

提要

微信ANDROID客户端的架构演进史，可以说是一个典型ANDROID应用在从小到大的成长过程中的“踩坑”与“填坑”的历史。从1.0版本安装包的354KB，到今天5.3版本的24.1MB，从最开始两三个码农的突击作业，到今天的“集团军”开发力量，微信的体量在不断增大，开发同学遇到的“成长的烦恼”也越来越多：

- * 为什么微信收消息又延迟了？为什么我得每次打开微信才收到消息？
- * 为什么我的微信无法安装了？为什么微信启动越来越慢了？
- * 为什么我的eclipse突然无法debug微信了！？如何把编译速度提升80%？
- * 如何在一个月左右的周期内排入5个迭代？如何并行发布3个以上代码线的客户端版本？
- * 如何减小因为增加开发人力而带来的资源损耗？

ANDROID系统先天的弊端与产品需求研发过程的矛盾，推动着客户端架构演进史这架车轮不断向前滚动。不断调整进化的架构，在为微信未来的高速成长保驾护航。欢迎各位和我们一起来了解微信ANDROID客户端的架构演进过程。



微信ANDROID客户端架构演进及其 对开发流程的影响

赵原
2014年7月



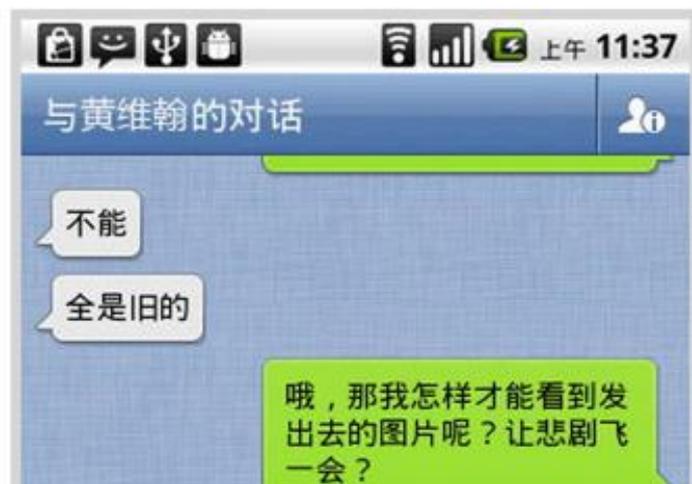


发布日期：2011-01-24

发布版本：微信1.0 for Android(测试版) [点击下载](#)

基于Android平台的腾讯微信服务，带给您全新的消息体验，您可以使用微信快速收发消息，即时拍照分享，随时随地联系身边的朋友。支持基于Android平台的手持终端设备。

