2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

XML을 이용한 SDP DB 생성기

이창환[0] 오세만
동국대학교 컴퓨터공학과
(yich, semanoh)@dongguk.edu

SDP DB Generator Using XML

Chang-Hwan Yi[0] Se-Man Oh
Dept. of Computer Engineering, Dongguk University.

요 약

최근의 네트워크는 유선에서 무선으로 발전하고 있다. 무선 네트워크 기술은 여러 가지가 있다. 그 중 좁은 영역에서 사용하고 개인적인 용도로 알맞은 것이 블루투스(Bluetooth)이다. 블루투스 기술 표준은 데이터를 전송하는 단순한 통신 방법만을 정의한 것이 아니라, 몇 가지 응용에 대한 표준도 정의되어 있다. 이 응용은 모든 블루투스 기기가 전부 구현해야 하는 것이 아니라, 그 중 기기에 알맞은 것만을 선택해서 구현을 할 수 있도록 되어 있다. 그렇다면 블루투스 기기에서 다른 기기가 제공하는 서비스가 무엇인지를 알아낼 수 있는 방법이 필요하게 된다. 다른 기기의 서비스를 알아내는데 사용되는 블루투스 기술이 SDP 계층 (Service Discovery Protocol Layer) 이다.

SDP 계층은 프로토콜만으로 작동 가능한 것이 아니라, 블루투스 기기에서 제공 가능한 서비스와 서비스 속성을 정의한 내부 데이터베이스를 참조해서 작동하게 된다. 이 내부 데이터베이스는 블루투스를 구현하는 사람마다 모두 다르게 구현하고 있다. 그래서 블루투스 서비스와 서비스 속성에 관한 정보는 글과 간단한 도표로만 정의되고 있는 상황이다. 블루투스 서비스와 서비스 속성 정보를 글과 도표가 아닌 XML을 이용한 문서로 표현을 하는 방법이 나타났었다. 그러나 블루투스 기기에서 직접 서비스와 서비스 속성을 기술한 XML 문서를 바탕으로 SDP를 작동시키는 것은 블루투스 기기에 XML 파서를 포함시켜야 한다는 것을 말한다. 대체로 작은 CPU 성능과 적은 메모리를 가지고 있는 블루투스 기기에서는 XML 파서를 포함하는 큰 부담이 된다.

이에 본 논문에서는 보편적으로 사용될 수 있는 블루투스 서비스와 서비스 속성을 기술한 XML 문서에서 블루투스 기기에 적합한 내부 정보를 생성하는 생성기를 설계하고 구현을 하였다.

1. 서 론

최근의 네트워크는 유선에서 무선으로 발전하고 있다. 건물 구내에서 주로 사용되는 랜도 유선에서 무선으로 발전하고 있고, 무선 전화를 사용한 무선 통신도 각광을 받고 있다. 네트워크의 유선에서 무선으로 발전 방향에 따라 이외의 여러 기술들이 나타나고 있다. 여러 기술 중 좁은 영역에서 개인적인 용도의 무선 네트워크를 구성하기 위해서 나온 것이 블루투스(Bluetooth)이다.

다른 무선 네트워크 기술과 비교를 하였을 때 블루투스는 단순히 통신 방법만을 정의한 것이 아니라 블루투스를 활용을 위한 방법도 정의하고 있다. 예를 들면, 블루투스와 자주 비교되는 무선 랜은 무선 랜 장비간에 통신 방법만을 정의하고 있다. 그러나 블루투스는 기기간의 통신 방법만이 아니라 헤드셋(Headset)이나 다이얼업 네트워킹(Dial-up networking), 팩스(FAX), 랜 엑세스(Lan access) 등을 같은 응용도 같이 정의하고 있다. 그리고 이런 응용을 위해 사용되는 중간 기술도 정의하고 있다.

이와 같이 블루투스 장비는 여러 응용을 위한 서비스를 다른 기기에 제공하거나 다른 기기가 제공하는 서비스를 이용할 수 있다. 그러나 모든 블루투스 장비가 스펙에 정의된 모든 응용을 구현할 것이 아니라는 것이다. 장비에 따라 적당한 응용만을 구현하고 이와 관련된 서비스를 다른 기기에 제공하고 있다. 그러므로 다른 기기가 제공하는 서비스를 이용하려고 하는 기기는 주위에 자기가 원하는 서비스를 제공하는 기기가 있는지를 먼저 찾아 낼 수 있는 기술이 필요하게 되었다. 또한 서비스를 제공하는 기기는 다른 기기가 자기가 제공하는 서비스가 무엇인지를 문의하였을 때, 이를 알려줄 수 있는 기술이 필요하다. 이를 위해 블루투스에서는 서비스 디스커버리 프로토콜(Service Discovery Protocol)과 서비스 디스커버리 애플리케이션 프로파일(Service Discovery Application Profile)을 정의하고 있다.

