Journal of The Korea Society of Computer and Information
Vol. 18, No. 3, March 2013
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2013.18.3.011

# 클라우드 기반의 N-Screen 에뮬레이터 설계 및 구현

이 원 주*, 이 정 표**, 윤 용 익***

# A Design and Implementation of N-Screen Emulator Based on Cloud

Won Joo Lee*, Jung-Pyo Lee**, Yong Ik Yoon***

## 요 약

본 논문에서는 클라우드 기반 N-Screen 에뮬레이터의 한계점을 해결하기 위해 새로운 클라우드 기반 N-Screen 에뮬레이터 설계 방법을 제안하고 구현한다. 이 방법은 서버에서 에뮬레이터의 실체 없이 웹 서비스를 이용하여 브라우저로 에뮬레이터의 모습을 확인할 수 있도록 한다. 이것은 개인용 컴퓨터 또는 모바일 환경에 구애받지 않고 동일한 서비스가 가능하도록 한 것이다. 또한, 다양한 디바이스의 각기 다른 웹 브라우저 엔진을 따로 에뮬레이팅 하기 위해 WebKit 엔진을 각 디바이스의 특성에 맞게 수정하고 관리한다. 그리고 기존의 설계 방법에서는 한 화면에 보여줄 수 있는 에뮬레이터 수가 2, 3개로 제한적 이지만, 새로운 설계 방법은 한 서버 당 100개 이상의 에뮬레이터를 구동시킬 수 있도록 성능을 향상시켰다.

▸ Keywords : N-Screen, 클라우드 컴퓨팅, HTML5

## Abstract

In this paper, we propose a new design scheme of N-Screen emulator based on Cloud and then implement the emulator, in order to solve the critical point of N-Screen emulator based on Cloud. This method, without the emulator in the server, will be able to confirm the features of the emulator with a browser using Web Service. This means that the identical service is possible without regard to personal computer or mobile environment. Also, in order to emulating each different web browser engine of the various devices separately, we revise and manage the WebKit engine to be suitable to the characteristics of each device. In the previous design method, the

• 제1저자 : 이원주 교신저자 : 윤용익
• 투고일 : 2013. 1. 3, 심사일 : 2013. 1. 15, 게재확정일 : 2013. 1. 23.
* 인하공업전문대학 컴퓨터정보과 교수(Dept. of Computer Science, Inha Technical College)
** 케이티하이텔 플랫폼사업부(Division of Paltform Business, KTH)
*** 숙명여자대학교 멀티미디어과학과 교수(Dept. of Multimedia Science, Sookmyung Women's University)
※이 논문은 한국콘텐츠진흥원 "2011년 콘텐츠산업기술지원사업"의 지원으로 연구된 결과입니다.

number of emulators which can be shown in a monitor is restricted to 2 or 3. However, we show that the proposed design method can improve the performance of server to the extent that this method could operate more than 100 emulators per each server.

▸ Keywords : N-Screen, Cloud Computing, HTML5

## I. 서 론

최근에는 스마트폰, 스마트 태블릿 PC, 스마트 TV 등 여러 스마트 디바이스에서 동작하는 디지털 콘텐츠가 다양하게 개발되고 있다. 스마트 콘텐츠의 종류로는 eBook, 음악, 게임, 동영상 등 여러 종류가 있으며, 각 스마트 콘텐츠에 대한 다양한 사용자 요구는 증가하고 있다[1]. 다양한 사용자 요구에 대응하기 위해서는 스마트 콘텐츠 형태를 단일화 함으로써 스마트 디바이스에서 동일한 서비스를 제공해야 한다. 최근에는 다양한 스마트 플랫폼에서 동일한 스마트 콘텐츠 서비스를 제공하기 위해 HTML5 기반의 웹 어플리케이션 형태의 스마트 콘텐츠를 개발하는 추세이다[2].

