HTML5 기반 스마트 TV 플랫폼 표준의 원격 앱 제어 기술과 구현[1]

이동훈, 홍성욱, 김호년, 박동영
한국정보통신기술협회
{ dhlee, swhong, hykimfnd, dypark }@tta.or.kr

Remote Application Control Technology and Implementation of HTML5 based Smart TV Platform Standard

Dong-Hoon Lee, Sung-Wook Hong, Ho-Youn Kim, Dong-Young Park
Telecommunications Technology Association

요 약

본 논문은 2016 년 6 월 TTA 개정 표준 "HTML5 기반 스마트 TV 플랫폼(TTAK.KO-07.0111/R2)"에서 스마트폰, 태블릿 등 컴패니언 디바이스를 통해 스마트 TV 에 설치된 서비스 애플리케이션(이하, 앱)을 원격으로 조회, 실행, 종료 등의 제어를 지원하는 표준 기술을 소개한다. 이 기술은 표준에서 이미 정의된 멀티스크린 기술을 바탕으로 컴패니언 디바이스(모바일, 테블릿 등)를 통해 스마트 TV 를 인지하고 해당 TV 의 IP 주소 정보를 인식하여 JSON-RPC 프로토콜을 이용한 함수 형식 제어 요청을 통해 정보를 전달하는 방식으로 스마트 TV 수신기는 앱 정보 제공, 특정 앱 실행, 앱 종료 등 다양한 원격 앱 제어 기능을 서비스한다. 또한, 해당 원격 앱 제어 기술을 PC 기반의 가상환경인 스마트 TV 2.0 에뮬레이터 구현으로 표준기술에 유효성 검증과 서비스 응용 사례를 설명한다.

1. 서론

“ HTML5 기반 스마트 TV 플랫폼” [1][2][3] 표준은 제조사와 플랫폼 사업자 중심으로 파편화된 스마트 TV 앱 생태계 한계를 극복하고 표준 기반의 공통 플랫폼 기술을 통해 앱 생태계를 활성화 촉진시키기 위해 제정된 개방형 스마트 TV 플랫폼 기술 규격이다. 이 표준은 2014 년 스마트 TV 플랫폼에 대한 기본 기능과 멀티스크린, 적응형 스트리밍 등의 부가 기능을 지원하는 스마트 TV 1.0 표준으로 정의되었으며, 2016 년에는 원격 앱 제어, 모션·음성 등 고급 입력, 콘텐츠 동기화, 결제 등의 고급 기능 지원하는 스마트 TV 2.0 표준으로 개정되었다.

본 논문은 원격 앱 제어 기술은 스마트 TV 가 스마트 폰이나 테블릿과 같은 컴패니언 디바이스와 상호 연동하는 멀티스크린 서비스[4]를 보완하기 위해서 개발된 기술을 소개한다. 기존의 멀티스크린 서비스는 컴패니언 디바이스와 상호 연동하는 스마트 TV 앱을 사용자가 직접 스마트 TV UI 를 통해서 실행시켜야 했으므로, 컴패니언 디바이스의 사용자 경험이 스마트 TV 로 자연스럽게 전달되지 못하고 단절되는 한계를 가지고 있다. 이를 보완하고자 원격 앱 제어 기술은 두 디바이스 간 끊김없는(seamless) 사용자 경험을 지원하기 위해 컴패니언 디바이스를 통하여 스마트 TV 에 설치된 앱을 원격으로 직접 제어할 수 있는 기술을 지원한다. 컴패니언 디바이스는 스마트 TV 에 설치되어 있는 스토어 앱의 정보를 조회할 수 있고, 이 정보를 활용하여 적절한 앱을 실행하거나 종료할 수 있는 등의 제어가 가능하다. 컴패니언 디바이스는 스마트 TV 1.0 표준의 멀티스크린 기술을 활용하여 스마트 TV 디바이스의 존재를 확인하고 IP 주소를 알 수 있으며, 또한 스마트 TV 에 설치된 앱 제어를 위해 표준에서 정의된 JSON-RPC[5] 프로토콜 기반의 메시지 교환할 수 있다. 본 논문에서는 TTA 표준에서 정의하는 원격 앱 제어를 지원하기 위한 멀티스크린, JSON-RPC 프로파일, 메시지 및 프로토콜 정의 등의 핵심 기술을 상세하게 설명하며, 더불어 스마트 TV 2.0 에뮬레이터의 구현을 통한 원격 앱 제어 기술의 응용 사례를 소개한다.

