推送系统架构设计 V1.1

目录

1 综述

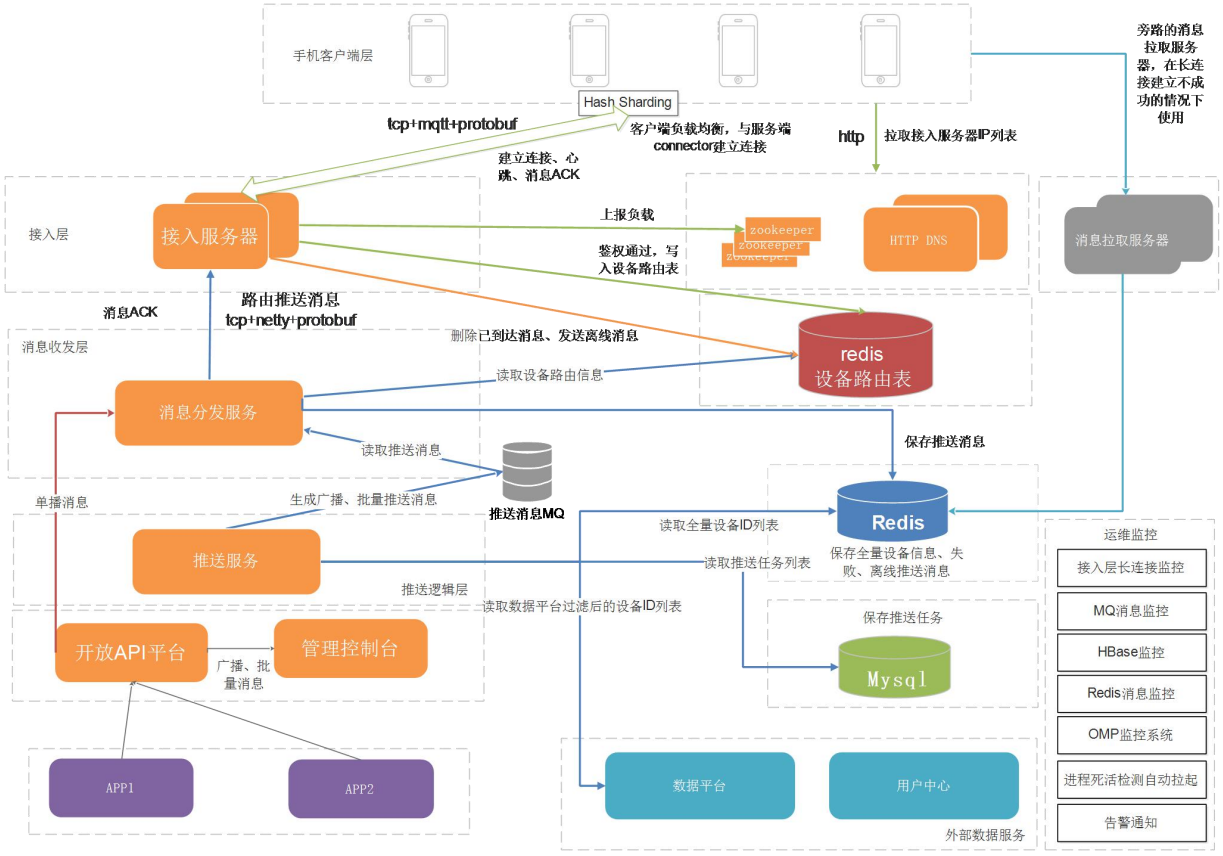
本文档是推送平台的系统设计文档，包括平台系统架构设计、接口设计、组网结构、数据模型、核心流程等部分。

本文档的阅读者包括开发、测试、维护相关人员。

2 术语说明

概念	含义
接入层	负责建立与客户端的长连接、认证鉴权、消息推送与接收、智能心跳、HTTP DNS、消息拉取服务器：消息拉取服务器可以在长连接建立不成功的情况下给用户使用
消息分发层	负责根据消息的路由信息分发消息到指定的接入层服务器
推送逻辑层	负责根据推送任务生成具体的推送消息。
数据层	存储设备路由信息、推送任务、设备全量信息、失败与离线消息
管理控制台	给业务运营人员提供的管理推送任务的界面
开放 API 平台	将推送能力通过 API 提供给业务使用
外部系统	数据平台：提供用户标签的相关接口，可以拉取到标签分类、标签名称列表、标签到 imei 的映射关系，同时提供标签之间的合集、并集、排除的服务 用户中心：用户中心需要提供一套全局唯一设备 ID 的方案、需要维护用户 ID 到设备 ID 的映射关系。