스마트 콘텐츠 개발자는 다양한 스마트 디바이스에서 스마트 콘텐츠를 개발하고 테스트를 진행해야 한다. 하지만, 현실적으로 다양한 스마트 디바이스를 만족시키는 스마트 콘텐츠 개발은 물리적인 시간과 비용을 증가시키는 문제점이 있다. 이러한 문제점을 해결하기 위한 하나의 방법은 스마트 콘텐츠 개발을 위한 기본 저작 도구로 N-Screen 에뮬레이터를 개발하는 것이다. 따라서 본 논문에서는 클라우드 기반의 N-Screen 에뮬레이터를 설계하고 구현하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. II장에서는 멀티 플랫폼용 N-Screen 에뮬레이터에 대하여 소개하고 문제점을 설명한다. III장에서는 본 논문에서 제안하는 클라우드 기반의 N-Screen 에뮬레이터 설계에 대하여 설명한다. 그리고 IV장에서는 클라우드 기반의 N-Screen 에뮬레이터 구현에 대하여 설명하고, V장에서 결론을 맺는다.

## II. 관련 연구

멀티 플랫폼용 N-Screen 에뮬레이터는 스마트 콘텐츠 개발자들이 많이 사용하는 Windows, iOS, Linux 기반의 플랫폼을 지원한다. 이러한 플랫폼을 지원하기 위해서는 각 플랫폼용 에뮬레이터를 각각 별도로 사용하는 것 보다 멀티 플랫폼용 개발 도구를 사용하여 개발 효율성을 높이는 것이 필요하다. 본 논문에서는 N-Screen 에뮬레이터 설계 및 개발에 필요한 요소 기술에 대하여 설명한다.

### 1. Qt

Qt는 GUI 프로그램 개발을 위한 크로스 플랫폼 위젯 툴킷이다[2][3]. Qt는 C++를 주로 사용하지만, Python, Ruby, C, Perl, 파스칼과도 연동된다. 수많은 플랫폼에서 동작하며, 국제화를 지원한다. SQL 데이터베이스 접근, XML 처리, 스레드 관리, 단일 크로스 플랫폼 파일 관리 API를 제공한다. Qt는 다음 플랫폼을 지원한다.

- Linux/X11: X 윈도 시스템(UNIX/Linux)을 위한 Qt
- 맥 OS X: 맥 OS X를 위한 Qt
- 윈도우: 마이크로소프트 윈도우를 위한 Qt
- 임베디드 Linux: PDA, 스마트폰 등 임베디드 플랫폼을 위한 Qt
- 심비안: 심비안을 위한 Qt

### 2. WebKit

WebKit은 애플에서 오픈소스로 개발하고 있는 web content 엔진 또는 web rendering 엔진이다[2][4]. WebKit은 애플 사파리와 구글 크롬 등에 채용된 엔진이며 2012년 기준으로 전 세계 브라우저 시장의 36%를 차지하고 있다. 또한 WebKit은 아마존 킨들, 아이폰, 안드로이드폰, 블랙베리 타블릿 OS와 HP의 webOS에도 널리 사용되는 엔진이다.

WebKit의 구성요소는 WebCore, JavaScriptCore, Drosera, SunSpider 등이 있다. WebCore는 WebKit 프로젝트에 의해 개발된 HTML 및 SVG 레이아웃, 렌더링, DOM(Document Object Model) 라이브러리이다[5]. JavaScriptCore는 WebKit 기능을 위한 자바스크립트 엔진을 제공하는 프레임워크이다[6]. 최신 JavaScriptCore는 기존의 JavaScript 인터프리터에서 원시 기계코드로 변경하

여 실행함으로써 속도를 향상시켰다. Drosera는 WebKit의 나이트 빌드에 포함되어 있던 자바스크립트 디버거이다[7]. SunSpider는 화면 그리기, 암호화, 텍스트 조작 등의 작업에서 자바스크립트 성능 측정을 위한 벤치마크 모음이다[8].