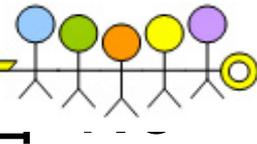
快速消息：极速轻快的楼层式对话，带给您飞一般的聊天体验。



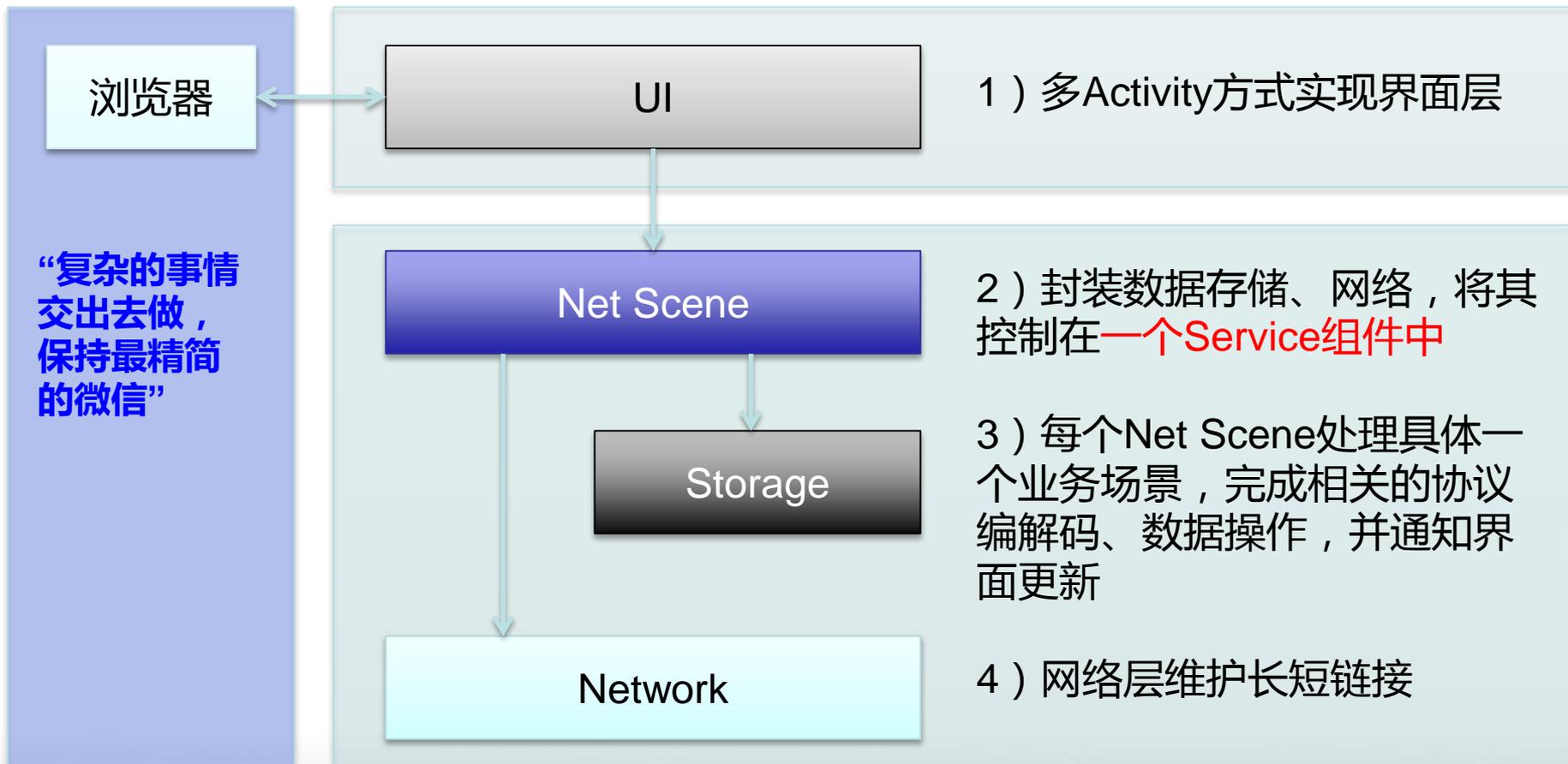


微信客户端架构V1

微信



火龙果•整理
uml.org.cn





“成长的烦恼”

