한국컴퓨터정보학회지
제20권 제1호, 2012. 6.

멀티 플랫폼용 에뮬레이터 기술

• 이원주(인하공업전문대학 컴퓨터정보과)
• 이정표(케이티하이텔 플랫폼사업부)
• 김민태(케이티하이텔 iOS팀 및 Web플랫폼Lab)

I. 서론

스마트 콘텐츠란 스마트폰, 스마트 태블릿 PC, 스마트 TV 등 여러 스마트 디바이스에서 동작하는 디지털 콘텐츠이다. 스마트 콘텐츠의 종류로는 eBook, 음악, 게임, 동영상등 여러 종류가 있으며, 각 스마트 콘텐츠의 사용자 요구는 시간이 갈수록 급속히 높아지며, 다양해지고 있다[1]. 이런 다양해지는 사용자 요구를 대응하기 위해서는 스마트 콘텐츠의 형태를 단일화 시켜 다양한 스마트 디바이스에서 일률적으로 동일한 서비스를 제공하는 서비스의 필요성이 대두되고 있다[2]. 다양한 디바이스, 다양한 스마트 플랫폼에서 동일한 스마트 콘텐츠 서비스를 제공하기 위한 방법으로, 최근에는 HTML5 기반의 웹 어플리케이션(web application)형태의 스마트 콘텐츠 모델이 주목을 받고 있다.

소프트웨어 공학적 관점에서 웹 애플리케이션 또는 웹 앱은 인터넷이나 인트라넷을 통해 웹 브라우저에서 이용할 수 있는 응용 소프트웨어를 말한다. 웹 애플리케이션의 장점으로는 수천만 대의 디바이스에 소프트웨어를 설치하지 않아도 웹 애플리케이션을 유지 관리할 수 있다는 점이 장점 중의 하나이다. 웹 애플리케이션은 음악과 다양한 제스처를 지원하는 스마트 eBook, 게임, 대용량 콘텐츠 등 다양한 형태의 스마트 콘텐츠로 제공될 수 있다.

이런 이유로, 스마트 콘텐츠를 제작하는 제작자 입장에서는 한 개의 스마트 콘텐츠를 제작할 때, 다양한 스마트 디바이스에서 개발 및 테스트를 진행하여만 한다. 하지만, 물리적인 시간과 비용 등의 이유로 시중에 나와 있는 모든 스마트 디바이스를 만족시키는 개발 및 테스트는 사실상 불가능에 가깝다. 따라서 스마트 콘텐츠 제작을 위한 기본 저작 도구로서 멀티 디바이스를 뜻하는 N-스크린 에뮬레이터의 필요성이 높아져 가고 있다.

본 연구에서는 웹에 기반을 둔 HTML5 웹 어플리케이션 뷰어로서 클라우드 기반 N-스크린 에뮬레이터 개발에 필요한 멀티 플랫폼 에뮬레이터 기술을 소개한다.

II. 멀티 플랫폼용 에뮬레이터 기술

멀티 플랫폼용 N-스크린 에뮬레이터는 개발자들이 가장 많이 사용하는 Windows, iOS, Linux 기반의 플랫폼을 지원한다. 이러한 세가지 플랫폼을 지원하기 위해서는 각 플랫폼용 에뮬레이터를 따로 만드는 것 보다 멀티 플랫폼용 개발 도구를 사용하여 개발의 효율성을 높이고 개발자의 요구 사항에 집중하는 것이 필요하다. 멀티 플랫폼용 개발 도구는 여러 가지가 있지만, 본 연구에서는 멀티 플랫폼용 개발 도구로 광범위하게 사용되며 노키아의 지원으로 지속적인 유지 보수와 업그레이드가 되고 있는 Qt를 소개한다.

1. Qt

Qt는 GUI 프로그램 개발을 위한 크로스 플랫폼 위젯 툴킷이다[3]. Qt는 C++를 주로 사용하지만, 파이썬(Python), 루비(Ruby), C, 펄(Perl), 파스칼과도 연동된다. 수많은 플랫폼에서 동작하며, 상당히 좋은 국제화를 지원한다. SQL 데이터

베이스 접근, XML 처리, 스레드 관리, 단일 크로스 플랫폼 파일 관리 API를 제공한다.

