SyncML 적합성 및 상호 연동성 시험 연구

*이종필[0] *최 훈 **윤대균
*충남대학교 공과대학 컴퓨터공학과 분산시스템 연구실, **㈜네오스텝스
{jpyi,hchoi}@ce.cnu.ac.kr, dkyoon@neosteps.com

Implementation of the SyncML Conformance Test And Interoperability Test Environment

Jong-Pil Yi Hoon Choi DaeKyun Yoon
Dept. of Computer Engineering, Chungnam National Univ., Neosteps

요 약

SyncML 은 무선 분산환경에서 멀티플 네트웍과 플랫폼, 그리고 장치간 공유 데이터를 서로 일치(동기)시키는 산업계 표준 프로토콜로 떠오르고 있는 신기술이다.지금까지 무선 데이터 동기화는 지역적인 제약이 따라왔고 서로 다른 프로토콜을 사용하는 플랫폼간의 통신이나 데이터 교환이 불가능하였다. 이에 SyncML 기술을 이용한 무선 데이터 동기방식의 표준이 제정되었으며 이를 구현한 서로 다른 플랫폼간의 연동성과 적합성을 테스트하는 것이 필수적인 요소로 떠오르고 있다. 이 논문에서는 SyncML 을 구현한 멀티플 플랫폼간의 적합성 및 연동성을 테스트하기 위한 방안과 사례를 제시한다.

1. 서론

글로벌 마케팅 시대에 있어서 휴대 단말간 무선 통신의 중요성이나 그 보편화가 엄청난 속도로 진행되고 있는 이 시점에 각 단말이나 네트워크 간의 데이터의 상호 운용성을 보장하는 일이 큰 이슈로 떠오르고 있다.

이에 SyncML 이라는 데이터 동기방식의 표준을 정함으로써 글로벌 무선 데이터 통신 환경에서 이기종 클라이언트 단말과 서버간에 데이터 통신을 활성화시키는 노력이 진행되고 있다.

SyncML 표준은 이종 시스템 간의 공통적 정보 표현 언어 (Synchronization Markup Language) 의 제정과 그 정보의 교환 절차를 규정한 것이다. 그러나, 단말이나 서버가 SyncML 규격대로 구현되었다고 하더라도 규격과의 적합성이 객관적으로 검증되고 또 다른 시스템과의 상호 연동성이 검증되지 않는다면 이 시스템은 사용이 불가하다. 따라서 SyncML 표준 협의체는 SCTS(SyncML Conformance Test Suite)를 정의하고 이를 통하여 적합성이 검증되면 "SyncML Conformant" 인증을 부여한다[5]. 또한, Sync Fest에서 상호 운용성을 테스트 하여 통과하여야 SyncML 로고를 사용할 수 있는 권한을 획득하게 하였다. SIC(SyncML Interoperability Committee)는 이 적합성과 상호 운용성을 검증하는 전체 과정을 감독한다[8].

이 논문에서는 본 연구팀에서 구현한 서버에 대해 SyncML 적합성과 상호 운용성을 테스트를 하기위한 환경구현 상황과 자체적으로 수행한 테스트 내용을 기술한다. SIC에서 공개한 테스트 Suite을 만족하는 툴을 개발하여 SCTS(SyncML Conformance Test Suite)를 적용하였다.

본 논문의 2장에서는 SyncML 을 간략히 소개하고 SyncML 구조를 분석하며 이를 구현한 플랫폼을 테스트하기위한 단계로 적합성 검증과 상호 운용성 검증에 대해 알아본다. 3장에서는 실제 적합성 검증을 위한 테스트를 수행하는 방안과 테스트의 과정을 분석, 제시한다. 그리고 4장에서는 적합성 테스트를 통과한 product에 대해 상호 연동성을 테스트하는 과정과 결과를 분석한다. 마지막으로 5장에서 향후 연구방향을 제시함으로써 본 논문을 마무리한다.

저자 이종필은 BK21 정보통신인력양성사업의 지원으로 연구를 수행하

2. SyncML 소개

SyncML 은 XML(Extensible Markup Language) 기반의 데이터 동기화 프로토콜 표준이다. 이동 전화나 PDA 또는 노트북 등과 같이 이동 가능한 데이터 단말기를 통해서 서버나 타 단말과 교환해야 하는 데이터의 동기화 절차이며 이를 위한 시스템 구조[2]는 그림 1 과 같다.