产品侧

- 一到两周的快速版本迭代
- 追求更好的用户体验
- 试错与新功能的快速叠加

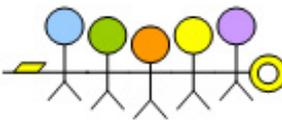


问题：

- 代码、安装包、内存体积膨胀
- 用户环境的复杂
- 系统组件的缺陷



微信的第一个大问题



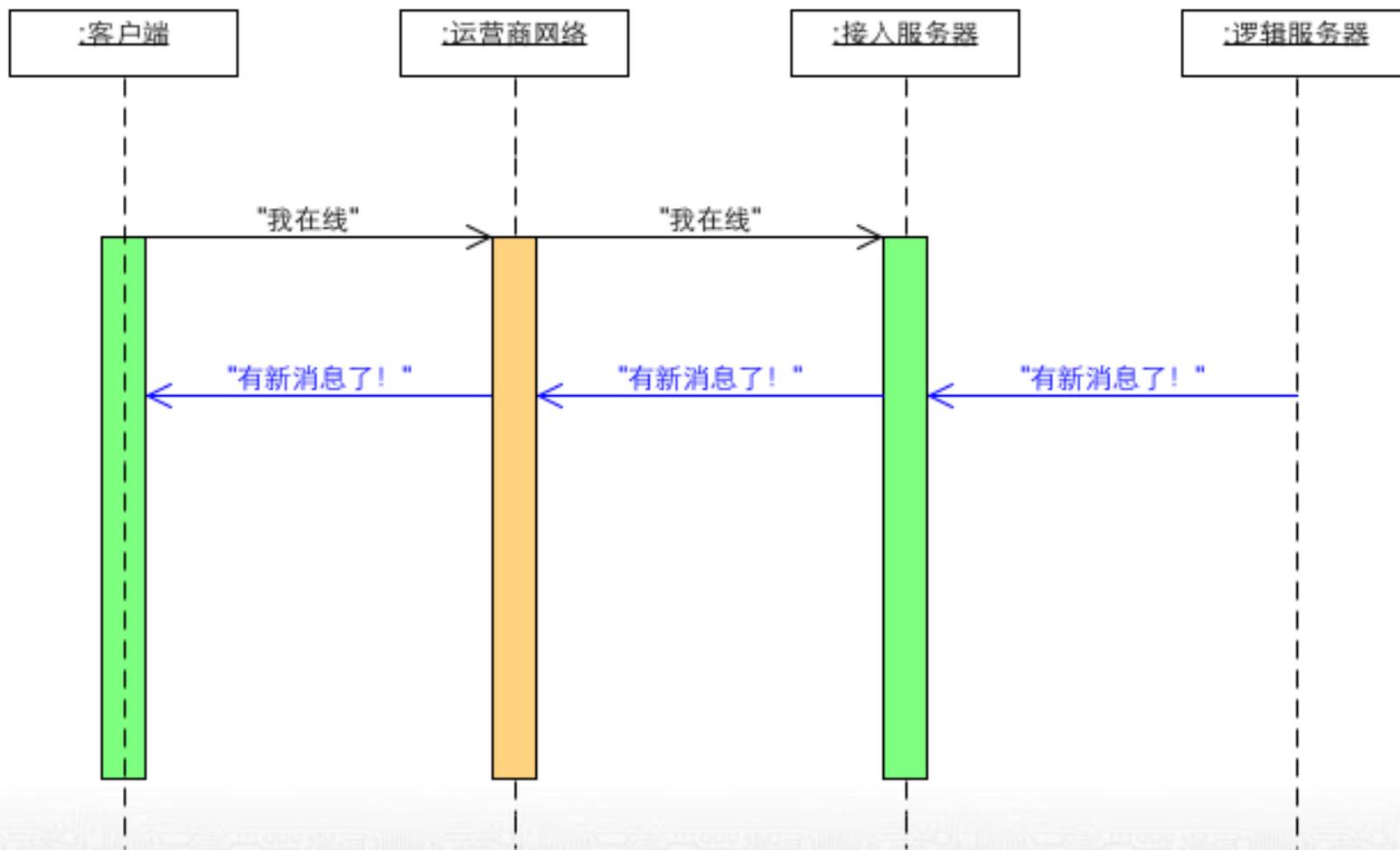
消息推送不及时

微信作为一款取代短信的即时通信程序，消息推送的及时性是早期遇到的最大的问题

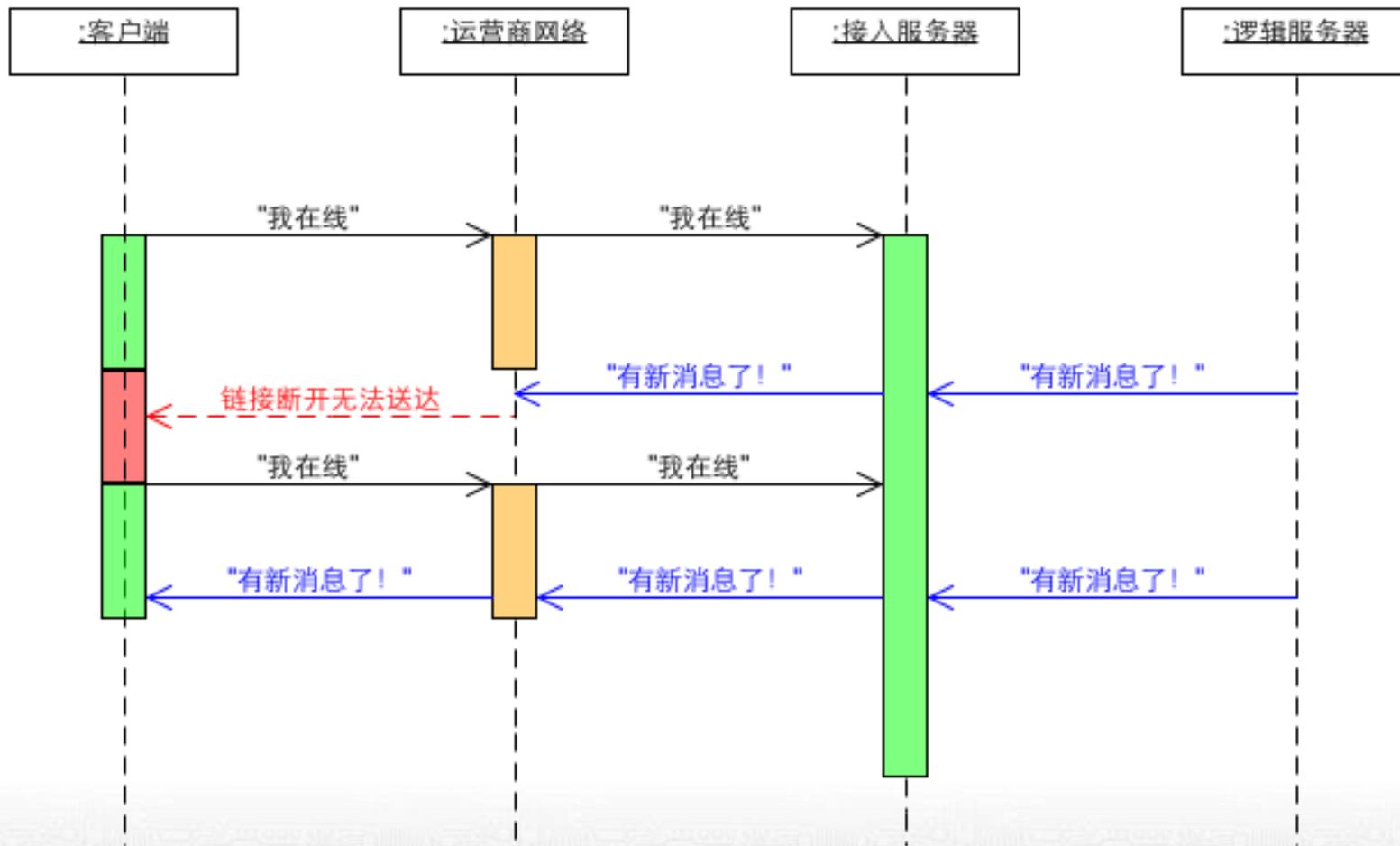
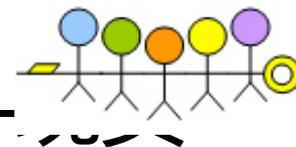
- 1.5的时代是没有GCM/C2DM通知机制的
- 国内网络的特殊性，需要维持准确的心跳周期
- 微信的膨胀与Android的进程与内存回收机制的矛盾



