**xmpp HTTP协议**

[XEP-0124: Bidirectional-streams Over HYPERLINK "http://www.xmpp.org/extensions/xep-0124.html"Synchronous HYPERLINK "http://www.xmpp.org/extensions/xep-0124.html"HTTP](http://www.xmpp.org/extensions/xep-0124.html) 内容:

1. <body/>元素是任何通过BOSH传递xml流的根元素 。
2. 对Create Session Request/Create Session Response的xml文本做了详细的规定。
3. Sending and Receiving XML Payloads
4. Inactivity/ Overactivity

**BOSH Connection阐述和分析**

1、该标准为双向同步数据提供一个模拟层。借助这个标准，可以与一个 XMPP 服务器建立一个较长的 HTTP 连接（时长一分钟或两分钟,不同浏览器不同）。如果新数据在那个期间到达，则 HTTP 请求返回数据并关闭；否则，该请求延迟一段时间后失效，向HttpIM Client发送inactivity数据并关闭。不管是哪种情况，一旦一个请求关闭，另一个请求连接将立即重新建立。 通过这种方式，HttpIM Client和服务器始终维持着一个连接，就像一个长tcp长连接一样。

2、Bosh连接过程中，不会在httpIM Client端设置cookie或http header.

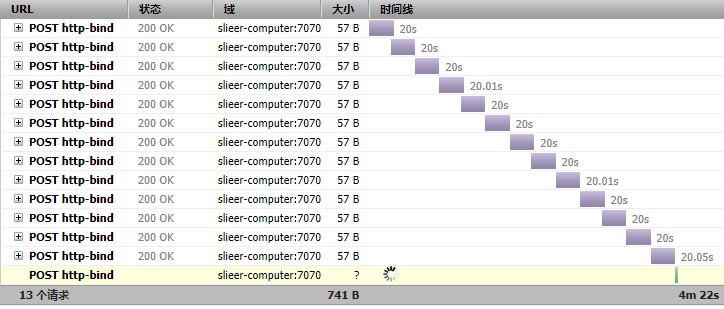
3、根据openfire插件jappix 的测试：

当客户端向xmpp server 发送数据时，立即开启一个新的请求。

在httpIM Client空闲时，会持续向xmpp Server发出请求，请求被Jetty被阻塞 20秒后超时，向客户端响应。见HttpConnection.waitForResponse

当其它的客户端如spark向httpIM Client发送数据时，唤醒当前的请求(HttpConnection.deliverBody),response返回数据到http IM Client.如果当前求response数据已发送，会在下一下response中立即返回。

从firefox 观察httpIM client在空闲时的发送请求的情况：



Openfire为实现上述功能，采用了jetty Continuations(和tomcat 的comet类似) 异步Http技术，能让Http连接挂起，直到超时或者异步事件发生时，Http连接可以恢复。

Continuation介绍：当一个Java filter或是Servlet处理一个AJAX请求，可以使用Continuation对象有效地中断(suspend)请求并释放当前的线程。当中断超时或是在Continuation对象上调用resume方法时，这个请求会被唤醒(resume)。

见文档：

<http://langyu.iteye.com/blog/707713>

<http://www.cnblogs.com/kylindai/archive/2009/11/01/1593749.html>

**架构**

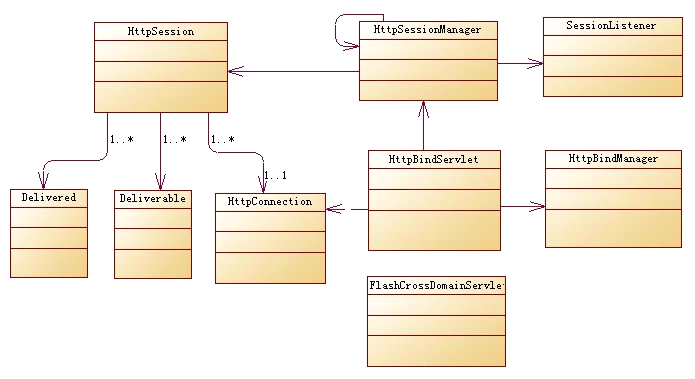
Server  
        |  
        |  [unwrapped data streams]  
        |  
     HTTP CM  
        |  
        |  [HTTP + <body/> wrapper]  
        |  
     Client

**Openfire bosh类和关系图**

HttpConnection HttpSession HttpBindServlet HttpBindManager

HttpSessionManager FlashCrossDomainServlet

类关系图：



**Xmpp http加载过程**

在Xmpp Server启动时，

当ConnectionManagerImpl模块被加载时，加载HttpBindManager ，同时启动jetty Http Server,并载入HttpBindServlet，FlashCrossDomainServlet，ResourceServlet

其中HttpBindServlet 在7070端口上接受web IM client 的请求，

FlashCrossDomainServlet 为web IM client提供了Flash 方式的跨域访问机制。当然也可以采用apache设置代理的方式。

当SessionManager模块被加载时，加载HttpSession，用来创建httpSesion.

