Evan
Only let oneself become strong enough, good enough, can afford the life that you want to.

Android实现推送方式解决方案

标签： android 推送 解决方案

2016-01-03 00:08 190人阅读 评论(0) 收藏 举报

分类： 移动开发（38）

本文介绍在Android中实现推送方式的基础知识及相关解决方案。推送功能在手机开发中应用的场景是越来越来了，不说别的，就我们手机上的新闻客户端就时不j时的推送过来新的消息，很方便的阅读最新的新闻信息。这种推送功能是好的一面，但是也会经常看到很多推送过来的垃圾信息，这就让我们感到厌烦了，关于这个我们就不能多说什么了，毕竟很多商家要做广告。本文就是来探讨下Android中实现推送功能的一些解决方案，也希望能够起到抛砖引玉的作用。^_^

1.推送方式基础知识：

在移动互联网时代以前的手机，如果有事情发生需要通知用户，则会有一个窗口弹出，将告诉用户正在发生什么事情。可能是未接电话的提示，日历的提醒，或是一封新的彩信。推送功能最早是被用于Email中，用来提示我们新的信息。由于时代的发展和移动互联网的热潮，推送功能更加地普及，已经不再仅仅用在推送邮件了，更多地用在我们的APP中了。

当我们开发需要和服务器交互的应用程序时，基本上都需要获取服务器端的数据，比如《地震应急通》就需要及时获取服务器上最新的地震信息。要获取服务器上不定时更新的信息，一般来说有两种方法：第一种是客户端使用Pull（拉）的方式，就是隔一段时间就去服务器上获取一下信息，看是否有更新的信息出现。第二种就是 服务器使用Push（推送）的方式，当服务器端有新信息了，则把最新的信息Push到客户端上。这样，客户端就能自动的接收到消息。

虽然Pull和Push两种方式都能实现获取服务器端更新信息的功能，但是明显来说Push方式比Pull方式更优越。因为Pull方式更费客户端的网络流量，更主要的是费电量，还需要我们的程序不停地去监测服务端的变化。

在开发**Android**和iPhone应用程序时，我们往往需要从服务器不定的向手机客户端即时推送各种通知消息。我们只需要在**android**或IPhone的通知栏处向下一拉，就展开了Notification Panel，可以集中一览各种各样通知消息。目前**iOS**平台上已经有了比较简单的和完美的推送通知解决方案，我会在以后详细介绍IPhone中的解决方案，可是Android平台上实现起来却相对比较麻烦。

最近利用几天的时间对Android的推送通知服务进行初步的研究，也希望能和大家共同探讨一下。

2. 几种常见的解决方案实现原理：

1）轮询(Pull)方式：应用程序应当阶段性的与服务器进行连接并查询是否有新的消息到达，你必须自己实现与服务器之间的通信，例如消息排队等。而且你还要考虑轮询的频率，如果太慢可能导致某些消息的延迟，如果太快，则会大量消耗网络带宽和电池。

2）SMS(Push)方式：在Android平台上，你可以通过拦截SMS消息并且解析消息内容来了解服务器的意图，并获取其显示内容进 ，不错的想法，我就见过采用这个方案的应用程序。这个方案的好处是，可以实现完全的实时操作。但是问题是这个方案的成本相对比较多动公司缴纳相应的费用。我们目前很难找到免费的短消息发送网关来实现这种方案。

3）持久连接(Push)方式：这个方案可以解决由轮询带来的性能问题，但是还是会消耗手机的电池。**ios**平台的推送服务之所以工作的很好，是因为每一台手机仅仅保持一个与服务器之间的连接，事实上C2DM也是这么工作的。不过刚才也讲了，这个方案存在着很多的不足之处，就是我们很难在手机上实现一个可靠的服务，目前也无法与IOS平台的推送功能相比。