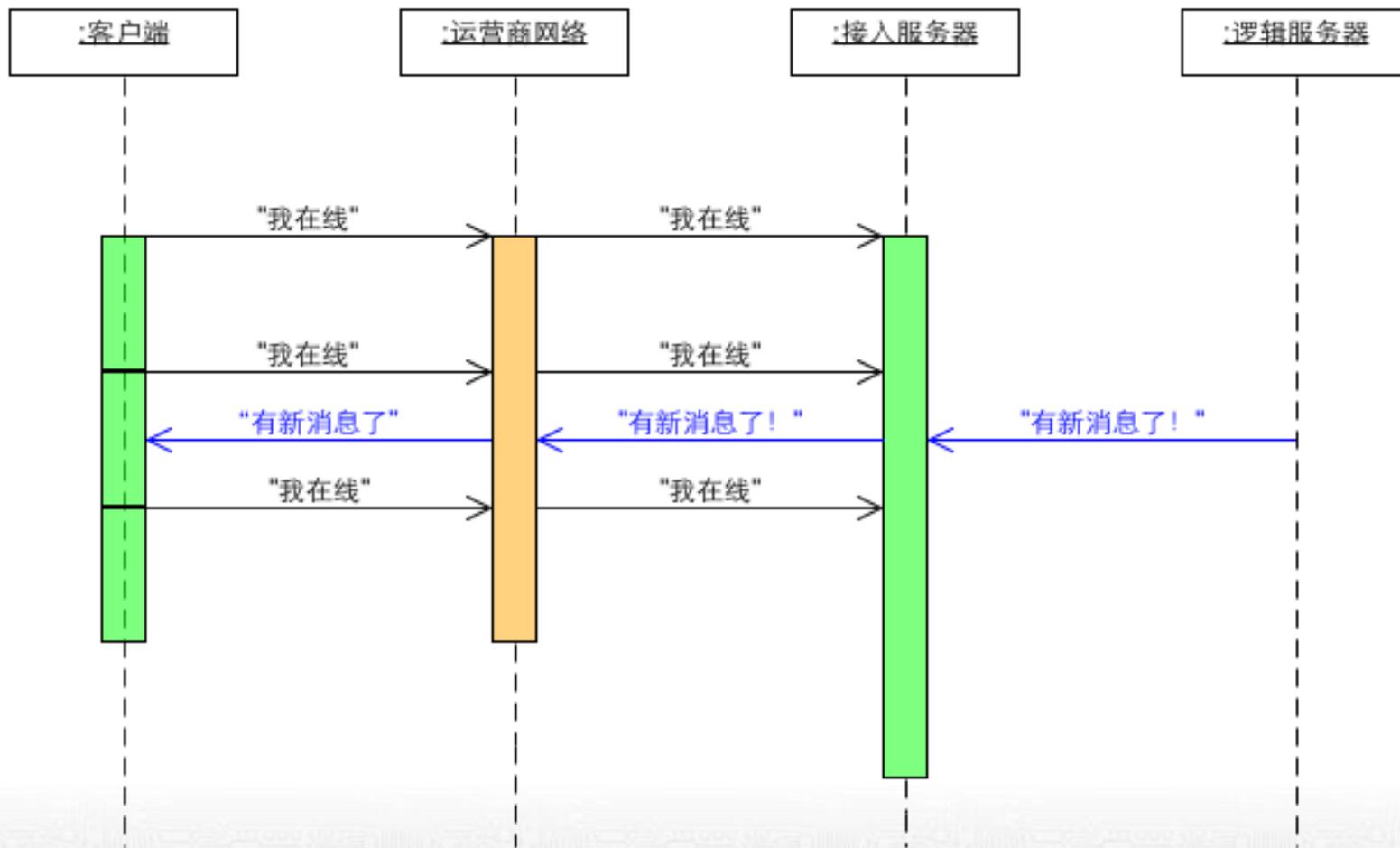
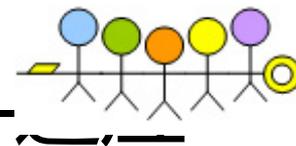
国内的移动网络环境



