웹 기반 멀티스크린 환경을 위한 장치 검색 방안 연구

전수용, 김근형
동의대학교 게임애니메이션공학전공
e-mail:zldejs22@naver.com, geunkim@deu.ac.kr

A Study on Device Discovery for Web-based Multi Screen Environment

Su-Yong Jun, Geun-Hyung Kim
Game Engineering Major, Dong-Eui University

요 약

유비쿼터스 환경에서 네트워크에 연결된 디바이스들이 연계하여 멀티스크린 서비스를 제공하기 위해서는 네트워크에 연결된 디바이스에서 사용가능한 서비스 유/무, 활성 상태, 서비스 상세 정보를 수집하는 디바이스 검색기술은 수집한 정보를 토대로 디바이스간 통신을 통해 서비스 상태의 공유/변경을 위해 필요하다. 본 논문에서는 유럽의 MediaScape 프로젝트의 웹 기반 디바이스 검색 기술을 분석하였으며 웹 브라우저에 구현되지 않은 블루투스 서비스 검색 기능을 안드로이드폰과 PC용 네이티브 어플리케이션의 형태로 RESTful 서버로 구현하여 웹 브라우저의 에이전트와 통신을 통해 웹 브라우저에서 장치 내 디바이스의 서비스 검색을 할 수 환경을 제공한다.

1. 서론

컴퓨팅 기술과 5G 기술과 같은 광대역 무선 네트워킹 기술의 발전으로 스마트폰, 테블릿, 스마트 TV등 다양한 웨어러블 디바이스와 IoT 센서와 같은 다양한 디바이스들이 인터넷을 통해서 서로 연결되어 사람, 사물, 공간이 연결된 초 연결사회가 도래하였다. 이러한 초 연결사회에서는 사람 간 소통, 사물 간 소통, 사람과 사물 간 소통이 이루어 질 것이며 이러한 소통이 원활하게 이루어지기 위해서는 시간적 공간적 제약 없이 자유롭게 융/복합 콘텐츠를 생성하고 유통할 수 있는 멀티스크린 서비스 환경을 위한 스마트 미디어 기술이 필요하다.

본 논문에서는 개방형 웹 플랫폼 기반 멀티 디바이스 생태계에서 IoT 기술과 융합하여 사용자가 보유한 멀티 디바이스 환경과 서비스 이용 환경을 인지하고 사용자 중심의 최적의 서비스 경험을 제공하는데 필요한 장치 검색 기술로서 유럽의 MediaScape 프로젝트의 장치 검색 기술을 분석하고 웹 브라우저에서 제공하지 않는 블루투스 장치 검색 기능을 네이티브 RESTful 서버 형태로 구현하여 웹 브라우저를 위한 장치 검색 API를 보완하였다.

장치 검색 기술은 다음의 요구사항을 갖는다. 웹 브라우저와 네트워크 웹 소켓(Network Web Socket) [1], Restful API를 제공하는 서버를 구현한다. 단, 안드로이드폰의 경우 디바이스 정보를 수집할 수 있는 권한을 부여해야한다.

2. MediaScape 장치 검색 구조

장치 검색 기능은 사용 중인 디바이스 내 서비스를 검색하는 Discovery-Self 기능과 로컬 네트워크 내 사용자 디바이스들의 서비스를 수집하는 Discovery - Multi 기능으로 분류한다. Self는 사용 디바이스 에이전트로부터 서비스 존재 유/무(Presence), 활성 상태(service), 상세 정보(Extra)를 수집하고 Multi는 네트워크 웹 소켓을 사용하여 공유객체를 통해 장치 검색 정보를 공유한다.

<표 1> 장치 검색 서비스 관련 기능

MediaScape Object	
Discovery - Self	Discovery - Multi
Discovery	Agent Context
	Discovery Agent Context
	Shared State
	Mapping Service
	Application Context

MediaScape 객체가 제공하는 기능은 다음과 같다.

- Discovery Agent Context: 사용 중인 디바이스의 서비스를 검색하고 다른 디바이스의 서비스를 수집한다.
- AgentContext: Discovery Agent Context로부터 얻은 자신의 디바이스 정보를 맵 형태로 관리할 컨트롤 블록을 생성하고 하나의 컨트롤 블록마다 고유의 에이전트 ID를 등록하고 서비스 정보에 따라 유효한 핸들러를 관리한다.
- Shared State: 공유 상태의 MediaScape 객체의 디바이스 정보를 저장한 맵을 송/수산하는 역할을 수행하며 네트워크 웹 소켓 연결 여부, 공유 가능 상태, 콜백, 데이터 관련 핸들러 조건을 검사한다.

