Arquitectura de Software con websocket para aplicaciones web multiplataforma

Luis Vivas, Horacio Muñoz Abbate, Mauro Cambarieri,
Nicolás García Martínez, Marcelo Petroff

Laboratorio de Informática Aplicada - Universidad Nacional de Río Negro
{lvivas, hmunoz, mcambarieri, ngarcia, mpetroff}@unrn.edu.ar

Resumen. Este trabajo presenta una arquitectura de software para aplicaciones que requieran comunicación en tiempo real y bidireccional entre servidor y clientes. Websocket aporta un cambio sustancial en las aplicaciones que necesitan ser notificadas por el servidor. Se explica el aporte de Html5 y la estandarización de Websocket además del uso en distintos dispositivos móviles y limitaciones. Al final del trabajo se presenta una implementación de dicha arquitectura en un sistema de voto legislativo en la provincia de Río Negro.

Abstract. This paper presents a software architecture for applications requiring two-way communication in real time between server and clients. Websocket provides a substantial change in the applications that need to be notified by the server. Html5 and the contribution of standardization addition WebSocket use different mobile devices and limitations are explained. At the end of the paper an implementation of this architecture is presented in a system of legislative vote in the province of Río Negro.

Keywords: móviles, multiplataforma, software libre, javaEE, HTML5, websocket, JSR-356

1 Introducción

En la actualidad las aplicaciones web avanzan en todo tipo de proyectos. El aporte de Html5 mejora la usabilidad de las aplicaciones web, permitiendo lograr aplicaciones capaces de reemplazar las tradicionales aplicaciones de escritorio.

Una nueva forma de comunicación entre servidores y clientes permiten obtener mejores resultados en nuestras aplicaciones web.

La necesidad de tener una comunicación en tiempo real entre servidor y clientes cada día es mayor. Podemos citar como ejemplo juegos online multijugador,

donde cada acción de cada cliente tiene que ser replicada a todos los participantes del juegos al mismo tiempo.

El Protocolo de WebSocket [1] permite la comunicación de dos vías entre un cliente a un servidor remoto. El protocolo consiste en lo que se define como un apretón de manos (en ingles handshake) donde se genera el enlace, seguido de mensajes básicos a través de la capa TCP.

La principal motivación de este trabajo es, ¿Cómo implementar websocket en una arquitectura Java Empresarial (Java EE - Enterprise Edition) [2]?.

También definimos como objetivo primario el de comprender y analizar en detalle de la especificación y el estándar de websocket sobre la arquitectura antes mencionada.

Nuestra hipótesis parte de que websocket es la mejor forma de conseguir una aplicación web con comunicación bidireccional entre servidor y clientes, además posee soporte en gran diversidad en dispositivos móviles.

2 Conceptos Generales

2.1 Paradigma request/response

El paradigma request/response (petición/respuesta) [3] es la base de cualquier aplicación web. Un cliente genera una petición de una página o recurso y el servidor le responde con los datos. Sobre esta base se desarrolla cualquier arquitectura web.

En el 2005 con la ayuda de AJAX[1] [4], las aplicaciones web comienzan a mejorar la usabilidad y eficiencia. Esta tecnología provee peticiones parciales, lo que permite no consultar toda la página, sino fragmentos de ella, optimizando la carga en la red. Aun así, todas las peticiones las solicita el cliente a través de HTTP. Sin una petición previa el servidor no puede enviar una respuesta a los clientes.

Existen tecnologías que permiten que el servidor envíe datos al cliente al detectar algún cambio, como podría ser una notificación. A continuación se presentan dos técnicas que resuelven este requerimiento:

- **AJAX Sondeo (polling)**: Consiste en realizar constantemente peticiones al servidor preguntándole si se ha producido algún evento que requiera una actualización en la vista. En un chat, desde la pantalla donde llegan los posibles mensajes que recibe el usuario por parte de otros usuarios, se estarían enviando peticiones HTTP al servidor preguntándole si tengo mensajes.
- **Comet Sondeo largo** (**long polling**) : Consiste en realizar una única petición al servidor de forma que éste responde diciendo que va a devolver la respuesta en trozos. La petición queda abierta hasta que el servidor responda con todas las porciones de respuesta que solicita el cliente, que no son otra

[1] Ajax es el acrónimo de asincronico javascript y XML

cosa que eventos que se producen en el servidor notificando un cambio de estado en el cliente.

Estas soluciones tienen el problema de generar muchas peticiones HTTP, por los cual cuando aumenta el número de clientes aumenta rápidamente el tráfico de red. Además de tener una alta latencia entre que se genera el nuevo evento y se notifica al cliente. Para una aplicación que requiera notificaciones en tiempo real estas soluciones son inviables.

