

FIVE SIGNS YOU NEED HTML5 WEBSOCKETS



A KAAZING WHITEPAPER

FIVE SIGNS YOU NEED HTML5 WEBSOCKETS

A KAAZING WHITEPAPER

HTML5 Web Sockets is an important new technology that helps companies build engaging, interactive real-time web applications quickly and reliably. Sure, HTML5 Web Sockets may be the best thing since sliced bread, but is this new technology right for you?

This document identifies five types of web applications that will benefit from HTML5 Web Sockets. For each of these scenarios, we will also show you where Kaazing WebSocket Gateway can be of additional value. So, without further ado... give me five!

THE FIVE SIGNS

1. Your application has data that must flow bi-directionally
2. Your application must scale to large numbers of concurrent users
3. Your application must extend TCP-based protocols to the browser
4. Your application developers need an API that is easy use
5. Your application must extend SOA over the Web and in the Cloud

1. YOUR APPLICATION HAS DATA THAT MUST FLOW BI-DIRECTIONALLY

When the Web was first conceived, it focused on document retrieval. Users requested a URL and the server delivered an object (for example, a web page or an image file).

Today, the Web also comes to us. Servers want to let us know when they have something for us—a stock update or a message from a friend. Unfortunately, due to the Web’s architecture, the client must still initiate communication using half-duplex (request/response) HTTP. Even relatively static applications need a way to communicate asynchronously between the client and the server for common tasks such as spell checking or search auto-completion.

In an effort to simulate full-duplex communication over half-duplex HTTP, developers have contrived several clever tricks and techniques using two connections: one for the downstream and one for the upstream. Many of

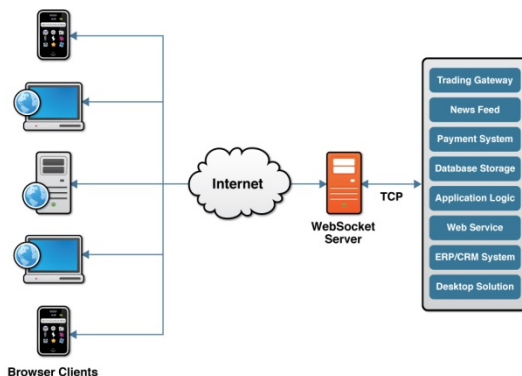


Figure 2: HTML5 Web Sockets delivers a full-duplex communication model for the Internet in which browsers can communicate directly with TCP-based back-end services without the need for convoluted server side logic that translates the TCP-based protocols to the Web.

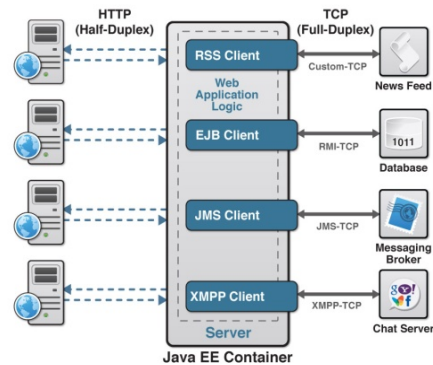


Figure 1: HTTP’s half-duplex nature prevents clients from talking directly to the back-end services they want to reach. Application servers are actively involved in the translation, interpretation, and reformatting of every message flowing between the browser and the back-end.

these techniques use polling or long-polling to simulate server-initiated push. The maintenance and coordination of two connections introduces significant resource consumption overhead and adds lots of complexity. Simply put, HTTP was not designed for real-time, full-duplex communication.

HTML5 Web Sockets delivers a full-duplex communication model for the Internet. This opens up entirely new application models without the burden and overhead of earlier approaches. If you are building a web application that needs full-duplex, asynchronous communication between clients and servers, you need HTML5 Web Sockets.

Kaazing WebSocket Gateway is a platform for building Web applications that require low latency, full-duplex communications between an end node—such as a browser or desktop—and a server. Kaazing WebSocket Gateway supports the native Web Socket protocol, but the Kaazing libraries also leverage all possible browser features to emulate WebSocket functionality in older browsers that do not support Web Sockets. Kaazing’s uses the best possible connection, whether or not clients and intermediate devices support the latest protocols.

2. YOUR APPLICATION MUST SCALE TO LARGE NUMBERS OF CONCURRENT USERS

If you are building a web application for a large number of concurrent users, you are going to face resource contention. Every one of those users will establish a connection to your back-end application. Typically, each of those connections will include the overhead of HTTP's verbose request-and-response protocol, as well as the possible establishment and teardown of connections.

