WebSocket 기술동향 분석 및 활용 방안

박우람*, 박석천**, 김두영***
*가천대학교 일반대학원 모바일소프트웨어학과
**가천대학교 컴퓨터공학과 정교수(교신저자)
***(주)메타빌드
e-mail : crzsnail@gmail.com

Analysis of Technical Trend and Utilization Method of WebSocket

Uram Park*, Seok-Cheon Park**, Do-Young Kim***
*Dept of Mobile Software, Gachon University
**Dept of Computer Engineering, Gachon University(Corresponding Author)
***Ltd. MetaBuild

요 약

최근 웹 서비스는 실시간 양방향 통신으로 높은 효율성과 전이중 통신이 주요 이슈가 되고 있으며, 이는 웹소켓의 활용성이 증대되는 흐름으로 이어지고 있다. 기존 데이터 통신은 반이중 통신으로 과도한 트랙픽과 불필요한 오버헤드를 발생시키는 단점이 있었다. 그러나 웹소켓은 반이중 통신의 단점을 해소하고, HTML5의 주요 기술중 하나로 웹표준은 물론, 표준 프로토콜로서의 확장성 등의 장점을 가지고 있다. 이에 따라 본논문에서는 기존 웹 기반 서비스인 실시간 통신, 제한된 하드웨어 플랫폼 서비스, 미디어 콘텐츠와 같은 대용량 서비스 제공 등에 웹소켓의 활용 방안에 대해 제안한다.

Ⅰ. 서론

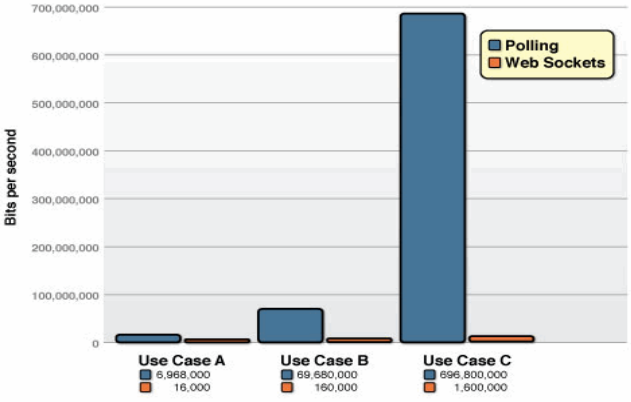
초기 웹의 탄생 목적은 문서 전달과 하이퍼링크를 통한 문서 연결이었다. 웹을 위한 HTTP는 이러한 목적에 매우 부합하는 모델이다. 사용자 문서가 위치한 서버에 URL을 통해 문서를 요청하면 서버는 웹페이지나 이미지 파일을 응답 하였다. 그러나 웹이 발전할수록 문서공유 이외에 동적인 표현과 뛰어난 상호동작을 요구하게 되었고, 이를 위해 Flash, Java Applet, ActiveX, Silverlight등이 개발 되었다. 하지만 이러한 기술들은 웹에서 화려한 동작과 뛰어난 상호작용을 보장하지만 순수 웹 환경이 아니라 별도의 런타임 플러그인 형태로 브라우저에 설치해야 사용 가능하다.

그러한 이유로 플러그인 없는 일관되고 표준화된 웹 응용 환경이라는 가치하에 순수 웹 환경에서 실시간 양방향 통신을 위한 여러 스펙이 개발 되었다. 폴링 기술을 사용한 DHTML, iframe, Ajax 와 롱폴링 기술을 사용한 Comet 과 같은 기술들이 등장 하였다. 또한 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.

웹 소켓은 폴링 방식과 비교했을 때 클라이언트가 서버에 데이터를 요청하고, 서버가 브라우저에 데이터를 응답하는 것이 별다른 제약 없이 동기식으로 이루어지므로 클라이언트와 서버 사이에 지연시간을 낮출수 있다.

Polling 방식과 WebSockt 방식의 네트워크 오버헤드를 비교하면 (그림1.1) 과 같다. 네트워크 오버헤드의 트래픽, 불필요한 오버헤드 등을 비교 했을 때 웹소켓이 폴링 방식보다 효율성이 뛰어나다는 것을 알 수 있다[3].