Qt는 자체 페인팅 엔진과 컨트롤을 사용하며, 실행되는 플랫폼의 모습을 최대한 따라한다. 따라서 Qt는 플랫폼에 의존적인 코드를 거의 사용하지 않았으므로 서로 다른 플랫폼으로 이식하는 것이 쉬웠다. 최신 버전의 Qt는 서로 다른 플랫폼의 자체 API를 사용해서 Qt 컨트롤을 그리므로 이전 버전의 Qt에는 적용되지 않는다. wxWidgets 같은 다른 플랫폼에 의존하는 함수를 사용하는 그래픽 툴킷 들은 그 나름대로의 디자인을 가지고 있다.

Qt는 다음 플랫폼을 지원한다.

- 리눅스/X11 — X 윈도 시스템(유닉스 / 리눅스)을 위한 Qt
- 맥 OS X — 맥 오에스엑스를 위한 Qt
- 윈도우 — 마이크로소프트 윈도우를 위한 Qt
- 임베디드 리눅스 — PDA, 스마트폰 등의 임베디드 플랫폼을 위한 Qt
- 심비안 - 심비안을 위한 Qt

각각 플랫폼에는 세 종류의 에디션이 있다. 첫째 GUI 프레임워크는 네트워크와 데이터베이스를 제외한 순수 GUI 개발 에디션으로 데스크톱 라이트 라고도 한다. 둘째 풀 프레임워크는 상업용 개발을 위한 완전한 에디션이다. 셋째 오픈 소스는 오픈 소스 개발을 위한 완전한 에디션이다.

2. WebKit

웹킷(WebKit)은 애플에서 오픈소스로 개발하고 있는 웹 컨텐트(web content) 엔진 또는 웹 렌더링(web rendering) 엔진이다[4]. 웹킷은 애플 사파리와 구글 크롬 등에 채용된 엔진이며 전 세계 브라우저 시장의 36%(2012년 기준)을 차지하고 있다. 웹킷은 또한 모바일 브라우저에서도 많이 채택되고 있는데, 아마존 킨들을 포함하여 애플의 아이폰, 구글의 안드로이드폰, 블랙베리 타블릿 OS와 HP의 webOS에도 널리 사용되어 지는 엔진이다.

웹킷은 애플이 컨퀴러 엔진 프로젝트(Konqueror Browser engine project)에서 파생시켜 만든 프로젝트이지만, 현재는 애플, 구글, 노키아, 삼성등 여러 기업에서 같이 주도하여 만드는 거대한 오픈소스 프로젝트이다. 또한 웹킷은 Mac OSX, Windows, Linux등 멀티 플랫폼을 지원한다.

웹킷은 원래 Mac OS X을 위해 개발되었기 때문에, 웹킷을 사용한 브라우저는 Mac OS X 전용의 것이 많았지만, 구글 Chrome(Chromium) 등, Linux 및 Windows 용 웹 브라우저에 WebKit을 채택한 브라우저가 최근 많이 출시되고 있다. 더욱이 최신의 웹킷은 데스크톱 브라우저에 그치지 않고 모바일 플랫폼에서도 활용되고 있다.

노키아는 자사의 심비안 OS의 기본 내장 브라우저에 웹킷을 채택하여 이용하고 있다. Adobe는 Flash, Flex, HTML, 자바스크립트, Ajax 기술을 사용하여 풍부한 인터넷 응용 프로그램을 구축하는 크로스 플랫폼 런타임인 Adobe AIR에서 HTML 및 자바스크립트를 처리하는 엔진으로 WebKit을 채택 하여 사용하고있다. 구글은 구글 Chrome과 모바일 플랫폼 Android에서 사용하고 있다. WebKit/GTK+는 다양한 Web 브라우저와 메일 클라이언트 등으로 이용 되고 있다.