2.2 Websocket

En 2011, la IETF (Grupo de trabajo en ingeniería de internet - The Internet Engineering Task Force) [5] estandarizó el protocolo RFC[2]-6455 [6] referido al protocolo websocket. Desde entonces, la mayoría de los navegadores web implementaron clientes que soportan este protocolo. También se desarrollaron librerías java para implementar websocket desde otros clientes como por ejemplo aplicaciones Android[3] o JavaFX[4].

Una solución al paradigma antes mencionado es utilizar una única conexión TCP para el tráfico bidireccional, esto es el protocolo websocket.

Websocket define un interfaz de programación de aplicaciones (Application Programming Interface - API) que establece conexiones socket entre un servidor y navegador web. Permite una conexión persistente entre el cliente y el servidor, y ambas partes pueden empezar a enviar mensajes en cualquier momento. En la figura 1 se puede ver una comparativa con Ajax Sondeo que se explicó anteriormente.

---

[2] RFC significa petición de comentarios del inglés Request For Comments

[3] Android es un Sistema Operativo, basado en Linux, optimizado para dispositivos móviles

[4] JavaFx herramientas para desarrollo de aplicaciones Java con enriquecimiento en su interfaz gráfica

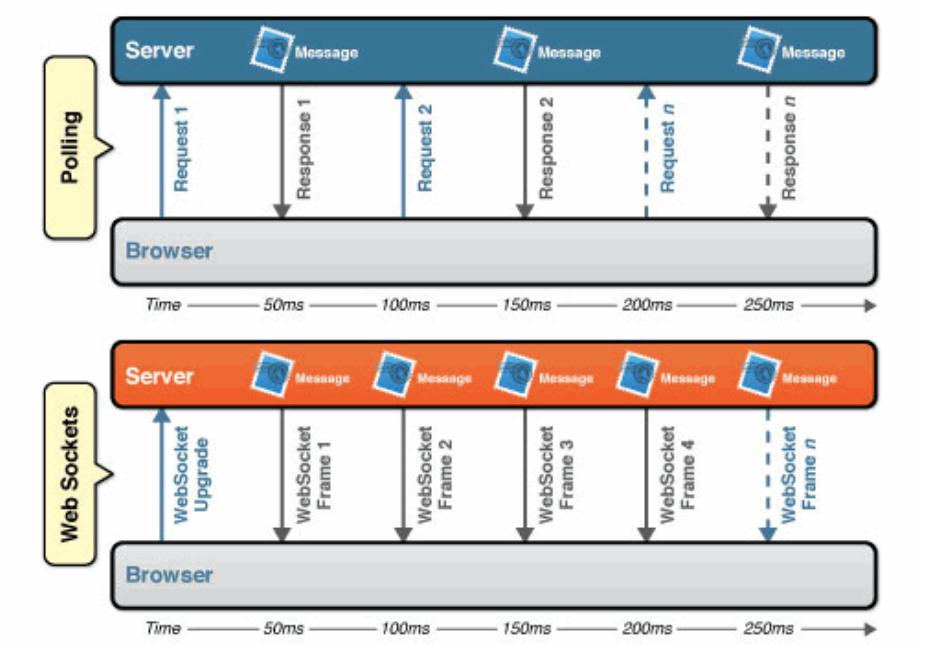

Fig. 1. En la figura se puede ver como un navegador web (browser) establece una conexión con el servidor remoto a través de websocket y Ajax Polling en una comparativa, en la misma se puede ver la mejora en los tiempos de respuesta y la baja de latencia entre mensajes [7].

Otro aspecto importante de websocket es la optimización del uso de red respecto otras tecnologías en la figura 2 se puede ver una comparativa. Para cuantificar la mejora vamos a retomar el paradigma request/response y estudiar cuanta información se transmite por cada mensaje.

Entre el encabezado de la petición y el encabezado de la respuesta tenemos un total de 871 bytes como promedio, teniendo en cuenta un mensaje de menos de 20 caracteres. Supongamos tres casos distintos:

- Caso A, con 1.000 cliente: 871 x 1,000 = 871.000 bytes = 6.6 Mbps[5]
- Caso B, con 10.000 cliente: 871 x 10.000 = 8.710.000 bytes = 66 Mbps
- Caso C, con 100.000 cliente: 871 x 100.000 = 87.100.000 bytes =665 Mbps

Es una cantidad de información innecesaria que viaja por la red. Si pudiéramos transmitir sólo los datos ahorrariamos gran parte del tráfico. Así es como funciona websocket, solo viajan los mensajes una vez establecida la conexión. Cada mensaje nos consume 2 bytes, entonces para los casos anteriores utilizando websocket quedaria asi:

[5]Mega bits por segundo (Mbps).

