통합 디지털 방송 환경을 위한 Program Event 에 대한 정보를 포함하는 Table의 통합 parsing 기법

신윤호*, 이혁준**, 김정선*
*한양대학교 컴퓨터공학과
**한양대학교 컴퓨터공학과
e-mail: *iashinyh@hanyang.ac.kr, **breeze45@hanyang.ac.kr

Strategy to parsing Table about Event Information for integrated digital broadcasting environment

Yun-ho Shin*, Hyuk-Joon Lee**, Jung-sun Kim*
*Dept of Computer Science Engineering, Hanyang University
**Dept of Computer Science Engineering, Hanyang University

요 약

여러 표준이 존재하는 디지털 방송 환경에서 각 표준별로 Electronic Program Guide(EPG)를 얻기 위한 Program Event에 대한 정보를 포함하는 Table 파싱 기법이 다르다. 이를 해결하고 통합 디지털 방송 환경을 만들기 위해서 XML 파일을 이용한 테이블 통합 파싱 기법을 제안한다.

1. 서론

디지털 방송의 목적은 사용자에게 아날로그 방송 보다 더 높은 품질의 영상과 음성을 제공하고, 함께 전송되는 부가 데이터를 통해 사용자에게 유용한 정보를 제공하는 것이다.

이와 같은 장점으로 인해, 세계 각국에서는 아날로그 방송을 디지털 방송으로 전환하는 계획을 세우고 있다. 이미 독일에서는 2008년 디지털 전환을 완료하였고, 영국에서는 2007년부터 순차적으로 전환하여 2012년 말까지 디지털 전환의 최종 완료를 목표로 하고 있다. 또한 프랑스와 일본에서도 2011년 디지털 전환을 완료할 예정이다[1].

우리나라도 디지털 전환 계획을 세우고 있다. 방송통신위원회에 따르면, 2013년부터 모든 아날로그 방송이 종료되고, 디지털 방송으로 대체 된다고 한다. 2010년 7월 27일에 울진군 지상파 아날로그 TV방송은 종료단계에 돌입했다고 방송통신 위원회는 밝혔다[2].

이와 같이 디지털 방송으로의 전환이 세계 각국에서 이루어지고 있다. 하지만 각국에서 사용하는 표준은 DVB-MHP, OpenCable-OCAP, ATSC-ACAP등으로 여러 가지이다. 따라서 셋톱박스를 제조하는 기업은 각 표준에 맞는 셋톱박스를 개발해야한다. 이를 해결하기 위해 통합 디지털 방송 환경을 구축해야한다. 본 논문에서는 통합 디지털 방송 환경에 적용될 수 있는 Program Event Information 데이터를 포함하는 Table의 통합 파싱 기법에 대해서 제안하고자 한다.

* 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임 (No. 20100000215)

Program Event는 드라마, 스포츠와 같은 방송 프로그램을 통칭하는 단어이다. 이 정보를 가지고 있는 Table의 명칭이 각 표준마다 조금씩 다르지만, 이후부터는 공통 용어로 Event Information Table(EIT)이라고 할 것이다.

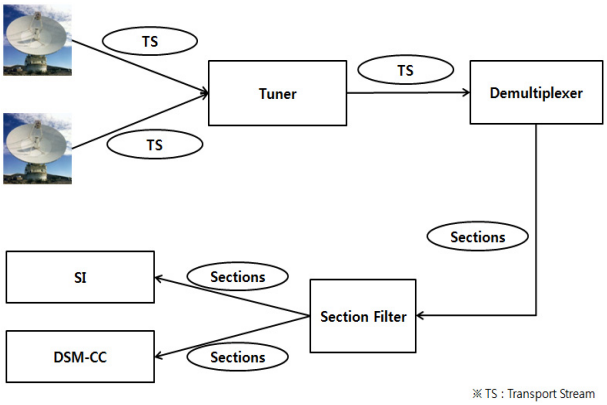
본 논문은 다음과 같이 전개된다. 2장에서는 EIT와 같은 Service Information(SI) table을 얻기 위한 디지털 방송 환경의 공통 과정인 Transport Stream이 Section으로 변하는 과정에 대해서 살펴보고, 3장에서는 각 표준별 EIT의 차이점에 대해서 살펴본다. 그리고 4장에서는 EIT 통합 파싱 기법에 대해서 살펴보고, 마지막으로 5장에서 결론을 낸다.