웹킷의 중요 구성요소는 웹코어(WebCore), 자바스크립트코어(JavaScriptCore), Drosera 이다.

2.1 웹코어(WebCore)

웹코어는 WebKit 프로젝트에 의해 개발된 HTML 및 SVG 레이아웃, 렌더링, Document Object Model (DOM) 라이브러리이다[5]. WebCore 완전한 소스 코드는 LGPL 라이센스로 공개중이다. WebKit 프레임 워크는 WebCore 및 JavaScriptCore를 Wrapping하고 C++ 기반 WebCore 렌더링 엔진과 JavaScriptCore 스크립트 엔진에 Objective-C application programming interface (API)를 제공함으로써, Cocoa API 기반 응용 프로그램에서 쉽게 참조할 수 있게 한다. 최신의 버전은 크로스 C++ 플랫폼 추상화를 포함하고 있으며, 또한 각종 port 그리고 추가 API를 제공한다.

2.2 자바스크립트코어(JavaScriptCore)

자바스크립트코어(JavaScriptCore)는 웹킷 기능을 위한 자바스크립트 엔진을 제공하는 프레임워크이다[6]. 자바스크립트코어는 KDE의 KJS 라이브러리에서 파생된 이후 많은 기능 향상이 이루어졌지만, 2008년 6월 2일 웹킷은 기존의 자바스크립트코어를 새로운 자바스크립트코어인 스퀴럴피시(Squrrel Fish) 로 교체했다고 발표했으며, 동년 9월 18일에는 스퀴럴피시 익스트림(SqurrelFish Extream)의 상위버전을 발표했다. 새로운 자바스크립트코어는 기존의 자바스크립

트 인터프리터에서, 원시 기계코드로 변경하여 실행되는 것으로 구조를 변경하였기 때문에 많이 속도가 향상되었다.

2.3 Drosera

Drosera은 WebKit의 나이트 빌드에 포함되어 있던 자바스크립트 디버거이다[7]. Drosera의 이름은 식충 식물 (즉 버그를 먹는다)의 끈끈이주걱 속의 학명에서 붙여졌다. Drosera는 Web Inspector에 포함된 디버깅 기능으로 대체되고 있다.

2.4 SunSpider

SunSpider는 자바스크립트 사용 (화면 그리기, 암호화, 텍스트 조작 등)와 관련된 작업의 자바스크립트 성능을 측정하기 위해 만들어진 벤치마크 모음이다[8]. SunSpider는 다른 브라우저 개발자들이 브라우저 사이의 자바스크립트 성능을 비교하는 데 사용하고 있다.

3. Smart Device 관리

3.1 UA(User Agent)

유저 에이전트란 사용자 측에서 작동하는 소프트웨어를 가리킨다[9]. 가령, 이메일 리더는 메일을 위한 유저 에이전트다. 유저 에이전트는 클라이언트-서버 컴퓨팅 시스템 내에서 일어나는 커뮤니케이션에 사용되는 네트워크 프로토콜의 클라이언트 역할을 한다. HTTP에서는 유저 에이전트 클라이언트 소프트웨어를 유저 에이전트로 정의한다. HTTP 커뮤니케이션 규격은 다양한 애플리케이션에서 사용된다. 전통적인 웹 브라우저 뿐만 아니라 서치 엔진 로봇, 모바일 폰의 각종 애플리케이션과 가전제품의 조작 패널에도 사용되며, 여기에 사용되는 소프트웨어 모든 의미에서 유저 에이전트로 칭할 수 있다.

HTTP를 사용하여 World Wide Web에 있는 자원의 취득 등을 하는 유저에이전트 혹은 HTTP 유저 에이전트라고 부른다. HTTP 유저 에이전트는 가져올 리소스를 HTML 렌더링 엔진으로 렌더링 하는 웹 브라우저와 리소스를 검색하고 데이터베이스화 하는 크롤러 등이 존재한다.

