웹소켓을 활용한 웹 서비스 이동 연구

최헌회, 김근형*
동의대학교 디지털미디어공학과
*동의대학교 영상정보공학과
e-mail : hunhoi@deu.ac.kr, geunkim@deu.ac.kr*

A Study on the Web Service Migration Based on Websocket

Hun-Hoi Choi, Geun-Hyung Kim*
Dept. of Digital Media Engineering, Dong-Eui University
*Dept. of Visual information Engineering, Dong-Eui University

요 약

본 논문은 Node.js 와 HTML5 의 웹소켓(Websocket)을 활용하여 사용자가 브라우저를 통해 이용 중인 웹 서비스의 콘텐츠를 다른 단말로 이동하는 구조와 알고리즘을 서술한다. 제안된 구조와 알고리즘을 통해 다양한 플랫폼과 단말 간 서비스가 이동됨을 알 수 있다. 기존의 폴링(Polling)방식을 사용한 웹 서비스 이동과 본 논문에서 제시한 알고리즘의 실험 결과를 통해 제시한 알고리즘의 성능이 우수함을 보였다. 사용자 단말의 트래픽은 5~13 배 가량 줄었으며, 응답 시간은 폴링방식에 비해 빠른 응답시간을 나타냈다. 실험에서는 자바스크립트를 사용하여 확장성이 높은 Node.js 와 HTML5 의 새로운 표준인 웹소켓을 활용하였으며, 제시한 구조와 알고리즘이 앞으로 웹 서비스 이동 시 다양한 단말 간 서비스 이동에 효과적으로 전달될 수 있음을 보였다.

1. 서론

무선 네트워크의 대역폭 증가와 스마트폰의 보급 등 급변하는 기술의 발전에 따라 언제 어디서나 웹 서비스를 이용하고자 하는 욕구가 증가하고 있다. 이러한 유비쿼터스 환경에서 기존 단말에서 이용 중인 서비스를 다른 단말로 이동하는 서비스의 연속성 제공을 위해 다양한 연구가 이루어졌다.

앞으로 다양한 종류의 단말 출시와 더불어 새로운 단말을 이용한 서비스 이용은 더욱 증가할 것이며, 현재 단말마다 웹 서비스를 그대로 적용하기 어렵기 때문에 단말의 디스플레이 크기와 접속 네트워크 특성을 고려하여 단말 특징에 맞는 서비스를 제공하는 방법이 필요하다. 이와 같은 동적인 표현과 뛰어난 상호작용이 요구되었고 이 때문에 Flash[1], Java Applet[2], ActiveX[3], Silverlight[4] 등이 개발되었다. 그러나 폴링과 롱폴링(long-polling)방식은 사용자가 서버에 일정 시간 간격으로 재요청 메시지를 전달하는 비효율적인 기술을 사용한다. 폴링은 클라이언트가 서버에 주기적으로 이벤트가 발생했는지 요청하면 서버는 응답하며 이벤트가 발생하면 응답과 동시에 데이터를 전달받는 방식이다. 롱폴링은 폴링을 개선한 방법으로 클라이언트가 서버에 요청하면 서버는 바로 응답하지 않고 이벤트가 발생할 때까지 기다린 후 응답하는 구조이다. 이러한 기술들은 순수 웹 환경이 아니라 별도의 런타임 플러그인(plug-in)형태로 브라우저에 설치해야 사용 가능한 기술이다.

HTML5 에서는 플러그인 없이 일관되고 표준화된 웹 응용 환경을 위한 웹소켓 [5]을 개발하여 채팅, 게임, 주식 차트와 같은 실시간이 요구되는 응용 프로그램의 개발을 한층 효과적으로 구현할 수 있도록 하였다. 웹소켓은 HTML5 표준안에 추가된 API 로 웹 상에서 양 방향 통신을 구현하기 위한 네트워크용 통신 규약이다. 이는 기존 Ajax, Comet 과 같은 폴링 방식에서 탈피한 순수한 웹 환경에서의 실시간 양방향 통신 기능을 제공한다. 하지만, HTML5 의 다른 기능과 달리 클라이언트의 코드만으로 실행되지 않기 때문에 브라우저뿐만 아니라 웹 서버도 기술을 지원해야 한다. Node.js[6]는 서버 측 작업을 자바스크립트 특성에 따라 스레드 기반으로 처리할 수 있어, 제한된 프로세서 환경에서 실시간 애플리케이션의 성능과 속도를 향상할 수 있다.