2. Service Information Table을 얻기 위한 디지털 방송 환경의 공통 과정

EIT와 같은 Service Information(SI) table 데이터는 방송사에서 보내주는 MPEG-2 표준의 Transport Stream(TS)에 실려 오게 되고, 그 TS를 demultiplexing하여 만들어진 Section들로 SI table을 만들게 된다.

(그림 1)을 참조하여 자세히 TS가 SI로 전달되는 과정을 살펴보도록 한다.

여러 방송국에서는 영상과 음성, SI table 등의 데이터를 담은 MPEG-2 표준인 TS를 송출한다. 그러면 각 가정에 있는 디지털 TV나 셋톱박스에서 방송사에서 송출한 TS를 수신하고 사용자가 보고자하는 방송의 TS를 디지털 TV나 셋톱박스에 내장된 Tuner에서 선택을 한 후, Demultiplexer에게 보낸다.



(그림 1) 디지털 방송 환경에서 Section이 SI로 전달되는 기본 흐름도

Demultiplexer는 Tuner로부터 넘어온 TS를 TS packet단위인 188byte로 demultiplexing을 해서 TS packet들을 만든다. 그리고 각 TS packet에 있는 packet_id(PID) 데이터를 참조하여 같은 PID를 가지고 있는 TS packet들을 모아서 Section들을 만들고 Section Filter에게 보내주게 된다.

Section Filter는 Demultiplexer로부터 Section들을 받아서, SI 모듈보다 상위 Layer인 Application layer에서 요청하는 packet_id(PID)와 table_id(TID)에 해당하는 Section만을 걸러서 Section의 TID를 보고 표준별 정해져 있는 TID 값에 따라서 SI Section이면 SI로, DSM-CC Section이면 DSM-CC로 Section을 보내준다.

3. 각 표준별 EIT의 비교

앞 장에서 모든 디지털 방송 표준의 공통 과정에 대해서 살펴보았고, 이 장에서는 공통 과정을 통해 얻은 Section들을 이용하여 만들 수 있는, 각 표준에서 정의하는, EIT의 차이점을 살펴본다.

디지털 방송 표준은 DVB-MHP, OpenCable-OCAP, ATSC-ACAP 등의 다양한 표준이 존재 한다. 하지만 모든 표준에서 Program Event에 대한 정보를 테이블 형태로 제공한다. 테이블 데이터를 만들기 위해서는 테이블에 해당하는 Section이 필요하다. 해당 Section은 테이블의 PID와 TID로 구할 수 있다. 따라서 테이블 데이터를 얻기 위해서 가장 필요한 정보는 해당 테이블의 PID와 TID 값이다. 위에서 제시한 3개의 표준에서 제공하는 EIT의 PID와 TID에 대해서 <표 1>에 정리했다.

DVB-MHP 표준에서는, DVB 표준 문서에 따르면, EIT라는 테이블을 제공하고 있고, EIT에 대한 PID와 TID의 값도 정해져서 제공한다. EIT의 PID 값은 0x0012, TID 값은 0x4E(event information section - actual transport stream, present/following), 0x4F(event information section - other transport stream, present/following), 0x50-0x5F(event information section - actual transport stream, schedule), 0x60-0x6F(event information section - other transport stream, schedule)이다[3].

<표 1> 표준별 EIT의 PID와 TID

Standard	Table	PID	TID	Description
DVB-MHP	EIT	0x0012	0x4E	event_information_section - actual_transport_stream, present/following
			0x4F	event_information_section - other_transport_stream, present/following
			0x50 to 0x5F	event_information_section - actual_transport_stream, schedule
			0x60 to 0x6F	event_information_section - other_transport_stream, schedule
OpenCable-OCAP	AEIT	per MGT	0xD6	
ATSC-ACAP	EIT	per MGT	0xCB	

OpenCable-OCAP에서는 Aggregate Event Information Table(AEIT)라는 테이블을 제공하고 있고, 이 테이블의 PID는 Master Guide Table(MGT)라는 테이블 안에 명시되어 있으며, TID는 0xD6로 정해져있다. 따라서 AEIT 데이터를 얻기 위해서는 AEIT의 PID가 필요하고, 이 PID를 얻기 위해서는 먼저 MGT 데이터를 구한 후, AEIT의 PID를 얻을 수 있다. MGT의 PID는 0x1FFC, TID는 0xC7이다[4].