The HTTP request and response model suffers from a significant amount of overhead. When retrieving a large document from a server, a few hundred bytes of HTTP header overhead is not a big deal. Consider what happens, however, when each message sent to a client is only a few bytes. For example, a stock price update that is only 20 characters long. The majority of the data transmitted in this case is unnecessary HTTP header overhead (up to 2000 bytes for the single stock price update), making the communication highly inefficient.

HTML5 Web Sockets specifies new, vastly more efficient way of communicating between clients and servers that is far less taxing on the application, and easier for the underlying network infrastructure to handle. Replacing hundreds of HTTP header bytes with just two WebSocket frame bytes leads to a massive reduction in unnecessary network throughput (up to 1000:1) as shown in Figure 3. Additionally, Web Sockets' lack of continuous polling dramatically reduces latency. All of this means that a single WebSocket server can deal with many more users at once, reducing the total cost of ownership (TCO) dramatically.

HTML5 Web Sockets provides such a dramatic improvement from the old, convoluted hacks that simulate a full-duplex connection in a browser that it prompted Google's Ian Hickson—the HTML5 specification lead—to say:

"Reducing kilobytes of data to 2 bytes...and reducing latency from 150ms to 50ms is far more than marginal. In fact, these two factors alone are enough to make Web Sockets seriously interesting to Google."

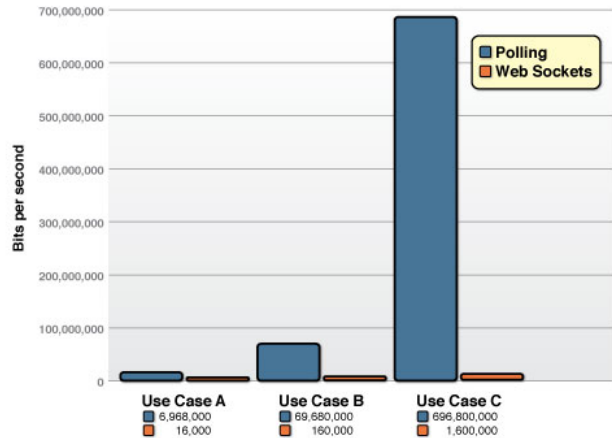


Figure 3: Comparison of unnecessary network throughput related to HTTP overhead between a polling Comet and Ajax solution (blue) and a WebSocket solution (orange) for three use cases: 1,000, 10,000, and 100,000 concurrent clients respectively at one-second intervals. In the 100,000 user case, the unnecessary network throughput is 665 Mbps versus 1.5 Kbps (less is better here!).

Kaazing WebSocket Gateway's architecture is based on Staged Event-Driven Architecture (SEDA), and it leverages Java New I/O (NIO) for enhanced Java networking functionality. For example, instead of assigning a single thread per request, the server shares threads for optimal performance. This allows the Kaazing Platform to achieve unparalleled levels of concurrent communication at extremely low message latency.

3. YOUR APPLICATION MUST EXTEND TCP-BASED PROTOCOLS TO THE BROWSER

Many web applications need to connect end users to information from back-end services. These services are contained in legacy systems, or travel across enterprise message buses via APIs and protocols such as TIBCO EMS, JMS, RMDS, AMQP, XMPP, and Stomp.

Some applications may be composed of several subsystems, each using a different application protocol. For example, one subsystem requires a publish/subscribe programming model listening and responding to the changing prices of inventory items, another subsystem is receiving a large volume of database events pushed from a column-based persistence engine, and yet another subsystem has to enable chat and chat rooms.

By using Web Sockets, your application can avoid silo-ed solutions for each subsystem. You no longer have to use hacks to push data to a browser or use CPU and network-intensive polling to simulate publish/subscribe. The W3C and IETF standards bodies have provided an elegant way to use full-duplex network communication over the Web with HTML5 Web Sockets. Additionally, since WebSocket traffic flows over standard HTTP ports (80 and 443); there is no need to open additional ports on corporate firewalls to allow full-duplex communication.