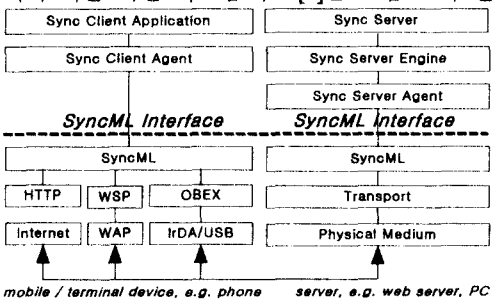


그림 1. SyncML 프로토콜 구조

였음.

2.1. SyncML 서버 구조

그림 2는 본 연구팀에서 구현한 SyncML 서버 어플리케이션 계층을 도시한 것이다.

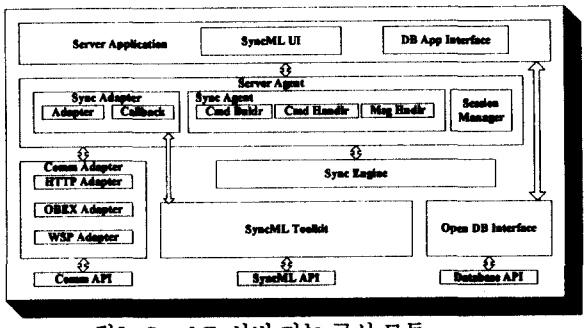


그림2. SyncML 서버 기능 구성 모듈

본 연구팀은 SyncML 기반의 서버를 개발하기 위해 기존의 SyncML 구조를 그 역할에 맞춰 분리 독립시켰다.

여기서 중요한 모듈의 기능들을 살펴보면 아래와 같다.

- **SyncML Toolkit** : SyncML API를 통해서 들어온 메시지를 인코딩하고 디코딩하는 역할을 수행한다. 이때 XML 기반과 WBXML 기반을 제공한다. 이 모듈을 통해서 Sync Adapter로 메시지를 전달한다.
- **Sync Adapter** : 툴킷을 통해 들어온 메시지를 처리하기 위한 감독자의 역할을 수행한다. 각 토큰의 의미에 따라서 해당 처리를 위해 Sync Agent를 호출하게 된다.
- **Sync Agent** : 각 커맨드에 대한 처리와 생성, 메시지의 유지 및 세션 유지 등의 핵심적인 내용을 담당한다.
- **Sync Engine** : 데이터 동기화를 위한 어플리케이션 독립적인 작업을 수행한다.

2.2. SyncML Conformance Test

적합성 검증은 SyncML 표준 규격을 기본으로 구현한 product를 대상으로 각 Test Case를 시험 대상에 적용하여 그 결과를 예측된 결과와 비교 분석하여 프로토콜 구현의 적합성 여부를 판별하는 것이다.

그림 3에서 SyncML Conformance Testing 을 위한 절차를 기술하였다. 이 절차를 위해서 각 개발자들은 SIC(SyncML Interoperability Committee)로부터 SICS(SyncML Implementation Conformance Statement Proforma)를 받아서 이에 맞춰 테스트를 진행하여야 한다.

SyncML 적합성(Conformance) 을 검증받기 위해서는 SIC로부터 받은 SICS에 기록된 절차대로 테스트를 수행하고 그 결과가 모두 성공하였다는 가정아래 이를 SIC에 보고하고 이를 SIC 가 검토하고 분석한 후 통과되면 Vendor는 적합성 테스트를 수행한다. 이 테스트가 정해진 Test Case에 맞춰 진행되고 그 결과물을 다시 SIC 가 검토하여 통과되면 SIC 는 Vendor 에게 "SyncML Conformant" 라는 인증을 부여한다 [5].

2.3. SyncML Interoperability Test

그림 4. 에서 SyncML Interoperability Testing 을 위한 절차를 기술하였다.