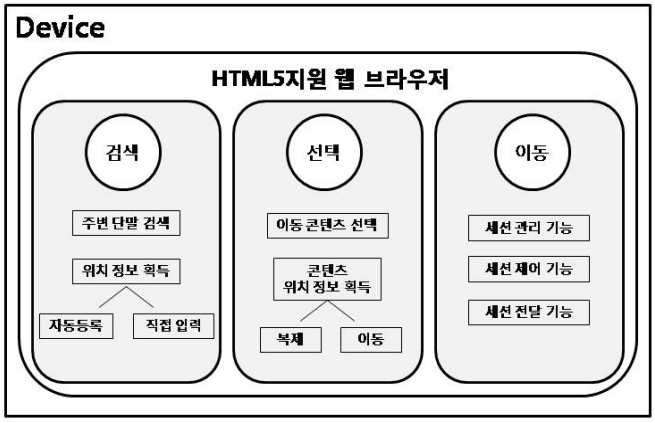
본 논문의 구성은 다음과 같다. 2 절은 본 논문에서 제시한 웹 서비스 이동을 위한 플랫폼 구조를 설명하고, 3 절은 기존 폴링 방식의 웹 서비스 이동과 본 논문에서 제시한 알고리즘의 이동 실험 결과를 설명 하며, 4 절에서 결론을 맺는다.

2. 웹 서비스 이동을 위한 플랫폼 구조

DLNA(Digtal Living Network Alliance)[7]와 3GPP(3rd Generation Partnership Project)[8]에서 단말 간 서비스 이동에 대한 관련 연구 및 표준화를 진행하고 있다. DLNA 사용자가 댁내에서 DLNA 호환 단말을 이용하여 단말간 서비스를 이동할 수 있는 기술을 제공하고 있으며, 3GPP 는 주로 모바일 단말 간 서비스이동 기

능을 표준화하고 있다.

제안한 서비스 이동을 위한 기능은 그림 1 과 같다. 3 가지 기능으로 단말 검색, 이동 콘텐츠 선택, 콘텐츠 이동 기능이 있다. 단말 검색 기능은 사용자가 콘텐츠를 전달하기 위해 대상 단말을 찾는 기능이다. 이때 각 단말은 자신의 단말에 대한 정보를 데이터베이스에 저장한다. 저장 내용은 단말의 종류, 위치, 사용자 ID, 제품 식별 번호를 저장한다. 콘텐츠 선택 기능은 웹 상에서 사용자가 이동 하려는 콘텐츠를 선택하고 분리하는 기능이다. 이동하고자 하는 콘텐츠를 선택한 뒤 전달받을 단말에 복제 또는 이동하는 기능을 수행한다. 콘텐츠 이동 기능은 앞서 선택된 콘텐츠의 전송과 그 외 부가적인 기능을 한다. 이동 전에는 현재 연결된 단말의 상태, 재생 정보, 전달 방법에 대한 정보를 제공하고, 이동 후에는 전달한 콘텐츠의 컨트롤 권한 설정에 대한 정보를 제공한다.

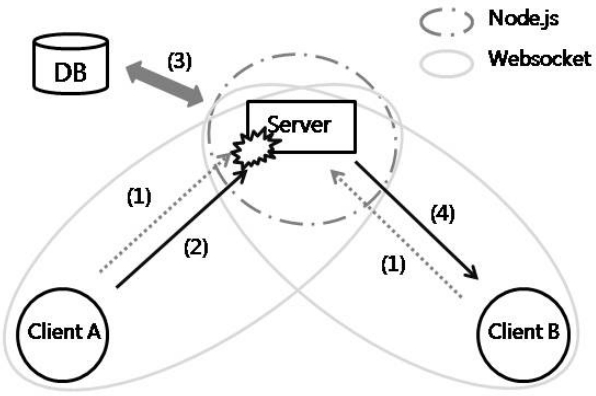


(그림 1) 서비스 이동에 필요한 기능.