HTTP는 요청에 각 유저 에이전트의 고유 이름을 유저 에이전트 문자열로 응답한다. 이 문자열에는 "User-Agent :" 접두어를 가진 HTTP 요청의 일부분을 형성한다. 또한 일반적으로 응용 프로그램 이름, 버전, 호스트 운영 체제 및 언어 등의 정보를 포함한다. 웹 크롤러와 같은 로봇의 경우 웹 담당자가 로봇 운영자와 연락을 취할 수 있도록 로봇은 종종 URL과 전자 메일 주소를 포함한다. 이 정보는 서버 측에서 다양한 용도로 이용되고 있다. 예를 들어, 액세스 해석에 방문자가 사용하는 Web 브라우저 정보를 결정하는 근거가 되고 있는 것을 예로 들 수 있다. 또한 사용자 에이전트 문자열은 “로봇 배제 기준”을 사용하여 특정 페이지 또는 웹 사이트의 일부에서 크롤러를 제거하는 기준 중 하나이다. 이렇게 웹 전문가가 그들의 웹사이트의 특정 부분이 특정 크롤러가 수집하는 데이터에 포함되어야하지 않거나 특정 크롤러가 너무 많은 대역폭을 소비하고 있다고 느낀 경우 그들의 페이지를 방문하지 않도록 요청할 수 있다. 그러나 유저 에이전트 문자열은 어디 까지나 HTTP 클라이언트의 요청에서 가능하다. 일부 웹 브라우저는 사용자가 쉽게 유저 에이전트 문자열을 변경할 수 있는 기능을 가진다.

3.2 MediaQuery

미디어 타입(media type)은 단말기의 종류에 따라 각각 다른 스타일시트를 적용하게 하는 기능이며 CSS 2.1 부터 추가되었다. 하지만 실제로 많이 사용되지 않았는데, 미디어 타입만으로는 해당 기기의 특성을 정확히 파악하여 알맞은 스타일을 적용시키기 어려웠던 문제점이 있었기 때문이다. CSS3에는 미디어 타입을 개선하여, 더 구체적인 조건에서 필요한 스타일을 정확하게 적용할 수 있도록 확장하였는데, 이를 미디어 쿼리(media query)라고 한다[10]. 미디어쿼리에서 사용할 수 있는 미디어 타입은 다음과 같다.

- all : 모든 미디어 타입
- aural : 음성 합성 장치
- braille : 점자 표시 장치
- handheld : 손으로 들고 다니면서 볼 수 있는 작은 스크린에 대응하는 용도
- print : 인쇄용도
- projection : 프로젝터 표현용도
- screen - 컴퓨터 스크린을 위한용도
- tty : 디스플레이 능력이 한정된, 텔렉스(teletype), 터미널, 또는 수동 이동 장치 등, 고정 피치(fixed-pitch:폭이 일정) 글자를 사용하는 메디아를 위한 의도 “tty” 메디

아 타입에서 제작자는 픽셀(pixel) 단위를 사용하여서는 안 됨
- tv : 음성과 영상이 동시 출력되는 TV와 같은 장치
- embossed - 페이지에 인쇄된 점자 표지 장치

III. 클라우드 기반 에뮬레이터 기술

1. X11

X Window System(X11)은 GUI 환경의 구현을 위한 소프트웨어와 네트워크 프로토콜이다[11]. 현재 주로 X11버전을 쓰고 있으며 X.org에서 이를 오픈 소스 버전으로 구현하여, 레드햇을 비롯한 대부분의 리눅스 배포판에서 기본 GUI 시스템으로 사용되고 있다.

X 윈도 시스템은 서버와 클라이언트의 네트워크 모델을 지원하여 X 서버와 X 클라이언트가 서로 분리되어 작동되도록 설계되어 X 서버의 프로그램을 X 윈도 클라이언트가 이용할 수 있는 것이 큰 특징 중의 하나이다. X 윈도에서는 아이콘, 메뉴, 패널을 자유자재로 사용할 수 있게 해 주는 창 관리자와 오픈 데스크톱 환경으로 KDE, Gnome, Xfce 등을 사용할 수 있도록 지원하고 있다.