当ClearspaceManager模块被加载，用来集中管理连接，HttpBindManager也被加载，以管理http 所占用的资源。

**响应请求图**

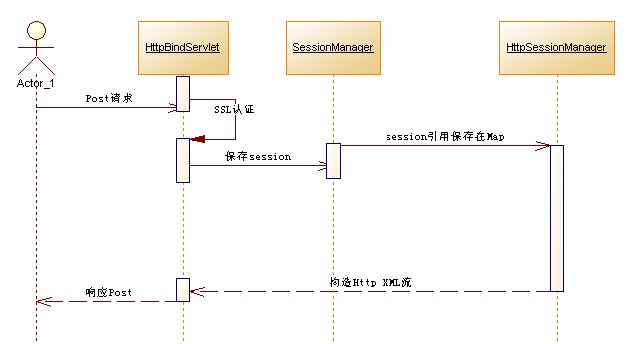
见：<webim.vsd>

最终处理http第次请求的是 HttpSessionManager开启一个新的HttpPacketSender线程，负责iq, presence,message处理，放在HttpSession中，key为RequestId

HttpBindServlet. respond 方法负责构建响影的过程及向客户端发出响应，响应的内容来自HttpSession中，key为RequestId对应的值。

**Session管理**

创建Session



httpIM session 和TCP IM(如spark) session创建和删除统一管理（org.jivesoftware.openfire. SessionManager）。别外，所有HttpIM session在org.jivesoftware.openfire.http.

HttpSessionManager的 ConcurrentHashMap 中存入了一个引用。

**Bosh client**

|  |  |  |
| --- | --- | --- |
| Web im | 编程语言 | 备注 |
| Sparkweb | Actionscript | 默认采用flash socket |
| openfire 插件Jappix | Java/php | 采用BOSH |

注意：Sparkweb 采用xiff， <http://www.igniterealtime.org/projects/xiff/index.jsp>

较为不错的xmpp client Javascript框架：Strophe.js，JSJaC.js

Jappix 是一个不错的xmpp http client。登录一定要采用服务器名称登录，比如90，只能采用以下地址：<http://m1.mobee.im:7070/jappix/>

**跨域访问**

方法一

flXHR + crossdomain.xml

方法二

Proxy Server比如 Aapache, Nginx

Xmpp Server和proxy配置

* 7070端口代理

Openfire 在 <http://localhost:7070/http-bind> 维护了一个 HTTP 绑定 URL，以便通过 BOSH 访问。要在端口 80 上使用这个 URL，您必须配置 Apache HTTP Server 以将一个 URL 转发到这个位置。为此，您需要启动代理模块。

* Xmpp Server配置

需要打开OpenFire的http-binding.具体为登录后台管理界面。Server->Server Settings->Http Binding:

这一项打勾 Enabled - Clients can connect to this server using HTTP binding.

下面Script Syntax ：

**这一项打勾 Enabled** - Allows BOSH clients with limited access to connect to the server

* proxy配置

打开您的 http.conf Apache 配置文件,移除以下行的前导井字符（#）。行如下:

|  |
| --- |
| LoadModule proxy\_http\_module modules/mod\_proxy\_http.so  LoadModule proxy\_module modules/mod\_proxy.so |

并且在文件的末尾加上：

|  |
| --- |
| # XMPP proxy rule  ProxyRequests Off  ProxyPass /xmpp-httpbind <http://127.0.0.1:7070/http-bind/>  ProxyPassReverse /xmpp-httpbind <http://127.0.0.1:7070/http-bind/> |

URL：/xmpp-httpbind。这个 URL 是 strophe.js（您稍后将用到的客户端 JavaScript 框架）分配给一个用于设置 BOSH 端点的变量的值。

重启服务器。

<http://www.cnblogs.com/hannover/archive/2010/11/14.html>