- Caso A, con 1.000 cliente: 2 x 1,000 = 2.000 bytes =0,015 Mbps
- Caso B, con 10.000 cliente: 2 x 10.000 = 20.000 bytes =0.153 Mbps
- Caso C, con 100.000 cliente: 2 x 100.000 = 200.000 bytes = 1.526 Mbps

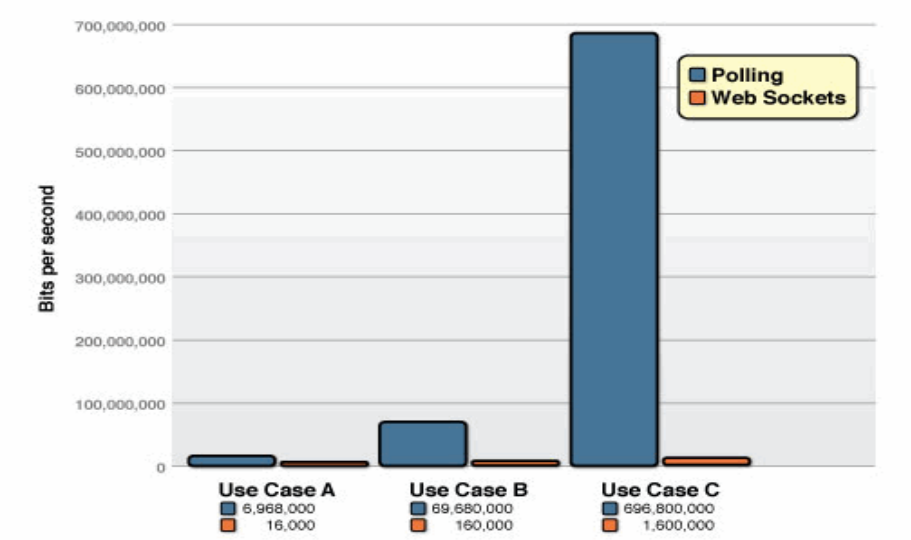

Fig. 2. En la siguiente grafico podemos ver otra comparativa entre websocket y ajax sondeo. En este caso en función al tráfico de red con distintas cantidades de usuarios, en A 1.000 clientes, para B 10.000 clientes y C 100.000 clientes. [8]

2.3 Html5 y Websocket

Html5 fue diseñado para que el desarrollo de aplicaciones web sea más fácil y más natural que sus predecesores. Hace las aplicaciones web más usables y elimina la necesidad de utilizar complementos. Entre las principales ventajas que introduce Html5 podemos citar: CSS3 para los estilos, funcionalidades fuera de línea y almacenamiento local y base de datos sqlite, acceso a multimedia y conectividad. Dentro de conectividad incluye tecnologías como websocket.

Beneficios de utilizar Html5 y websocket podemos mencionar:

- **Rendimiento**: Permite que la comunicación en tiempo real mucho más eficiente, ahorra ancho de banda, potencia de la CPU y la baja latencia.
- **Simplicidad**: Permite simplificar radicalmente la comunicación en aplicaciones web en tiempo real.
- **Estándares**: WebSocket es un protocolo de red subyacente que le permite construir otra norma protocolos en la parte superior de la misma. La mayoría de las aplicaciones AJAX consisten en

componentes de cliente y servidor fuertemente acoplados. WebSocket apoya el concepto de protocolos de más alto nivel, que puede evolucionar de forma más flexible, con clientes y servidores independientemente uno del otro.

2.3 Especificación en Arquitectura Java Empresarial

JSR[6]-356 [9] es la especificación para implementar websocket en Java. Esta es la base para nuestra implementación de websocket dentro de la arquitectura Java Empresarial. En la figura 3 se presenta un esquema de relación de la especificación entre clientes y servidor.

Cada implementación del protocolo websocket que pretende ser compatible con JSR-356 debe implementar esta API. Como consecuencia de ello, los desarrolladores pueden escribir sus aplicaciones basadas en websocket independientemente de la aplicaciones subyacentes.

Dicha especificación es parte del estándar Java EE 7 [10]; por lo tanto, todos los servidores de aplicaciones compatibles Java EE 7 tendrán una implementación del protocolo websocket que se adhiere al estándar JSR-356. También define una API de cliente de Java, que es un subconjunto de la API completa requerida en Java EE 7.