### 3. Smart Device 관리

User agent는 클라이언트에서 실행되는 소프트웨어를 가리킨다[9]. 이메일 리더는 메일을 위한 User agent 이다. User agent는 클라이언트-서버 컴퓨팅 시스템에서 네트워크 프로토콜의 클라이언트 역할을 한다. HTTP에서는 User agent 클라이언트 소프트웨어를 User agent로 정의한다. HTTP 프로토콜은 다양한 애플리케이션에서 사용된다. 전통적인 웹 브라우저 뿐만 아니라 서치 엔진 로봇, 모바일 폰의 각종 애플리케이션과 가전제품의 조작 패널에도 사용되며, 여기에 사용되는 소프트웨어를 모든 의미에서 User agent라 할 수 있다[2].

미디어 타입은 단말기의 종류에 따라 각각 다른 스타일시트를 적용하게 하는 기능이다. 하지만 미디어 타입 만으로 해당 기기의 특성을 정확히 파악하여 알맞은 스타일을 적용하기에는 어려움이 있다. CSS 3에서는 미디어 쿼리(media query)를 사용하여 구체적인 조건에서 필요한 스타일을 정확하게 적용할 수 있도록 확장하였다[10]. 미디어쿼리에서 사용할 수 있는 미디어 타입은 다음과 같다.

- all: 모든 미디어 타입
- aural: 음성 합성 장치
- braille: 점자 표시 장치
- handheld: 휴대용 작은 스크린에 대응하는 용도
- print: 인쇄용도
- projection: 프로젝터 표현 용도
- screen: 컴퓨터 스크린을 위한 용도
- tty: 디스플레이 능력이 한정된, teletype,
- tv: 음성과 영상이 동시 출력되는 TV와 같은 장치
- embossed: 페이지에 인쇄된 점자 표지 장치

## III. 클라우드 기반 N-Screen 에뮬레이터 설계

### 1. 시스템 구성

클라우드 기반 에뮬레이터는 N-Screen 에뮬레이터의 실행 화면을 이미지화하여 일반 브라우저에서 에뮬레이터로 사용할 수 있도록 시스템을 구성한다. 이때 VNC(Virtual Network Computing) 기술은 에뮬레이터의 실행 화면을 이미지화 하는 기능을 제공한다. 클라우드 기반 N-Screen 에뮬레이터 서비스는 그림 1과 같다.

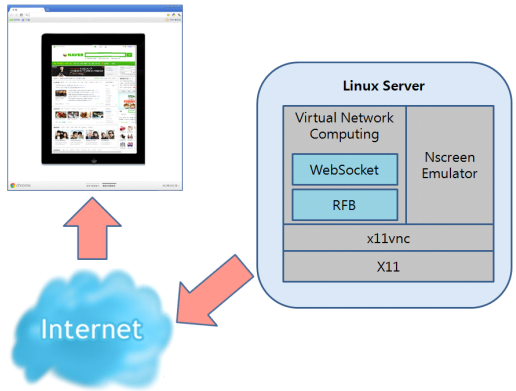



그림 1. 클라우드 기반 N-Screen 에뮬레이터 서비스
Fig. 1. N-Screen emulator service based on Cloud.

그림 1에서 클라우드 기반 N-Screen 에뮬레이터의 서버를 구성하는 요소는 다음과 같다. X11은 사용자 인터페이스를 규정하지 않는다[11]. X11은 개별 클라이언트 프로그램의 관리 하에 있기 때문에 X 기반 환경의 외형은 다양하며 프로그램마다 인터페이스가 다르다. X는 OS의 Kernel 부분을 포함하지 않고 응용 프로그램 계층 구축의 기반이며 네트워크 연결을 통해서 사용되도록 설계한다. X 시스템은 클라이언트-서버 모델로 각종 클라이언트와 통신한다. X11 시스템 구조는 그림 2와 같다.

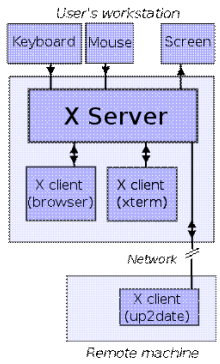



그림 2. X11 시스템 구조도
Fig. 2. X11 system architecture.