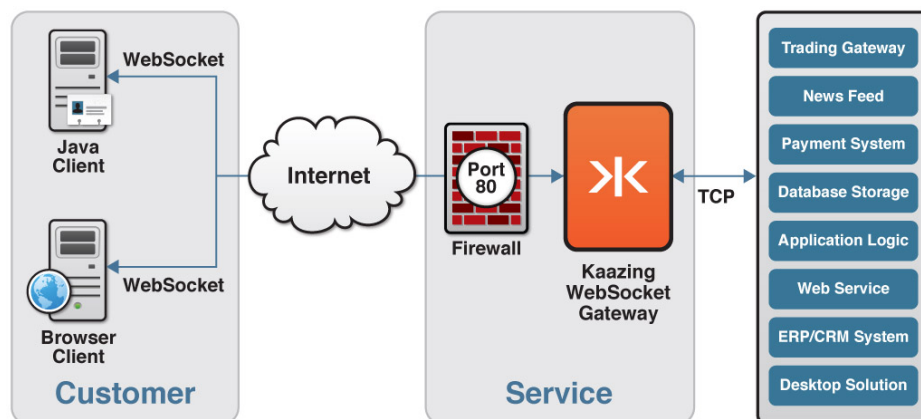


Figure 4: HTML5 Web Sockets defines a full-duplex communication channel that operates through a single socket. This enables web pages and other clients to have two-way communication with a remote host over the Web. In this figure, Kaazing WebSocket Gateway seamlessly extends TCP-based protocols to the Web.

Kaazing Websocket Gateway is easy to configure and install. Once deployed, the gateway supports many popular protocols transparently and efficiently right out of the box. For example, Kaazing WebSocket Gateway supports XMPP/Jabber, IRC, TIBCO JMS, Stomp, AMQP, RMDS, and many other protocols in a wide variety of client technologies, such as JavaScript, Adobe Flex, Microsoft Silverlight, and Java/JavaFX.

Developers can use familiar APIs to call native methods to communicate with back-end services in familiar client technologies. This approach significantly reduces the learning curve and development time for projects that must offer back-end communication to clients.

4. YOUR APPLICATION DEVELOPERS NEED AN API THAT IS EASY TO USE

To deliver a compelling, usable application, developers rely on rich client platforms such as Adobe Flex (Flash), Microsoft Silverlight, Java/JavaFx, and JavaScript. However, connecting these rich clients to real-time data over the Web can be challenging. Developers often have to create their own client- and server-side communication libraries; essentially reinventing the wheel to overcome some of the inherent limitations of HTTP.

Consider how much time and effort is required to create a reliable two-way communications protocol, and to connect an application server to back-end systems. Testing and securing applications built on top of this protocol is difficult, because it is harder to pinpoint the problem in a proprietary protocol. Moreover, the work is application-specific, thwarting all attempts to re-use it.

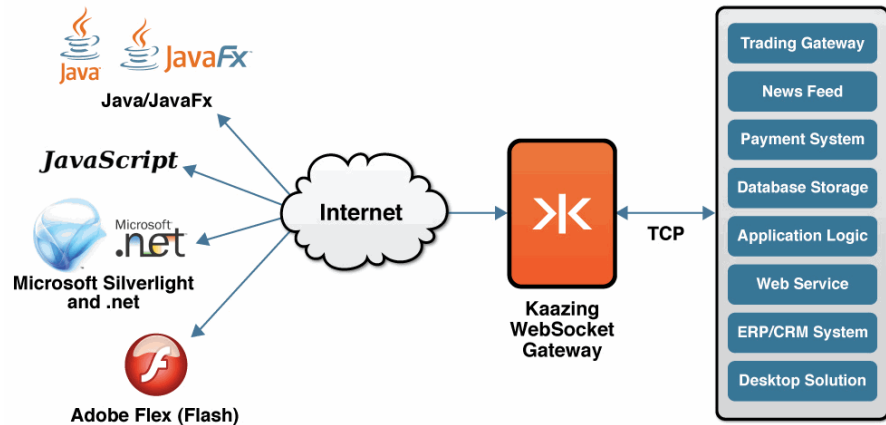


Figure 5: Kaazing WebSocket Gateway not only supports many TCP-based protocols; it also provides support for a variety of client technologies, such as Adobe Flex (Flash), Microsoft Silverlight and .net, JavaScript, and Java/JavaFx.

HTML5 Web Sockets offers a single, standard interface against which to develop. That means developers can spend less time building and testing communication protocols, and more time designing a superb client-side experience. HTML5 Web Sockets eliminates most of the custom development work that engineers have to do to create a fast, secure, two-way application.

Kaazing WebSocket Gateway works with a variety of Rich Internet Applications and client-side frameworks (in addition to JavaScript), ensuring that you can develop in the platform of your choice. These client-side technologies include Adobe Flex (Flash), Microsoft Silverlight, Java/JavaFX, and JavaScript. Kaazing WebSocket Gateway also provides integration with PHP and Google Web Toolkit.