(그림 1.1) Polling방식과 WebScoket방식의 네트워크 오버헤드 비교

W3C에서 HTML의 차기 주요 버전으로 HTML5를 채택하였다. 이는 데이터 통신을 위해 웹소켓을 사용가능성이

높아진 것을 의미하고, 웹 표준화를 위해서 비표준 인터넷 웹 환경인 ActiveX, Flash, Sliverlight 등의 별도 프로그램 설치하여 제공되었던 멀티미디어와 확장 기능들을 HTML5 기술이 대체하고, 개방형 웹 서비스 제공을 위해 플랫폼간의 의존도를 감소와 시킬뿐 아니라 PC, 스마트폰, 태블릿 PC 등 기존 기기와 상관없이 서비스를 통해 사용자에게 편의성을 제공할 수 있다.

본 논문의 구성은 1장 서론에 이어 2장에서 웹소켓에 대해 기술하고, 3장에서 기술 동향 분석 후, 4장에서 웹 소켓 활용 방안을 제안하고, 5장에서 결론으로 맺는다.

Ⅱ. 관련 연구

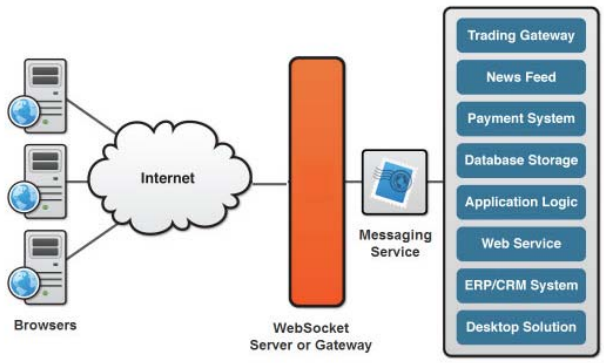
2.1 WebSocket 소개

웹소켓은 컴퓨터 네트워크용 통신 규약의 하나이다. 웹소켓은 단일 TCP 접속을 통하여 전이중 방식으로 브라우저와 지속적으로 연결되어 실시간 데이터를 주고 받을 수 있는 HTML5에서 제공하는 API이다. 인터넷 표준화 단체인 W3C(World Wide Web Consortium)와 IETF가 웹서버와 웹브라우저 간의 통신을 위한 규정을 정의한 양방향 통신 기술규약으로써 API는 W3C에서 책정[1]을 맡고 있고, 웹소켓 프로토콜은 IETF가 책정을 맡아 RFC 6455 표준[2]으로 지정하였다.

2.2 WebSocket 프로토콜

웹소켓은 TCP 소켓 통신처럼 전이중 통신을 제공하지만 바이트스트림 메시지 흐름을 가능하다는 점에서 TCP와 차이점을 보인다. 웹소켓 이전에는 80 포트를 사용하는 Comet 채널을 통해 전이중 통신을 사용하였다. 그러나 TCP 핸드셰이크 및 HTTP 헤더의 용량이 메시지의 용량에 비해 비효율적이다[5].

(그림2.2)는 WebSocket 프로토콜 아키텍처를 나타낸다. 그림을 살펴보면 클라이언트가 웹브라우저 통해 웹소켓 서버에 데이터를 요청하면 웹소켓은 백엔드에서 제공하는 다양한 서비스에 소켓 연결을 통해 클라이언트에게 제공한다.



(그림 2.2) 웹 소켓 프로토콜의 아키텍처

웹소켓을 구현하기 위해서는 일반적인 TCP 소켓 통신처럼 서버와 클라이언트간 데이터 교환이 이루어지는 형태이다. 따라서 다른 HTML5 스펙과는 달리 클라이언트 코드만으로 실행이 불가 하다. 클라이언트에서는 웹소켓이 제공하는 자바스크립트 API를 이용해 서버에 연결하고 데이터를 송/수신하는 코드를 구현해야 하며 서버에서는 웹소켓 프로토콜에 맞는 전용 장치가 구축되어 있어야 한다.

