

### **Name attributes of the HTTP protocol which makes it difficult to use for real time systems.**

A hard real-time system (also known as an immediate real-time system) is hardware or software that must operate within the confines of a stringent deadline. The application may be considered to have failed if it does not complete its function within the allotted time span.

Examples of attributes of the HTTP protocol that makes it difficult to use for real time systems could be:

- Cache-Control: max-age=0

- \*This instructs the user agent that the content is stale and should be validated before use.

- if-Modified-Since

- \* Allows a 304 Not Modified to be returned if content is unchanged.

- If-Unmodified-Since

- \* Only send the response if the entity has not been modified since a specific time.

- If-Match

- \* Only perform the action if the client supplied entity matches the same entity on the server. This is mainly for methods like PUT to only update a resource if it has not been modified since the user last updated it.

- If-None-Match

- \* Allows a 304 Not Modified to be returned if content is unchanged.

The reason why these could be examples is that they are used as a common way to prevent old content from being shown to the user without validation. It just tells the browser and proxies to validate the cache content with the server before using it. Thereby sending a no-cache value thus instructs a browser or proxy to not use the cache contents merely based on "freshness criteria" of the cache content.

The main point of this question is that because the HTTP protocol is so big and it takes a lot of information with it to make each call it is slow.

In other words the protocol contains a lot of information - information that is an overhead.

One of the core principles of the HTTP protocol, is the Request - Response model. A client sends a request to the server and the server then comes with an appropriate response. This is a great construction for a traditional website where the client do an action and gets a response back. Since HTTP is stateless, every single request must have some information about what it want to do, where it should be done and who is doing it and so on. All this information, that needs to be send every time, adds up to a lot of bytes which affects the speed of the request - response time.

## **What is the WebSocket protocol, and how is it different from HTTP communication, what advantages has it over HTTP?**

WebSocket is a protocol providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket [API](#) in Web IDL is being standardized by the W3C.

WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. The WebSocket protocol makes more interaction between a browser and a website possible, facilitating the real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. The communications are done over TCP port number 80, which is of benefit for those environments which block non-web Internet connections using a firewall. Similar two-way browser-server communications have been achieved in non-standardized ways using stopgap technologies such as Comet.

The WebSocket protocol is currently supported in most major browsers including Google Chrome, Internet Explorer, Firefox, Safari and Opera. WebSocket also requires web applications on the server to support it. Unlike HTTP, WebSocket provides full-duplex communication. Additionally, WebSocket enables streams of messages on top of TCP. TCP alone deals with streams of bytes with no inherent concept of a message. Before WebSocket, port 80 full-duplex communication was attainable using Comet channels; however, Comet implementation is nontrivial, and due to the TCP handshake and HTTP header overhead, it is inefficient for small messages. WebSocket protocol aims to solve these problems without compromising security assumptions of the web.

The WebSocket protocol specification defines `ws` and `wss` as two new uniform resource identifier (URI) schemes that are used for unencrypted and encrypted connections, respectively. Apart from the scheme name and fragment (`#` is not supported), the rest of the URI components are defined to use URI generic syntax.

Using the Google Chrome Developer Tools, developers can inspect the WebSocket handshake as well as the WebSocket frames.

## **What's the advantage of using libraries like Socket.IO, SockJS, WS, over pure WebSocket libraries in the backend and standard APIs on frontend? Which problems do they solve?**

Socket.io can be used as a wrapper for WebSocket, but provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O. It allows you to send/emit messages by specifying an event name.

Furthermore it simplifies the usage of WebSockets, and provides failovers to other protocols in the event that WebSockets are not supported on the browser or server. Ex. It will test WebSocket compatibility and if it's not supported it will use Adobe Flash, AJAX, or an iFrame. Socket.io supports a very large set of browsers: Internet Explorer 5.5+, Safari 3+, Google Chrome 4+, Firefox 3+, Opera 10.61+, iPhone Safari, iPad Safari, Android WebKit & WebOs WebKit.

SockJS is a JavaScript library (for browsers) that provides a WebSocket-like object. SockJS gives you a coherent, cross-browser, Javascript API which creates a low latency, full duplex, cross-domain communication channel between the browser and the web server, with WebSockets or without.

In short terms ws is a node.js websocket library. Ws is a simple to use WebSocket implementation, up-to-date against RFC-6455, and probably the fastest WebSocket library for node.js.

The reason why you would use libraries like the above on top of WebSocket is that it is still a young technology and not fully implemented in all browsers. However, by using WebSocket with libraries it fallbacks, whenever WebSocket is not available. A library that has become very popular in this domain is socket.io which comes with a client and a server implementation of the protocol and includes fallbacks.

### **What is Backend as a Service, Database as a Service, why would you consider using Firebase in your projects?**

#### **BaaS**

Backend as a Service, or BaaS (sometimes referred to as mBaaS) is a cloud computing category that consists of companies that make it easier for developers to setup, use and operate a cloud backend for their mobile, tablet and web apps.

Backend as a Service provides web and mobile app developers with a way to link their applications to backend cloud storage while also providing features such as user management, push notifications, and integration with social networking services. These services are provided through customized Software Development Kits (SDKs) and Application Programming Interfaces (APIs).

Cloud backend as a service platforms enable app developers to store, manage and access their data through such cloud. Moreover, they are able to develop backend features for their apps in minutes, rather than weeks.

Source: <http://www.baasbox.com/what-is-backend-as-a-service/>

#### **DBaaS**

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data.

DBaaS delivers database functionality similar to what is found in relational database management systems (RDBMSes) such as SQL Server, MySQL and Oracle. Being cloud-based, on the other hand, DBaaS provides a flexible, scalable, on-demand platform that's oriented toward self-service and easy management, particularly in terms of provisioning a business' own environment. DBaaS products typically provide enough monitoring capabilities to track performance and usage and to alert users to potential issues. The products can also generate at least some degree of data analytics.

Disadvantages to the DBaaS model include a lack of control over network performance issues, such as unacceptable latency and application failures. Furthermore, some DBaaS products don't support capabilities of the typical RDBMS, such as data compression and table partitions. Before committing to DBaaS, it's essential to assess your specific requirements and ensure they are satisfactorily addressed. DBaaS is one of a growing number of examples of cloud-based services including Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) and Software as a Service (SaaS). Databases that are administrated through this model are sometimes referred to as cloud databases.

**Explain and demonstrate a “real time” system using either; your own server and a library like Socket.IO or Firebase.**

See Example