X 윈도가 그래픽 사용자 인터페이스 환경으로서 기존의 GUI 시스템과 다른 점은 네트워크 프로토콜(X프로토콜)을 기반을 둔 클라이언트와 서버 모델의 네트워크 지향 그래픽 시스템이라는 점이다.

X 응용 프로그램은 자신이 클라이언트로써 네트워크로 X 서버에 접속하여 X 서버에게 명령 서비스(예를 들면, '굴림 글꼴을 출력' 또는 '마우스 포인터를 여기로 옮겨라' 등등)를 요청하면, X 서버는 이러한 명령 요청을 받아 글꼴 서버를 통해서 굴림 글꼴을 보여 주거나, 응용 프로그램에서 마우스 포인터의 위치를 이동시켜 준다. 이와 같이 X 클라이언트는 X 서버에서 동작하면서 서버에게 명령을 전달하고, X 서버는 클라이언트에게 명령 요청의 결과를 화면에 출력해 주거나 키보드, 마우스, 터치스크린과 같은 사용자 입력을 클라이언트에게 제공해 주는 역할을 한다.

X의 클라이언트/서버의 네트워크 지향 시스템 구조는 로컬 상의 X 서버로부터 다른 시스템의 X 클라이언트의 요청을 받아들일 수 있는 또 다른 특징을 제공해 준다.

이것은 다른 시스템에 있는 응용 프로그램을 로컬 상의 X에서 실행시킬 수 있음을 의미하는 것으로써 예를 들어, 로컬 상의 X 시스템에서는 파이어폭스가 설치되어 있지 않더라도 다른 원격 시스템에 파이어폭스가 설치되어 있다면 원격 시스템에 텔넷이나 SSH로 접속하여 로그인한 후 파이어폭스를 실행하면 로컬 X 서버에서 제공하는 화면 출력을 이용하여 원격 시스템에서 파이어폭스를 사용할 수 있다는 말이다. 이때 X 응용 프로그램의 디스플레이는 로컬 X 서버에서 처리되기 때문에, 원격 시스템에서는 X서버가 없어도 상관이 없다.

2. VNC/noVNC

VNC(Virtual Network Computing) 는 AT&T를 통해 무료로 공개 된 오픈 프로그램이다[12]. 이것은 서버 클라이언트 소프트웨어로서 어디에 있든지, 컴퓨터(서버)에서 로컬 네트워크를 통하거나, 인터넷(TCP/IP)을 통해 완벽히 다른 컴퓨터의 상태(클라이언트)를 조정할 수 있게 한다. 예를 들어 재택근무시, 멀리 떨어진 곳의 서버를 관리 하거나, 컴퓨터실에서 작업을 할 때, 이 프로그램은 매우 유용하다. 비밀번호 보안, SSH(secure Shell)을 통한 보안 유지 등이 이것의 연결을 통해 가능해진다.

VNC는 컴퓨터 환경에서 RFB 프로토콜을 이용하여 원격으로 다른 컴퓨터를 제어하는 그래픽 데스크톱 공유를 한다. 자판과 마우스 이벤트를 한 컴퓨터에서 다른 컴퓨터로 전송시켜서 네트워크를 거쳐 그래픽 화면을 갱신하는 방식을 제공한다. 이 VNC 기술을 이용한 VNC 클라이언트에는 대표적으로 아래와 같은 것들이 있다.

- UltraVNC
- noMachine NXServer/NXClient
- RealVNC
- noVNC

이것들 중, noVNC는 웹기반의 VNC Client로서, HTML5와 캔버스, 웹소켓 등으로 구성되어 있으며, 인터넷 익스플로러, 구글 크롬 브라우저, 애플 사파리, 모질라의 파이어폭스 등 여러 가지 브라우저에서 VNC Client의 동작을 할 수 있으며, 모바일 브라우저(안드로이드, iOS등)에서도 지원이 된다.