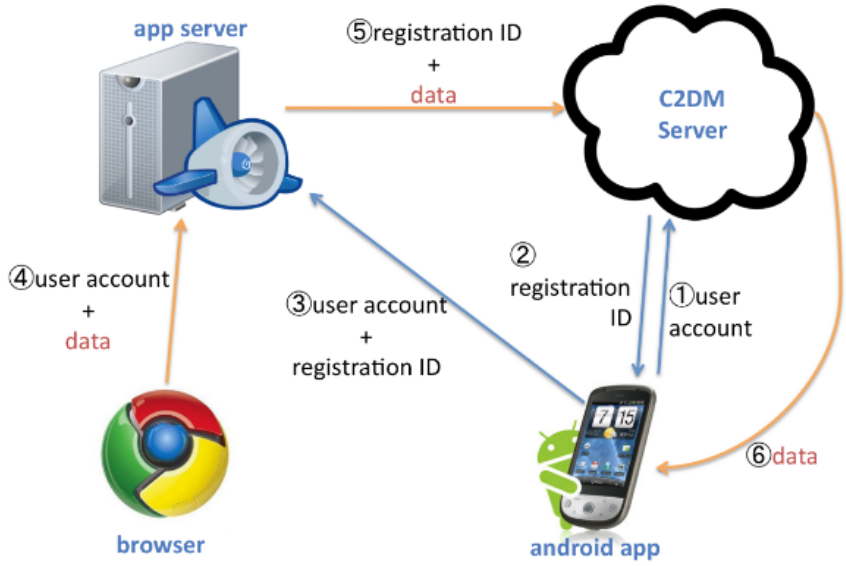
Android**操作系统**允许在低内存情况下杀死系统服务，所以我们的推送通知服务很有可能就被操作系统Kill掉了。 轮询(Pull)方式和SMS(Push)方式这两个方案也存在明显的不足。至于持久连接(Push)方案也有不足，不过我们可以通过良好的设计来弥补，以便于让该方案可以有效的工作。毕竟，我们要知道GMail，GTalk以及GoogleVoice都可以实现实时更新的。

3.第一种解决方案：C2DM云端推送功能。

在Android手机平台上，Google提供了C2DM（Cloudto Device Messaging）服务，起初我就是准备采用这个服务来实现自己手机上的推送功能，并将其带入自己的项目中。

Android Cloud to Device Messaging (C2DM)是一个用来帮助开发者从服务器向Android应用程序发送数据的服务。该服务提供了一个简单的、轻量级的机制，允许服务器可以通知移动应用程序直接与服务器进行通信，以便于从服务器获取应用程序更新和用户数据。C2DM服务负责处理诸如消息排队等事务并向运行于目标设备上的应用程序分发这些消息。关于C2DM具体使用过程，大家可以去查阅相关的资料，在这里先让我们了解下大致方案情况。

下面是C2DM操作过程示例图:



但是经过一番研究发现，这个服务存在很大的问题：

1）C2DM内置于Android的2.2系统上，无法兼容老的1.6到2.1系统;

2）C2DM需要依赖于Google官方提供的C2DM服务器，由于国内的网络环境，这个服务经常不可用，如果想要很好的使用，我们的App Server必须也在国外，这个恐怕不是每个开发者都能够实现的;

3) 不像在iPhone中，他们把硬件系统集成在一块了。所以对于我们开发者来说，如果要在我们的应用程序中使用C2DM的推送功] 的这种硬件厂商平台，比如摩托罗拉、华为、中兴做一个手机，他们可能会把Google的这种服务去掉，尤其像在国内就很多这种，把 的 服务去掉。买了一些像什么山寨机或者是华为这种国产机，可能Google的服务就没有了。而像在国外出的那些可能会内置。