Pre-built libraries for popular clients make it easy to connect to a wide range of back-end servers while using the language and paradigms of the frameworks that development teams are familiar with.

5. YOUR APPLICATION MUST EXTEND SOA OVER THE WEB AND IN THE CLOUD

Your enterprise Service-Oriented Architecture (SOA) application or SOA product works well on an internal enterprise network, but you now have to deploy your high-performance distributed software over the Web—and through the quagmire of firewalls and proxy servers along the way—to leverage all the advantages of a web infrastructure for your demanding customers and even more-demanding management.

With Web Sockets, your applications can open a standardized, bi-directional connection over the Web. Developers can leverage this connection to extend messages from an SOA inside the firewall to an external SOA such as a high-performance Enterprise Service Bus (ESB) or a web-based Supply Chain in a cloud environment.

HTML5 Web Sockets are more efficient and WebSocket applications can handle more concurrent users and a greater message volume, with less infrastructure. That is good for any application that has to deal with capacity constraints, as the number of servers required is lower. Where this really makes a difference, however, is in on-demand computing environments—in public and private clouds. In a cloud-computing model, capacity is elastic: you pay for what you use.

Traditionally, there are two ways of handling growing demand. The first is to scale “vertically”, buying a bigger machine, adding RAM, and so on. Modern web applications do not use this approach, however; they scale “horizontally”—that is, to handle more load, you add more machines at each of the tiers in the application.

Web applications deployed on elastic computing platforms must be especially aware of their resource consumption. In a cloud, the addition of machines is often automated, so there is no upper limit on resource consumption. Suddenly, inefficient code translates directly into a higher bill at the end of the month, because more virtual machines were spun up to handle the traffic. In these environments, the efficiency of HTML5 Web Sockets is particularly compelling.

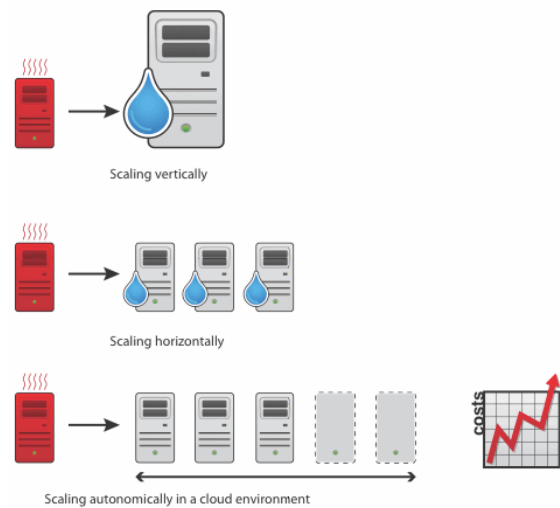


Figure 6: Horizontal scaling in an elastic environment, such as the cloud emphasizes the importance of efficient application design.

Kaazing WebSocket Gateway allows you to extend TCP-based SOA protocols over the web so you can easily use programming models such as publish/subscribe and instant messaging. Out of the box, the Kaazing Platform offers a collection of APIs such as JMS/STOMP, XMPP, IRC and others. In addition, the Gateway is compatible with a long list of message brokers (for example, Tibco, RabbitMQ, Apache ActiveMQ, MQSeries, and so on) that allow programmers to focus on application functionality rather than the low-level WebSocket details. Finally, Kaazing WebSocket Gateway’s unique architecture makes it easy to configure a virtual private (TCP) connection between mutually exclusive networks or in the cloud, allowing you to extend the reach of any enterprise system with ease.

ABOUT KAAZING

Kaazing is a web infrastructure software company that enables companies to quickly deliver massively scalable, real-time enterprise web applications and significantly reduce operational costs by simplify back-end server infrastructure.

Kaazing's patent-pending, WebSocket Acceleration™ technology is the only solution in the marketplace today that facilitates full-duplex web communication, which makes it possible to extend any TCP-based messaging protocols to the Web seamlessly and reliably, without buying expensive new hardware.

Unlike older, non-standard technologies that rely on “polling” or “long-polling,”—complex and inefficient techniques that result in higher latency and sky-rocketing server infrastructure costs—Kaazing delivers ultra-high application performance at the lowest operational cost.

RELATED LINKS

- Kaazing: <http://www.kaazing.com>
- Kaazing Technology Network: <http://tech.kaazing.com>
- Websockets.org: <http://www.websockets.org>