3. Node.js

Node.js는 V8 (자바스크립트 엔진) 위에서 동작하는 이벤트 처리 I/O 프레임워크이다[13]. 웹 서버와 같이 확장성 있는 네트워크 프로그램 제작을 위해 고안되었다.

Node.js는 파이썬으로 만든 Twisted, 펄로 만든 펄 객체 환경, 루비로 만든 이벤트머신과 그 용도가 비슷하다. 대부분의 자바스크립트가 웹 브라우저에서 실행되는 것과는 달리, Node.js는 서버 측에서 실행된다. Node.js는 일부 CommonJS 명세를 구현하고 있다. node.js의 창시자 Ryan Dahl이 발표한 design goal을 정리하면 다음과 같다.

- Function은 직접 I/O에 연결되지 않는다.
- 저수준을 지향하며 스트리밍으로 모든 것을 처리하되, 데이터 버퍼링을 강제하지 않는다.
- TCP, DNS, HTTP등의 중요한 프로토콜에 대해서 빌트인 된 형태로 지원된다.
- 다양한 HTTP 기능들을 지원한다.
- 클라이언트 사이드 자바스크립트 프로그래밍과 비슷하며 동시에 과거의 유닉스 시스템 프로그래밍과도 친숙하다.

다음은 Node.js의 구조이다.

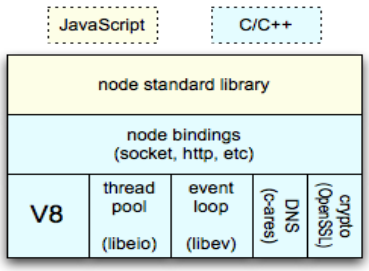

그림 1. Node.js 구조

- V8 : 자바스크립트 엔진

그림 1에서 Google V8 자바스크립트 Engine은 Google이 개발하는 오픈 소스 JIT Virtual Machine 형식의 자바스크립트 실행 엔진이다. 문맥에 따라서는 단순히 V8라고 불리는데 이름은 또한 "V8"로 알려진 V형 8기통 엔진에서 유래한다. Google Chrome과 Android Browser에서 사용된다.

ECMAScript(ECMA-262) 3rd Edition 규격, C++로 작성되어있다. 독립적으로 실행이 가능하며, C++로 작성된 응용 프로그램의 일부로 작동할 수 도 있다. 예를 들어, Firefox는 인터프리터에서 실행하여 통계 정보를 먼저 중간 코드로 변환하고 그 위에 JIT 컴파일을 하지만 V8에서는 중간 코드 없이 인터프리터도 탑재하지 않고 처음 실행할 때 컴파일한다. V8의 어셈블러는 Strongtalk 어셈블러를 기반으로 하고 있다.

- Event loop 라이브러리
- libeio : 비동기 I/O 라이브러리
- c-ares : 비동기 DNS를 지원하는 오픈 소스
- openSSL : 암호화 라이브러리

OpenSSL은 SSL 프로토콜 TLS 프로토콜의 오픈 소스 구현이다. 라이브러리 (C 언어로 쓰여져있다)는 기본적인 암호화 기능과 다양한 유틸리티 함수를 구현하고 있다. 다양한 컴퓨터 언어로 OpenSSL 라이브러리를 사용할 수 있도록 래퍼도 여러 종류가 있다. OpenSSL은 Eric A. Young과 Tim Hudson에 따르면 SSLeay(1998 년 12 월 개발자가 RSA Security에 이동했기 때문에 개발은 종료되었다.)을 기본으로 하고 있다.

OpenSSL을 사용할 수 있는 플랫폼은 거의 모든 Unix 계열 (Solaris 및 Linux, Mac OS X, BSD 포함), OpenVMS, 그리고 Microsoft Windows이다. IBM의 System i (iSeries/AS400) 포팅 버전도 있다.