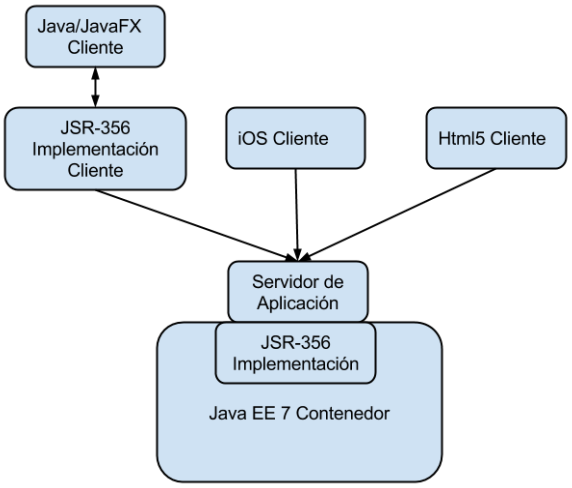

Fig. 3. Representa un servidor escrito en Java, y los detalles del protocolo websocket son manejados por la implementación JSR-356 que figura en el contenedor Java EE 7. Un cliente de Java/JavaFX, que podría ser un cliente android, puede confiar en

[6] JSR significa solicitud de especificación java del inglés Java Specification Request

cualquier aplicación cliente compatible con JSR-356 para el manejo de las cuestiones de protocolo websocket específico. Otros clientes (cliente iOS o un cliente de HTML5) pueden usar otras implementaciones no Java que cumplan con RFC-6455 con el fin de comunicarse con el servidor de aplicaciones [11].

3 Implementación websocket

Para explicar la implementación de websocket expondremos un ejemplo práctico de un sistema de voto. Podemos ver en la figura 4 una representación del esquema del ejemplo a desarrollar.

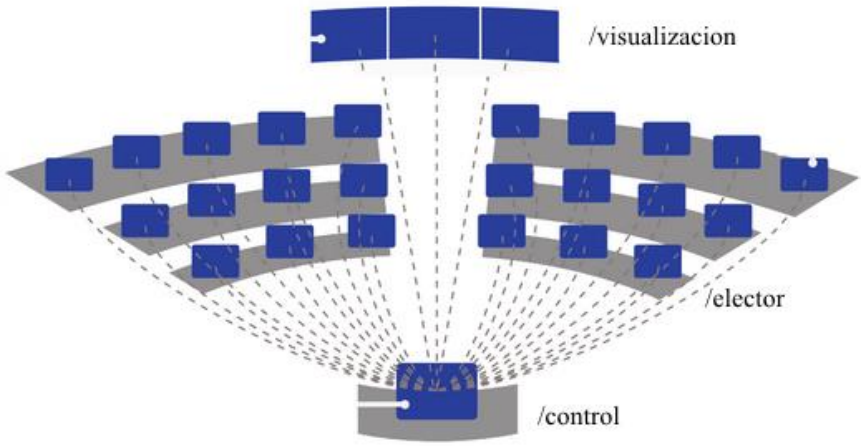

Fig. 4. Esquema de conectividad, donde se puede dividir en tres tipos de clientes. Pantallas de visualización (un solo cliente), electores (46 clientes) y puesto de control (3 clientes).

Todos los clientes están conectados en tiempo real con websocket a través de un explorador web. Los electores con tabletas android 4.0 mediante una aplicación web basado en el motor de gecko (webview)[12]. Versiones anteriores a android 4.4 no soportan la especificación de websocket. El puesto de control y la visualización utilizan el explorador Chrome versión 36.

Se crearon tres servicios websocket mediante la especificación JRS-356, uno para cada de cliente. La especificación provee la anotación @ServerEndpoint para definir el punto de acceso al recurso, con el parámetro valor (value) definimos el nombre del canal websocket, este valor tendría que ser representativo, en nuestro caso:

- @ServerEndpoint(value = "**/visualizacion**")
  public class wsEndpointVisualizacion{}
- @ServerEndpoint(value = "**/control**")

public class wsEndpointControl{}

- @ServerEndpoint(value = **"/elector/{nombreUsuario}"**)
  public class wsEndpointElector{}

El servicio de websocket para electores tiene un parámetro que nos permite identificar a un elector en particular, usamos el nombre de usuario que inició la sesión.

