신기술 해설

SynML

이 재 영*

● 목 차 ●

1. 서 론

현재 사용자들은 하루가 멀다하고 출시되는 다양한 형태의 휴대 단말의 홍수 속에 살고 있다. 이미 휴대폰은 필수품이 되어버렸고 다양한 형태의 PDA들도 점차 그 보급이 늘어가고 있는 추세이다. 모바일 컴퓨팅과 다양한 통신 기기들이 인기를 끌고 있는 배경은 이 들이 사용자로 하여금 언제 어디서나 필요로 하는 정보를 얻을 수 있게 해주기 때문이다. 특히 고속의 무선 데이터 통신이 가능해짐에 따라 사용자들은 시간과 공간의 제약에서 벗어나 자유롭게 자신의 정보를 접근, 수정, 저장, 전송할 수 있게 되었고 손에 들고 있는 기기로부터 정보와 응용에 대한 편재형 접근과 이 정보에 대한 즉각적인 접근과 업데이트를 할 수 있게 된 것이다.

그러나 대부분의 사용자에게 휴대 단말은 단지 일시적인 데이터 입출력 및 조작의 기기일 뿐이며 실질적인 데이터의 저장소는 유무선 네트워크에 접속되어있는 PC와 같은 기기이다. 따라서 휴대 단말 사용자는 정기적으로 데이터 저장소와의 데이터 동기화(data synchronization)를 통하여 데이터의 일치성을 확보하여야 한다. 사용자들이 동기화하기 원하는 데이터는 개인 PC를 저장소로 하는 개인적인 것 뿐만 아니라 이메일, 그룹웨어가 제시하는 다양한 형태의 공동 작업 데이터 등을 망라한 네트워크화된 데이터들이다. 휴대 단말을 통한 응용과 정보를 사용할 수 있는 것과 사무실 또는 네트워크와의 응용 및 정보의 업데이트를 동기화 할 수 있는 것이 pervasive하며 단절된 컴퓨팅의 인기도와 사용도의 핵심이다.

하지만 오늘날 우리는 매우 다양한 형태의 휴대 컴퓨팅을 즐길 수 있음에도 불구하고 다음과 같은 이질적인 기대를 동시에 만족할 수 없다.

- ㅇ 어떠한 이동 기기와도 동기화 할 수 있는 네트워크화된 데이터
- ㅇ 어떠한 네트워크화된 데이터와도 동기화할 수 있는 이동 단말

도리어 세상에는 독자적이며 폐쇄적인 데이터 동기화 프로토콜들이 난무하고 있다. 개개의 이러한 프로토콜들은 한정된 전송로와 한정된 기기에 국한되어있으며 극히 일부의 네트워크화된 데이터에만 접근할 수 있다. 단일화된 동기화 표준의 부재는 사용자와 기기 제조업체, 응용 개발자 및 서비스 제공자에게 많은 문제를 제기하고 있다.

* (주) 모빅스 대표이사

SyncML은 단일, 공통 데이터 동기화 프로토콜을 개발하고 보급하여 업계 전체가 사용할 수 있도록 해주는 새로운 industry initiative이다. SyncML initiative는 에릭스, IBM, 로터스, 모토롤러, 노키아, 팜, 사이온, 스타피쉬소프트웨어 등의 회사가 주도하고 있으며 수백개의 업체가 참여하고 있다. 국내의 3RSoft, 삼성 등도 클라이언트와 서버를 개발하여 적합성 및 호환성 테스트를 통과하였다.

2. 휴대 단말과 데이터 동기화

2.1 데이터 동기화(Data Synchronization)

근본적으로 모바일 사용자는 항상 네트워크 및 네트워크 상에 저장된 데이터와 접속되어 있지 않다. 사용자들은 데이터를 네트워크에서 추출하여 자신의 이동 단말에 저장하고 이에 접근하여 조작하며 주기적으로 다시 네트워크에 접속하여 변화된 로컬 데이터를 네트워크 상의 저장소에 저장하게 된다. 이 때 사용자들은 네트워크화된 데이터에 가해진 변화에 대해서도 알게 된다. 때때로 네트워크화된 데이터에 대한 여러 업데이트간의 충돌을 해소하여야 한다. 이렇듯 업데이트를 교환하고 충돌을 해소하는 결합 작업을 데이터 동기화라고 한다. 즉 데이터 동기화는 두 개의 데이터 집합을 동일하게 보이도록 하는 행위이다[1]. 이동 단말의 경우 동기화는 이동 단말에 국지적으로 저장된 데이터에 적용된다. SyncML은 데이터에 제한을 두지 않으며 이는 어떠한 형태의 데이터도 수용할 수 있다는 것으로 범용의 데이터 동기화 프로토콜이 가져야 할 기본적인 기능이다.