- node bindings
  - C/C++로 구현된 시스템 바인딩 레이어(여기에서 socket, http등의 통신 기능 이 제공되지만 DOM에 관한 기능은 제공되지 않는다.)
- node standard library
  - 유저가 실질적으로 V8의 기능을 사용하기 위해 사용하게 될 자바스크립트 라이브러리, 이를 통해 node bindings에 접근할 수 있다.

4. WebSocket

웹소켓 (WebSocket)은 컴퓨터 네트워크의 통신 규격의 하나이다[14]. 인터넷 표준화 단체인 W3C와 IETF가 웹 서버와 웹 브라우저 사이의 통신을 위한 쌍방향 통신 기술 규격이며, 웹소켓 API는 W3C가, 웹소켓 프로토콜은 IETF가 표준화 작업을 하고 있다. 프로토콜은 RFC 6455에 정의되어 있으며. TCP위에서 동작한다.

웹소켓은 XMLHttpRequest의 결점을 해결하는 기술로서 개발되어 현재 Comet 등에 대체하는 것을 목표로 하고 있다. 이른바 Ajax 애플리케이션은 서버와 클라이언트 사이의 데이터 교환이 자주 발생하지만 기존 XMLHttpRequest는 어디 까지나 브라우저 측에서 서버에 데이터 송신 요구를 내는 수단에 불과하며, 서버 측에서 클라이언트로 데이터를 푸시하는 것이 어렵다. 한편 Comet에서는 서버 측에서의 푸시가 가능하지만, 대부분의 구현에서는 양방향 통신 요구가 발생할 때마다 TCP 핸드셰이크 요청을 다시 할 필요가 있고, HTTP 연결을 위해 커넥션을 점유하는 동안 동일한 서버에 연결하는 다른 응용 프로그램의 동작에 영향을 미칠 수 있는 등 또 다른 문제가 야기된다. 이에 대해 웹소켓에서는 서버와 클라이언트가 한번 연결을 해제한 후 필요한 통신을 해당 연결에 전용 프로토콜을 사용하여 동작한다. 기존 방식에 비해 새로운 연결을 할 필요도 없을뿐더러 HTTP 연결은 다른 프로토콜을 사용하는 등의 이유로 통신 손실이 감소되고, 하나의 연결에서 모든 데이터 송수신이 가능하므로 동일한 서버에 연결 다른 응용 프로그램에 영향이 적은 등의 장점이 있다. 원래는 HTML5 표준의 일부로 개발이 진행되고 있었지만, 후에 HTML5에서 분리되어 현재는 단일 프로토콜로서 표준 작업이 진행되고 있다.

웹소켓 인터페이스는 다음과 같이 정의된다.

```
[Constructor(DOMString url, optional
 (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
  readonly attribute DOMString url;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;

  // networking
  [TreatNonCallableAsNull] attribute Function? onopen;
  [TreatNonCallableAsNull] attribute Function? onerror;
  [TreatNonCallableAsNull] attribute Function? onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] optional unsigned short code,
                     optional DOMString reason);

  // messaging
  [TreatNonCallableAsNull] attribute Function?
                              onmessage;
  attribute DOMString binaryType;
  void send(DOMString data);
  void send(ArrayBuffer data);
  void send(Blob data);
};
```

그림 2. 웹 소켓 인터페이스

웹소켓 연결을 설정하기 위해 클라이언트는 먼저 핸드쉐이크 요청을 보낸다. 그리고 서버는 응답을 리턴 한다. 웹 브라우저에서 다음 요구 사항을 서버에 보낸다.

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

그림 3. 웹 소켓 서버 메시지

위 클라이언트 측으로 부터의 메시지로부터 서버는 아래와 같은 응답 메시지를 보내준다.

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

그림 4. 웹 소켓 클라이언트 메시지

또한 웹소켓은 스키마로서 ws와 wss 두 개의 새로운 URI 스키마를 정의 한다.

IV. 결론

기존의 애플리케이션 에뮬레이터는 각 플랫폼 당, 각 디바이스당 1개씩만을 지원했다. 하지만 스마트 콘텐츠의 특성상 OSMU(One Source Multi Use)의 조건을 충족시키기 위해 한 개의 저작도구에 반드시 복수개의 에뮬레이터가 지원되어야만 한다.