그림 2에서 X Server는 그래픽 출력 요청을 받아 Keyboard, Mouse, Screen 등의 사용자 입력을 X client에 전송한다. X 프로토콜은 운영체제에 독립적이기 때문에 UNIX, Windows에서 사용할 수 있다. 특히, Windows에서는 소프트웨어 및 하드웨어(펌웨어)에서 X 프로토콜을 처리하는 X 터미널이 존재한다. X 터미널은 UNIX 시스템이 고가일 때 GUI를 제공하는 장치로 널리 사용되었다.

원격 클라이언트 프로그램을 로컬 서버에 게시하려면 터미널 창을 열고 telnet 또는 ssh에서 원격 클라이언트 응용 프로그램 또는 shell을 실행하여 입출력 장치를 로컬로 지정하여 클라이언트를 시작한다. 클라이언트 응용 프로그램은 로컬 서버와 연결된 로컬 컴퓨터의 디스플레이와 입력 장치를 사용하여 동작한다.

X11VNC는 VNC(Virtual Network Computing) 서버 프로그램이다[12]. X11VNC는 지속적으로 X 서버 프레임 버퍼로부터 데이터를 받아 클라이언트 측으로 데이터 접근을 허용한다. 이것은 사용자가 네트워크 또는 인터넷을 통해 원격 컴퓨터에서 자신의 X11 개인용 컴퓨터(KDE, GNOME, XFCE 등)을 제어할 수 있다. 또한, 웹캠이나 TV 튜너 카드, iPAQ, Neuros OSD, Linux 콘솔, 그리고 맥 OS X 그래픽 디스플레이와 같은 비 X11 프레임 버퍼 장치의 데이터를 수신할 수 있다. X11VNC 원격 제어를 위한 별도의 디스플레이를 생성 하지 않는다. 대신 실시간으로 UNIX와 같은 컴퓨터의 모니터에 표시된 기존의 X11 디스플레이를 사용한다. X11VNC는 사용자가 접속 암호를 설정하거나 UNIX ID와 암호를 사용할 수 있도록 보안 기능을 가지고 있다. 보안 SSL(Secure Socket Layer) 링크를 통하여 연결하기 위한 옵션이 있다. SSL 자바 VNC 뷰어 애플릿은 웹 브라우저에서 보안 연결을 가능하도록 한다.

N-Screen Emulator는 Qt로 만들어진 멀티플랫폼, 멀티 디바이스용 Web app 전용 뷰어이다. Canvas element는 HTML5의 일부 동적 2 차원 비트맵 이미지 렌더링을 위한 HTML 요소이다. Mac OS X v10.4 내에서 WebKit 구성 요소로 Dashboard 위젯과 Safari에서 응용 프로그램을 강화하기 위해 애플이 최초로 도입하였다. canvas element는 width와 height 속성으로 그리기 가능한 영역을 지정한다. 자바스크립트 코드는 해당 영역에서 다른 일반적인 2 차원 API와 비슷한 API를 사용하면 게임, 애니메이션, 그래프 작성, 이미지 구축 등과 같은 동적 그래픽을 생성할 수 있다. API는 상태 관리, 변형, 합성, 색상 및 스타일 라인 캡과 접합, 그림자, 직사각형, 경로, 포커스 관리, 문자열, 픽셀 조작, 이미지 변환 등을 지원한다.

VNC는 RFB와 WebSocket으로 구성되어 있다. RFB는 그래픽 사용자 인터페이스에 대한 원격 액세스를 위한 간단한 프로토콜이다. 이 프로토콜은 프레임 버퍼 수준에서 작동하기 때문에 X11과 Windows 3.1/95/NT, Macintosh 등 모든 윈도우 시스템에 적합하다. WebSocket은 RFB에서 받은 실행 화면을 브라우저 클라이언트로 전송하는 기능을 제공한다.

## 2. N-Screen 에뮬레이터 구조

N-스크린 에뮬레이터는 그림 3과 같은 구조로 설계한다.

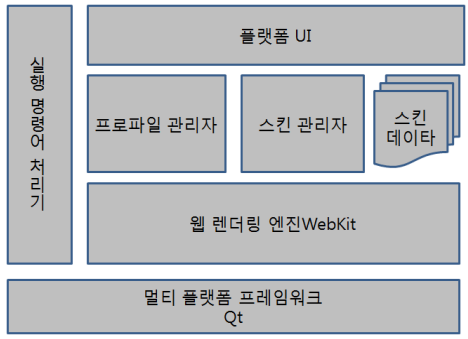



그림 3. N-Screen 에뮬레이터 구조
Fig. 3. N-Screen emulator architecture.