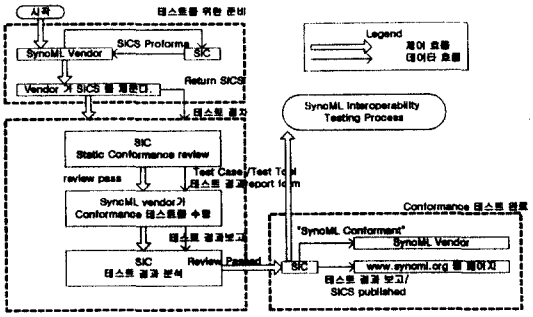


그림3. SyncML Conformance Testing Process

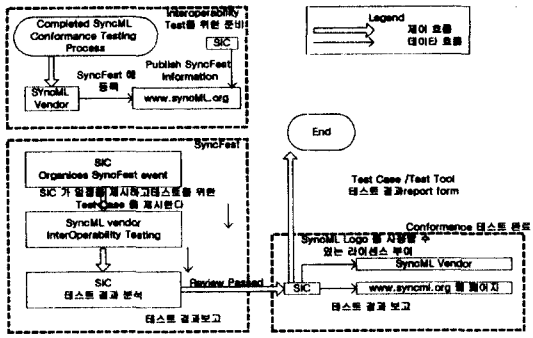
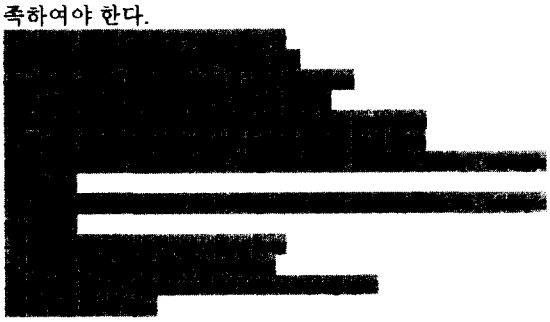


그림 4. SyncML Interoperability Testing Process

SyncML 상호 운용성(Interoperability)을 검증받기 위해서는 그림 4에서 기술하고 있는 절차를 따라야 한다. 즉, 우선적으로 적합성 검증을 통과한 Product는 SyncFest에 등록하고 정기적으로 열리는 SyncFest에서 SIC 의 감독 하에 상호 운용성을 테스트하게 된다. 이후 SIC 는 그 결과를 분석하고 결과가 통과되면 "SyncML logo"를 사용할 수 있는 권한을 부여한다[8].

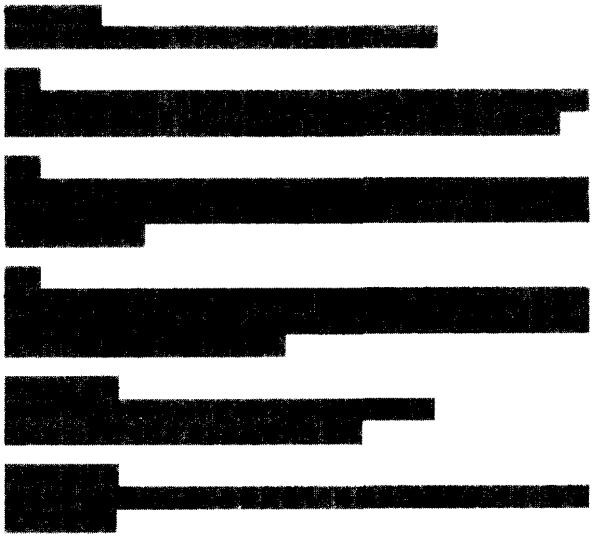
3. 적합성 검증 및 결과 분석

SyncML 적합성을 검증받기 위해서는 우선 SyncML Group에서 제시하는 기본 12가지 Test Case[7]를 모두 만족하여야 한다.



Two-Way 란 클라이언트와 서버간에 상호 메시지를 주고 받을 수 있다는 시나리오[3]이며, 이 위에 다양한 상황을 테스트하게 된다.

각각의 Test Case 는 다음과 같은 구조로 가진다.



위 Test Case #1 을 수행하는 단계에서 클라이언트와 서버간의 상태를 로그파일에 저장한 후 기대되는 결과를 얻었는지 확인하였다.

```
Running normal session.....
Initiating SCTS Session.....Ok
Source State : Initiate
Target State : Initiate
Waiting for client.....Ok
Beginning Exchange.....Ok
Receiving document info.....Ok
Receiving document.....Ok
Processing document.....
Handle Start Message.....Ok
Handle Put.....Ok
Handle Get.....Ok
Handle Alert.....Ok
Handle Alert.....Ok
Handle End Message.....Ok
Processing document.....Ok
Building document.....
Building Start Message.....Ok
Building Status.....1.0.0 1.1.0
1.2.0 1.3.0 1.4.0 Ok
Building Results.....1.2.1 Ok
Building Alert.....Ok
Building Alert.....Ok
Building End Message.....Ok
Building document.....Ok
Sending document info.....Ok
Sending document.....Ok
Ending Exchange.....Ok
Closing Connection.....Ok
Source State : Sync
Target State : Sync
Waiting for client.....Ok
Beginning Exchange.....Ok
Receiving document info.....Ok
Receiving document.....Ok
Processing document.....
Handle Start Message.....Ok
Handle Status.....Ok
Handle Status.....Ok
Handle Status.....Ok
Handle End Message.....Ok
Processing document.....Ok
Building document.....
Building Start Message.....Ok
Building Status.....2.0.0 Ok
Building Results.....Ok
Building End Message.....Ok
Building document.....Ok
Sending document info.....Ok
Sending document.....Ok
Ending Exchange.....Ok
Closing Connection.....Ok
Source State : Terminate
Target State : Map
Terminating SCTS Session.....Ok
Normal session successful !
```