임의의 데이터 집합 α와 β를 데이터 집합 α', β'으로 변환시키는 변환 조작을 각각 ΩA, ΩB라고 할 때 데이터 동기화를 다음과 같이 데이터 집합과 변환 조작을 원소로 갖는 집합으로 정의할 수 있다.

$f \equiv \{\alpha,\ \beta,\ \Omega A,\ \Omega B\}$ *where* $\Omega A\,\alpha = \Omega B\,\beta$

데이터 동기화에는 크게 다음과 같은 두 개의 요소 기술이 있다.

- ㅇ변환된 데이터의 교환
- ㅇ데이터 불일치의 해결

2.2 동기화 프로토콜(Synchronization Prototol)

데이터 동기화 프로토콜은 이동 단말이 네트워크에 접속하여 구성한 데이터 동기화 세션 중의 통신을 위한 워크플로우를 정의한다. 이는 반드시 기록들의 명명(naming)과 인식(identification)을 지원하여야 하고 로컬 데이터와 네트워크 데이터를 동기화 하기 위한 공통된 프로토콜 명령어를 지원하여야 한다. 동기화 충돌(synchronization conflict)을 인식하고 해결할 수 있도록 하여야 한다.

데이터 동기화 프로토콜(Data Synchronization Protocol)은 데이터 동기화 세션을 위한 통신 방법이며 이에는 다음과 같은 요소가 있다.

- 기록들의 명명(naming)과 인식(identification)
- 공통의 프로토콜 명령
- 동기화 충돌의 identification과 해소(resolution)

이러한 목적을 달성하기 위해 동기화 프로토콜은 다음과 같은 특성을 갖추어야 한다.

- 유선 및 무선 네트워크 구간에서도 효과적으로 동작할 것
- 다양한 전송 프로토콜을 지원할 것
- 임의의 네트워크화된 데이트를 지원할 것
- 다양한 응용으로부터의 데이터 접근을 가능하게 할 것
- 휴대 단말의 자원 부족에 따른 제약점을 해소할 것
- 기존의 인터넷과 웹 기술들을 기반으로 구현될 것
- 모든 종류의 단말을 수용하는 가장 공통적으로 요구되는 동기화 기능을 기본 기능만으로 지원

할 것

이 중 무선 네트워크의 지원은 프로토콜이 해결하여야 할 대부분의 문제들을 지니고 있으며 이는 다음과 같다.

- 높은 네트워크 대기 시간
  네트워크 대기 시간은 데이터를 일시적으로 저장, 분석, 전송하면서 발생한다. 높은 네트워크 대기 시간을 가지고 있는 무선 네트워크는 robust한 전송 프로토콜을 필요로 한다.
- 제한된 대역폭
  제한된 대역폭과 무선 단말의 낮은 처리 능력을 보완하기 위하여 WBXML과 같이 데이터와 명령어들의 이진 인코딩 기법들을 필요로 한다.
- 상대적으로 높은 패킷 비용
  단말과 네트워크화된 데이터 사이의 요청-응답 상호작용의 수를 최소화하여야 한다. 가장 효율적인 프로토콜은 단일 요청-응답 쌍으로 메시지를 송수신하여야 할 것이다. 이러한 모델이라면 휴대 단말로부터의 요청은 국지적 데이터에 대한 모든 업데이트를 포함할 것이다. 이에 대한 요청은 업데이트와 업데이트간의 모든 충돌이 확인되고 아마도 해결되어 있어야 할 것이다.
- 데이터와 연결성의 낮은 신뢰도
  네트워크와의 간헐적인 접속 기능을 지원하려면 공통 데이터 동기화 프로토콜은 일련의 동기화 작업 중에 발생하는 접속 중단에 대처할 수 있어야 한다. 접속 중단이 발생하더라도 프로토콜은 단말과 네트워크화된 데이터 저장소가 일정하게 유지되어야 하며 접속이 다시 이루어지면 작업이 계속되어져야 한다.