- Mapping Service: 공유가 가능한 사용자 정보를 관리하는 것으로 Context Server를 통해 App Id를 수집하여 리스트에 저장하고 이벤트를 수신할 경우 신뢰성을 검사한다. Context Server로부터 전달받은 App Id는 에이전트 아이디, 사용자 아이디, auto Clean 정보로 구성되며 해당 서비스를 시작하기 전에 app.js를 통해 생성되고 서버로 전달된다.
- Application Context: 서버로부터 전달받은 공유 상태의 다른 디바이스 정보를 클론 객체로 remote Agent Context를 생성하여 다른 디바이스의 앱을 통해 변경된 정보를 취득하여 업데이트한다. 새로운 디바이스가 추가될 때마다 remote Agent Context를 생성하고 서버를 통해 다른 앱의 클론 객체들과 구독 관계를 맺는 역할을 한다.

3. 검색 대상

사용 중인 디바이스에서 서비스를 검색하는 방법은 웹 브라우저의 객체와 네이티브 에이전트(Native Agent)를 통해 얻는 방법이 있다. 아래 표 2는 검색 방법에 따라 얻는 서비스 정보를 나타낸다[4].

<표 2> 장치 검색 방법에 따른 검색 서비스

navigator	battery, camera, audio, getlocation, language, vibration, connection, Type
screen	orientation, screen size
window	device/user proximity, touch screen

【브라우저 객체를 통한 장치 검색】

Native Agent	bluetooth, upnp

【네이티브 에이전트를 통한 장치 검색】

두 가지의 방법을 활용하여 브라우저 객체를 통해 먼저서비스를 검색한 후, 서비스를 찾지 못할 경우 네이티브 에이전트를 사용한다. 네이티브 에이전트를 사용하지 않고 크롬 개발자 환경을 통한 웹 블루투스, UPnP 서비스 정보를 얻을 수 있으며 제한적이고 보안상 취약한 http가 아닌 https 통신을 사용하여 복잡한 인증 절차를 거치는 것이 문제점이다. 따라서 기존의 네이티브 에이전트를 사용하여 처리 한다

4. 네이티브 에이전트 개발

핵심기술은 크게 '서비스 부트(Service Boot)', '블루투스 서비스' 방식이 존재한다. 서비스 부트 방식은 디바이스 부트 과정을 통해 서비스를 수집하는 에이전트가 생성, 시작되며 '디바이스/센서' 정보를 감지한다. 감지하는 디바이스 정보로는 '배터리, 카메라, 회전, 각도, 스크린 사이즈'가 있고 센서 정보는 'GPS, 근접성 센서, NFC' 정보가 존재하며 감지한 서비스 정보들은 모두 RESTful API를 통해 HTTP 메소드로 전달받을 수 있도록 RESTful 서버로 구현된다[5][6].

http://localhost:8182/discoveryagent/battery/extra
localhost:8182 - 8182 포트를 통해 서버와 통신
discoveryagent - 호스트 네임
battery/extra - 요청 URL (배터리 extra 정보를 Get)

```
{"extra":[{"level":"94%"}]}
```

【Restful API를 사용한 배터리 사용량 체크】

(그림 1) RESTful API를 활용한 디스커버리 정보

위 그림은 Restful API를 이용하여 얻은 배터리 Extra 정보이다. 그러나 블루투스와 같은 특정 서비스의 경우 안드로이드 환경에서 검색 가능한 API를 사용하여 서비스를 감지하기 때문에 네이티브 에이전트를 구현할 때 서로 다른 API를 사용한다. 또 PC 환경을 위해서는 cpprest SDK를 통해 RESTful 서버를 구현하고 윈도우 환경에서의 블루투스 디바이스를 검색하는 bluetothapis 라이브러리를 이용해 JSON 데이터를 전달하도록 구현하였다.

5. 결론

MediaScape 프로젝트의 장치 검색(Discovery) 기술은 상용 디바이스와 상호작용이 쉬운 장점이 있다. 그러나 모든 서비스 정보를 웹 브라우저 객체만을 통해 수집할 수 없으며, 현재 네이티브 에이전트를 통해 안드로이드 이외의 디바이스의 서비스를 수집할 수 없다. 해결책으로 디바이스마다 모든 서비스를 감지할 수 있도록 안드로이드와 PC를 위한 새로운 네이티브 에이전트를 개발하였다. 향후 아이폰과 스마트 TV와 같은 디바이스로 확대 적용하고 사물 인터넷 센서도 지원하는 장비 검색 기능 개발과 스마트 미디어 서비스를 개발할 계획이다.

Acknowledgement

이 논문은 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.NRF-2017R1D1A1B0303035074)

참고문헌

[1] https://en.wikipedia.org/wiki/WebSocket
[2] https://github.com/mediascape/discovery-self
[3] https://github.com/mediascape/discovery-multi
[4] https://github.com/mediascape/discovery-self/tree/master/API
[5] Restlet Library 홈페이지
[6] BlueCove API 홈페이지