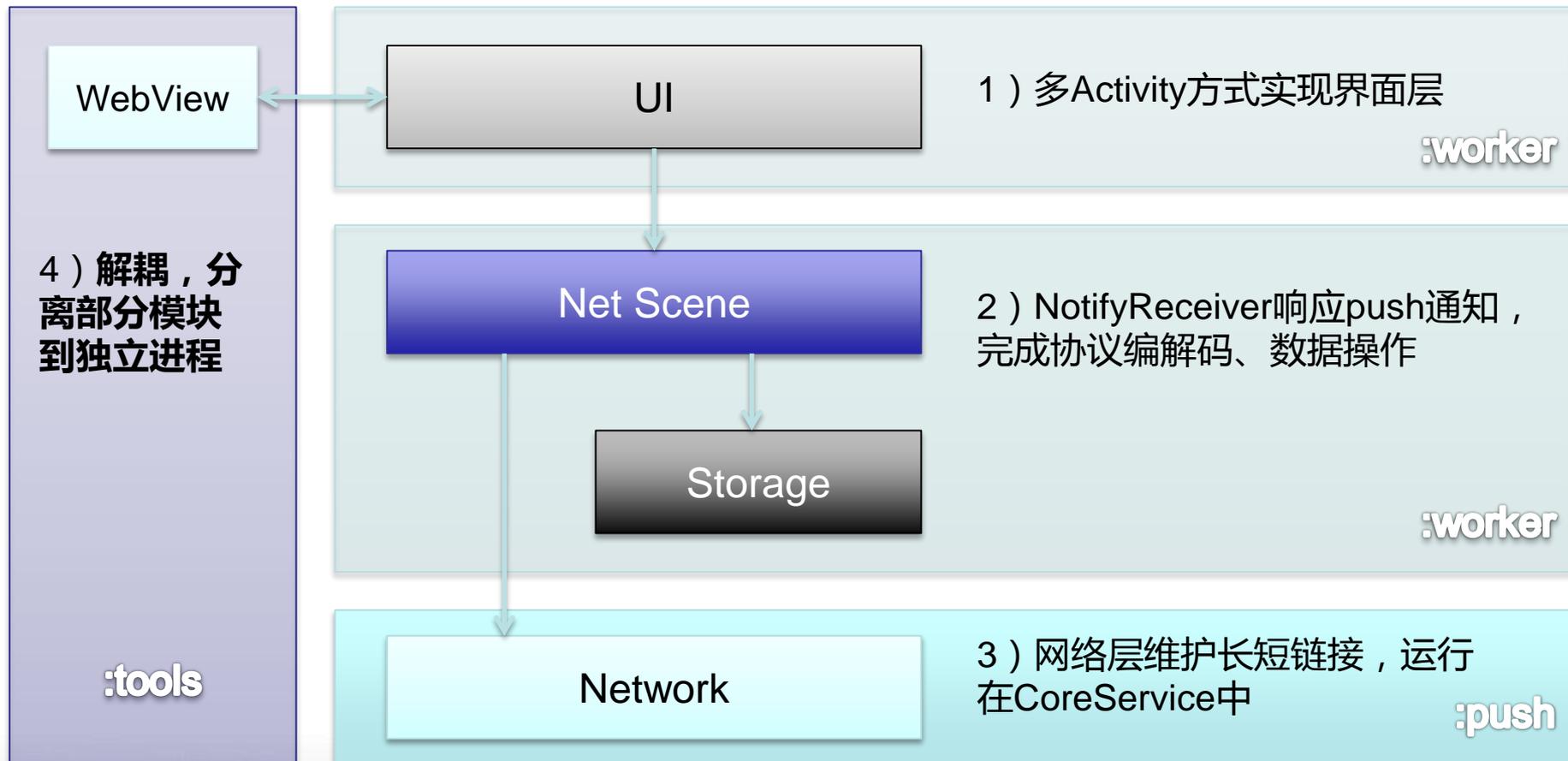
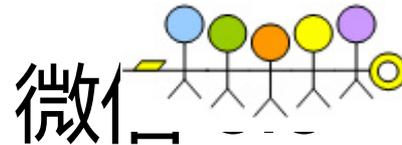
国内的移动网络环境



国内的移动网络环境



微信客户端架构V2 —— 微信





鲜花 VS 鸡蛋

优点：

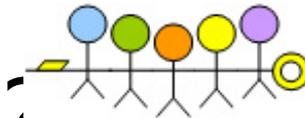
- 内存消耗降低
- 推送稳定性大幅提升
- 耗电降低

缺点：

- 启动速度变慢
- 缓存失效
- 系统资源的消耗实际上的增加



“微信无法发布了！”



·dexopt在2.3以下系统上遇到的无法安装问题

Tag	Text
dalvikvm	creating instr width table
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/ISms;' has an earlier definition; blocking out
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/ISms\$Stub\$Proxy;' has an earlier definition; blocking out
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/ISms\$Stub;' has an earlier definition; blocking out
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/ITelephony;' has an earlier definition; blocking out
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/ITelephony\$Stub\$Proxy;' has an earlier definition; blocking out
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/ITelephony\$Stub;' has an earlier definition; blocking out
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/SmsRawData\$1;' has an earlier definition; blocking out
dalvikvm	DexOpt: 'Lcom/android/internal/telephony/SmsRawData;' has an earlier definition; blocking out
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/ISms;': multiple definitions
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/ISms\$Stub\$Proxy;': multiple definitions
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/ISms\$Stub;': multiple definitions
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/ITelephony;': multiple definitions
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/ITelephony\$Stub\$Proxy;': multiple definitions
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/ITelephony\$Stub;': multiple definitions
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/SmsRawData\$1;': multiple definitions
dalvikvm	DexOpt: not verifying 'Lcom/android/internal/telephony/SmsRawData;': multiple definitions
dalvikvm	DexOpt: couldn't find field Landroid/graphics/BitmapFactory\$Options;.inMutable
dalvikvm	GC_FOR_MALLOC freed 2K, 1% free 2049K/2051K, external 0K/0K, paused 9ms
dalvikvm	LinearAlloc exceeded capacity (5242880), last=208
dalvikvm	VM aborting





问题定位

·2.3 (gingerbread)

[platform/dalvik.git]/vm/LinearAlloc.c, line 72

```
71 /* default length of memory segment (worst case is probably "dexopt") */  
72 #define DEFAULT_MAX_LENGTH (5*1024*1024)  
73
```

·master (4.x)

[platform/dalvik.git]/vm/LinearAlloc.cpp, line 72

```
71 /* default length of memory segment (worst case is probably "dexopt") */  
72 #define DEFAULT_MAX_LENGTH (16*1024*1024)  
73
```