본 논문의 구성은 다음과 같다. 2 장에서는 원격 앱 제어의 기반 기술인 스마트 TV 1.0 표준의 멀티스크린 기술 소개하고, 3 장에서는 원격 앱 제어의 핵심기술인 JSON-RPC 기반 메시지 정의와 전송 프로토콜의 정의를 상세히 설명한다. 이어서 4 장에서는 참조규격으로써 DIAL[6]에 대한 소개 및 스마트 TV 2.0 원격 앱 제어 기술의 차이점 소개, 5 장은 에뮬레이터를 통한 원격 앱 제어 구현과 그 응용 사례를 보인다. 마지막으로 6 장에서 결론 및 추진 방향을 제시한다.

2. 멀티스크린 표준 기술

스마트 TV 1.0 표준의 멀티스크린 기술은 스마트 TV 에서 실행되는 앱이 스마트 폰이나 태블릿과 같은 컴패니언 디바이스와 연동하여 다양한 멀티스크린 서비스를 실현하기

[1] 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [B136-16-1002, 방통융합 기반기술 테스트 환경구축]

위한 기술 요구사항을 정의한다. 멀티스크린 서비스를 위하여 스마트 TV 플랫폼과 컴패니언 디바이스는 서로의 존재를 인식하고 네트워크로 연결되어야 하며, 스마트 TV 플랫폼은 UPnP[7], mDNS[8], Web Service Discovery[9] 등 다양한 네트워크 기술 표준을 기반으로 멀티스크린 기술을 지원 한다. 이는 스마트 TV 1.0 표준이 스마트 TV 플랫폼과 컴패니언 디바이스간 연결을 위하여 특정한 Zero Conf. 네트워크 기술을 규정하지 않고, 단지 수신기가 지원하는 네트워크 기술을 통해 디바이스를 발견하고 표준에서 정의된 Javascript 형태의 멀티스크린 API 가 실행되기 위한 최소한의 요구사항만을 정의하고 있기 때문이다.

그림 1 은 표준에서 제안하는 스마트 TV 플랫폼의 멀티스크린 구조를 나타낸다. 그림과 같이 멀티스크린 커넥션 레이어는 수신기가 지원하는 Zero Conf. 네트워크의 디바이스 발견 기능을 이용하여 컴패니언 디바이스의 정보를 조회하고 디바이스간 멀티스크린 서비스를 위한 세션을 맺는 기능을 수행하는 부분으로 이를 제공하는 공통 API 는 네트워크 기술에 상관없이 독립적으로 동작함을 보장한다. 반면, 멀티스크린 서비스 레이어는 디바이스간 세션을 기반으로 동작하는 서비스 API 의 호출을 통해 서비스에 필요한 다양한 정보를 JSON 포맷으로 교환할 수 있는 기능을 제공한다.

멀티스크린 서비스를 위해 정의되는 공통 API 와 서비스 API 는 스마트 TV 1.0 표준의 애플리케이션 확장 API[10]의 하위 인터페이스로 정의되어 있으며, 개발자는 window 객체의 속성으로 접근하여 API 를 호출함으로써 멀티스크린 기술을 서비스에 활용할 수 있다.

또한, 스마트 TV 1.0 표준은 멀티스크린 기술을 위하여 스마트 TV 앱과 연동하고자 하는 컴패니언 디바이스의 요구사항을 정의하고 있다. 우선, 컴패니언 디바이스는 W3C 웹소켓 API[11]를 지원하여야 하며 이를 통해서 서비스 세션을 맺을 수 있어야 한다. 또한, 멀티스크린 서비스 세션에서 서버의 역할을 수행하여야 하며, 서비스에 맞게 적절한 인바운드(In-bound) 포트로 스마트 TV 플랫폼의 접속 요청에 대기해야 한다. 스마트 TV 에서 실행되는 앱은 API 호출을 통해 스마트 TV 플랫폼으로 컴패니언 디바이스에 대한 서비스 세션 연결을 요청할 수 있고, 이 때 스마트 TV 플랫폼은 접속하고자 하는 컴패니언 디바이스의 주소와 포트를 이용하여 웹소켓 기반의 세션을 생성하게 된다. 이후 서비스 API 의 메시지 송수신 함수를 통해서 멀티스크린 서비스를 위한 JSON 메시지를 교환할 수 있다.