2.3 다양한 전송 프로토콜과 매체 지원

유무선 네트워크는 다양한 프로토콜과 매체를 사용하며 동기화 프로토콜은 다음과 같은 전송 프로토콜 및 매체 위에서 유기적으로 작동하여야 한다.

- HTTP
- WSP (Wireless Session Protocol)
- OBEX (Bluetooth, IrDA, 또는 다른 다양한 국지적 연결)

이 들외에도 동기화 프로토콜은 다음의 통신 프로토콜 상에서도 동작하여야 한다.

- SMTP, POP3, IMAP
- 순수 TCP/IP 네트워크
- 폐쇄적인 무선 통신 프로토콜

효과적인 동기화 프로토콜은 위의 전송 프로토콜 상에서 구현될 수 없는 기능을 바탕으로 구현되어서는 안되며 전송 프로토콜이 제공하는 기능들과 중첩되는 것을 최소화하여야한다.

신뢰성 있는 요청-응답 모델이 위의 모든 전송 프로토콜에 적용될 수 있으며 효율적인 동기화 프로토콜을 이러한 모델을 바탕으로 구현할 수 있다. 또한 프로토콜 단위를 위한 MIME(Multi-Purpose Internet Mail Extensions) 타입을 정의하는 것은 위의 전송 프로토콜 상의 변환을 허용하게 된다. 전송 프로토콜의 선택적인 개량은 보안과 압축 기능을 가져야 한다.

2.4 Client/Server

동기화 작업에 참여하는 개체들은 클라이언트와 서버의 역할을 각각 수행하여야 한다. 클라이언트와 서버는 각각의 변환 로그를 유지하며 동기화를 실행할 때 변환 로그를 바탕으로 어떠한 데이터가 변하였는지를 결정한다. 변환 로그는 "치환(replace)", "추가(addition)", "삭제(deletion)"와 같은 조작을 반드시 기록하여야 한다. 데이터 저장소의 각각의 데이터는 유일한 식별자를 갖으며 클라

이언트와 서버 각각은 내부 사용을 위해서 각자의 사적 식별자를 사용할 수 있다. 다만 서버는 이러한 사적 식별자간의 매핑을 위한 매핑 테이블을 관리할 책임을 갖고 있다.

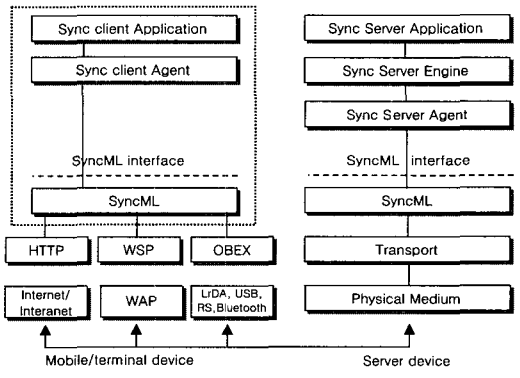
3. SyncML

3.1 개요

SyncML은 Synchronization Markup Language의 약자로써 네트워크나 기기의 종류에 상관없이 데이터들을 동기화 하고자하는 공통의 언어로써 IBM, 로터스, 모토로라 등의 업체가 주도하고 있는 Industry Initiative이다. SyncML은 설계 단계에서부터 네트워크에 간헐적으로 접속하는 휴대 단말과 네트워크에 상시 접속되어있는 네트워크 서비스간에 사용되는 것을 고려하였다. 또한 SyncML은 peer-to-peer 데이터 동기화를 위해서도 사용될 수 있다. 특히 SyncML은 네트워크 서비스와 기기가 동기화 하고자 하는 데이터를 다른 포맷으로 저장하거나 다른 소프트웨어 시스템을 사용하는 상황도 고려하여 설계되어졌다.

SyncML은 "모든 종류의 데이터의 동기화", "다양한 전송 프로토콜과의 연동", "보안", "상호연동성"을 목표로 하며 데이터 Representation 프로토콜과 동기화 프로토콜, 그리고 트랜스포트 바인딩의 구성요소로 되어있다(그림 1).