이 SDP는 프로토콜과 프로파일의 정의만으로 작동하는 것이 아니고, 기기가 제공하는 서비스와 서비스 속성에 대한 정보도 있어야 한다. 이 정보의 저장 형태는 SDP 계층의 구현에 따라 다양한 형태를 가지게

된다. 여러 형태 중 XML 문서로 SDP 정보를 저장하는 방법도 있다. 이미 많이 사용되는 XML 기술을 사용하기 때문에 XML 문서로 SDP 정보를 저장하면, SDP 계층 내에서 뿐만 아니라 다른 곳에서도 쉽게 SDP 정보를 사용할 수 있는 장점이 있다.

XML 문서로 SDP 정보를 저장되면 XML 문서에서 필요할 때마다 정보를 읽어 SDP 계층을 구현 할 수 있다. XML 문서로 SDP 정보를 기술하게 되면 다른 곳에서도 쉽게 SDP 정보를 사용할 수 있고, 정보를 쉽게 다른 개발자들과 주고 받을 수 있는 장점이 있다. 그러나 XML 문서에서 정보를 읽어야 하기 때문에 SDP 계층에 XML 파서가 있어야 하는 것이 단점이다. 일반적으로 블루투스를 구현하려는 장비는 PC에 비해 상대적으로 작은 메모리와 적은 컴퓨팅 파워를 가지고 있다. 이런 장비에 SDP 계층의 구현을 위해 XML 파서를 탑재하는 것은 힘든 일이다.

이에 본 논문에서는 SDP 정보를 XML 문서로 표시했을 때의 장점만을 가지고, 장비가 XML 문서를 직접 다루면서 생기는 단점을 제거하기 위해 XML 문서를 SDP 계층에서 쉽게 사용할 수 있는 형태로 생성하는 생성기를 구현하려고 한다.

2. 배경 연구

2.1. SDP

SDP는 각 기기가 제공할 수 있는 서비스를 다른 기기에 알리거나 알아내기 위해서 사용되는 프로토콜이다. SDP 패킷의 기본 형태는 [그림 1]과 같다.

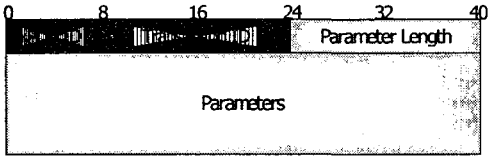

그림 1. SDP 패킷 구조

[그림 1]에서 PDU ID는 SDP 패킷의 종류를 나타내는 것으로 [표 1]과 같이 7 종류가 있다.

종류	값
SDP_ErrorResponse	0x01
SDP_ServiceSearchRequest	0x02
SDP_ServiceSearchResponse	0x03
SDP_ServiceAttributeRequest	0x04
SDP_ServiceAttributeResponse	0x05
SDP_ServiceSearchAttributeRequest	0x06
SDP_ServiceSearchAttributeResponse	0x07

표 1. SDP 패킷의 종류

Transaction ID는 SDP 서버에 대한 요청의 ID 값을 나타낸다. 이 값을 한 요청에 대해 여러 번의 응답이 오는 경우를 처리하기 위해서 사용이 된다. Parameter Length는 패킷에 실리는 Parameter의 길이를 나타낸다. Parameter에는 패킷 종류에 따라 다양한 정보가 실린다.

만약 응답의 길이가 한 번에 보낼 수 있는 패킷의 길이보다 긴 경우에는 응답이 여러 패킷으로 나누어질 수도 있다. 이런 경우에는 [그림 2]와 같은 형태의 패킷을 사용한다.