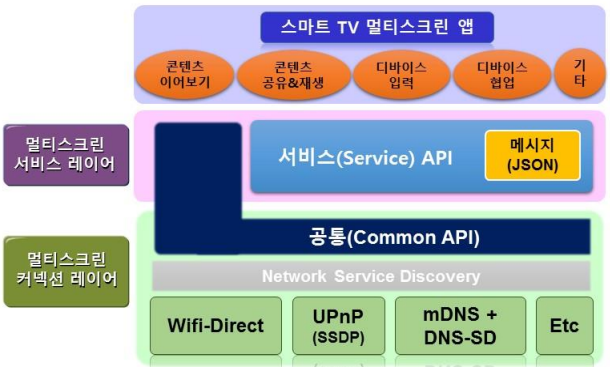

그림 1. 스마트 TV 1.0 플랫폼의 멀티스크린 구조

3. JSON-RPC 기반 원격 앱 제어

스마트 TV 2.0 표준의 원격 앱 제어는 스마트 TV 1.0 의 멀티스크린 기술을 기반으로 하는 응용 기술의 일종이다. 컴패니언 디바이스를 통해 원격으로 스마트 TV 의 앱을 제어하기 위해서 우선 컴패니언 디바이스는 2 장에서 정의한 멀티스크린 기술을 통해 스마트 TV 의 존재를 확인하고 스마트 TV 의 IP 주소를 인식한다. 원격 앱 제어를 지원하는 스마트 TV 는 JSON-RPC 서버 역할을 수행하며, 표준에서 정의하는 메시지 포맷에 따라 컴패니언 디바이스에서 전달되는 원격 앱 제어 관련 요청을 처리한다.

JSON-RPC 는 메시지에 대한 인코딩 형식을 JSON 으로 채택한 원격 프로시저 콜(Remote Procedure Call, RPC)의 일종으로, 소수의 데이터 타입과 커맨드만을 정의하는 경량의 프로토콜이다. 스마트 TV 2.0 표준은 JSON-RPC 2.0 기술을 준용하며, 원격 앱 제어를 지원에 적합하게 요청과 응답 메시지를 정의하여 사용한다. JSON-RPC 2.0 의 규격에서 요청(Request) 객체는 아래 표 1 과 같은 속성으로 구성된다.

스마트 TV 2.0 표준에서 원격 앱 제어를 지원하기 위한 JSON-RPC 요청 메시지의 함수(method) 형식은 다음과 같다.

method_name@module_name

이 함수의 형식에서 함수명(method_name)과 모듈명(module_name)은 표준의 확장 API 에서 정의하고 있는 함수명 및 계층구조와 동일하게 아래 표 2 같이 정의된다.

반면, JSON-RPC 규격에서 응답(Response) 메시지의 정의는 아래 표 3 과 같은 4 개의 속성으로 구성이 된다.

표 1. JSON-RPC 2.0 규격의 요청 객체의 속성

이 름	설 명
jsonrpc	JSON-RPC의 버전, “2.0”
method	원격 앱 제어를 위한 함수의 명
params	함수에 따른 인자의 목록
id	요청 메시지를 구분하는 id 값

표 2. JSON-RPC 요청 함수명과 모듈명 정의

함수명	설 명
getStoreApps @storeApp.application.tvExt	스마트 TV 에 설치된 앱의 정보 조회. params 속성은 “”.
createApplication @appmgr.application.tvExt	스마트 TV 의 특정한 앱을 실행. params 속성은 앱의 URL(appURL) 전달.
destroyApplication @appmgr.application.tvExt	현재 실행 중인 스마트 TV 앱을 종료 params 속성은 “”.