3 系统整体架构



说明：

系统主要分四层。

- 接入层：提供 OPPO 手机的接入，负责建立与客户端的长连接、认证鉴权、消息推送与接收、智能心跳、HTTP　DNS、消息拉取服务器。
- 消息分发层：负责根据消息的路由信息分发消息到指定的接入层服务器；
- 推送逻辑层：负责根据推送任务生成具体的推送消息，保存到消息队列。
- 数据层：存储设备路由信息、推送任务、设备全量信息、失败与离线消息。
- 管理控制台：给业务运营人员提供的管理推送任务的界面。
- 开放 API 平台：将推送能力通过 API 提供给业务使用。
- 外部系统：数据平台：提供用户标签的相关接口，可以拉取到标签分类、标签名称列表、标签到 imei 的映射关系，同时提供标签之间的合集、并集、排除的服务。用户中心：用户中心需要提供一套全局唯一设备 ID 的方案、需要维护用户 ID 到设备 ID 的映射关系。

4 数据模型

这里主要为了描述核心的数据实体以及其关系，具体字段请参考数据库设计。

4.1 推送任务（Mysql+Redis）

字段	类型	描述	必选	备注
push_task_id	int	推送任务 ID	是	自增,PK
push_type	int	推送类型	是	1：推送通知 2：透传消息 3：点对点消息
appid	int32	推送消息的应用 ID	是	
show_mode	int32	展示方式	是	1：文本， 2：图片暂时保留
title	string	标题	否	推送通知是必选
content	string	内容	是	
click_action_type	int32	点击操作类型	否	推送通知是必选
click_action_url	string	点击操作地址	否	
action_parameters	string	操作参数，以&分隔	否	
start_date	Date	展示开始时间，精确到天	是	
end_date	Date	展示结束时间，精确到天	是	
balance_time	int32	散列时间（单位分钟）	是	
time_ ranges	string	可展示时段 08:00-23:00,多个以&分隔	是	
logo	string	通知栏图标，Cdn 地址，完整地址	可选	

4.2 设备路由表（Redis）

字段	类型	描述	备注
client_id reginstrtation	string	设备唯一标识符	
server_ip	string	接入服务器 IP 地址	
connect_create_time	int	连接创建时间	
heart_beat_time	int	最近一次心跳时间	