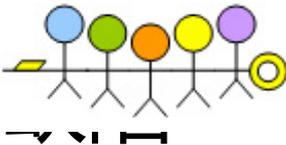


问题定位

- 重新编译内核，打印出dexopt过程中对LinearAlloc的引用

Tag	Text
dalvikvm	[Loaded Landroid/view/animation/AccelerateDecelerateInterpolator; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Ljava/sql/Time; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Landroid/app/TimePickerDialog; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/widget/TimePicker\$OnTimeChangedListener; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/media/Ringtone; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/widget/TimePicker; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/webkit/GeolocationPermissions\$Callback; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/view/Surface; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/media/ThumbnailUtils; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Ljava/io/BufferedWriter; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Landroid/database/DataSetObservable; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/database/Observable; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/database/ContentObservable; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Ljava/lang/UnsupportedOperationException; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Ljava/text/Collator; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Ljava/text/CollationKey; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Landroid/os/MemoryFile; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Landroid/os/ParcelFileDescriptor; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Ljava/util/concurrent/locks/ReentrantLock; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Ljava/util/concurrent/locks/Lock; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Ljava/util/WeakHashMap; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Landroid/util/Log; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Ljava/util/TreeMap; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Ljava/util/SortedMap; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Ljavax/crypto/CipherInputStream; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Landroid/database/SQLException; from DEX 0x365b0 (cl=0x0)]
dalvikvm	[Loaded Ljavax/crypto/CipherOutputStream; from DEX 0x173f0 (cl=0x0)]
dalvikvm	[Loaded Lorg/apache/harmony/dalvik/NativeTestTarget; from DEX 0x173f0 (cl=0x0)]
dalvikvm	DexOpt: load 9471ms, verify 21814ms, opt 1056ms

影响微信的两个关键系统



单dex 65535方法数限制：

Android在早期设计时留下的问题，dex文件中方法id用16位整型来标记，单个dex文件中的方法数因此无法超过65536

- 在编译中由dx过程触发
- 导致问题：eclipse中无法debug



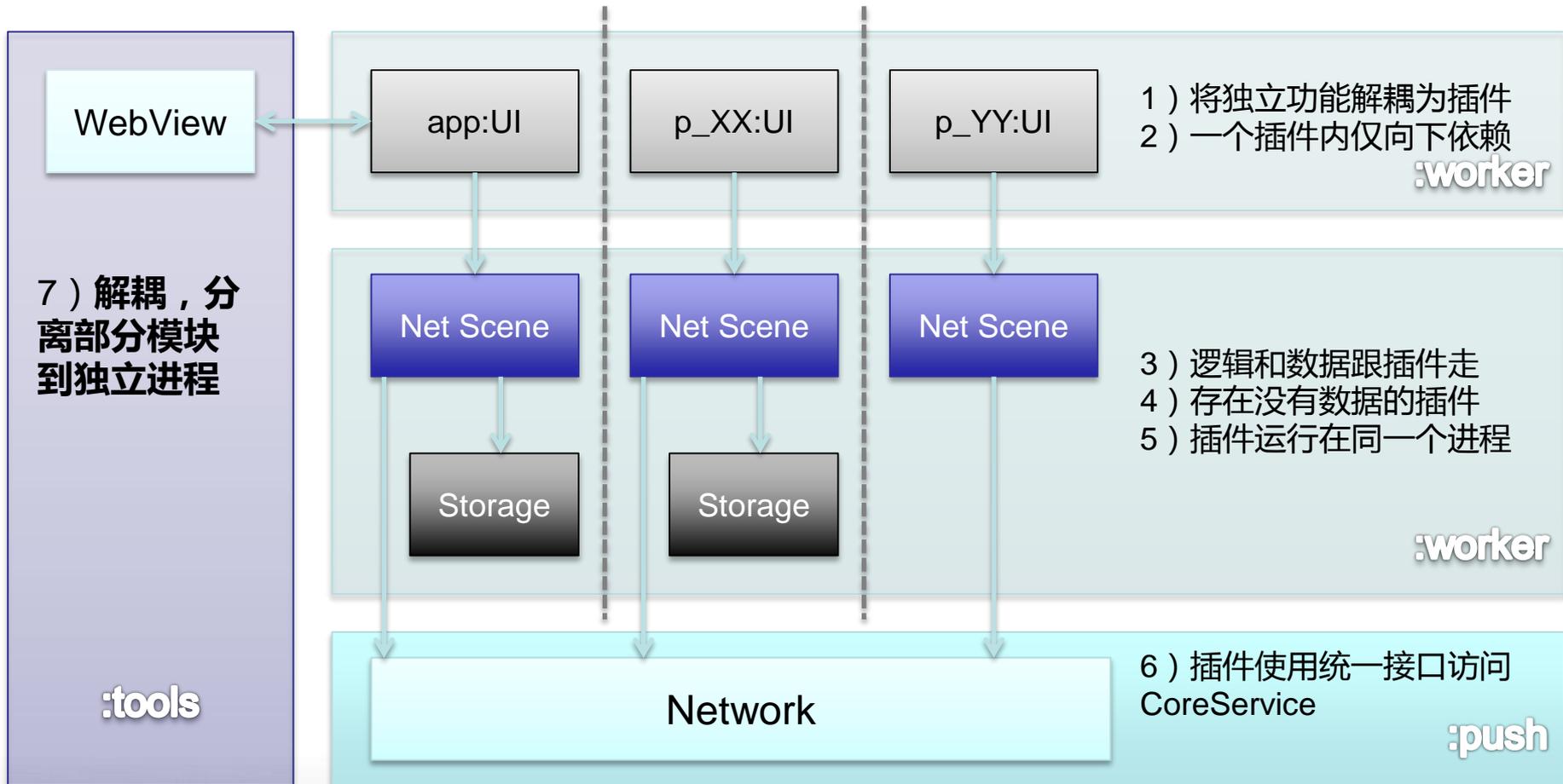
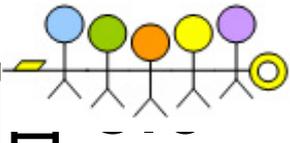
线性分配器限制：