실제 구현한 시스템의 구조는 그림 2 와 같다. Node.js 는 사용자가 전송한 메시지를 해석하여 데이터베이스에서 전달 받은 사용자를 검색 후 전달한다. 이때 전달 받을 사용자가 오프라인이면 데이터베이스에 저장하여 사용자가 온라인 상태가 되면 알려주는 기능을 한다. 웹소켓은 서버와 사용자 사이에서 입력한 메시지를 전달하는 역할을 한다. 서비스 이동의 흐름은 (1)번에서 각 사용자는 서버를 통해 자신의 단말 정보를 데이터베이스에 저장하고, 클라이언트 A 는 (2)번 메시지를 통해 서버에 이동을 요청한다. 서버는 (3)번에서 사용자를 검색하여 (4)번 메시지를 통해 클라이언트 B 에게 메시지를 전달한다.

<표 1> 브라우저별 테스트 결과.

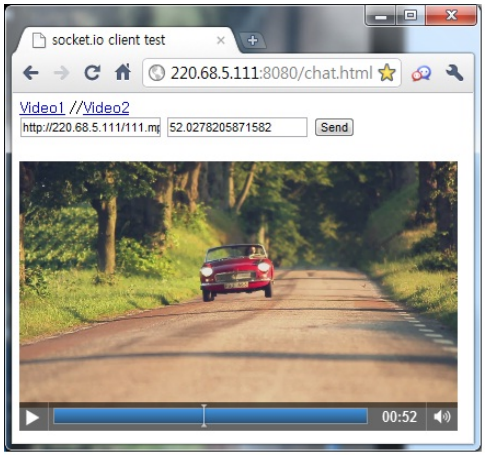
플랫폼	브라우저	작동 여부
PC	Internet Explorer 9	O
	Chrome 10	O
	Firefox 3.6	O
	Safari 5.04	O
	Opera 11.01	X
스마트폰	iOS Safari 4.2	O
	Android Browser 2.3	X



(그림 2) 웹 서비스 이동 구조.

PC 에서의 브라우저별 테스트 결과는 표 1 과 같다. 오페라를 제외한 모든 브라우저에서 상호간 서비스 이동이 됨을 확인했다. 모바일 단말은 애플의 사파리에서만 웹소켓을 지원하고 있다.

그림 3 은 이종 단말간 서비스 이동 모습을 나타낸다. 실험은 PC 와 스마트폰에서 보던 비디오를 상호간 이동하는 것을 실험하였다. 실험을 통해 PC 와 스마트폰과 같이 이종 단말 간 외에도 동종 단말 간 에서도 서비스 이동이 가능함을 알 수 있었다.



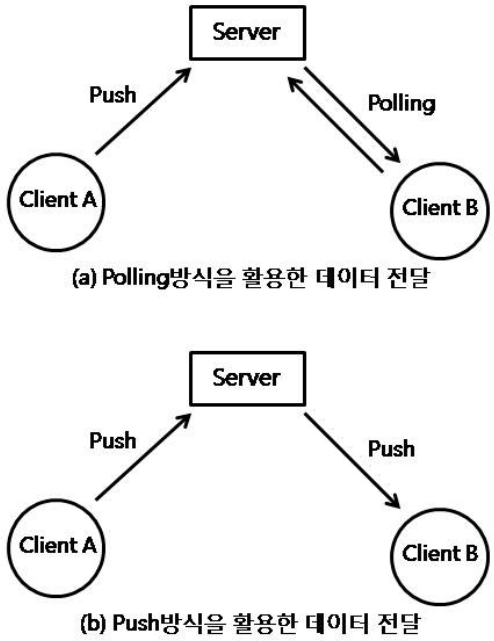
(a) Google 의 Chrome 브라우저에서 비디오재생 중인 모습

(b) (a)에서 보던 비디오를 스마트폰으로 전송하여 같은 시점을 재생중인 모습

(그림 3) 이종 단말 간 서비스 이동 모습.