웹소켓 서버를 위한 Java, JavaScript, PHP, Python 등 다양한 언어로 오픈소스가 온라인에서 제공 되고 있으며, 주요 구현 서버로는 jWebSocket , Socket.IO-node 등이 있다.

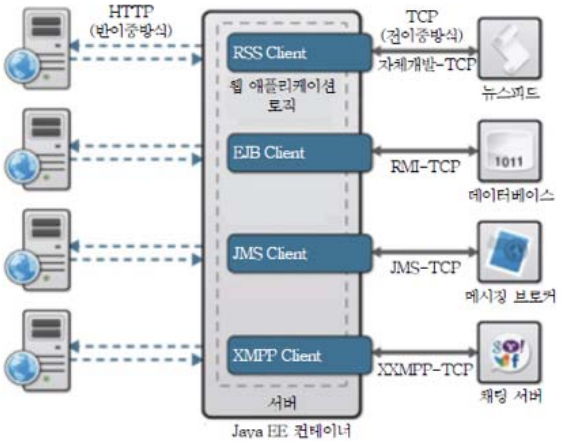
Ⅲ. WebSocket 기술 동향 분석

3.1 HTTP

HTTP(HyperTextTransferProtocol)는 World Wide Web 상에서 정보를 주고 받을 수 있는 프로토콜이며, 주로 HTML 문서를 주고 받는 데에 사용된다. TCP와 UDP를 사용하며, 80번 포트를 사용한다.

HTTP는 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜이다. 예를 들면, 클라이언트인 웹 브라우저가 HTTP를 통하여 서버로부터 웹 페이지나 그림 정보를 요청하면, 서버는 이 요청에 응답하여 필요한 정보를 해당 사용자에게 전달하게 된다. 이 정보가 모니터와 같은 출력 장치를 통해 사용자에게 나타나는 것이다. HTTP를 통해 전달되는 자료는 http: 로 시작하는 URL(인터넷 주소)로 조회 할 수 있다[4].

(그림2.1)은 HTTP와 TCP의 서버와 정보교환시 사용되는 데이터의 종류를 비교한 내용이다. 웹 브라우저가 단지 HTTP만 사용하는 것에 비해 백엔드 서버는 TCP, JMS, XXMPP 등 다양한 프로토콜을 사용할 수 있으므로, 지금까지는 양자의 사이(미들웨어)에 모든 메시지를 변환, 해석, 재포맷하는 다양한 애플리케이션 서버가 필요 하다.



(그림2.1) HTTP와 TCP 데이터 통신 비교

3.2 Polling / Long-Polling 구현기술

iframe, Ajax, Comet 등 Polling 방식의 구현기술은 WebSocket이 등장하기 클라이언트에게 실시간 통신과 문

서공유 외의 웹서비스를 제공하기 위한 방안이었다. Polling 기술 중 하나인 Ajax의 경우는 HTTP Request/ Response 방식을 사용하여 클라이언트에게 실시간 통신과 같은 서비스를 제공하기 위해 클라이언트가 서버에게 주기적으로 데이터 요청을 통해 응답을 주고 받는 방식이다. 그러나 Polling의 주기가 짧을수록 서버의 성능에 부담을 주고 과도한 트래픽을 발생 시킨다.

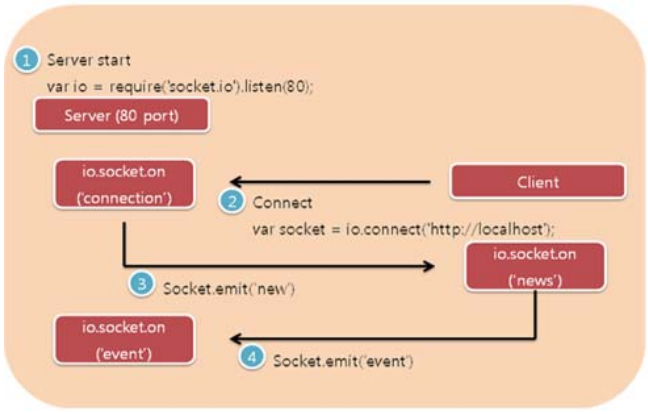
이후 Comet 방식은 Ajax의 문제점을 해결하고자 클라이언트가 요청 후에는 서버는 접속을 유지하다가 이벤트가 발생하면 데이터를 클라이언트에게 보내고 세션을 종료하는 방식을 사용한다. 클라이언트는 응답 받은 데이터를 종료하고 다시 서버에 접속하는 방식을 반복한다.

