作者：谢泽帆  
链接：https://www.zhihu.com/question/23015432/answer/34652229  
来源：知乎  
著作权归作者所有，转载请联系作者获得授权。

最后分享下一篇关于各种Android消息解决方案的分类总结。

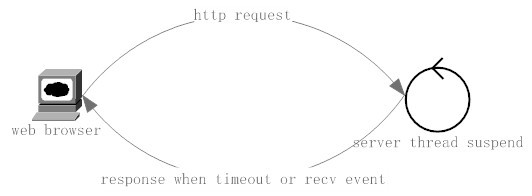
本文主旨在于，对目前Android平台上最主流的几种消息推送方案进行分析和对比，比较客观地反映出这些推送方案的优缺点，帮助大家选择最合适的实施方案。  
  
**方案1、**使用GCM服务（Google Cloud Messaging）  
简介：Google推出的云消息服务，即第二代的G2DM。  
优点：Google提供的服务、原生、简单，无需实现和部署服务端。  
缺点：Android版本限制（必须大于2.2版本），该服务在国内不够稳定、需要用户绑定Google帐号，受限于Google。  
  
**方案2、**使用XMPP协议（Openfire + Spark + Smack）  
简介：基于XML协议的通讯协议，前身是Jabber，目前已由IETF国际标准化组织完成了标准化工作。  
优点：协议成熟、强大、可扩展性强、目前主要应用于许多聊天系统中，且已有开源的Java版的开发实例androidpn。  
缺点：协议较复杂、冗余（基于XML）、费流量、费电，部署硬件成本高。  
  
**方案3、**使用MQTT协议（更多信息见：[MQTT: MQ Telemetry Transport](//link.zhihu.com/?target=http://mqtt.org/" \t "_blank)）  
简介：轻量级的、基于代理的“发布/订阅”模式的消息传输协议。  
优点：协议简洁、小巧、可扩展性强、省流量、省电，目前已经应用到企业领域（参考：[Software - runtimes, APIs and libraries for MQTT](//link.zhihu.com/?target=http://mqtt.org/software" \t "_blank)），且已有C++版的服务端组件rsmb。  
缺点：不够成熟、实现较复杂、服务端组件rsmb不开源，部署硬件成本较高。  
  
**方案4、**使用HTTP轮循方式  
简介：定时向HTTP服务端接口（Web Service API）获取最新消息。  
优点：实现简单、可控性强，部署硬件成本低。  
缺点：实时性差。

## 实时推送的几种实现方式

相较于手机端的消息推送（一般都是以Socket方式实现），WEB端是基于HTTP协议，很难像TCP一样保持长连接。但随着技术的发展，出现了WebSocket，Comet等新的技术可以达到类似长连接的效果，这些技术大体可分为以下几类：

**1）短轮询**。页面端通过JS定时异步刷新，这种方式实时效果较差。

**2）长轮询**。页面端通过JS异步请求服务端，服务端在接收到请求后，如果该次请求没有数据，则挂起这次请求，直到有数据到达或时间片（服务端设定）到，则返回本次请求，客户端接着下一次请求。示例如下：

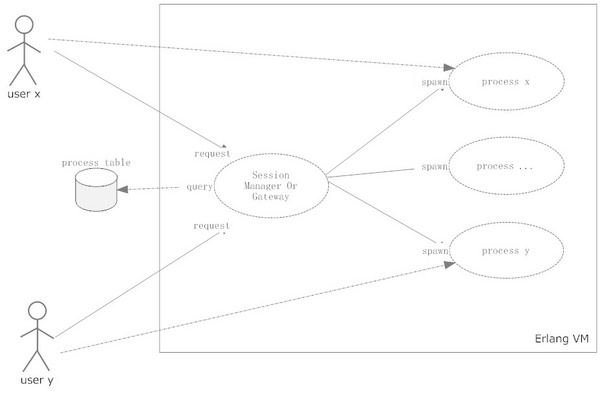


**3）WebSocket**。浏览器通过WebSocket协议连接服务端，实现了浏览器和服务器端的全双工通信。需要服务端和浏览器都支持WebSocket协议。

以上几种方式中，方式1实现较简单，但效率和实时效果较差。方式2对服务端实现的要求比较高，尤其是并发量大的情况下，对服务端的压力很大。方式3效率较高，但对较低版本的浏览器不支持，另外服务端也需要有支持WebSocket的实现。Worktile的WEB端实时消息推送，采用的是XMPP扩展协议XEP-0124 BOSH( <http://xmpp.org/extensions/xep-0124.html>)，本质是采用方式2长轮询的方式。踢踢网则采用了WebSocket连接RabbitMQ的方式实现，下面我会具体介绍如何用这两种方式实现Server Push。

## 运行时环境准备

服务端的实现中，无论采用Ejabberd还是RabbitMQ，都是基于Erlang语言开发的，所以必须安装Erlang运行时环境。Erlang是一种函数式语言，具有容错、高并发的特点，借助OTP的函数库，很容易构建一个健壮的分布式系统。目前，基于Erlang开发的产品有，数据库方面：Riak（Dynamo实现）、CouchDB， Webserver方面：Cowboy、Mochiweb， 消息中间件有RabbitMQ等。对于服务端程序员来说，Erlang提供的高并发、容错、热部署等特性是其他语言无法达到的。无论在实时通信还是在游戏程序中，用Erlang可以很容易为每一个上线用户创建一个对应的Process，对一台4核8个G的服务器来说，承载上百万个这样的Process是非常轻松的事。下图是Erlang程序发起Process的一般性示意图：



如图所示，Session Manager(or Gateway)负责为每个用户（UID）创建相对应的Process， 并把这个对应关系（MAP）存放到数据表中。每个Process则对应用户数据，并且他们之间可以相互发送消息。Erlang的优势就是在内存足够的情况下创建上百万个这样的Process，而且它的创建和销毁比JAVA的Thread要轻量的多，两者不是一个数量级的。