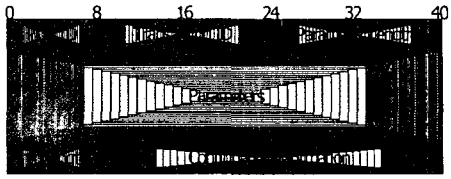

그림 2. 응답이 나누어질 때의 SDP 패킷 구조

[그림 2]에서 ContinuationInformaiton의 필드의 길이는 InfoLength 필드의 길이에 따라서 결정이 된다. 그리고 각 패킷의 종류에 따라 그 패킷에만 있는 필드의 항목이 있을 수 있다.

SDP DB에서 사용되는 서비스 속성 정보를 나타내는 형태는 [그림 3]와 같다.

그림 3. 서비스 속성 정보 구조

또한 속성 정보의 종류에는 28 종류가 있다. 속성 정보는 [그림 4]와 같은 헤더가 처음에 나타난다.

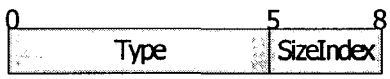

그림 4. 속성 정보 헤더

속성 정보는 헤더 다음에 헤더의 SizeIndex 필드 값에 따라 가변 길이의 데이터가 온다. Type 필드의 값과 SizeIndex의 값은 각각 [표 2], [표 3]와 같다.

종류	값
Nil, the null type	0
Unsigned Integer	1
Signed twos-complement Integer	2
UUID, a universally unique identifier	3
Text string	4
Boolean	5
Data Element Sequence	6
Data Element Alternative	7
URL, a uniform resource locator	8

표 2. Type 필드의 값

설명	값
1 Byte, Type 필드가 0인 경우에는 0 Byte	0
2 Bytes	1
4 Bytes	2
8 Bytes	3
16 Bytes	4
필드 다음에 8 bits의 길이 값이 옴.	5
필드 다음에 16 bits의 길이 값이 옴.	6
필드 다음에 32 bits의 길이 값이 옴.	7

표 3. SizeIndex 필드의 값

2.2. XML 문서 구조.

사용할 XML 문서의 구조는 공개된 블루투스 스택인 Axis OpenBT Stack[6]에서 사용하는 XML 문서 구조를 사용할 것이다. 이 문서를 사용하는 이유는 현재 XML 형태로 SDP 정보를 기술하는데 유일하게 사용이 되는 문서이기 때문이다. 다른 사람들과 XML 문서로 된 SDP 정보를 주고 받기 위해서는 많이 사용되는 문서 구조를 따를 것이 좋기 때문이다. 이 XML 문서 구조는 DTD 형태로 구성이 된 것은 아니고, 잘 정형화된 (Well-Formed) XML 문서이다. 그러나 DTD는 없지만 다음과 같은 문서 구조를 가지고 있다. 문서는 크게 다음과 같은 세 부분으로 구성되어 있다.

- bluetoothSDP
- SDPBrowsingRegister
- SDPTranslationRegister

각 부분을 자세히 설명하면, bluetoothSDP 부분은 여러 프로파일에 관한 정보를 저장하는 부분이다. 이곳에 저장이 되는 프로파일에 관한 정보는 다음과 같은 형태로 저장이 된다.

```
<PROFILE_NAME ServiceRecordHandle = "0xXXXXXXXX">
  <ServiceRecordHandle Parameter0 = "0x0aXXXXXXXX">
  </ServiceRecordHandle>
  <ServiceClassIDList NbrOfEntities = "1">
    <PROFILE_NAME_1> </ PROFILE_NAME_1>
    <PROFILE_NAME_2> </ PROFILE_NAME_2>
  </ServiceClassIDList>
  <ServiceName>...</ServiceName>
</PROFILE_NAME>
```

SDPBrowsingRegister 부분은 SDP 데이터를 브라우징할 때, 입력으로 들어오는 16진수에 해당하는 이름을 정의하고 있다. 이 부분은 다시 Protocols, ServiceClasses, AttributeIdentifierCodes에 관한 정보를 저장하는 세 부분을 가지고 있다. SDPTranslationRegister 부분은 SDPBrowsingRegister 와는 반대로 bluetoothSDP 부분에 정의된 엘리먼트 (element)나 속성(attribute)의 이름을 SDP에서 사용하는 형태로 생성하는데 사용이 된다.

3. 설계 및 구현

3.1. 전체 구조

[그림 5]는 본 논문에서 제안한 생성기의 전체 구조도이다.