线性分配器(LinearAlloc)大小限制，dalvik虚拟机用来加载类的堆内存大小，在代码中硬编码，2.3以下是5M，2.3以上是8M

- 在apk安装时，通过dexopt过程触发
- 导致问题：微信无法安装

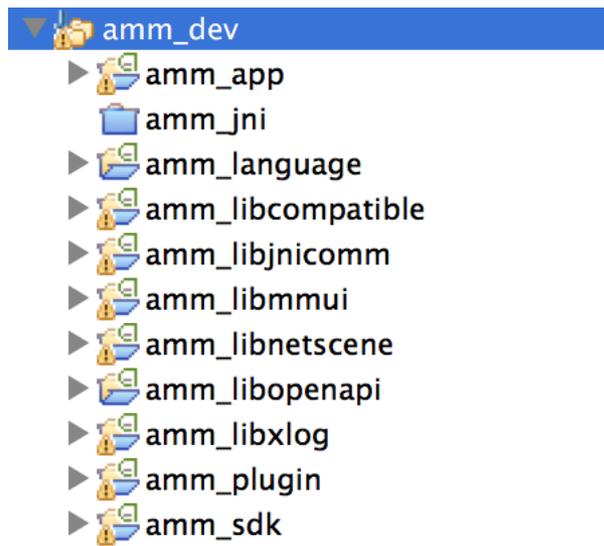


微信客户端架构V3



多工程分离

核心工程组 + 插件



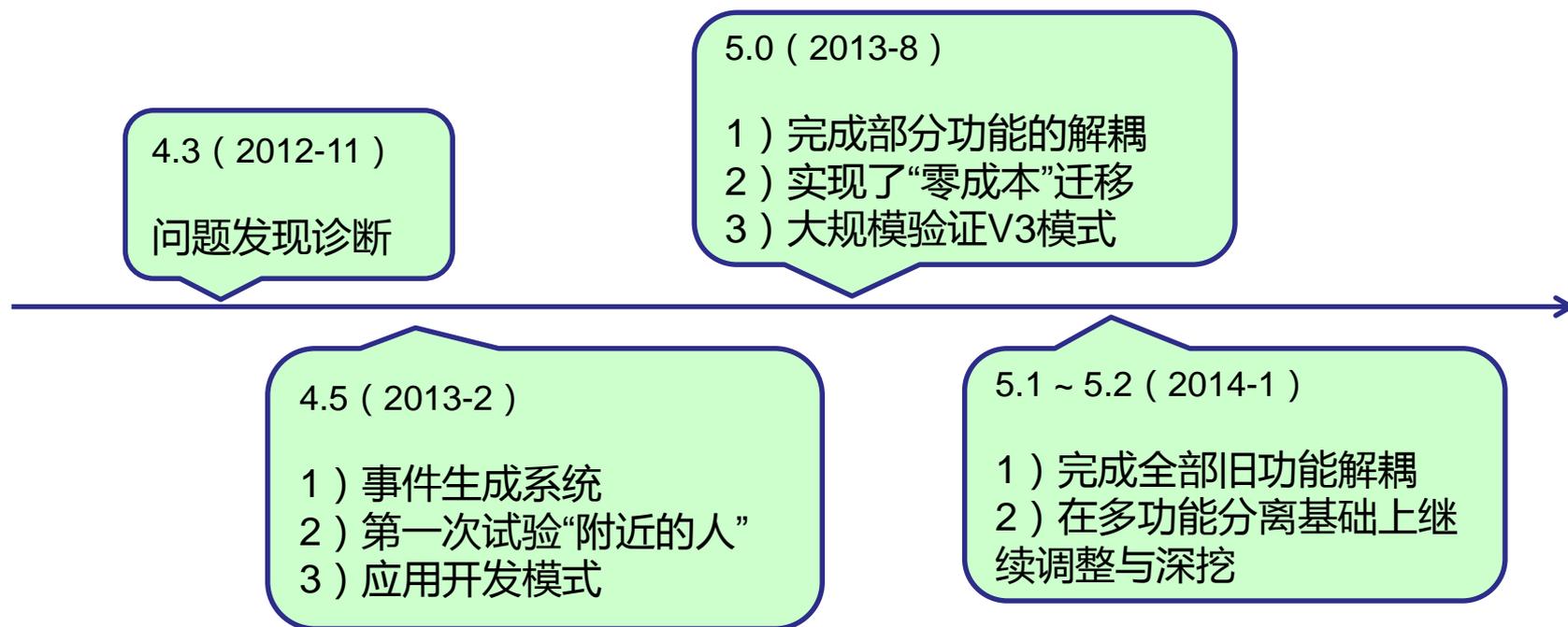
- p_accountsync
- p_bottle
- p_brandservice
- p_emoji
- p_emoticon
- p_ext
- p_favorite
- p_gallery
- p_gwallet
- p_masssend
- p_nearby
- p_qqmail
- p_qqsync
- p_radar
- p_readerapp
- p_sandbox
- p_scanner
- p_shake
- p_shoot
- p_shoot_stub
- p_sns
- p_sysvideo
- p_talkroom
- p_traceroute
- p_voip
- p_wallet
- p_webview
- p_whatsnew





