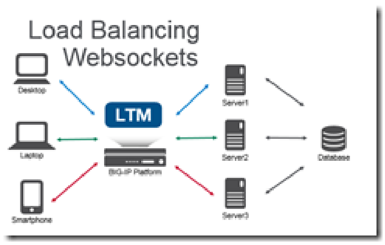
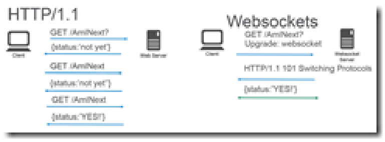
Load Balancing WebSockets

Eric Chen, 2015-31-07



An introduction to WebSockets and how to load balance them. WebSockets creates a responsive experience for end-users by creating a bi-directional communication stream versus the one-way HTTP stream. For example, when you're waiting at the deli counter you need to take a number. An HTTP method of checking your status in line would be to periodically take your number up to the deli counter to see if you're next in line. The WebSocket method for notification would be to have someone shout out the number to you when you're next. One of these methods is more convenient!

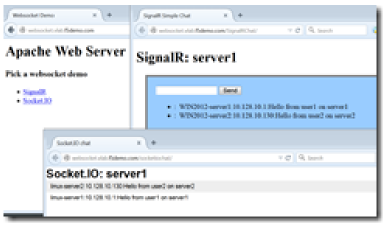
HTTP is a stateless protocol. It looks like a series of request/responses that originate from the client to the server. WebSockets is a bi-directional protocol that allows the client to send requests to the server AND allows the server to push responses to the client.



On the BIG-IP with LTM the default HTTP profile has supported the WebSocket upgrade header since 11.4.0. It is possible to use a FastL4 profile to treat all the traffic as TCP, but you lose some resources like the ability to set X-Forwarded-For headers to provide visibility to the client IP when using SNAT, cookie persistence (avoid issues when client IP changes), and the ability to route traffic based on the HTTP request. Given the long duration of a WebSocket connection; you can also utilize pool member connection limits and least connection load balancing to ensure an even distribution of traffic across multiple nodes.

General tips for the backend servers is to ensure that the servers are stateless (any server can generate a response for any client) or share state. SignalR (ASP.NET) has a nice introduction to scaling out (don't forget to use the same MachineKey across IIS servers). Socket.IO (Node.JS) has helpful documentation that covers utilizing multiple nodes (Redis works well as a provided adapter). Not all clients will support WebSocket natively, and/or web proxy/firewalls may not allow these connections. Fallback mechanisms exist for both SignalR/Socket.IO to allow communication without support for WebSockets (via HTTP).

Using these tips to load balance WebSockets you can create a highly available service of WebSocket servers or create a demo that combines an Apache web, Node.JS Socket.IO, and SignalR ASP.NET server under a single URL!



Apache Web Server

Pick a websocket demo

- SignalR
- Socket.IO

SignalR: server1

- : WIN2012-server1 10.128.10.1 Hello from user1 on server1
- : WIN2012-server2 10.128.10.130 Hello from user2 on server2

Socket.IO: server1

linux-server2 10.128.10.130 Hello from user2 on server2
linux-server1 10.128.10.1 Hello from user1 on server1