그림 3에서 플랫폼 UI는 Windows, Mac, 그리고 Linux에서 해당 플랫폼의 기본 UI 가이드에 따라 구현한다. 그리고 에뮬레이터 실행 중에 다른 스마트 콘텐츠를 재생할 수 있도록 URL 입력 기능을 제공하며, 가로, 세로 전환을 위한 메뉴도 제공한다.

프로파일 관리자는 에뮬레이터 실행시 전달 받은 프로파일 정보를 사용하여 웹 렌더링 엔진이 대상 스마트 디바이스의 브라우저와 같은 기능들을 가질 수 있도록 세부 기능 조정을 담당한다.

스킨 관리자/스킨 데이터는 폰, 패드 그리고 TV 해상도의 스킨을 지원하며 해당 해상도에 맞는 그래픽 데이터를 가지고 있다.

웹 렌더링 엔진은 스마트 콘텐츠를 실제로 재생하는 역할을 담당한다. 현재 대부분의 스마트 디바이스용 브라우저가 웹킷을 기반으로 하고 있기 때문에 에뮬레이터도 WebKit을 기반으로 작성되었다.

실행 명령어 처리기는 에뮬레이터 실행시 다양한 입력 옵션 처리를 담당한다. 입력 요소로는 디바이스 종류, 가로/세로 모드, 프로파일 정보 등이 있다.

멀티 플랫폼 프레임워크는 하나의 소스 코드로 다양한 플

랫폼을 대응하기 위한 기반 프레임워크로 Qt를 사용한다.

## 3. N-Screen 에뮬레이터 내부 구성

클라우드 기반 N-Screen 에뮬레이터는 크게 웹 브라우저 기반 클라이언트와 VNC 기반 서버로 구성한다.

웹 브라우저 기반 클라이언트는 브라우저를 사용하여 클라우드에서 실행되는 에뮬레이터를 출력하는 모듈로 HTML/자바스크립트/CSS를 사용하여 설계한다. 모듈은 초기 화면을 담당하는 HTML/CSS 파일과 서버와의 연동 및 화면 출력을 담당하는 자바스크립트 파일들로 구성된다.

실제 에뮬레이터의 화면은 canvas 태그를 이용하여 출력한다. 출력은 Display 모듈에서 담당하며 X11VNC 서버에서 전송하는 이미지 파일을 디코딩해서 보여주는 것이 핵심 작업이다. 서버와의 통신은 WebSocket이 담당한다. WebSocket 모듈로 실제 전송되는 데이터는 RFB 프로토콜로 자바스크립트로 작성된 RFB 모듈에서 인코딩과 디코딩을 제공한다.

클라우드에서 실행되는 에뮬레이터와 브라우저 사이의 연계는 VNC 기반의 서버에서 담당한다. 에뮬레이터는 X11 라이브러리를 사용하여 화면 출력을 제공하고, VNC 서버의 한 종류인 X11VNC는 X11의 출력을 받아 브라우저에서 사용할 수 있는 양식으로 인코딩을 한다. 인코딩된 화면 출력은 RFB 프로토콜을 사용하며 WebSocket을 이용하여 브라우저와 통신하도록 한다.

## 4. 성능 향상을 위한 클라우드 기반의 N-Screen 에뮬레이터 설계

프로토타입 클라우드 기반 N-Screen 에뮬레이터의 구현 한계점은 성능과 서비스이다. 프로토타입의 클라우드 기반 N-Screen 에뮬레이터는 기본적으로 웹 서비스를 하기 위해 X11, RFB 같은 VNC 기술을 사용한다. VNC 기술은 기본적으로 시스템의 GUI를 이미지로 만들어 외부로 서비스 하는 기술이기 때문에 성능의 한계점이 도출되고 있다.