그러나 이러한 Polling, Long-Polling 방식은 기존의 HTTP방식의 한계점을 벗어나지 못한다는 점에서 불필요한 트래픽 발생을 야기한다.

3.3 WebSocket 구현기술

웹소켓을 구현한 기술로는 첫째, Socket IO가 있다. Socket IO는 자바스크립트를 이용하여 브라우저 종류에 상관 없이 실시간 웹을 구현할 수 있도록 한 기술이다.

(그림3.1)에서 보면 서버에서는 클라이언트의 요청을 받고 emit이라는 함수를 사용하여 클라이언트로 다시 결과값(data)를 반환하면 클라이언트는 on 함수를 통해 그 결과를 받고 다시 서버쪽으로 emit 함수로 다시 결과를 주면 서버는 on 함수를 통해 요청을 받아들인다. 이와 같이 Socket IO를 사용하면 지속적인 요청이 아니라 이벤트 발생시에만 서버와 클라이언트가 동작하여 서버의 과부하를 줄이면서 실시간 데이터의 신뢰도를 높이게 된다.



(그림3.1) 소켓 IO 데이터 플로우

둘째, jWebSocket은 웹에서 양방향 고속 통신을 위한 오픈소스 솔루션이며, TCP 소켓 기술과 HTML5 웹 소켓을 기반으로한 혁신적인 스트리밍 및 통신 응용 프로그램을 만들 수 있다. jWebSocket은 W3C와 IETF 플록시 및 방화벽에 상호작용이 가능 하며 크로스 플랫폼 브라우저, 모바일 및 웹 애플리케이션 등 포괄적인 서비스 제공이 가능하다. jWebSocket은 WebSocket Server, WebSocket Client, Client, Flash Bridge 모듈을 이용해서 서버와 클라이언트간의 데이터 통신과 다양한 플랫폼에 사용가능 하다.

웹소켓 구현 기술은 위의 두가지 외에도 php, ruby, c++ 등 다양한 언어로 개발되어 있고, 모바일에서도 웹소켓 구현 오픈소스가 제공되고 있다.

결론적으로 웹 기반 서비스의 확장성과 데이터 전송 효율성을 향상시킬 수 있는 웹소켓을 활용한다면 효율적인 데이터 전송 서비스 제공과 기존 다양한 언어로 개발된 서비스와의 접목적 측면에서 활용가치가 높다는 것을 확인할 수 있다.

Ⅳ. WebSocket 활용 방안 제시

웹소켓의 사용 가능 분야는 웹을 기반으로 하는 서비스 애플리케이션에 적용하는 것이 주요할 것으로 판단된다.

실시간 웹서비스를 제공해야 되는 분야인 스마트 TV, 미디어 플레이 등의 시각적 통신을 하는 곳에서도 활용하면 좋다. 기존 플러그인 형태의 웹서비스를 지양하고 HTML5, 자바스크립트를 활용해서도 충분히 기존 서비스보다 낮은 데이터 전송으로 동일한 서비스 제공이 가능하기 때문이다. 이때 폴링 방식이 아닌 웹소켓을 활용한다면 클라이언트측에서 데이터 통신 중단 현상을 획기적으로 줄일 수 있으며, 양질이 좋은 서비스가 가능하다.

모바일, 웨어러블 컴퓨팅과 같은 하드웨어의 제한성을 가지는 플랫폼에서 웹서비스를 제공할 때도 웹소켓을 활용한다면 무선 트래픽의 감소 효과와 효율적인 데이터 전송이 가능할 것으로 판단된다.