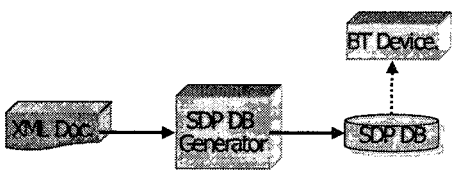

그림 5. 전체 시스템 구조도.

이미 만들어진 SDP 정보를 가지고 있는 XML 문서를 SDP DB 생성기가 입력으로 읽어, 블루투스 기기가 사용할 수 있는 형태의 DB로 생성하는 일을 한다.

3.2. 구현 환경

구현 환경은 SDP DB 생성기의 구현 환경과 생성된 DB 정보를 사용하는 SDP 계층의 구현 환경이 다르다.

SDP DB 생성기의 구현 환경은 다음과 같다.

- 장비: 일반 PC
- 운영체제: Windows 2000
- 개발도구: Visual C++ 6.0
- XML 라이브러리: Xerces C++ Parser

SDP 계층의 구현 환경은 다음과 같다.

- 장비: iPAQ (PDA)
- 운영체제: Windows CE 3.0
- 개발도구: Platform Builder 3.0

원래 Axis에서 사용하는 XML 파서는 expat - XML Parser Toolkit이다. 그러나 DB 생성기에서 사용하는 XML 파서는 Xserces C++ 파서를 사용하고 있다. 이 파서는 최신 XML 기술을 지원하기 때문에 추후 SDP 정보를 표현하는 XML 문서에 최신 XML 기술을 적용할 때 쉽게 적용을 하기 위해서 이다.

3.3. 내부 DB 구조.

생성할 SDP DB의 구조는 [그림 6]와 같이 각 항목이 항목의 연관 관계에 따라 트리 형태를 가지고 있다.

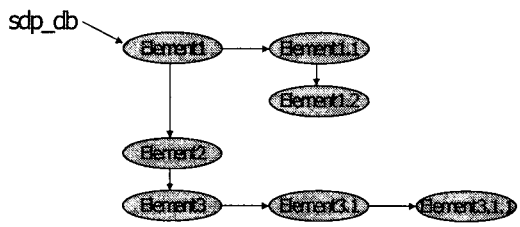

그림 6. SDP DB의 내부 구조.

트리의 각 항목을 구성하는 구조체는 다음과 같다.

```
typedef struct sdp_record {
  u32 uuid;
  u32 data;
  struct sdp_record *child;
  struct sdp_record *next;
} sdp_record;
```

uuid는 항목이 어떤 속성인지를 나타내기 위해 사용이 되고, data는 속성의 정보를 나타낸다. 만약 길이가 4 byte인 경우에는 정보가 저장된 영역에

대한 포인터를 형 변환해서 넣는다. child와 next는 트리를 구성하기 위해 사용한다.

4. 실행 결과

다음은 시리얼 포트 프로파일에 대한 SDP 정보를 XML 문서로 나타낸 것이다.

```
<?xml version="1.0"?>
<root>
<bluetoothSDP>
  <SerialPort ServiceRecordHandle = "0x0100ffff">
    <ServiceRecordHandle Parameter0 = "0x0a0100fff>
    </ServiceRecordHandle>
    <ServiceClassIDList NbrOfEntities = "1">
      <SerialPort> </SerialPort>
    </ServiceClassIDList>
    <ProtocolDescriptorList NbrOfEntities = "2">
      <L2CAP type = "DES"> </L2CAP>
      <RFCOMM type= "DES" Parameter0 = "0x0801">
      </RFCOMM>
    </ProtocolDescriptorList>
    <ServiceName>COM3</ServiceName>
  </SerialPort>
</bluetoothSDP>

<SDPBrowsingRegister>
  <Protocols
    SDP = "0x0001"  RFCOMM = "0x0003"
    L2CAP = "0x0100"> </Protocols>
  <ServiceClasses SerialPort = "0x1101"> </ServiceClasses>
  <AttributeIdentifierCodes
    ServiceRecordHandle = "0x0000"
    ServiceClassIDList = "0x0001"
    ProtocolDescriptorList = "0x0004"
    ServiceName = "0x0100">
  </AttributeIdentifierCodes> </SDPBrowsingRegister>

<SDPTranslationRegister
  L2CAP = "190100"  RFCOMM = "190003" SDP = "190001"
  ServiceDiscoveryServerServiceClassID = "191000"
  SerialPort = "191101"> </SDPTranslationRegister>
</root>
```