또한, VNC 기술에서 사용하는 RFB는 X11에서 출력하는 GUI 이미지를 그대로 서비스를 하기 때문에 한 화면 당 실행할 수 있는 에뮬레이터의 수가 정해져 있다. 또한 실행중인 에뮬레이터 화면에 다른 화면이 겹쳐진다면 사용자에게 잘못된 화면까지 서비스가 된다.

이러한 클라우드 기반 N-Screen 에뮬레이터의 한계점을 해결하기 위해 본 논문에서는 새로운 클라우드 기반 N-Screen 에뮬레이터 설계 방법을 제안한다. 기존 설계 방법과 새로운 설계 방법의 차이점은 다음과 같다. 기존 설계 방법은 Qt로 개발된 에뮬레이터를 VNC 기술을 이용하여 웹 서비스를 제공하는 형태로 어떤 로컬 컴퓨터에서도 실행된다. 하지만, 새로운 설계 방법은 서버에서 에뮬레이터의 실체 없이 웹 서비스를 이용하여 브라우저로 에뮬레이터의 모습을 확인할 수 있도록 한다. 이것은 데스크 탑 혹은 모바일 환경에 구애받지 않고 동일한 서비스가 가능하도록 한다. 그리고 새로운 설계방법은 다양한 디바이스의 각기 다른 웹 브라우저 엔진을 따로 에뮬레이팅 하기 위해 WebKit 엔진을 각 디바이스의 특성에 맞게 수정하고 관리한다. 또한, 기존의 설계 방법에서는 한 화면에 보여줄 수 있는 에뮬레이터 수가 2, 3개로 상당히 제한적 이었지만, 새로운 설계안으로는 한 서버당 100개 이상의 에뮬레이터를 구동시킬 수 있다.

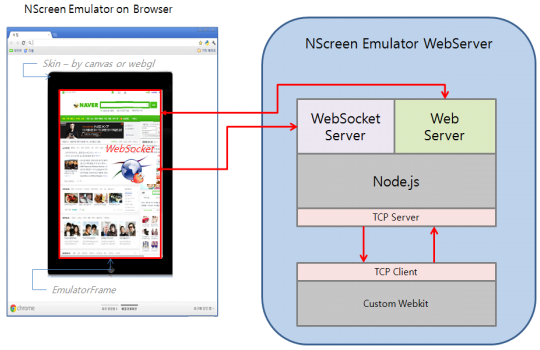



그림 4. 제안하는 클라우드 기반 N-Screen 에뮬레이터 구성도
Fig. 4. Proposed N-Screen emulator architecture based on Cloud.

그림 4 제안하는 클라우드 기반의 N-Screen 에뮬레이터는 Node.js를 적용하여 성능 향상을 목표로 한다. Node.js는 바이너리로 컴파일 되는 많은 코어 모듈과 함께 제공된다[13]. 그것은 네트워크 비동기 래퍼인 네트워크 모듈의 다른 경로 또는 파일 시스템 버퍼 타이머보다 일반적인 스트림과 같은 기본적인 모듈을 포함한다.

### 1) Node.js를 이용한 웹서버

웹서버는 Node.js의 확장 모듈인 express.js를 사용한다. express.js는 Node.js의 이벤트 중심의 동작 특성을 최대한 살려 고성능 웹서버를 구축할 수 있다. express.js의 설치 방법은 "npm install -g express"이다. express.js를 사용한 간단한 웹서버 구축 예는 그림 5와 같다.

```
var app = require('express').createServer();
app.get('/', function(req, res){
  res.send('hello world');
});
app.listen(3000);
```

그림 5. express.js를 사용한 웹서버 구축
Fig. 5. Web server implementation using express.js.

express.js를 사용한 웹서버 구동 방법은 node app.js 이다.