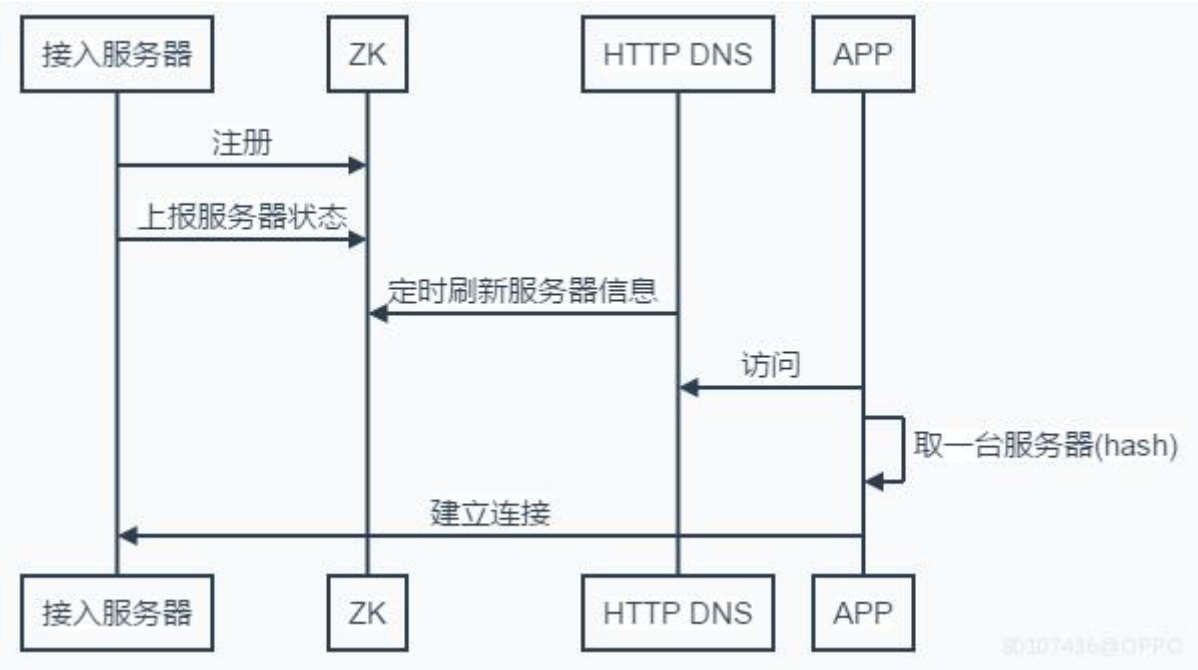
4.3 离线消息（Redis）

client_id	string	设备唯一标识符	

push_task_id_list	list	推送任务 Id 列表	

5 核心流程

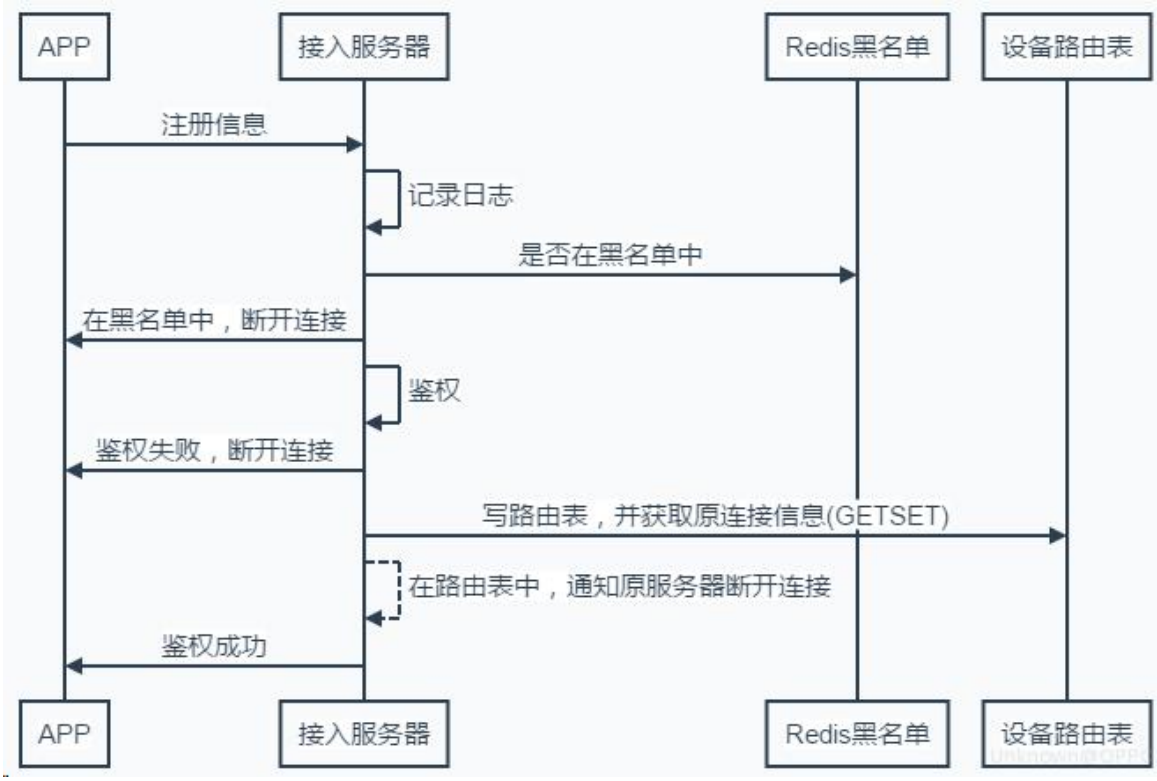
5.1 负载均衡流程



Ip	电信运营商	权重
XXX.XXX.XXX.XXX	cm（中国移动）	0.2

1. APP 内置 HTTP DNS 的 IP 列表，在访问 HTTP DNS 的时候对多个 IP 做 hash，选择其中一个。
2. APP 端在选择服务器的时候，需要根据前`N`个`IP`进行随机计算，并且每个`IP`会有不同的加权
3. 获取 IP 列表的策略: 为空 或 建立连接失败次数/总 IP 列表数量>=05。