多工程分离——从V2到V3

- 小步快跑，平稳迁移





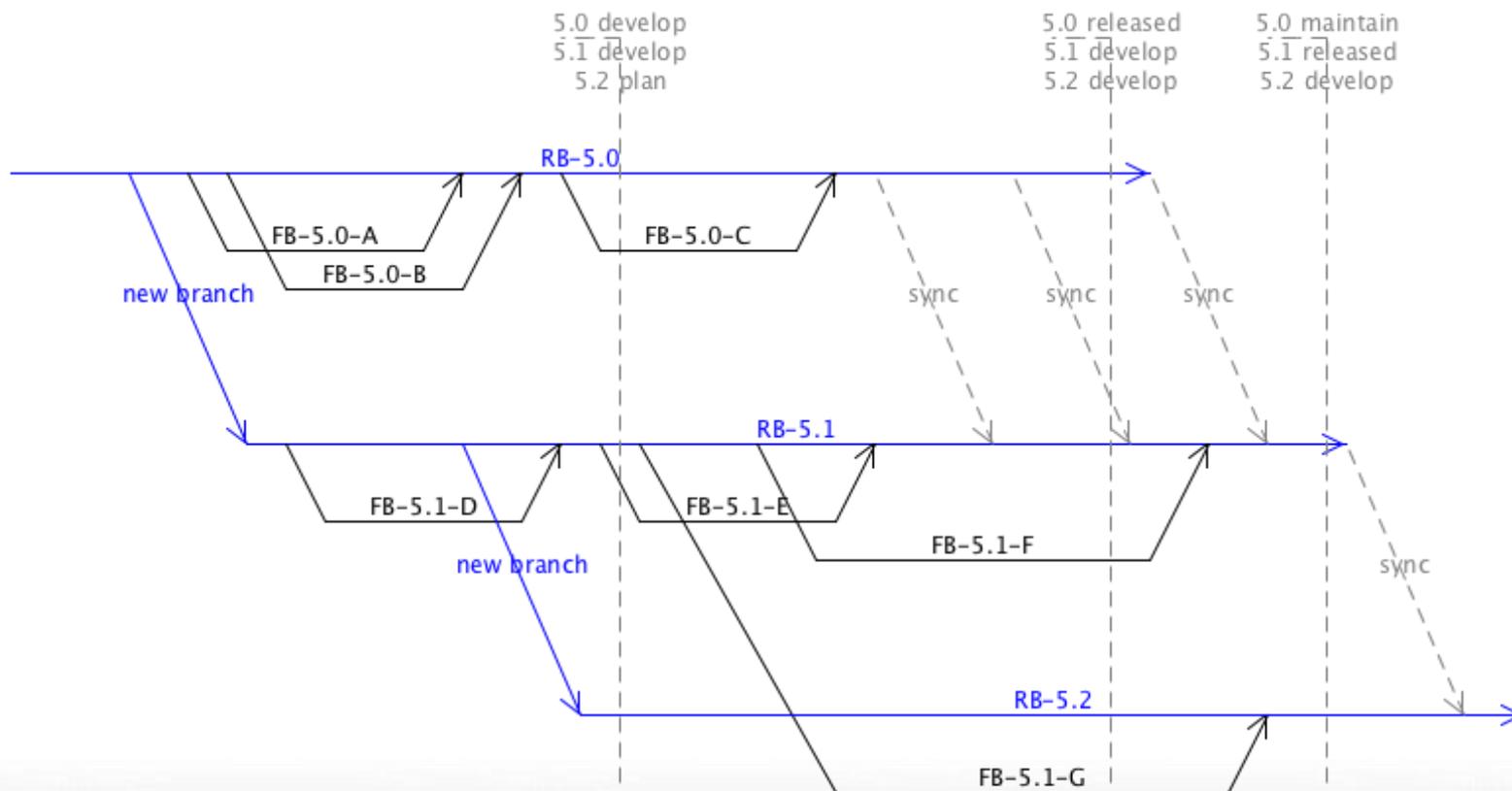
多工程分离——开发模式的改变

客户端架构	V1、V2	V3
团队规模	小型	中到大型
代码管理模式	<ul style="list-style-type: none">• 单trunk主线• 1~2个Release Branches	<ul style="list-style-type: none">• 3~4个瀑布型分支• 若干Feature Branches
版本迭代与发布	<ul style="list-style-type: none">• 单一迭代• 每1~2周发布一个版本	<ul style="list-style-type: none">• 1个半月完成1个版本发布• 每个版本有4~5个迭代• 同时存在3~4个版本发布
特点	<ul style="list-style-type: none">• 简单• 小团队作战• 快速迭代试错• 快速发布修复问题	<ul style="list-style-type: none">• 高并发• 满足随时变更的产品需求，且不影响版本发布计划• 对用户影响小





多工程分离——开发模式的改变





多工程分离——收获

子项目	描述
PluginClassLoader	“零成本”V3架构插件dex加载
PluginResourceLoader	“零成本”V3架构插件资源加载
autogen	自动化事件代码生成器，辅助解耦
buck+	基于FB开源项目buck系统的改造，增加LinearAllocCalculator和DexMethodsCalculator输出，讲问题提前到编译期预警
eclipse2buck	自动生成buck脚本
fastbuild	基于eclipse2buck和buck+的快速编译系统，结合jenkins实现快速自动构建。编译时间缩短50%~75%



我们还有更多可以拿出来说的的事儿.....



火龙果•整理
uml.org.cn

~ 敬请期待 ~