- XML 기반 데이터 Representation 프로토콜
  데이터 동기화를 수행하기 위해 필요한 모든 데이터, 메타 데이터 및 명령어 등의 표현을 위한 XML DTD를 규정한다.
- 동기화 프로토콜
  데이터 Representation 프로토콜은 데이터 동기화 작업에 참여하는 개체간에 교환하는 잘 정의된 메시지들의 집합으로 정의되어진다. 메시지들은 텍스트 문서 마크업의 업계 표준인 XML 문서로 나타내지며 이는 데이터, 데이터 추가, 삭제, 변경 명령문 및 기타 상황 정보 등을 담고있다. SyncML representation 프로토콜은 MIME content type으로도 표현될 수 있다. SyncML representation 프로토콜은 요청/반응 명령어 체계에 기반한 데이터 동기화 모델뿐 아니라 무조건적인 푸시 명령어 체계에 기반한 데이터 동기화 모델도 지원한다. 또한 SyncML representation 프로토콜은 SyncML 패키지라는 개념도 가지고 있다. SyncML 패키지는 묶여진 데이터 동기화 조작들을 수행한다. 이러한 개념적인 데이터 동기화 패키지는 단일 SyncML 메시지에 담겨지거나 다수의 SyncML 메시지에 담겨진 다수의 데이터 동기화 조작의 배치 조작을 허용한다.
- 동기화 프로토콜을 위한 트랜스포트 바인딩
  SyncML Transport binding으로 현재 HTTP, WSP와 OBEX가 규정되어 있으나 원래의 SyncML 데이터 Representation 프로토콜과 동기화 프로토콜이 기저의Transport 형식과 독립적이므로 SyncML 메시지는 향후 다른 전송 프로토콜과의 Binding이 가능하다.



(그림 1) SyncML 프로토콜 Architecture

SyncML은 다음과 같은 동기화 방식을 지원한다.

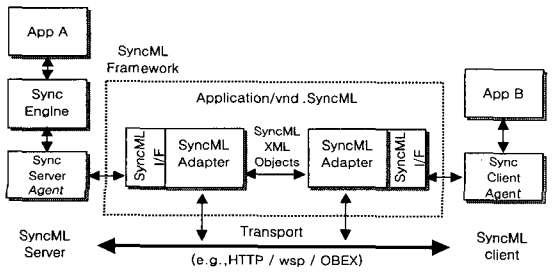
- 양방향 동기화
  클라이언트와 서버간의 데이터 불일치가 발생

한 경우

- 서버 또는 클라이언트로부터의 단방향 동기화
  클라이언트 또는 서버가 국지적인 데이터의 업데이트를 단방향으로 상대에게 보내는 경우
- 서버의 알람에 의한 동기화
  서버가 동기화 세션을 시작하는 경우로써 모든 경우에 적용된다.

3.2 SyncML Framework

(그림 2)는 SycnML 프레임워크를 보여주고 있다. 응용 A는 네트워크 서비스로써 네트워크 기기인 응용 B와 데이터 동기화를 하고자 한다. 서비스와 기기는 HTTP와 같은 동일한 네트워크 트랜스포트로 연결되어 있으며 응용 A는 데이터 동기화 프로토콜 구현인 "Sync Engine" 프로세스를 사용한다. 이 데이터 동기화 프로토콜은 클라이언트 응용이 "Sync Server" 네트워크 자원을 사용되도록 구현되어 진다. "Sync Server Agent"는 "Sync Engine"의 네트워크 접근을 제어하고 클라이언트 응용으로부터의 또는 응용으로의 통신을 담당한다. "Sync Server Agent"는 "SyncML I/F"라는 인터페이스의 함수들을 사용하여 이러한 기능들을 수행한다. "SyncML I/F"는 "SyncML Adapter"의 API이다. "SyncML Adapter"는 SyncML 포맷의 객체들을 주고 받기 위하여 송신자와 수신자가 사용하는 논리적 프로세스이며 또한 네트워크 전송계층과 인터페이스 하는 프레임워크의 entity로써 응용 A와 응용 B간의 네트워크 연결을 생성하고 유지한다. 응용 B는 "Sync Client Agent"를 사용하여 네트워크를 접근하고 "SyncML I/F"를 통하여 "SyncML Adapter"를 사용한다. 실제적인 서버와 클라이언트는 이러한 개별적인 구성요소로 구현될 필요는 없으며 이러한 프레임워크는 공통 데이터 프로토콜을 구현하기 위해 필요한 개념적인 것이다.