멀티 플랫폼용 에뮬레이터가 지원하는 요소로서는 해상도의 변화에 따라 자동으로 콘텐츠의 모습이 변해야 한다. 예를 들어 스마트 콘텐츠가 유저에이전트나, 미디어 쿼리등에 의해 CSS 작업이 되어 있다면 그 해상도에 맞춰 콘텐츠는 스

스로 모습을 달리한다. 이러한 결과는 멀티 플랫폼용 에뮬레이터 기술 구현에 필수적이다.

본 연구에서는 다양한 디바이스의 에뮬레이터를 설계하고 구현하는데 필요한 기술을 조사하였다. 멀티 플랫폼용 에뮬레이터 구현에 필요한 기술로 Qt, WebKit과 스마트 디바이스 관리를 위한 UA, MediaQuery를 설명하였다. 그리고 클라우드 기반 에뮬레이터 구현을 위해 X11, VNC/noVNC, Node.js, WebSocket 에 대하여 설명하였다.

이러한 기술과 HTML5 기술을 최대한 활용한다면 좀 더 빠르고, 안정적인 멀티 플랫폼용 에뮬레이터를 구현할 수 있을 것이다.

참고문헌

[1] 윤용익, 김스베틀라나, “N-Screen 표준화 고려사항 및 전략의 등장,” 정보과학회지, 제29권, 제7호, 16-22쪽, 2011년 7월.

[2] 김화숙, 이현진, 조기성, “N-Screen 서비스 현황 및 연구 개발 이슈,” 정보과학회지, 제29권, 제7호, 9-15쪽, 2011년 7월.

[3] http://qt.nokia.com/

[4] http://www.webkit.org/

[5] http://www.oss.kr/57257

[6] http://trac.webkit.org/wiki/JavaScriptCore#

[7] http://trac.webkit.org/wiki/Drosera#

[8] http://www.webkit.org/perf/sunspider/ sunspider.html

[9] http://en.wikipedia.org/wiki/User_agent

[10] http://www.w3.org/TR/css3-mediaqueries/

[11] http://en.wikipedia.org/wiki/X_Window_System

[12] http://www.csd.uwo.ca/staff/magi/doc/vnc/

[13] http://nodejs.org/

[14] http://en.wikipedia.org/wiki/WebSocket

※ 이 논문은 한국콘텐츠진흥원의 “2011년 콘텐츠산업 기술지원사업”의 지원으로 연구된 결과입니다.

저 자 소 개

이 원 주

1989: 한양대학교
전자계산학과 공학사.
1991: 한양대학교
컴퓨터공학과 공학석사.
2004: 한양대학교
컴퓨터공학과 공학박사.
현 재: 인하공업전문대학
컴퓨터정보과 부교수.
관심분야: 병렬처리시스템, 성능분석,
모바일컴퓨팅,
클라우드컴퓨팅,
N-Screen 서비스

이 정 표

1998: 인하대학교
물리학과 이학사.
2002-2008: Teleca Korea
수석연구원.
2010: Teleca Korea 기술이사
현 재: 케이티하이텔(주)
플랫폼사업부 부장
관심분야: Web & 모바일 플랫폼,
모바일컴퓨팅,
클라우드컴퓨팅,
N-Screen 서비스

김 민 태

1991: 정석항공공업고등학교
항공 전자과.
1995~1997: 한국신용평가
1997~1999: 보앤앨소프트
개발 팀장
1999~2000: 한국기술표준원
연구원
2000~2004: (주)투비웨이
개발 선임
2004~2011: (주)이미지클릭
개발 팀장
현 재: 케이티하이텔 개발실
iOS팀 및 Web플랫폼
Lab 팀장
W3C HTML5 KIG UI
그룹장
관심분야: Web & 모바일 플랫폼,
모바일컴퓨팅,
클라우드컴퓨팅,
N-Screen 서비스