ATSC-ACAP의 경우는 OCAP표준과 유사하다. ACAP에서는 EIT 테이블을 제공하고, 이 테이블의 PID는 OCAP과 같이 MGT 테이블을 통해서 얻을 수 있고, TID는 0xCB로 정해져 있다. MGT의 PID와 TID는 각각 0x1FFB와 0xC7이다[5].

4. EIT 통합 파싱 기법

3장에서 본 바와 같이, 각 표준마다 Program Event에 대한 정보를 얻는 방법이 다르다. 이것을 통합으로 다룰 수 있는 기법으로, XML 파일을 이용하는 통합 파싱 기법을 제안한다. 자세한 이야기는 이 장을 세분화하여 살펴본다. 4.1절에서는 XML 파일을 이용하여 표준별로 EIT를 요청하는 방법을 통합적으로 관리 할 수 있는 XML 파일 문법에 대해서, 4.2절에서는 테이블의 Section Format에 대한 XML 파일 문법에 대해서, 4.3절에서 XML 파일에 있는 데이터를 이용하여 실제 table을 만드는 방법에 대해서 설명한다.

4.1 표준별 EIT 요청 통합 관리를 위한 XML 파일 문법

표준별 EIT를 통합적으로 요청하기 위해 만든 XML 파일은 (그림 2)와 같다.

standard별로 관리를 해야 하기 때문에 최상위 태그로 standards 태그를 두었고, 여기에는 여러 개의 standard 태그가 포함될 수 있다. standard 태그는 name이라는 attribute를 통해 어떤 표준인지 구별이 가능하다. 이 attribute로 인해 표준에 맞는 테이블 정보를 뽑아올 수 있다. standard 태그 안에는 table 태그가 하나이상 포함

```
<standards>
    <standard name="DVB">
        <table>
            <name>EIT</name>
            <pid>0x0012</pid>
            <tid>0x4E</tid>
            <tid>0x4F</tid>
            <tid>0x50~0x5F</tid>
            <tid>0x60~0x6F</tid>
        </table>
    </standard>
        <standard name="OCAP">
        <table>
            <name>MGT</name>
            <pid>0x1FFC</pid>
            <tid>0xC7</tid>
        </table>
        <table>
            <name>AEIT</name>
            <pid></pid>
            <tid>0xD6</tid>
        </table>
    <standard name="ATSC">
        <table>
            <name>MGT</name>
            <pid>0x1FFB</pid>
            <tid>0xC7</tid>
        </table>
        <table>
            <name>EIT</name>
            <pid></pid>
            <tid>0xCB</tid>
        </table>
    </standard>

        ....
</standards>
```

(그림 2) 표준별 EIT의 PID와 TID를 위한 XML 구조

될 수 있고, 각 table 태그 안에는 name, pid, tid 태그가 존재하게 된다. name 태그는 테이블의 이름을 의미하는데 테이블 이름을 이용해 Cache에 이미 테이블이 저장되어 있는지 확인할 수 있다. 만약 Cache에 저장되어 있다면 PID, TID로 요청할 필요 없이 저장되어 있는 테이블의 값에서 해당 정보를 얻어오면 되고, 저장되어 있지 않다면 pid와 tid 태그에 있는 값을 이용해 Section Filter에게 요청하게 된다.

만약 위에서 제시한 표준이 아닌 다른 표준에서 EIT를 얻기 위해 2개의 테이블 데이터를 얻어야 한다면, 순서대로 두 개의 테이블에 대한 PID와 TID를 정의하고 그 다음으로 EIT에 대한 PID와 TID를 정의해 놓으면 된다.