위 경우에서 보여지는 메시지를 분석해보면 아래와 같다. 클라이언트와 서버간 상호 메시지 교환은 다음 순서로 일어난다.

아래와 같은 클라이언트와 서버간의 메시지 교환을 확인하였고 이를 통해서 12단계의 Test Case를 통과해서 적합성을 검증하였다.

```
client  ->  server  ->  client  ->  server
+ <SyncHdr>   + <SyncHdr>   + <SyncHdr>   + <SyncHdr>
  </SyncHdr>    </SyncHdr>    </SyncHdr>    </SyncHdr>
- <SyncBody>  - <SyncBody>  - <SyncBody>  - <SyncBody>
+ <Put>       + <Status>    + <Status>    + <Status>
  </Put>        </Status>     </Status>     </Status>
+ <Get>       + <Status>    + <Status>    <Final />
  </Get>        </Status>     </Status>     </SyncBody>
+ <Alert>     + <Status>    + <Status>    </SyncML>
  </Alert>      </Status>     </Status>
+ <Alert>     + <Status>    <Final />
  </Alert>      </Status>     </SyncBody>
  <Final />   + <Status>    </SyncML>
  </SyncBody>   </Status>
  </SyncML>   + <Results>
                </Results>
              + <Alert>
                </Alert>
              + <Alert>
                </Alert>
                <Final />
                </SyncBody>
                </SyncML>
```

4.상호 운용성 검증 및 결과분석

SyncML Logo를 사용할 수 있는 권한을 얻기 위해서는 SyncFest에서 SyncML 이 구현된 타 시스템간에 상호간 운용성을 테스트하게 된다. 본 연구팀은 개발한 서버의 상호 운용성을 시험하기 위해서, 국내에서 최초로 SyncFest에서 상호 연동성 시험에 통과한 ㈜네오스텝스의 클라이언트를 서버와 연동하여 시험을 수행하였다. 상호간 Test Case를 적용하여 연동 테스트를 수행함으로써 상호 연동성을 검증하였다. 이 때 진행되는 사항은 이미 그림4에 보였다[8]. 이 때도 물론 클라이언트와 서버간에 주고받는 메시지를 로그파일을 통해서 확인하였다.

5. 결론 및 향후연구방향

SyncML 을 구현한 제품이 글로벌 시장에서 우선권을 얻기 위해서는 SyncML Conformance Test 와 Interoperability를 통과하는 것이 필수적인 요소이다. 이에 각 Test Case를 분석하고 그 경우의 수에 대처하여 서버나 클라이언트를 개발하는 것이 절실하다고 하겠다. 추후에 아직 보고되지 않은 문제점들에 대한 대응 방안을 연구해야 할 것으로 보인다.

6. 참고문헌

[1]한국전자통신연구원, 정보통신 프로토콜 공학, 1998.1.30.

[2]SyncML Initiative, SyncML Sync Protocol version 1.0.1, 2001.5.30.

[3]SyncML Initiative, SyncML Representation Protocol Version 1.0.1, 2001.5.30.

[4]SyncML Initiative, SyncML Conformance Test Suite version 0.2, 2001.1.15.

[5]SyncML Initiative, SyncML Conformance Testing Process version 0.2, 2001.1.25.

[6]SyncML Initiative, SyncML Implementation Conformance Statement Proforma version 0.5, 2001.3.22.

[7]SyncML Initiative, SyncML Manual Test Cases version 0.3, 2001.2.2.

[8]SyncML Initiative, SyncML Interoperability Testing Process version 0.3, 2001.6.6.