실시간 예약 시스템, 통계 시스템, 분석시스템 등 클라이언트와 서버가 지속적으로 상호작용해야하는 분야에서도 웹소켓을 활용한다면 기존 통신방식에서 문제가 되었던 적은 메시지 데이터 통신에서 발생하는 과도한 오버헤드와 대용량 데이터 전송시 데이터의 유실 등에서도 오류를 획기적으로 줄일 수 있을 것이다.

본 논문에서 제안하는 내용은 크게 웹기반의 애플리케이션, 실시간 대용량 데이터 전송, 제한된 하드웨어 지속적 데이터 통신에서 양방향 통신이 가능한 웹소켓의 활용 방안에 대해 제시하였다. 차세대 플랫폼에서도 웹서비스는 지속적으로 이루어지고, 차세대 플랫폼간의 서비스에 대한 대안방법으로 웹이라는 유연한 기술을 웹소켓을 접목하여 지속적으로 활용한다면 본 논문에서 제안하는 방안이외에 타분야에서도 웹소켓 도입이 가능할 것이다.

Ⅵ. 결 론

웹 소켓의 등장 배경은 반이중 통신 방법으로 실시간 서비스를 제공하는데 있어 데이터 낭비가 대두되는 상황에서 데이터 효율성이 높은 기술을 도입하는데에 있다.

기존 웹 서비스에서는 서버측 푸시 기술을 Pilling / Long-Polling 기술로 지원해 주었지만 이는 완전한 전이중 통신의 기술이 아니어서 클라이언트의 요청을 통해야

만 서버에게 데이터를 전송 받는다. 따라서 불필요한 헤더와 높은 트래픽을 야기하는 단점을 지니고 있다.

이러한 문제 해결을 위해 HTML5의 웹소켓 기술이 필요하며, 이는 기술적으로 전이중 양방향 통신의 단일 소켓 연결 형식이다. 따라서 웹소켓을 사용하면 클라이언트는 한번의 요청만으로도 서버에게 지속적인 정보를 푸시 받을 수 있으며, 데이터의 종류와 상관 없이 주기적으로 필요한 데이터를 제공 받는다. 웹소켓의 장점은 실시간 통신의 높은 효율성과, 표준 프로토콜을 사용하는 점, HTML5의 기능, 웹소켓 API를 제공한다는 것이다.

기존 HTTP, Polling, Long-Polling 방식을 가지고 이용되었던 메시지 전송, 파일 전송, 미디어 콘텐츠 전송 등 웹 기반 서비스를 웹소켓 프로토콜을 활용하여 XMPP, STOMP 와 같은 상위 프로토콜 개념을 도입하여 활용한다면 웹 서비스를 다양한 분야에 접목시켜 서비스 제공이 가능하다.

HTML5의 웹소켓 기술을 주요 웹브라우저와 모바일 브라우저에서 제공하고, 최근 모든 플랫폼이 웹 기반 서비스를 지향한다는 점은 다양한 서비스를 제공할 것이라고 예상된다. 이에 따라 웹소켓 기술을 도입 한다면 실시간 데이터 통신과 다양한 데이터 서비스 제공시 높은 효율성과 간결한 송수신이 가능할 것이다.

사사의 글

본 연구는 2013년도 지식 경제부의 SW전문인력양성사업의 재원으로 정보통신산업진흥원의 고용계약형 SW석사과정 지원사업(HB301-13-1003)으로부터 지원받아 수행되었습니다.

참고문헌

[1] The WebSocket API Editor's Draft,
http://dev.w3.org/html5/websockets/.

[2] The WebSocket Protocol RFC 6455
http://datatracker.ietf.org/doc/rfc6455/

[3] HTML5 Web Sockets:
A Quantum Leap in Scalability for the Web
http://www.websocket.org/quantum.html

[4] 위키백과 Hypertext Transfer Protocol
http://ko.wikipedia.org/wiki/HTTP

[5] "HTML5 Web Sockets: A Quantum Leap in Scalability for the Web", Available: www.websocket.org/