 어느 한 쪽의 요청에 의해 발생하지 않고, 주기적으로 발생한다.
- Client 계층과 SyncServer 계층간의 통신은 제한을 두지 않으나, SyncServer 계층과 DataServer 계층간의 통신은 유선을 전제로 한다

4.2 SyncStore 설계

그림 2 는 MoIM-Sync 3 계층 구조에서 SyncStore 부분을 확대한 것이다. 응용프로그램은 SyncStore 가 제공하는 API 를 통해 Reconcilable 객체들을 접근하고 변경할 수 있다. StoreManager 는 이들 객체들에 대한 영속성(persistency)과 입출력을 제공하고, Update Monitor 는 객체들에 대한 update history 를 유지· 관리한다. 응용프로그램에서 SyncStore API 를 통해 동기화 요청을 하게 되면, Synchronizer 는 이 요청을 받아서 Update Monitor 로부터 새롭게 발생한 변경 리스트들을 얻어온 다음 SyncML 프로토콜을 통해 원격 Synchronizer 와 동기화를 수행한다. 주요 컴포넌트들에 대한 사항은 다음과 같다.

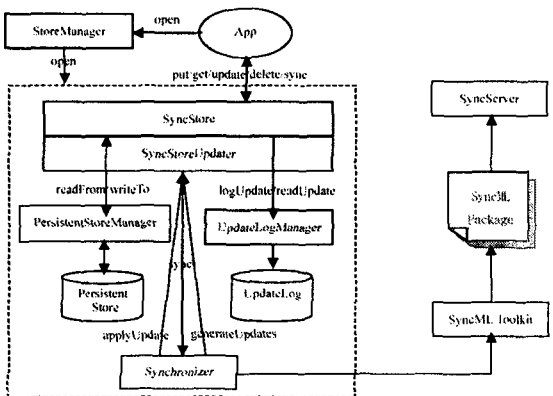

그림 2 SyncStore 내부 구조

4.2.1 Replicable 인터페이스

SyncStore 에 저장될 객체들은 응용 프로그램의 일부분인 클래스에 속한다. 예를 들어 개인 정보 관리 응용에서 명함, 주소록, 달력 등에 해당하는 클래스들이다. 응용 프로그래머는 이와 같은 클래스들로 하여금 Reconcilable 인터페이스를 구현하도록 선언한다. 이렇게 함으로써 각 클래스의 객체들이 SyncStore API 에 정의된 함수에 의해 SyncStore 에 저장· 관리 될 수 있게 된다.

4.2.2 SyncStore Manager

StoreManager 는 지역 장치(local device)에 대한 SyncStore 관리(administration)를 위한 클래스 이다. 이 클래스는 지역 장치에 존재하는 SyncStore 들에 대한 정보와 그들의 실제 영속 저장소(persistent storage)의 위치를 유지 관리한다.

4.2.3 SyncStore API

SyncStore 는 Reconcilable 객체들을 저장하기 위한 경량의 중복 가능한 영속 저장소 (lightweight replicable persistent store) 이다. 클라이언트-서버 응용에서 SyncStore 는 클라이언트가 단절되었을 때 변경된 응용 정보들을 접근하고 갱신할 수 있는 신뢰성 있고 효과적인 메커니즘을 제공함으로써 단절 연산(disconnected operation)기능을 갖는다. SyncStore 인터페이스는 응용프로그램에서 SyncStore 를 접근할 수 있도록 하기위한 API 를 제공한다.

4.2.4 SyncStoreUpdater

SyncStore 는 자신에게 저장되어 있는 객체들의 변경을 중복된 SyncStore 에 전달하기 위해, 객체들의 상태 변경 정보를 관리· 유지한다. Synchronizer 는 SyncStore 로부터 객체 상태 변경 정보를 얻어와서 원격 Synchronizer 에서 사용되는 전송/동기화 프로토콜에 따라 중복된 SyncStore 에게 전달한다. 중복된 SyncStore 에서는 자신에게 저장되어 있는 Synchronizable 객체들에 대한 객체 상태 변경 정보와 전달 받은 객체 상태 변경 정보를 비교하여 적절히 동기화 한다. SyncStoreUdpater 인터페이스는 이와 같이 동기화 과정에서 Synchronizer 가 SyncStore 에 접근하여, 객체 상태 변경 정보들을 얻어내고 원격 Synchronizer 로부터 전달되어 온 원격 객체 상태 변경 정보를 local SyncStore 에 반영한다.

4.2.5 PersistentStoreManager

SyncStore 를 특정 영속 저장소 구현 메커니즘으로부터 분리 시킴으로써 SyncStore 가 임의의 장치에서 사용될 수 있도록 하기 위해서 PersistentStoreManager 인터페이스를 정의하였다. 즉, 영속 저장소로서 특정 file system 이나, DBMS, 혹은 nonvolatile memory 등이 사용된다 하더라도 PersistentStoreManager 인터페이스를 구현한 것이면 SyncStore 구현 클래스에서는 장치 독립적으로 영속 저장소를 접근할 수 있게 된다.

4.2.6 UpdateLogManager

Reconcilable 객체들의 가능한 상태 변경은 삽입, 갱신 그리고 삭제가 있다. 이와 같은 객체들의 상태 변경은 다수의 중복된 SyncStore 에서 병렬적으로 발생할 수 있으며, 다수의 중복된 SyncStore 에 포함된 특정한 두 SyncStore 들의 동기화는 임의의 형태로 이루어 질 수 있다. 이로 인해, 서로 다른 중복된 SyncStore 에 존재하는 동일한 객체들은 서로 다른 update history 를 가질 수도 있다. UdateLogManager 를 통해 SyncStore 에 존재하는 모든 Reconcilable 객체들에 대한 update history 를 로깅한다

4.2.7 Synchronizer

두 개의 Synchronizer 는 서로 협력하여 해당 호스트에 존재하는 SyncStore 에서 발생된 객체 상태 변경 정보를 교환시켜 줌으로써, 두 SyncStore 들이 서로 동기화 할 수 있도록 한다. 구체적인 동기화 프로토콜은 오직 협력하는 Synchronizer 에게만 알려지고 공유되며, 사용자에게는 투명해야 한다.