Para establecer la comunicación entre el servidor y los clientes se crea una instancia de websocket mediante Javascripts desde el lado del cliente.

```
var wsUri = "ws://servidor/elector/lvivas";

websocket = new WebSocket(wsUri);
websocket.onopen = function(evt) { ...};
websocket.onclose = function(evt) { ...};
websocket.onmessage = function(evt) { ...};
websocket.onerror = function(evt) { … };
```

En el código se ve parte de la implementación de websocket elector en lenguaje javascripts. La variable wsUri tiene la dirección del websocket de elector con el usuario lvivas. Después de generar la instancia de WebSocket, con la dirección se crea la conexión. Las funciones onopen, onclose, onmessage y onerror (cuando se abre, cierra, llega un mensaje y se produce un error) nos permiten reimplementar comportamiento en el cliente en función de nuestra necesidad. En onopen, podríamos informar al usuario que se establece una conexión con el servidor y también informar si la conexión falló definiendo una acción en la función onerror.

Siguiendo con el ejemplo de votación, desde el control se inicia una nueva votación, con el siguiente método que se ejecuta en el servidor Java:

```
wsEndpointElector.enviarMensaje(“INICIO_VOTACION”);
```

Todos los electores conectados al canal **/elector/{nombreUsuario}** van a recibir el mensaje “INICIO_VOTACION”.

```
websocket.onmessage = function(evt) {
    if (evt.data == “INICIO_VOTACION”){
        //Actualizar datos de la votación
        //Cambiar a pagina de Votacion
    }
```

En el fragmento de código se implementa: la función onmessage, se captura el mensaje de inicio de votación y se ejecuta la acción implementada.

4 Conclusión y Trabajos futuros

Websocket es una herramienta fundamental en el desarrollo de aplicaciones web que requieren conectividad en tiempo real entre servidor y clientes, independientemente de la plataforma. Permite establecer mecanismos de sincronización en sistemas que requieran acceso exclusivo a recursos, resolviendo los principales problemas de la concurrencia en un esquema de tiempo real.

Esta arquitectura se implementó en la legislatura de Rio Negro, en un sistema de voto nominal durante el primer semestre de 2014. Este sistema permite además de votar proyectos de ley, administrar las asistencias de los legisladores. Cuando los legisladores se registran en el sistema, se genera el presente el cual es notificado en la pantalla de visualización. Para el proceso de votación es importante la sincronización para conseguir el Quórum[7], que permite o no iniciar la votación de un proyecto.

La experiencia empírica del uso de websocket fue satisfactoria, no sólo por los resultados obtenidos en tiempos de respuesta, baja carga en la red, eficiencia en el uso de CPU del servidor y dispositivos, sino también por la facilidad de implementación gracias a la especificación JSR-356 y los estándares antes mencionados.

En trabajos futuros vamos a desarrollar e investigar una arquitectura basada en Node.js[13]. Este tipo de arquitectura tiene la particularidad de ser altamente escalable por su característica asincrónica respecto a otros servidores de aplicación como Apache o servidores para Java Empresarial. Google actualmente tiene una plataforma en la nube (Google Cloud Platform) [14] para implementar esta arquitectura con excelentes resultado. La principal razón de esta arquitectura es permitir masividad de conexiones sin modificar los tiempos de respuesta o latencia.

Referencias

1. websocket. Apache . http://tomcat.apache.org/tomcat-7.0-doc/web-socket-howto.html (accedido 03/14).
2. Java Empresarial. Oracle. http://www.oracle.com/technetwork/java/javaee/overview/index.html (accedido 04/14).

[7] Número de individuos necesario para que un cuerpo deliberante tome ciertos acuerdos.

3. Paradigma Request/Response. W3C. http://www.w3.org/Library/User/Architecture/Request.html (accedido 05/14).
4. Ajax Professional ajax 2nd edition ISBN: 978-0-470-10949-6.
5. EITF http://www.ietf.org (accedido 06/14).
6. Especificación WebSocket. IETF http://tools.ietf.org/html/rfc6455 (accedido 07/14).
7. Latencia. Websocket.org http://www.websocket.org/quantum.html.(accedido 06/14).
8. Tráfico de red. Websocket.org http://www.websocket.org/quantum.html (accedido 06/14).
9. JSR-356 Java Community Process https://jcp.org/en/jsr/detail?id=356 (accedido 02/14).
10. Java EE 7. Oracle http://www.oracle.com/technetwork/java/javaee/overview/index.html (accedido 05/14).
11. Java EE 7 y relación con websocket. Oracle http://www.oracle.com/technetwork/articles/java/jsr356-1937161.html
12. Gecko Webview Mozilla https://wiki.mozilla.org/Mobile/GeckoView (accedido 05/14).
13. NodeJs http://nodejs.org/ (accedido 07/14).
14. Node.js Plataforma en la nuve de Google. Google https://cloud.google.com/developers/articles/real-time-gaming-with-node-js-websocket-on-gcp (accedido 06/14).