2) Node.js를 이용한 WebSocket 서버

WebSocket 서버는 Node.js의 확장 모듈인 socket.io를 사용한다[14]. socket.io를 이용한 WebSocket 서버도 웹서버와 마찬가지로 고성능인 반면 최소한의 구축 노력만이 필요하다. socket.io의 설치 방법은 npm install -g socket.io 이다. socket.io를 이용한 WebSocket 서버의 구축 예는 그림 6과 같다.

```
var app = require('express').createServer(),
    io = require('socket.io').listen(app);
app.listen(80);
app.get('/', function (req, res) {
  res.sendfile(__dirname + '/index.html');
});
io.sockets.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data) {
    console.log(data);
  });
});
```

그림 6. socket.io를 사용한 웹 소켓 서버 구축
Fig. 6. WebSocket server implementation using socket.io

socket.io를 이용한 WebSocket 클라이언트의 자바스크립트 구축 예는 그림 7과 같다.

```
script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', {my: 'data' });
  });
</script>
```

그림 7. socket.io를 사용한 웹 소켓 클라이언트 구축
Fig. 7. WebSocket client implementation using socket.io.

3) Node.js를 이용한 TCP 서버

새로운 설계방법의 TCP 서버는 WebSocket 서버와 마찬가지로 socket.io를 이용하여 Node.js 확장 모듈을 사용한다. 사용법은 위의 WebSocket 서버 사용방법과 동일하다.

브라우저 클라이언트와 WebSocket 서버와의 데이터 통신을 위해 브라우저 클라이언트와 WebSocket 서버는 에뮬레이터의 화면을 표시하기 위해 특정한 프로토콜이 필요하다. 해당 프로토콜의 간단한 프로토콜 내용은 다음과 같다.

① HELLO : 최초 접속 핸드쉐이킹 메시지.
② PROFILE : 최초 접속후 구동한 디바이스의 정보, 프로파일을 전송
③ RENDER_IMAGE : 커스텀 웹킷의 렌더링 이미지를 전송한다
④ OK : 정상 처리 메세지
⑤ ERR : 에러 메세지, Body에는 에러 내용이 있다.
⑥ BYE : 접속 종료 메시지

커스텀 WebKit과 Node.js와의 데이터 통신을 위해 커스텀 WebKit과 Node.js 서버간에도 렌더링 이미지를 별도 프로세스간 공유를 위해 TCP 소켓을 통해 렌더링 이미지를 공유한다. 이미지를 공유하기 위한 간단한 프로토콜의 내용은 다음과 같다.

① HELLO : 최초 접속 핸드쉐이킹 메시지. 커스텀 웹킷의 프로세스 관리의 기초 정보가 된다.
② PROFILE : 최초 접속후 구동한 디바이스의 정보, 프로파일을 전송
③ RENDER_IMAGE : 커스텀 웹킷의 렌더링 이미지를 전송한다. 이 메시지는 커스텀 웹킷측에서 Node.js측으로 주기적으로 전송한다.
④ OK : 정상 처리 메세지
⑤ ERR : 에러 메시지. Body에는 에러내용이 있다.
⑥ BYE : 접속 종료 메시지.

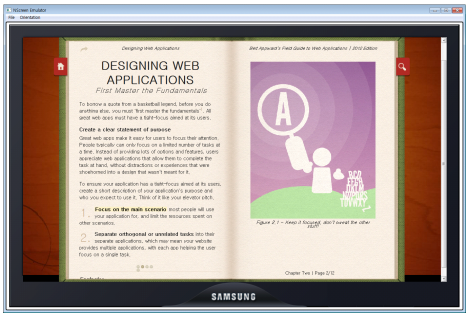
## IV. 클라우드 기반 N-Screen 에뮬레이터 구현

### 1. N-Screen 에뮬레이터

본 논문에서 구현한 프로토타입의 N-Screen 에뮬레이터 실행 결과 화면은 그림 8과 같다.


(a) iPhone(가로모드)


(b) iPad(가로모드)




(c) Smart TV(가로모드)

그림 8. 에뮬레이터 실행 결과
Fig. 8. Emulator execution result.

그림 9는 각 스마트 디바이스 구동 시 화면에 로딩 된 페이지로 N-Screen 에뮬레이터의 화면 해상도 및 디바이스 종류를 자동으로 인식하여 보여주는 eBook 형태의 스마트 콘텐츠이다. N-Screen 에뮬레이터는 화면에 나타나는 콘텐츠의 가로, 세로모드에 따라 화면의 해상도 변화로 인해 자동으로 해상도를 인식하여 스마트 콘텐츠의 모습을 변화시킨다.