표 3. JSON-RPC 2.0 규격의 응답 객체의 속성

이 름	설 명
jsonrpc	JSON-RPC의 버전, “2.0”
result	함수 호출 성공 시, 그 결과를 전달하는 속성
error	함수 호출 실패 시, 에러정보를 전달하는 속성
Id	대응하는 요청메시지와 동일한 응답메시지 ID

표 4. getStoreApps 함수의 result 속성 정의

이 름		설 명
Count		원격 앱 제어로 호출이 가능한 앱의 개수
appList		개별 앱의 리스트 정보
-	appID	스토어 앱의 앱 ID 정보
	appName	스토어 앱의 이름 정보
	appDesc	스토어 앱의 상세 정보
	appVer	스토어 앱의 버전 정보

스마트 TV 2.0 표준은 원격 앱 제어를 지원하기 위하여 응답 메시지의 속성 값들 중 result 속성값을 함수에 따라서 정의한다. createApplication 과 destroyApplication 함수를 호출하는 경우에는 그 값을 “ ” 로 설정하며, getStoreApps 의 경우에는 result 속성에 현재 설치된 스토어 앱의 정보가 전달된다. getStoreApps 함수의 응답 객체의 result 속성에 대한 JSON 문자열의 정의는 위 표 4 와 같다. 이때, 개별 앱에 대한 정보를 나타내는 속성은 표준의 확장 API 가 정의한 속성을 준수하여 동일한 의미와 값으로 지정되어야 한다.

스마트 TV 2.0 표준에서 JSON-RPC 기반 원격 앱 제어를 위해 교환하는 메시지의 예는 다음과 같다.

① (컴패니언 디바이스 → 스마트 TV) 앱 정보를 조회하기 위한 JSON 메시지 전달

```
{
    “jsonrpc” : “2.0”,
    “method” : “getStoreApps@storeApp.application.tvExt”,
    “params” : “”,
    “id” : “1234”
}
```

② (스마트 TV → 컴패니언 디바이스) 앱에 대한 정보를 조회하고, 요청에 대한 응답 메시지 전달

```
{
    “jsonrpc” : “2.0”,
    “result” : {
        “count” : 2,
        “appList” : [
            {
                “appID” : “tvapp://widget.0x11000100.0x0001”,
                “appName” : “TTATestApp”,
                “appDesc” : “TTA 시험용 앱”,
                “appVer” : “1.0”
            },
            {
                “appID” : “tvapp://widget.0x1100123F.0x0010”,
                “appName” : “Game”,
                “appDesc” : “월드컵 축구 게임”,
                “appVer” : “1.1”
            }
        ],
    “id” : “1234”
}
```

위와 같이 스마트 TV 2.0 에서 정의하고 있는 JSON-RPC 프로토콜 기반의 원격 앱 제어 메시지를 전달하는 전송 프로토콜은 HTTP 1.1[12]의 POST 방식이다. POST 요청의 헤더 정보에서 Content-Type 과 Accept 헤더는 반드시 “ application/json” 으로 지정되어야 하며, 응답 메시지 헤더의 Content-Type 값도 동일하게 지정되어야 한다. 또한, HTTP POST 를 통해서 메시지를 전달하는 JSON-RPC 의 메시지는 스마트 TV 디바이스가 제공하는 서비스 URL 을 통해서 전달되는데, 스마트 TV 2.0 규격은 스마트 TV 앱 원격 제어를 지원하는 스마트 TV 디바이스가 기본(Default) 서비스 URL 로 다음과 같은 형식의 URL 을 지원하도록 규정하고 있다. 따라서, 컴패니언 디바이스는 멀티스크린 API 를 통해서 조회한 스마트 TV 의 IP 주소를 활용하여 다음 주소의 형식으로 원격 앱 제어 메시지를 전달할 수 있다.

http://SMARTTV_IP_ADDR/appRemoteCtrl
(예. http://192.168.1.123/appRemoteCtrl)

4. Dial 기술 규격과의 비교

스마트 TV 환경에서 원격 앱 제어와 관련한 기술로써 DIAL(Discovery And Launch)를 참조할 수 있다. DIAL 은 미국의 온라인 동영상 서비스 업체인 넷플릭스가 개발한 개방형 멀티스크린 프로토콜로 스마트 폰이나 태블릿 등의 컴패니언 디바이스를 통해서 TV, 블루레이 플레이어, 셋톱박스 등과 같은 디바이스의 앱을 찾고 이를 실행할 수 있는 기술이다. DIAL 프로토콜은 UPnP 기술을 기반으로 디바이스 상호간에 DIAL 서비스를 탐색하기 위해서 정의된 DIAL Service Discovery 프로토콜과 특정한 디바이스에 대해서 앱의 정보를 조회하고 지정된 앱을 실행하거나 종료하기 위해서 정의된 DIAL REST Service 로 구분된다.