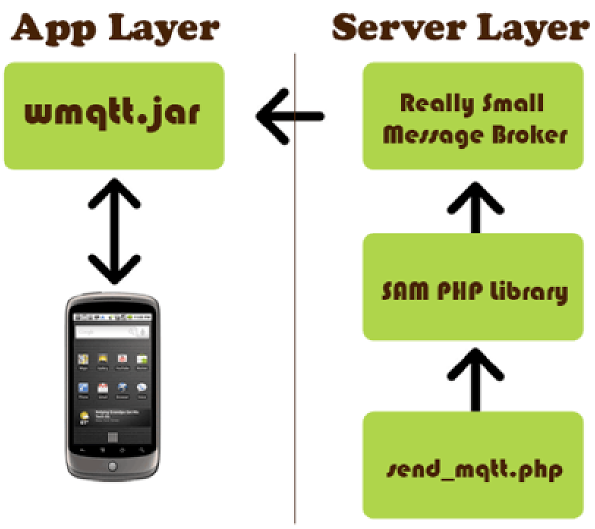
有了上述几个方面的制约，导致我最终放弃了这个方案，不过我想利用另外一篇文章来详细的介绍C2DM的框架以及客户端和App Server的相应设置方法，可以作为学习资源让我们有个参考的资料。 即然C2DM无法满足我们的要求，那么我们就需要自己来实现Android手机客户端与App Server之间的通信协议，保证在App Server想向指定的Android设备发送消息时，Android设备能够及时的收到。

4. 第二种解决方案：MQTT协议实现Android推送功能。

采用MQTT协议实现Android推送功能也是一种解决方案。MQTT是一个轻量级的消息发布/订阅协议，它是实现基于手机客户端的消息推送服务器的理想解决方案。

wmqtt.jar 是IBM提供的MQTT协议的实现。我们可以从这里（https://github.com/tokudu/AndroidPushNotificationsDemo）下载该项目的实例代码，并且可以找到一个采用**PHP**书写的服务器端实现（https://github.com/tokudu/PhpMQTTClient）。

架构如下图所示：



wmqtt.jar 是IBM提供的MQTT协议的实现。我们可以从如下站点下载（http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24006006）它。我们可以将该jar包加入自己的Android应用程序中。

5.第三种解决方案：RSMB实现推送功能。

Really Small Message Broker (RSMB) ，他是一个简单的MQTT代理，同样由IBM提供，其查看地址是：http://www.alphaworks.ibm.com/tech/rsmb。缺省打开1883端口，应用程序当中，它负责接收来自服务器的消息并将其转发给指定的移动设备。

SAM是一个针对MQTT写的PHP库。我们可以从这个http://pecl.php.net/package/sam/download/0.2.0地址下载它.

send_mqtt.**php**是一个通过POST接收消息并且通过SAM将消息发送给RSMB的PHP脚本。

6. 第四种解决方案：XMPP协议实现Android推送功能。

这是我希望在项目中采用的方案，因为目前它是开源的，对于其简单的推送功能它还是能够实现的。我们可以修改其源代码来适应我们的应用程序。

事实上Google官方的C2DM服务器底层也是采用XMPP协议进行的封装。XMPP(可扩展通讯和表示协议)是基于可扩展标记语言（XML）的协议，它用于即时消息（IM）以及在线探测。这个协议可能最终允许因特网用户向因特网上的其他任何人发送即时消息。关于XMPP协议我在上篇博文中已经介

绍，大家可以参考下文章：http://www.cnblogs.com/hanyonglu/archive/2012/03/04/2378956.html

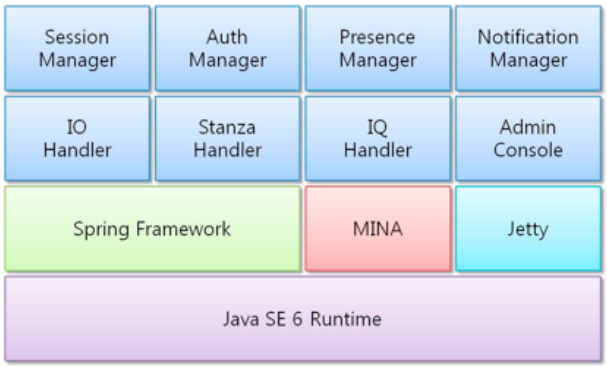
androidpn是一个基于XMPP协议的**Java**开源Android push notification实现，我会在以后的博文中详细介绍androidpn。它包含
和服务器端。经过源代码研究我发现，该服务器端基本是在另外一个开源工程openfire基础上修改实现的，不过比较郁闷的是androidpn的文档是由韩
语写的，所以整个研究过程基本都是读源码。