4.2. 테이블 Section Format에 대한 XML 파일 문법

2장에서 보았듯이, 테이블 데이터는 Section을 이용해서 만들어진다. 즉, 모든 테이블은 해당 테이블의 Section이 존재한다. 여기서 다수의 표준별 EIT 테이블에 대한 Section을 다 살펴볼 수 없으므로, (그림 3)에 나와 있는 [6], MPEG-2 표준의 Private Section의 Format을 보면서 XML 파일로의 변환 방법에 대해서 설명한다.

Private Section은 모든 표준에서 제공하는 테이블 Section의 기본 골격이라고 생각하면 된다. EIT를 포함한 모든 테이블 Section은 Private Section의 private_data_byte 부분을 변형해서 만들어졌으므로 같은 방법을 통해 XML 파일을 생성하면 된다[3][4][5].

(그림 3)의 Private Section을 살펴보면, table_id, section_syntax_indicator, private_indicator, reserved, private_section_length를 포함하고 있다. 그리고 section_syntax_indicator의 값이 0이라면 반복문인 for 루프의 반복 횟수만큼 private_data_byte 데이터를 가지게

Syntax	No. of bits	Mnemonic
private_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
private_section_length	12	uimsbf
if (section_syntax_indicator == '0') {		
for (i = 0; i < N; i++) {		
private_data_byte	8	bslbf
}		
}		
else {		
table_id_extension	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for (i = 0; i < private_section_length-9; i++) {		
private_data_byte	8	bslbf
}		
CRC_32	32	rpchof
}		
}		

(그림 3) MPEG-2 표준의 private_section

되고, for 루프의 반복 횟수를 나타내는 N값은 private_section_length의 값을 이용해 구할 수 있다. 만약, section_syntax_indicator의 값이 0이 아닌 1이라면 header 데이터들이 추가되고 private_data_byte 데이터를 갖게 된다. 여기서의 for 루프의 반복 횟수를 나타내는 값은 private_section_length의 값에서 추가된 header의 byte수와 CRC_32의 byte수를 뺀 값이 된다.

이것을 XML로 만든 결과물은 (그림 4)와 같다.

table 태그의 name attribute에는 table의 이름이 들어가게 된다. 예를 들어, EIT section을 나타내고자 한다면 <table name="EIT">로 넣어주면 된다. table 태그 안에는 private section에 나와 있는 변수들을 태그로 만들어서 넣게 되고, 각 변수들을 위한 태그들의 값으로는 해당 변수의 bit수를 넣어준다.

그리고 조건문인 if의 경우에는 condition과 value, equal이라는 attribute를 두어 조건을 검색하도록 도와준다. condition에는 비교 대상을 나타내고 value는 비교 대상과 비교할 값을 나타낸다. 그리고 equal attribute를 통해 condition의 값과 value값이 일치해야 참인지, 불일치해야 참인지를 나타낸다.

```
<standard>
    <table name="PRIVATE">
        <table_id>8</table_id>
        <section_syntax_indicator>1</section_syntax_indicator>
        <private_indicator>1</private_indicator>
        <reserved>2<reserved>
        <private_section_length>12</private_section_length>
        <if condition="section_syntax_indicator" value="0" equal="true">
            <for condition="private_secction_length">
                <private_data_byte>8</private_data_byte>
            </for>
        </if>
        <if condition="section_syntax_indicator" value="0" equal="false">
            <table_id_extension>16</table_id_extension>
            <reserved>2</reserved>
            <version_number>5</version_number>
            <current_next_indicator>1</current_next_indicator>
            <section_number>8</section_number>
            <last_section_number>8</last_section_number>
            <for condition="private_secction_length-9">
                <private_data_byte>8</private_data_byte>
            </for>
        </else>
    </table>
</standard>
```

(그림 4) private section의 XML 구조

반복문인 for의 경우에는 condition attribute를 통해 condition attribute에 나와 있는 값까지 반복할 수 있게 한다.