DIAL Service Discovery 프로토콜은 DIAL 서비스를 위한 URN 값으로 다음과 같이 정의한다.

urn:dial-multiscreen-org:service:dial:1

M-SEARCH 요청으로 디바이스를 검색하고, LOCATION, Application-URL 헤더 정보를 통해서 앱 제어 서비스를 제공하는 RESTful URL 의 값을 찾을 수 있게 된다. 그림 2 는 DIAL 의 Discovery 프로토콜의 동작 나타낸다.

UPnP 를 기반으로 앱의 제어를 위한 RESTful URL 을 알게 되면, 컴패니언 디바이스는 HTTP 의 GET, POST, DELETE 메서드를 이용하여 TV 와 같은 장비의 앱을 제어할 수 있다. 일반적인 RESTful 서비스와 유사하게 GET 방식으로 앱의 목록을 조회할 수 있고, POST 로 앱을 실행할 수 있으며, DELETE 로 실행중인 앱을 종료할 수 있다. 그림 3 은 DIAL REST Service 프로토콜의 동작을 나타낸다.

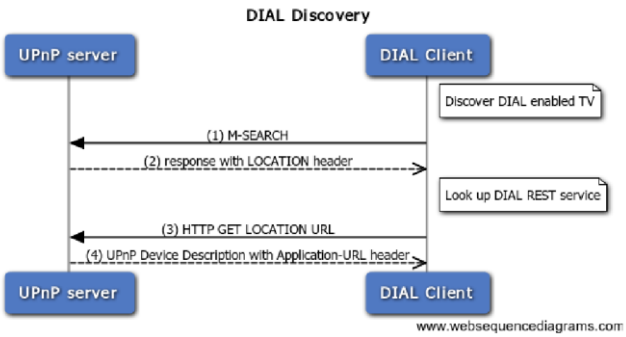

그림 2. DIAL Discovery 프로토콜의 동작

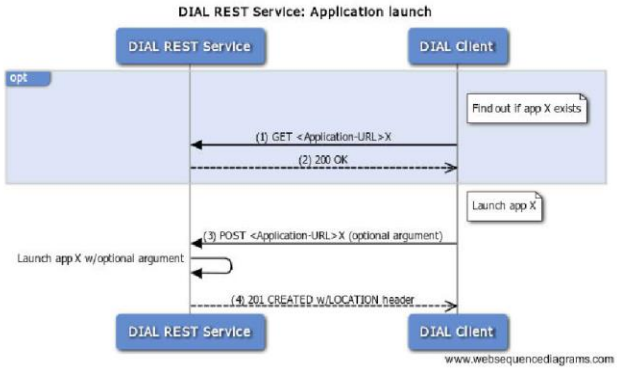

그림 3. DIAL REST Service 프로토콜의 동작

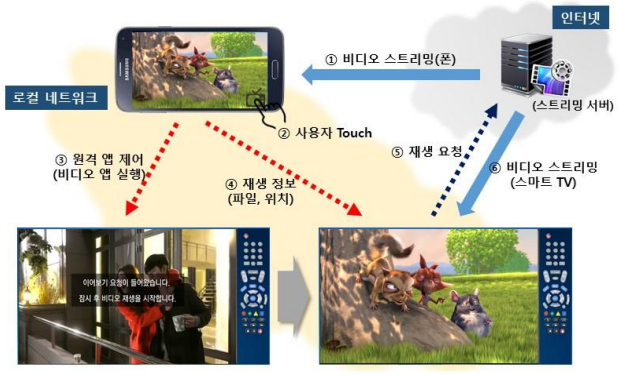

그림 4. 원격 앱 제어 기술의 응용 서비스 사례

이처럼, DIAL 규격은 Zero Conf. 네트워크로써 반드시 UPnP 를 지원해야 하는 반면, 스마트 TV 표준은 특정 Zero Conf. 네트워크에 종속 없이 독립적으로 동작한다. 또한 DIAL 의 REST Service 에서 정의된 제어 기능과 유사한 인터페이스가 스마트 TV 표준의 확장 API 에 이미 존재하기 때문에 새로운 API 가 아닌 기존 확장 API 형식과 유사하게 앱 제어 기능을 정의하는 방식으로 표준화가 결정되었으며, 이에 DIAL 이 아닌 JSON-RPC 프로토콜을 준용하여 스마트 TV 원격 앱 제어를 수행하도록 표준 규격이 정의되었다.