这是androidpn的项目主页：http://sourceforge.net/projects/androidpn/

androidpn实现意图如下图所示：

androidpn 客户端需要用到一个基于java的开源XMPP协议包asmack，这个包同样也是基于openfire下的另外一个开源项目smack，不过我们不需要自己编译，可以直接把androidpn客户端里面的asmack.jar拿来使用。客户端利用asmack中提供的XMPPConnection类与服 务器建立持久连接，并通过该连接进行用户注册和登录认证，同样也是通过这条连接，接收服务器发送的通知。

androidpn服务器端也是java语言实现的，基于openfire开源工程，不过它的Web部分采用的是**spring**框架，这一点与 openfire是不同的。Androidpn服务器包含两个部分，一个是侦听在5222端口上的XMPP服务，负责与客户端的 XMPPConnection类进行通信，作用是用户注册和身份认证，并发送推送通知消息。另外一部分是Web服务器，采用一个轻量级的HTTP服务器， 负责接收用户的Web请求。服务器架构如下：



最上层包含四个组成部分，分别是SessionManager，Auth Manager，PresenceManager以及Notification Manager。SessionManager负责管理客户端与服务器之间的会话，Auth Manager负责客户端用户认证管理，Presence Manager负责管理客户端用户的登录状态，NotificationManager负责实现服务器向客户端推送消息功能。

这个解决方案的最大优势就是简单，我们不需要象C2DM那样依赖操作系统版本，也不会担心某一天Google服务器不可用。利用
可以进一步的对协议进行扩展，实现更为完善的功能。 采用这个方案，我们目前只能发送文字消息，不过对于推送来说一般足够了，
通过推送得到所有的数据，一般情况下，利用推送只是告诉手机端服务器发生了某些改变，当客户端收到通知以后，应该主动到服务器
这样才是推送服务的完整实现。 XMPP协议书相对来说还是比较简单的，值得我们进一步研究。

但是在经过一段时间的**测试**，我发现关于androidpn也存在一些不足之处：

1. 比如时间过长时，就再也收不到推送的信息了。
2. 性能上也不够稳定。
3. 如果将消息从服务器上推送出去，就不再管理了，不管消息是否成功到达客户端手机上。

等等，总之，androidpn也有很多的缺点。如果我们要使用androidpn，则还需要做大量的工作。

至于详细使用过程，我们会在下个博文中再给大家介绍。

7.第五种解决方案：使用第三方平台。

第三方平台有商用的也有免费的，我们可以根据实现情况使用。关于国内的第三方平台，我感觉目前比较不错的就是极光推送。关于极光推送目前是免费的，我们可以直接使用。关于详细情况，大家可以查看它的主页：http://www.jpush.cn/index.jsp，这里不再详细描述。

关于MQTT的方案，国内最近出现基于 mqtt 的第三方解决方案 云巴 (http://yunba.io)，据了解是原 极光推送 CTO 创办的，有兴趣的朋友可以研究下。

关于国外的第三方平台我也见过几个：http://www.push-notification.org/。有兴趣的朋友可以查阅相关信息。使用第三方平台就需要使用别人的服务器，关于这点，你懂的。

8.第六种解决方案：自己搭建一个推送平台。

这不是一件轻松的工作，当然可以根据各自的需要采取合适的方案。

好了，以上是关于在Android中实现推送方式的基础知识及相关解决方案。

转载地址：http://www.cnblogs.com/hanyonglu/archive/2012/03/04/2378971.html

- 上一篇 Android之Android WebView常见问题及解决方案汇总
- 下一篇 android极光推送初步了解

相关文章推荐

- Android实现推送方式解决方案
- android推送解决方案（1）