(그림 2) SyncML Framework

3.3 SyncML Package와 Message

SyncML에서는 데이터 동기화 조작들이 개념적인 "SyncML Package"라는 단위로 묶여져 있다. SyncML Package는 하나 또는 그 이상의 데이터 동기화 semantics의 집합을 전달하기 위하여 필요한 "SyncML Message"의 개념적인 프레임이다. SyncML 메시지는 잘 짜여진 (well-formed) XML 문서이어야 하지만 적법한 (valid) XML일 필요는 없다. 이는 <SyncML>이라는 문서 개체 타입으로 인식된다. 이 요소개체가 SyncML 메시지의 부모 컨테이너역할을 수행한다.

SyncML 메시지는 <SyncHdr> 요소개체 타입으로 정의되는 헤더부와 <SyncBody> 요소개체 타입으로 정의되는 본문부로 구성되어 진다. SyncML 헤더는 해당 SyncML 메시지의 라우팅과 버전 정보를 포함하며 본문부는 하나 또는 그 이상의 SyncML 명령어들을 포함한다. SyncML 명령어들은 개별적인 요소개체 타입으로 정의되며 각각의 SyncML 명령어들은 해당 명령어를 기술하는 동기화할 데이터와 메타 정보와 같은 다른 요소개체 타입들의 컨테이너이다.

3.4 SyncML 명령어

SyncML은 다음과 같은 "요청" 명령어와

- Add, Alert, Atomic, Copy, Delete, Exec, Get, Map, Put, Replace, Search, Sequence, Sync

"응답" 명령어들을 정의한다

- Status, Results

명령어에 관한 자세한 설명은 프로토콜 정의를 참조하기 바란다[2].

SyncML 명령어들은 SyncML 조작들의 semantics를 완벽하게 정의하지 않는다. 예를 들어 어떤 문서를 응용의 데이터베이스에 "Add"하는 조작은 트랜잭션 요청을 큐에 "Add"하는 조작과 완전히 다른 의미를 갖는다. SyncML 명령어의 의미는 동기화 되는 데이터의 타입에 따라 결정된다. 이는 송신자가 수신자에게 전혀 의미 없는 조작을 요청할 수 있다는 것이며 이때 수신자는 송신자에게 반드시 에러 코드를 전송하여야 한다.

데이터 집합 α가 데이터 집합 β와 성공적으로 동기화 하였다는 것을 다음과 같이 표기하자.

$$\alpha \Rightarrow \beta$$

SyncML에서 $(\alpha \Rightarrow \beta)$가 참일 경우는 아래의 필요 충분 조건이 만족되었을 경우이다.

모든 식별자(identifier) ι에 대하여 다음을 만족하는 일대일대응 사상 (bijective mapping) μ가 존재한다.

- $\alpha(\iota)$가 존재하면 $\beta(\mu(\iota))$가 존재하고 $\beta(\mu(\iota))$가 $\alpha(\iota)$와 일치한다.
- $\alpha(\iota)$가 존재하지 않으면 $\beta(\mu(\iota))$가 존재하지 않는다.

이때 $(\alpha \Rightarrow \beta)$가 참이라고 해서 $(\beta \Rightarrow \alpha)$가 반드시 참이지는 않다. 즉, SyncML을 사용한 데이터 동기화는 대칭적이지 않다.

4. 임베디드 시스템, 무선인터넷 그리고 SyncML

4.1 임베디드 시스템과 SyncML

SyncML 문서는 데이터의 추가, 삭제, 변경 명령문과 같은 데이터 동기화 명령어들을 포함하며 저장 용량과 처리 능력이 일정하지 않은 단말들을 대상으로 함으로 최소한의 자원 사용을 목표로 한다. 이를 위하여 단축 naming convention을 사용한다. 예를 들어 "protocol version"을 <VerProto>라고 한다.