4.3. XML 파일을 이용하여 Section을 파싱하고 테이블을 만드는 과정

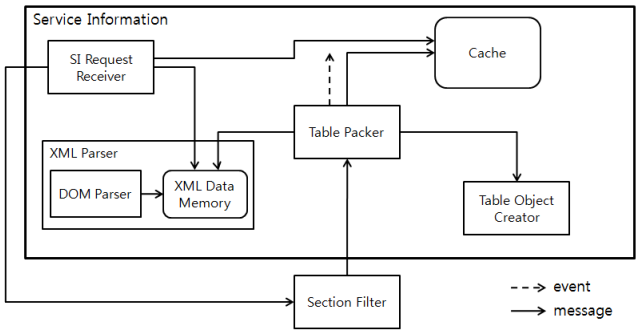
4.1절과 4.2절을 통해 두 종류의 XML 파일을 살펴보았고, 이 XML 파일들을 이용하여 Section을 파싱하는 과정은 (그림 5)와 같다. (그림 5)에 나와 있는 SI module들은 연구 중에 필요하다고 판단된 모듈들이다.

셋톱박스 부팅 시에 또는 표준 변경 시에 Setting되어 있는 표준에 해당하는 XML 파일들을 Dom Parser를 이용하여 파싱하고 XML Data Memory에 데이터를 저장해 놓는다.

Dom Parser는 Document Object Model Parser의 약자로, XML을 파싱하는 방법 중에 한 가지 방법이다. XML 파일 안의 데이터를 Tree형태로 파싱하는 Parser이다[7].

EIT와 관련된 서비스 요청은 SI module의 상위 layer인 Application에서 MHP API나 JavaTV API와 같은 Application을 위한 API를 통해 SI의 SI Request Receiver모듈에게 하게 된다. 그러면 SI Request Receiver는 XML Data Memory에 저장되어 있는, 현재 시청 하고자하는 표준에 해당하는, 4.1절에서 명시한 DVB-MHP의 경우, EIT 테이블의 PID와 TID를, OpenCable-OCAP 또는 ATSC-ACAP의 경우, MGT 테이블의 PID와 TID를 읽어와서 Section Filter에게 요청을 하게 되고, Section Filter가 PID와 TID에 해당하는 Section들을 걸러서 Table Packer에게 넘겨주게 된다. 그러면 Table Packer는 Section에 맞는 테이블 객체를 Table Object Creator로부터 가져오고, 테이블 객체에 저장할 데이터를 Section으로부터 얻기 위해 Table Section Format 데이터를 XML Data Memory로부터 가져와서 Section을 파싱한다. 파싱된 데이터를 테이블 객체에 채우고 만들어진 테이블은 Cache에 저장하고 테이블을 Cache에 저장했다고 이벤트를 가하게 된다.

여기서 OpenCable-OCAP, ATSC-ACAP 표준의 경우에는 Table Packer에서 발생하는 이벤트를 듣고 Cache에 저장되어 있는 MGT에 담겨있는 EIT의 PID와 XML Data Memory에 저장되어 있는 EIT의 TID를 받아와서 같은 작업을 반복하게 되고, 최종적으로 EIT의 데이터를 Application의 요청에 맞게 데이터를 만들어서 넘겨준다.



(그림 5) 테이블 데이터를 얻는 과정

5. 결론 및 향후 과제

디지털 방송 환경의 통합은 셋톱박스를 만드는 회사와 표준에 상관없이 방송을 보고자하는 사람들의 비용 절감을 위해 꼭 필요하다. 본 논문에서는 EIT에 한정하여서 연구를 하였지만, 이와 같이 동작하는 다른 테이블들에 대해서도 적용이 가능하다고 판단된다. 향후 과제로, 모든 디지털 방송 표준의 테이블들을 다룰 수 있는 통합 디지털 방송 환경을 위한 SI Architecture를 설계하는 것이 필요하다.

참고문헌

[1] 이종화, 정용찬, 김남두, 신호철 “KISDI 이슈리포트 주요국의 아날로그방송 종료 시범사업과 시사점” 09-08 정보통신정책연구원

[2] “울진군아날로그TV방송종료단계돌입자료(7.27).hwp” 방송통신위원회

[3] “ETSI EN 300 468 v1.9.1” ETSI

[4] “ANSI/SCTE 65 2008” SCTE

[5] “ATSC_SI” ATSC

[6] ”iso13818-1” ISO/IEC

[7] “http://www.w3.org/TR/2003/WD-DOM-Level-3-LS-20030619/” W3C