Android 推送模块技术选型及结果报告

版本	发布时间	修订章节	作者
v1.0	2017/06/13	初版	wz

目的：

1. 介绍 Android 客户端推送模块的技术选型过程
2. 介绍 Android 客户端推送模块的开发文档使用及注意事项

第一部分：推送平台技术选型

一、什么是推送？

一般用户使用 APP 时，都是用户主动操作去获取服务端的数据。**推送**则是服务端主动推送消息到App，用户属于被动获取，可以选择接收或者忽略。推送是一种不可或缺的运营工具。

二、推送能做什么？

- 向用户推送消息。如：运营活动等
- 唤醒长时间未使用但未卸载 **APP** 客户端的用户，增加用户留存的可能性
- 及时送达某些耗时操作的状态。如用户要下载某个东西，但耗时较长，可能切换到其他应用，当下载完成可以提示用户已下载完成。

三、 实现 Android 端推送要解决的问题

1. 问题

- 解决 **APP** 在运行状态下的推送问题
- 解决 **App** 进程在被系统杀死之后，仍然需要保证消息的送达问题

2.出现这些问题的原因

国内 **Android** 系统定制化严重，删除了 **Android** 原生系统提供的 **Google serice** 模块，没办法使用 **Android** 原生系统的 **GCM** 推送内部通道。

国内各大手机厂商为了保证手机的续航能力和使用的流畅性，会对长时间处于后台运行的进程进行清理，同时把自己的推送模块写入系统进程，来保证走自己的推送通道的应用消息的正常送达。而其他推送进程则直接被杀死。

四、如何解决推送问题

> 如果是面对国外的用户，则应用推送直接选择 **GCM** 即可。
>
> 如果是国内用户，则要复杂的多。下面是是针对国内用户不同体量公司的解决方案。

1. 对于大型公司

- App 处于前台运行的时候，走自己搭建的内部通道。客户端维护一条长连接连到自己的推送服务器，不依赖第三方的推送系统。
- App 处于后台运行但还没有被系统杀死之前，这时推送仍然走自己的内部通道。通过 “进程保活” 来减少被系统杀死的概率或者延长存活时间。
- App 长时间处于后台被系统杀死。走第三方的推送平台。

使用这种推送方案的好处：

- 更快的送达，保证消息的及时性。
- 走自己的通道能保证送达率。
- 走内部通道时数据只在内部流通，保证数据安全
- 一般会有专们的团队维护推送相关服务，有保障
- 内部通道加第三方通道结合的方式能最大化的保证消息 **送达率** 的问题。

一般大公司的策略是优先走内部通道，然后再选择走第三方

2. 对于初创公司

自己去搭建内部的推送平台相比于使用第三方来说性价比较低，所以一般使用第三方的推送平台来解决推送问题。

就目前而言，自己内部搭建推送服务性价比不高，因此，如何选择第三方推送服务十分重要。

五、 第三方推送平台的技术选型

1.下表为国内现有的排名靠前的第三方推送平台

平台	是否收费	说明
极光	有免费也有收费	
个推	有免费也有收费	
百度云推送	免费	
信鸽	免费	
华为推送	免费	
友盟推送	免费	集成有小米和华为推送通道
阿里云推送	收费	集成有小米和华为推送通道
小米推送	免费	

2.收集资料：

除了各平台的开发者文档之外，以下为社区关于如何选择第三方推送的讨论

- Android端外推送到底有多烦？
- https://github.com/android-cn/topics/issues/4#issuecomment-223264673
- Android推送SDK哪家好？
- 国内Top500Android应用分析报告
- 手机推送系统中的坑和解决方法

3.参与测试的手机机型

品牌	系统
小米 5	MIUI8.2(Android 7.0)
mx5	Flyme6.1.0.0A(Android 5.1)
LG	Android 7.0
三星	android 6.0
OPPO R9m	ColorOS 3.0.0(Android 5.1)
华为mete8	EMUI 5.0(Android 7.0)

手上无 **VIVO**手机无法测试 **Funtouch3.0(android 6.0)**

4. 调研结果

个推

	LG(原生)	三星	小米(MIUI)	魅族(Flyme)	OPPO(ColorOs)	华为(EMui)
app处于前台	√	√	√	√	√	√
返回键退出	√	√	√	√	√	√
杀掉APP	√	√	×	×	×	×
手机重启后	√	√	×	×	×	×

> 测试说明(下面的测试也相同)：
>
> （√：能收到推送，×：收不到推送，-：无意义的测试）
>
> 杀死APP操作：进入任务管理器页面，删除当前任务

在OPPO 手机上测试默认APP不显示推送，需要用户自己打开推送通知。集成时耗费时间较长，后续其他功能未做测试。没有开发者社区，出现问题没有地方反馈问题。 在国内第三方定制的机型上，存在诸多限制，要想杀掉APP和手机重启后能自启动APP，需要引导用户开启相关权限或者设置。

友盟推送

	LG(原生)	三星	小米(MIUI)	魅族(Flyme)	OPPO(ColorOs)	华为(EMui)
app处于前台	√	√	√	√	√	√
返回键退出	√	√	√	√	√	√
杀掉APP	√	√	×	×	×	×
手机重启后	√	√	×	×	×	×