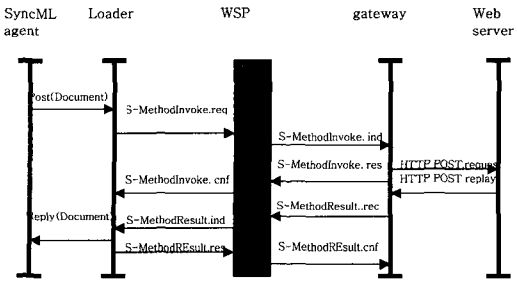
SyncML은 클라이언트와 서버간의 통신 시 capability exchange를 할 수 있도록 규정하고 있다. Capability exchange는 SyncML 서버와 클라이언트

```
<Meta>
<Format xmlns=' syncml:metinf' >xml</Format>
<Type xmlns=' syncml:metinf' >application/vnd.syncml-devinf+xml</Type>
<Data>
<DevInf xmlns=' syncml:devinf' >
<Man xmlns='syncml:devinf'>IBM</Man>
<Mod xmlns='syncml:devinf'>WorkPad<Mod>
<DevTyp xmlns='syncml:devinf'>pda</DevTyp>
<DevID xmlns='syncml:devinf'>J. Smith</DevID>
<FwV xmlns='syncml:devinf'>PalmOSv3.0</FwV>
<OEM xmlns='syncml:devinf'>Palm, Inc.</OEM>
</DevInf>
</Data>
</Meta>
```

가 각각 지원하는 기기, 사용자, 응용 등을 결정할 수 있는 능력이다. SyncML 서버의 입장에서는 SyncML 클라이언트로부터 기기 정보, 사용자 정보, 응용 정보 문서를 "Get" 명령어를 이용하여 얻을 수 있고 SyncML 클라이언트는 SyncML 서버로부터 동일한 방법으로 정보 문서를 얻을 수 있다. 이 문서들은 잘 알려진(well-defined) 기능들에 대한 지원에 관련된 정보를 담고 있다. 또한 SyncML 클라이언트는 "Put" 명령어를 사용하여 capability exchange 정보를 SyncML 서버로 푸시 할 수 있다. SyncML 표준은 capability exchange에 사용될 기기 정보를 담는 Device Information DTD를 정의하고 있다.[devinf.dtd]

4.2 WAP과 SyncML

SyncML은 WSP(Wireless Session Protocol)[4]와의 바인딩을 제공함으로써 WAP을 통한 데이터 동기화를 가능하게 해준다. WSP는 무선인터넷상에서 HTTP 1.1의 기능들을 제공하며 장기(long-lived) 세션, 데이터 푸시, capability negotiation, 서스펜드/리쥼과 같은 추가적인 기능들을 제공한다. WSP 관련 표준들은 비교적 긴 대기 시간과 낮은 대역폭을 지닌 bearer 네트워크에 최적화 되어있다. SyncML과 WSP와의 바인딩은 로더(Loader)라는 논리 레이어를 통해 이루어진다. 아래 그림은 SyncML 에이전트가 웹서버로부터 POST 요청을 통하여 문서를 수신하는 순서를 보여주고 있다.

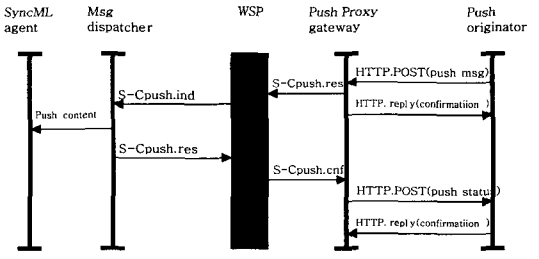


(그림 3) HTTP POST를 이용한 웹 서버 접근

HTTP와 대별되는 WAP의 특징적인 기능이 Push 기능이다. 서버로부터 클라이언트로의 데이터 전송은 HTTP에 정의된 기능만을 사용하도록 되어있다. 이 기능은 SyncML 동기화 프로토콜에 정의된 서버 Alerted 동기화의 경우에 사용될 수 있다. 푸시의 개시자(initiator)는 HTTP의 POST를 사용하여 XML 문서를 Push Proxy Gateway(PPG)로 보낸다. 푸시 메시지는 언제 푸시 메시지가 폐기되는지 등의 제어 정보를 담은 제어 entity와 클라이언트로 보내져야하는 내용부분으로 나뉘어진다.