4.2.8 SyncML

SyncML 은 IBM 을 비롯한 9 개의 동기화 관련 회사들이 주축으로 제안한 데이터 동기화 표준으로, 현재 수백개의 회사들이 SyncML 표준을 지지하거나 지원 제품을 개발 중에 있다. SyncML 의 목표는 업계 전반에서 사용할 수 있는 단일 데이터 동기화 방식을 제안하고, 이를 바탕으로 다양한 모바일 또는 일반 장치들이 서로 자신의 데이터를 동기화 할 수 있는 환경을 만드는데 있다. SyncML 표준은 다양한 장치들 사이의 상호 연동을 원활하게 하기 위해, 동기 작업 중 장치 사이에 전달되는 데이터의 형식을 표준화하였고, 형식은 텍스트 형식의 XML 을 기반으로 한다.

4.3 구현 및 시연 결과

MoIM-Sync 는 JDK 1.3 을 기반으로 크게 idplab.moim. datasync, idplab.moim.syncstore, idplab.webstore 이렇게 세 개의 패키지로 구성되었으며, 각각의 패키지는 하위 패키지를 포함하고 있다. 이 가운데 idplab.moim .datasync 패키지만이 응용 개발자들이 접근할 수 있는 공용 패키지 이다. 그림 3 은 MoIM-Sync 시스템의 시연 환경을 나타낸 그림이다. MoIM-Sync 의 어댑터 기반 3 계층 구조를 통해 SyncStore API 로 구현된 MoIM-Sync 클라이언트가 아닌 두 개의 MS outlook 모바일 클라이언트가 각각 MoIM-Sync 기반의 동기화 프로토콜, ActiveSync 를 사용하여 MS Exchange 서버를 통해 동기화 되고 있다. 이는 본 시스템의 구현 목표인 기존 동기화 시스템/서버 시스템들과의 연동 과정을 보여준다.

5. 결론

본 논문에서는 응용 개발자들에게 오직 데이터 동기화를 위한 추상화된 인터페이스만을 제공해주는 데이터 동기화 미들웨어 시스템인 MoIM-Sync 시스템의 설계 및 구현에 관해 기술하였다. 구현 언어로 Java 를, 동기화 프로토콜로 표준 동기화 프로토콜인 SyncML 을 사용함으로써 이형성 문제를 극복하였으며, 3 계층 구조를 통해 확장성 및 기존 동기화 시스템/서버 시스템들과의 연동 문제를 해결하였다

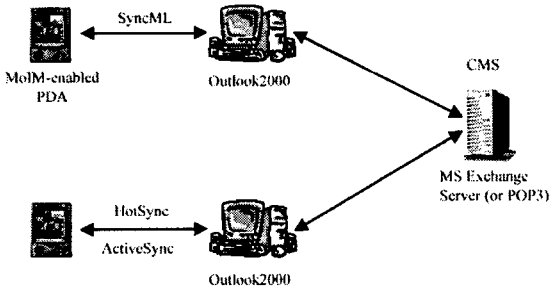

그림 3 시연 환경

참고문헌

[1] Butrico et al. "Enterprise Data Access from Mobile Computers: An end-to-end story", IEEE Data Engineering, RIDE workshop, 2000

[2] "Building an Industry-Wide Mobile Data Synchronization Protocol", SyncML White Paper, Version 1.0

[3] D. M. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. "Managing update conflicts in bayou, a weakly connected replicated storage system" In Proc. Of 5th ACM Symposium on Operating System Principles, pp. 172-182, Dec 1995

[4] H. Lei, K. Lee, M. Blount, and C. Tait. "Enabling ubiquitous database access with XML", In Proc. Of 1st International Conference on Mobile Data Access, Dec. 1999

[5] Montenegro, G. "MNCRS: Industry Specification for the Mobile NC" IEEE Internet Computing 2, No.1, pp. 73-77, Jan. 1998

[6] S. Davidson, H. Garcia-Molina, and D. Skeen. "Consistency in a partitioned network: A survey." ACM Computing Surveys, 17(3): 341-370, Sep. 1985

[7] Palm Computing. "Development Documentation." http://www.palm.com/devzone/docs.html, 1999

[8] J. Murray. "Inside Microsoft Windows CE", Microsoft Press, 1998

[9] Advance Systems. "Connecting Mobile Workers to Enterprise Information Systems", http://www.asl.com, 1999

[10] AvantGo Gorporation. "AvantGo Developer Guide", http://www.avantgo.com/DevCorner/DevGuide, 1999

[11] Puma Technology, "Puma Technology Developer Zone", http://www.pumatech.com/developer, 1999

[12] Synchrologic Corporation, "Solving the Mobile Computing Challenge: Data, Documents, and Software Distribution to Mobile Users", http://www.synchrologic.com/images/whitepapers/mobile_computing_whitepaper.html, 1999

[13] Oracle Corporation, "Oracle8iLite", http://www.oracle.com/mobile/o8ilite/index.html, 1999

[14] Sybase Corporation, "Sybase Mobile and Embedded Computing", http://www.sybase.com/mec, 1999