3. 기존 시스템과 비교

기존의 폴링 방식[9]은 구현이 간단하고 응답 및 요청이 중간에 끊어져도 새로운 요청이 생성되기 때문에 연결 유지가 쉽지만 일정 시간 간격으로 서버에 응답 및 요청을 하기 때문에 트래픽의 증가와 응답시간 지연과 같은 문제점이 발생한다. 그림 3 은 폴링과 푸쉬(Push) 방식의 구조 비교를 나타낸다. 폴링 방식은 서비스를 전달받는 사용자가 서버에게 반복적인 요청을 하고, 푸쉬 방식은 서버에서 사용자에게 즉시 전해 주는 구조이다. 최근 널리 보급된 스마트폰, 태블릿 PC 와 같은 무선 네트워크를 사용하는 단말에서 폴링방식을 사용하게 되면 단말 성능과 네트워크 연결속도가 PC 에 비해 낮으므로 불필요한 대역폭 낭비 및 성능저하와 같은 부작용을 초래할 수 있다. 반면 푸쉬 방식은 이벤트가 발생했을 때 즉시 서버에서 사용자에게 응답 및 데이터를 보내는 구조이며, 폴링 방식에 비해 사용자 단말에서 소요되는 트래픽이 적고 메시지가 서버에서 지연되는 시간 없이 빠른 응답 시간을 나타낸다.



(그림 3) 폴링과 푸쉬 방식의 구조 비교.

시간 간격에 따른 폴링 방식과 푸쉬 방식에 대한 단말의 트래픽과 평균 응답시간은 다음 표 2 와 같다. 소요 트래픽은 사용자가 재요청하는 페이지의 크기와 재요청 횟수를 계산한 결과이고, 평균 응답시간은 폴링 주기에 따른 평균 응답시간을 계산한 결과이다. 폴링 방식에서 시간간격에 따른 트래픽과 평균 응답시간은 반비례하며, push 방식은 polling 방식에 비해 트래픽 사용량과 응답시간이 적게 소요된다.

<표 2> 폴링과 푸쉬 방식의 비교표.

방식	폴링		푸쉬
시간 간격(초)	3	10	-
소요 트래픽 (Byte/분)	65,090	25,232	4663
평균 응답 시간(초)	1.5	5	즉시

4. 결론

본 논문에서는 HTML5 의 웹소켓과 Node.js 를 활용하여 사용자가 받고 있는 웹 서비스 중 일부 콘텐츠를 다른 단말로 이동하여 서비스 연속성이 보장됨을 보였다. 또한, 제시한 알고리즘을 통해 기존 폴링으로 구현한 시스템에 비해 효율적으로 서비스가 이동됨을 보였다. 제안한 구조의 소요 트래픽은 5~13 배 가량 줄었으며, 평균 응답 시간은 폴링 방식이 시간 간격에 따라 응답시간이 1~5 초 걸리는데 비해 이동 요청과 동시에 지연 없는 즉각적인 응답을 나타냈다. 앞으로 연구 진행 방향은 실험에서 이용한 비디오뿐만 아니라 웹 페이지 내 존재하는 사진, 문서, 음악과 같이 다양한 객체를 이동하는 연구를 진행할 것이며, 이와 더불어 서비스 이동 시 모든 단말에서 같은 UI(User Interface)와 UX(User eXperience)를 지원하기 위한 연구를 병행할 것이다.

참고문헌

[1] Adobe, Flex3 “Network and communication”.
[2] SUN, “JavaFX 1.3.1 Overview”.
[3] Microsoft, “Introduction to active controls”.
[4] Microsoft, “Silverlight overview”.
[5] W3C, HTML5 “The Websocket API”, 28 February 2011
[6] Rtan Danl, “Node.js Manual & Documentation”.
[7] DLNA, http://www.dlna.org.
[8] 3GPP, “Transparent end–to-end Packet-switched Streaming Service (PSS) - Protocols and codecs (Release 9),” March 2010.
[9] 최헌회, 최익성, 김근형, “HTML5 기반 HTTP 스트리밍 환경에서의 서비스 이동성 연구,” 한국멀티미디어학회 논문 제출.