## 2. 클라우드 기반 N-Screen 에뮬레이터

본 논문에서 구현한 클라우드 기반의 N-Screen 에뮬레이터를 구동한 화면은 그림 9와 같다.


그림 9. 브라우저로 서비스되는 클라우드 기반 N-Screen 에뮬레이터
Fig. 9. N-Screen emulator serviced by browser.

N-Screen 에뮬레이터는 실제 스마트 디바이스와 마찬가지로 멀티터치를 지원한다. 다만 사용자 PC에서는 터치가 지원이 안 되기 때문에 키보드와 마우스의 특수한 조합으로 가상 멀티터치를 구현하였다.


(a)Panning


(b)Pinching

그림 10. Panning & Pinching
Fig. 10. Panning & Pinching

## V. 결론

본 논문에서는 클라우드 기반 N-Screen 에뮬레이터의 한계점을 해결하기 위해 새로운 클라우드 기반 N-Screen 에뮬레이터 설계 방법을 제안하였다. 기존 설계 방법과 새로운 설계 방법의 차이점은 다음과 같다. 기존 설계 방법은 어떤 로컬 컴퓨터에서라도 구동될 수 있도록 Qt로 개발된 에뮬레이터를 VNC 기술을 이용하여 웹 서비스를 제공하는 형태이다. 하지만, 새로운 설계 방법은 서버에서 에뮬레이터의 실체 없이 웹 서비스를 이용하여 브라우저로 에뮬레이터의 모습을 확

인할 수 있도록 한다. 이것은 개인용 컴퓨터 또는 모바일 환경에 구애받지 않고 동일한 서비스가 가능하도록 한 것이다. 그리고 새로운 설계방법은 다양한 디바이스의 각기 다른 웹 브라우저 엔진을 따로 에뮬레이팅 하기 위해 브라우저 엔진인 WebKit 엔진을 각 디바이스의 특성에 맞게 수정하고 관리한다. 또한, 기존의 설계 방법에서는 한 화면에 보여줄 수 있는 에뮬레이터 수가 2, 3개로 상당히 제한적 이었지만, 새로운 설계방법은 한 서버 당 100개 이상의 에뮬레이터를 구동시킬 수 있도록 성능을 향상시켰다.

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

## 참고문헌


〔1〕 Y. I. Yoon, B. Kim, "N-Screen Service Standardization Based on Platform Type," KSCI Review, Vol. 20, No. 1, pp. 1-9, June 2012.

〔2〕 Won Joo Lee, Jung Pyo Lee, and Min Tae Kim, "N-Screen Emulator Technology for Multi Platform," KSCI Review, Vol. 20, No. 1, pp. 23-30, June 2012.

〔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*


## 저 자 소 개


**이 원 주**

1989: 한양대학교 전자계산학과 공학사.
1991: 한양대학교 컴퓨터공학과 공학석사.
2004: 한양대학교 컴퓨터공학과 공학박사.
현 재: 인하공업전문대학 컴퓨터정보과 교수.
관심분야: 병렬처리시스템, 모바일컴퓨팅, 성능분석, 클라우드컴퓨팅, N-Screen 서비스.
Email: wonjoo2@inhatc.ac.kr

**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
Email: tikilee@kthcorp.com

**윤 용 익**

1983: 동국대학교 통계학과 이학사.
1985: 한국과학기술원 전산공학과 공학석사.
1994: 한한국과학기술원 전산공학과 공학박사
현 재: 숙명여자대학교 멀티미디어과학과 교수
관심분야: 스마트사이니지, 스마트 클라우드 컴퓨팅, 모바일 멀티미디어 시스템, 분산시스템, 실시간 처리시스템, 미들웨어, 실시간 OS/DBMS, 상황인지 서비스, N-Screen 표준화, 모바일 클라우드
Email: yiyoon@sookmyung.ac.kr