**Web Interface and its problems**

When a user interacts with other users and applications on the network (through a computer), there are two basic types of interaction:

**Client Initiated**: Here the user makes a decision to do something. The client (the software on the computer that the user is using) will send protocol to a server on the network. Typically this forms a request/response pattern, with the client making a request and the server providing a response, although XMPP allows for other patterns.

**Server Initiated**: Here the user is passive. Something happens remotely; A server sends something to a client; The user is alerted or informed in some way. This may form either a request/response pattern or a pure server-push where no response is expected.

**Why a Different Approach is needed for XMPP?**

Server initiated interactions are central to the way XMPP operates. In particular:

* Messages may arrive, either from one of your buddies.
* The online status of one of your buddies may change.

Unless you have an appropriate browser window open, there is no mechanism for the user to be alerted in a standard Web browser. This is a reason why many XMPP users (and IM users in general) prefer to use a desktop client. A desktop application can give useful alerting, even when the primary window is minimized or obscured (e.g., by a "pop up" alert).

A more general problem is that the core Web (HTTP) protocol is client (browser) initiated. This means that there is no basic mechanism for a server web application to display information on the user’s screen in response to a server initiated event. This means that the simple application server architecture used for client initiated protocol will not work for XMPP.

**Strophe.js**

Strophe.js is an XMPP library for JavaScript. Its primary purpose is to enable web-based, real-time XMPP applications that run in any browser.

While most XMPP libraries and implementations are focused on chat-based applications, Strophe takes a grander view. It has been used to implement real-time games, notification systems, search engines, as well as traditional instant messaging.

**The Strophe Family**

There are currently two members of the Strophe family of libraries.

* **Strophe.js**

[Strophe.js](http://strophe.im/strophejs) is a JavaScript implementation targeting browser-based clients. It uses [BOSH](http://xmpp.org/about-xmpp/technology-overview/bosh/), a binding of XMPP to HTTP using long polling and WebSockets, a full-duplex single socket connection to a server. Strophe.js makes creating real-time web applications easy.

* **libstrophe**

[libstrophe](http://strophe.im/libstrophe) is a C library for XMPP clients and components. It has very minimal dependencies and was designed with both POSIX and Windows systems in mind.

Strophe.js is an XMPP library for JavaScript. Its primary purpose is to enable web-based, real-time XMPP applications that run in any browser. Since JavaScript had no facilities for persistent TCP connections, this library uses Bidirectional-streams Over Synchronous HTTP (BOSH) to emulate a persistent, stateful, two-way connection to an XMPP server.

**Problems with Strophe.js**

1. **Connection with Openfire Server:**
2. *Installing, Configuring and Testing Openfire Server on local machine:*

Openfire is a real time collaboration (RTC) server licensed under the Open Source Apache License. It uses the only widely adopted open protocol for instant messaging, XMPP (also called Jabber). Openfire is incredibly easy to setup and administer, but offers rock-solid security and performance. Openfire is an IM server that uses XMPP protocol. As a prerequisite, OpenJDK is a must have in the machine.

Download openfire\_3\_7\_1.tar.gz package. Extract the archive : tar -xzvf openfire\_3\_0\_0.tar.gz. Move extracted folder to /opt: mv openfire /opt

The command to start openfire server is :

$ sudo /opt/openfire/bin/openfire start.

Access openfire through : [http://localhost:9090](http://localhost:9090/)

It’ll take you through Openfire configuration wizard. Create a MySQL database named ‘openfire’. Under Database Settings, in Database URL enter : jdbc:mysql://localhost:3306/openfire.

In order to test openfire, you must create a user id. To do this, log in to the administrative console, click Users and Create New User. Once the user is created, you can test your Openfire server using an XMPP Client (Pidgin, Gajim, etc) and configuring the URL and port of your server.

1. *Configuring BOSH:*

BOSH (Bidirectional streams Over Synchronous HTTP) specified in XEP-0124 is the standardized way to do XMPP over HTTP. BOSH is simply a means of carrying the XMPP protocol over HTTP. The XML XMPP packets used in BOSH are the same as the ones used when standard XMPP is run over TCP. BOSH defines a simple framing to use these packets in HTTP protocol. For client initiated protocol the client simply sends packets over HTTP (using HTTP post). Server initiated protocol is handled by a technique called 'long polling'. In long polling, the web browser initiates a standard request, but does not expect a response back immediately. When the server has data to send, it will send it as a response to the request, and the client will immediately issue another request, thus keeping a request pending for the server to respond to.

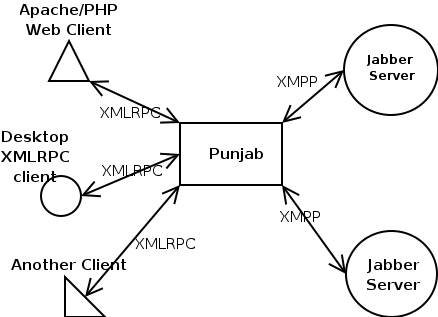
1. *Connecting with Strophe.js:*

[Strophe](http://strophe.im/strophejs/) is an XMPP library for Javascript which allows you to connect from a Web browser to an XMPP server. Download the latest [version](http://strophe.im/strophejs/) of this library and unzip it in your workspace. This package contains a couple of examples that can be used as reference. In my case, I will use examples/basic.html.

**The connection was established successfully with the Openfire Server using strophe.js**

1. **Connection with Facebook**
2. *Configuring Punjab Server as BOSH Connection Manager:*

Punjab is sort of a blackbox. It has a xmlrpc api that client's make method calls to and it then converts them into jabber messages and returns them based on the methods the client's call. What goes on in between does not really matter to the client developer.



A standalone BOSH Server i.e. Punjab was installed for connection to Facebook’s XMPP servers using strophe.js. Punjab was created to allow stateless systems a way to interact with [jabber](http://jabber.org/), a very stateful system. Jabber or XMPP is a powerful, extendable, and easy to use xml messaging protocol. It is mostly used for instant messaging, but can be used for many other applications.

1. *Connecting with Strophe.js:*

**The connection could not be established even after repeated trials to connect Facebook XMPP Server using strophe.js and the configured Punjab BOSH Connection Manager. The reason behind it is the deprecation of the Facebook Chat API which does not allows to connect to the Facebook XMPP Servers.**

1. **Connection with Gmail**
2. *Configuring Punjab Server as BOSH Connection Manager:*

Same procedure as in ‘Connection with Facebook’

1. *Connecting with Strophe.js:*

**Since Google has stopped GTalk and switched over to Hangouts, the XMPP connection to Google Server is denied using Strophe.js library as XMPP no longer is the underlying protocol.**