푸시 모델은 확인모드(confirmed)와 비확인(non-confirmed)모드의 두 가지 모드를 정의한다. 확인모드를 사용하기 위해서는 능동적인 WSP 세션이 필요하며 만약 세션이 존재하지 않으면 PPG는 세션 생성 요청을 클라이언트 내의 "Msg dispatcher"라는 논리적인 응용으로 보내게 된다. 이 응용이 세션을 생성하게 되고 푸시가 일어난다. 비확인 푸시 메시지는 세션을 필요로 하지 않는다.

(그림 4)에서 볼 수 있듯이 Msg dispatcher가 푸시 메시지를 수신하면 이 응용은 메시지의 응용 인식자를 참조하여 이를 적절한 에이전트에게 보내게 되며 에이전트는 메시지의 내용에 따라 서버로부터 추가적인 정보를 수신할지를 결정한다.



(그림 4) Push 시나리오

5. 결 론

무선인터넷과 PDA와 같은 휴대정보단말의 급속한 보급은 사용자로 하여금 언제 어디에서나 원하

는 정보에 접근할 수 있도록 해준다. 또한 사용자의 정보는 단일한 시스템에만 존재하지 않고 네트워크 상에 분산, 산재한다. 그러나 네트워크에 상시 접속되어 있는 시스템과는 달리 PDA와 같은 휴대 단말은 무선인터넷의 고비용으로 인하여 간헐적으로 네트워크상의 정보에 접근하게 된다. 따라서 사용자가 항상 휴대하는 휴대단말의 국지적인 정보와 네트워크 상에 존재하는 정보사이에는 차이점이 발생하게 된다. 또한 하나 이상의 휴대 단말 (셀룰러폰, PDA)을 휴대하는 사용자가 늘어남에도 불구하고 각각의 기기는 독자적인 데이터 동기화 프로토콜을 사용하여 데이터를 동기화 함에 따라 사용자는 자신이 소유하고 접근 가능한 모든 정보가 동기화 되도록 하기 위하여 부단한 노력을 기울여야 한다.

SyncML은 이러한 문제점을 해결하기 위하여 다수의 기업들이 제시한 새로운 industry initiative이다. SyncML은 유무선 환경에서의 효과적 동작, 다양한 전송 프로토콜 지원, 임의의 데이터 지원, 다양한 응용 수용, 휴대 단말의 자원 제약에 따른 문제 해소, 기존 인터넷 및 웹 기술 기반, 모든 단말 수용이라는 요구를 만족시키는 것을 목표로 한다. SyncML은 데이터 representation 프로토콜과 동기화 프로토콜, 그리고 transport binding의 구성요소로 되어 있다.

SyncML은 XML 기반의 Mark-up Language이다. SyncML은 Client/Server 모델을 사용하며 응용은 SyncML 어댑터를 통하여 다양한 전송 프로토콜을 사용하여 통신할 수 있다. 또한 WAP이 정의한 WSP을 사용하여 무선인터넷을 자연스럽게 수용할 수 있도록 설계되어있다.

SyncML은 현재 대다수의 휴대단말 관련 기업들의 지지를 받고 있으며 국내의 기업들도 서서히 제품들을 출시하는 상황이다. 앞으로 무선인터넷과 각종 휴대 단말의 보급이 확산됨에 따라 SyncML은 기업과 소비자 모두의 지원을 받을 것으로 예측되는 중요한 산업 표준이다.

참고문헌

[1] http://www.syncml.org "SyncML White Paper"

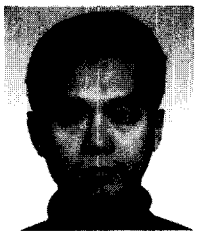
[2] http://www.syncml.org "SyncML Specification; Version 1.0.1-SyncML Representation Protocol, version 1.0.1"

[3] http://www.syncml.org "SyncML Device Information DTD; Version 1.0.1"

[4] http://www.wapforum.org "Wireless Session Protocol specification"

[5] http://www.wapforum.org "Push Architectual Overview"

저자약력



이 재 영

1988년 서울대학교 물리학과 졸업(이학사)
1990년 The Johns Hopkins University 물리학과 졸업 (이학석사)
1994년 The Johns Hopkins University 물리학과 졸업 (이학박사)
1994년-2000년 한국전자통신연구원 선임연구원
2001년-현재 주식회사 모빅스 대표이사
관심분야 : 보안 알고리즘, Embedded Linux
e-mail : leej@mobiggs.com