5. 원격 앱 제어 구현 및 응용 사례

스마트 TV 2.0 원격 앱 제어 기술에 대한 표준 유효성과 기능 검증은 스마트 TV 에뮬레이터를 통해서 구현하였다. 에뮬레이터는 스마트 TV 2.0 표준을 준수하는 스마트 TV 앱을 실제 수신기 디바이스가 없이 PC 환경에서 실행하고 검증하기 위하여 TV 수신기를 가상화하여 구현한 PC 용 프로그램이다. 에뮬레이터는 앱 개발 목적뿐만 아니라 신규 표준기술에 대한 유효성 검증용으로도 활용될 수 있으며, 본 논문의 원격 앱 제어 기술 또한 에뮬레이터를 활용하여 검증을 수행하였다.

그림 4 는 원격 앱 제어기술의 서비스 예를 나타낸다. 이처럼 컴패니언 디바이스(스마트 폰)를 통해 동영상을 시청하고 있는 사용자는 현재 재생 상태를 그대로 유지하면서 스마트 TV 를 통해서 영상을 재생할 수 있다. 컴패니언 디바이스인 스마트 폰에서 실행되는 앱에서 동영상을 시청하는 도중에 TV 모양의 아이콘을 터치하게 되면, 컴패니언 디바이스는 스마트 TV 에 JSON-RPC 요청 메시지를 통해 동영상을 재생하기 위한 앱을 조회하여 실행하게 되며, 컴패니언 디바이스에 의해서 실행된 스마트 TV 용 앱은 스마트 폰에서 실행되는 동영상의 스트리밍 주소와 현재 위치 정보를 전달받아 TV 스크린에 동일한 상태의 동영상 재생 기능을 제공한다.

6. 결론

스마트 TV 2.0 표준에서 정의하고 있는 JSON-RPC 기반의 원격 앱 제어 기술은 컴패니언 디바이스에서 스마트 TV 에 설치되어 있는 앱을 원격으로 조회, 실행, 종료 등의 제어를 할 수 있는 표준 기술이다. 이 기술은 JSON-RPC 2.0 프로토콜을 기반 함수 정의, DIAL 과의 비교를 통한 기존 확장 AIP 기반 표준 정의 등을 통해 스마트 TV 2.0 표준에 정의되었으며, 스마트 TV 에뮬레이터의 구현을 통해 검증하였다.

향후 TTA 는 HTML5 기반의 공통 플랫폼 기술을 스마트 TV 뿐만 아니라 클라우드, 스마트(디지털)사이니지 등 다양한 스마트 디바이스간 융합과 연동을 지원하는 스마트 미디어 플랫폼 기술에 대한 표준화를 추진하고 표준 적합성, 상호운용성 등 시험환경 구축 및 지원 방안을 통하여 표준기술 기반의 스마트 미디어 산업 활성화 및 육성에 기여할 것이다.

참고 문헌

[1] TTAK.KO.07-0111/R2, “HTML5 기반 스마트 TV 플랫폼”, 2016.06

[2] 이동훈, “HTML5 기반 스마트 TV 플랫폼 표준”, TTA Journal Vol.147, 2013.05

[3] 박동영, 이동훈, “스마트 TV, 이젠 표준과 손 잡아야 할 때 (개방형 생태계로의 진화를 위한 스마트 TV 표준화)”, SW 지식채널 칼럼 25 호

[4] 이동훈, 김호년, 황희선, 박동영, “HTML5 기반 스마트 TV 플랫폼 표준의 멀티스크린 적합성 시험 환경 구현”, 2013 년 방송공학회 하계학술대회논문집, 2015.06

[5] JSON-RPC, http://www.jsonrpc.org/

[6] DIAL, http://www.dial-multiscreen.org/

[7] UPnP Forum Homepage, http://www.upnp.org/

[8] Multicast DNS Homepage, http://www.multicastdns.org/

[9] W3C, “Network Service Discovery”, 2014.02, http://www.w3.org/TR/2014/WD-discovery-api-20140220/

[10] 이동훈, 김호년, 박동영, 이은향, “HTML5 기반 스마트 TV 플랫폼을 위한 확장 API 의 설계와 구현”, 2013 년 방송공학회 하계학술대회, 2013.06

[11] W3C, “The WebSocket API”, 2011.09, http://www.w3.org/TR/2011/WD-websockets-20110929/

[12] RFP 2616, IETF, Hypertext Transfer Protocol-HTTP/1.1, 1999.06