同个推结果一样。在应用第一次安装后接收推送有点慢。集成时很简单，直接引入一个module，然后再初始化即可，开发者文档及社区很完善，遇到问题能及时找到解决办法。

关于特殊机型的整理 http://bbs.umeng.com/thread-21334-1-1.html

> 注：需要使用消息路由功能，要按照下面的方式设置：
>
> 小米【MIUI7、MIUI8】 安全中心-授权管理-自启动管理，需把应用添加到允许自启动列表里 华为【EMUI 4】 手机管家-权限管理-自启动管理-关联启动，需允许应用被其他应用启动 （首次触发相互唤醒后，应用才会出现在关联启动的应用列表里） 魅族 【Flyme 5】 手机管家-权限管理-自启动管理-相互启动，需允许相互启动 （Flyme 5.1.9.0A及以上版本没有相互启动选项，只有自启动选项，需允许自启动） 【Flyme 6】 手机管家-后台管理，应用需勾选保持后台运行 VIVO【Funtouch OS_2.5】 i管家-软件管理-自启动管理-关联启动，需允许应用的关联启动 （首次触发相互唤醒后，应用才会出现在关联启动的应用列表里） 努比亚【Nubia UI v3.7.8及以上】 手机管家-授权管理-自启动管理，需允许自启动

极光推送

	LG(原生)	三星	小米(MIUI)	魅族(Flyme)	OPPO(ColorOs)	华为(EMui)
app处于前台	√	√	√	√	√	√
返回键退出	√	√	√	√	√	√
杀掉APP	√	√	×	×	×	×
手机重启后	√	√	×	×	×	×

结果同其他第三方相同。集成方式很简单，使用 android studio 开发的话直接一句代码依赖SDK，然后添加SDK配置。文档详细，社区很完善。

关于

小米推送

由于手上没有小米开发者帐号，测试不了

下面为网友的测试数据

	LG(原生)	三星	小米(MIUI)	魅族(Flyme)	OPPO(ColorOs)	华为(EMui)
app处于前台	√	√	√	√	√	√
返回键退出	√	√	√	√	√	√
杀掉APP	√	√	√	×	×	×
手机重启后	√	√	√	×	×	×

可以看到小米的推送在自己的机型上运行良好。

> 官网上说:小米推送服务走的系统级别的服务，在送达率和省点方面做了很多优化

华为推送

无华为开发者帐号，暂时无法测试

第二部分 推送方案

通过查看友盟数据中心的数据报告，出货量排名靠前的分别是：华为、OPPO、VIVO、小米、三星、魅族等。结合网上其他开发给的方案以及自身的测试最终选择以下的测试方案。

> 接入小米和友盟共同推送方案。策略是在 MIUI 上启用小米推送，不启用友盟推送；非 MIUI 上启用 友盟推送，废弃小米推送。鉴于没有华为开发者帐号暂时不考虑接入华为推送。

如何判断用户 ROM 是 MIUI 系统

```
    public static boolean isMiUi() {
            return !TextUtils.isEmpty(getSystemProperty("ro.miui.ui.version.name"));
        }

        public static String getSystemProperty(String propName) {
            String line;
            BufferedReader input = null;
            try {
                java.lang.Process p = Runtime.getRuntime().exec("getprop " + propName);
                input = new BufferedReader(new InputStreamReader(p.getInputStream()), 1024);
                line = input.readLine();
                input.close();
            } catch (IOException ex) {
                return null;
            } finally {
                if (input != null) {
                    try {
                        input.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
            return line;
        }
```

因为推送服务是采用另一个进程，所以application会被启动多次，因此必须要判断是哪个进程启动的application，这样就可以让主进程才需要的配置工作对push进程屏蔽。开发者需要做的就是判断是否是主进程，如果是那么就执行原本application中的代码。

```
/**
 * 注意：因为推送服务等设置为运行在另外一个进程，这导致本Application会被实例化两次。
 * 而有些操作我们需要让应用的主进程时才进行，所以用到了这个方法
 */
public static boolean isInMainProcess(Context context) {
```

```
    ActivityManager am = ((ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE));
    List<ActivityManager.RunningAppProcessInfo> processes = am.getRunningAppProcesses();
    String mainProcessName = context.getPackageName();
    int myPid = Process.myPid();
    for (ActivityManager.RunningAppProcessInfo info : processes) {
        if (info.pid == myPid && mainProcessName.equals(info.processName)) {
            return true;
        }
    }
    return false;
}
```

千万要注意！！！！！！！！

因为推送服务的复活会启动application，即唤醒应用。所以千万不要在application启动的时候进行网络请求的操作。因为一旦开始推送，大量用户的应用会被唤醒，然后再几分钟内会产生对于少数api的大量请求。后台服务很可能会被搞挂，所以千万要小心这点。这也就是为什么我强烈不推荐application中做很多事情的原因之一。可以考虑把这些请求转移到Activity中进行。

具体的代码：

- 友盟推送DEMO
- 极光推送DEMO
- 个推DEMO

注意：最终方案需要小米开发者帐号，暂时未给出相关代码。