이 문서를 본 논문에서 제안한 생성기를 사용하여 C 코드로 생성한 SDP DB 정보의 일부를 나타낸 것이다.

```
struct sdp_record SerialPort_ServiceName = {
  UUID_SERVICENAME, (u32)"COM3", NULL, NULL};
struct sdp_record SerialPort_ProtocolDescriptorList_RFCOMM =
{
  UUID_RFCOMM, 0x03, NULL, NULL};
struct sdp_record SerialPort_ProtocolDescriptorList_L2CAP = {
  UUID_L2CAP, 0x0100, NULL,
  &SerialPort_ProtocolDescriptorList_RFCOMM};
struct sdp_record SerialPort_ProtocolDescriptorList = {
  UUID_PROTOCOLDESCRIPTORLIST,
  0x0004,
  &SerialPort_ProtocolDescriptorList_L2CAP,
  &SerialPort_ServiceName};
...
struct sdp_record SerialPort = {
  UUID_SERIALPORT, 0x1101,
  &SerialProt_ServiceRecordHandle,
  NULL};

sdp_record *sdp_db;
```

본 논문에서 제안한 생성기를 사용하여 XML 문서를 SDP 계층에서 직접 사용 가능한 형태로 생성을 하였다. XML 파서를 내장한 SDP 계층과 내부 형태의 SDP 정보를 사용하는 SDP 계층의 크기를 다음과 같이 비교하였다. 크기 비교는 윈도 CE 3.0을 기준으로 바이너리 코드 크기와 관련 정보 파일의 크기를 합한 것이다.

종류	XML 파서를 사용하는 SDP 계층	본 논문에서 제안하는 형태를 사용한 SDP 계층
크기	1.5K (XML 파일) + 35K (SDP 계층) + 18K (XML 파서) = 54.5 K	0.8 K (SDP 정보파일) + 30K (SDP 계층) = 30.8 K

표 4. SDP 계층 크기 비교표 (윈도 CE 3.0용 기준)

[표 4]과 같이 XML 파서를 내장한 SDP 계층이 24.5KB이상 큰 것을 알 수 있다. 은 메모리가 작은 장비에서는 사용하기 힘든 크기임을 알 수 있다. 또한 정보 파일의 크기도 XML 문서에 비해 절반 정도 되는 것을 알 수 있다. 정보 파일의 크기와 XML 파일의 크기와의 차이는 XML 파일의 크기가 커질수도 커진다.

5. 결론 및 향후 연구 과제

본 논문에서는 SDP 정보를 XML 문서로 표현할 수 있는 방법을 소개하고, XML로 저장된 정보를 SDP 계층에서 사용할 수 있는 내부 형태로 생성하는 생성기를 설계하고 구현을 하였다.

생성기를 통해 SDP 정보를 XML 문서로 표현해서 얻을 수 있는 장점을 살리고, 블루투스 기기에 XML 파서가 들어가서 생기는 단점을 해결할 수 있었다.

향후에는 XML 문서를 개발자가 텍스트 편집기를 통해서 만드는 것이 아니라, SDP 정보를 손쉽게 XML 문서로 만들 수 있는 편집기를 개발 할 것이다. 또한 추후에 나오는 다양한 프로파일에 대한 내용과 기기 개발자가 직접 만드는 SDP 정보도 쉽게 XML 문서로 만들 수 있는 방법이 필요하다. 마지막으로 생성기에서 다양한 형태로 정보를 생성 할 수 있어야 한다. 이를 위해 다른 개발자들이 쉽게 DB 생성에 관련된 부분만을 쉽게 생성기에 추가할 수 있는 방법을 고려해야 한다.

참고문헌

[1] Bluetooth SIG, Bluetooth Protocol Architecture, 1999
[2] Bluetooth SIG, Bluetooth Core Specification 1.1, 2001
[3] Bluetooth SIG, Bluetooth Profile Specification 1.1, 2001
[4] Bluetooth SIG, Bluetooth Assigned Number 1.1 2001
[5] Bray, Sturman, Bluetooth: Connect Without Cables, PH PTR, 2001
[6] Axis OpenBT Stack, http://sourceforge.net/projects/openbt/
[7] Xerces C++ Parser, http://xml.apache.org/xerces-c/index.html
[8] expat-XML Parser Toolkit, http://www.jclark.com/xml/expat.html