5.2 设备接入流程



1. 接入服务器建立连接后，如果 3 秒内没有收到注册信息，判断连接失败，断开连接
2. 只有在鉴权成功后才处理 APP 发送的消息，之前所有消息全部丢弃
3. 设备路由表的所有写入、删除操作由接入服务器来发起。存在网络状况导致连接断开，接入服务器需要及时更新该表
4. 同一个客户端同时仅允许建立一个连接。

5.3 广播消息发送流程

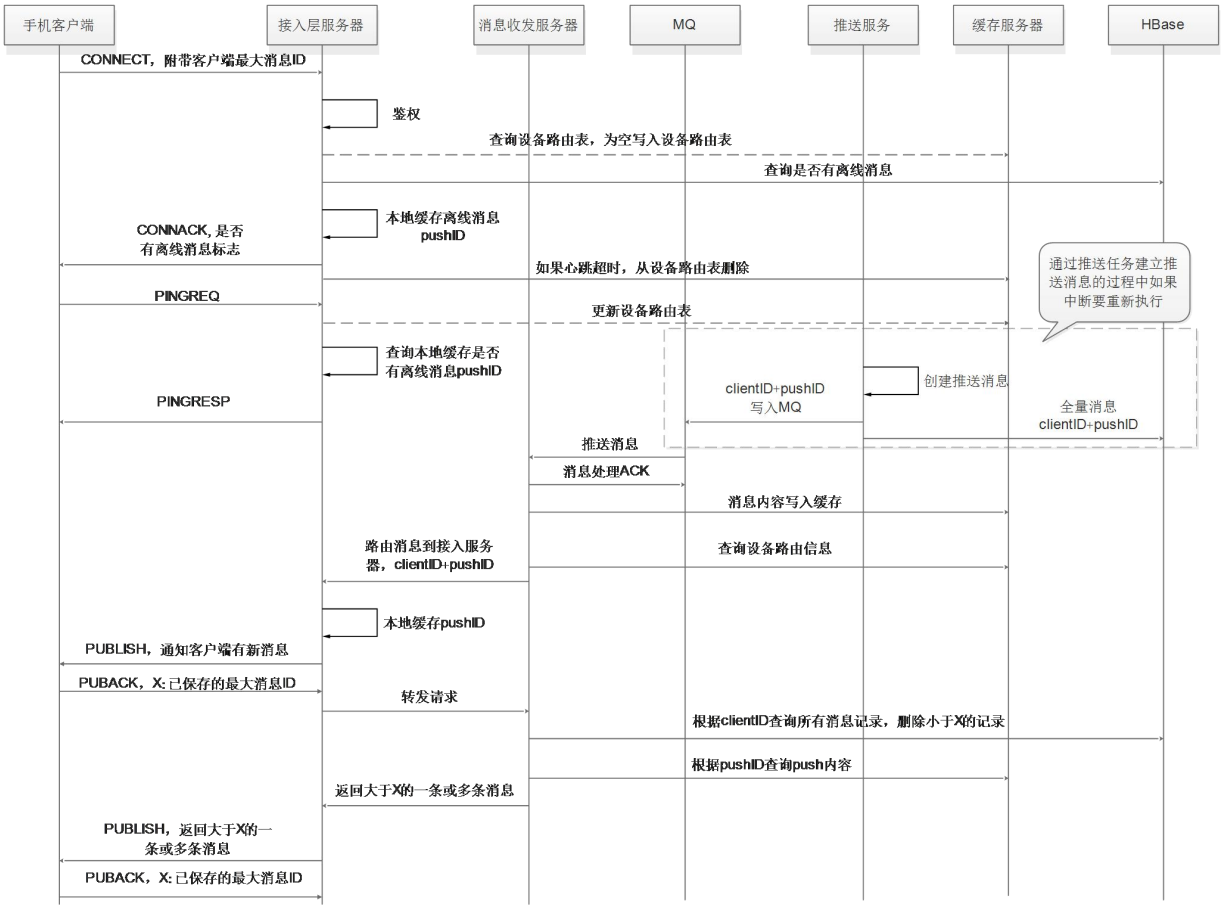


图 广播消息发送流程

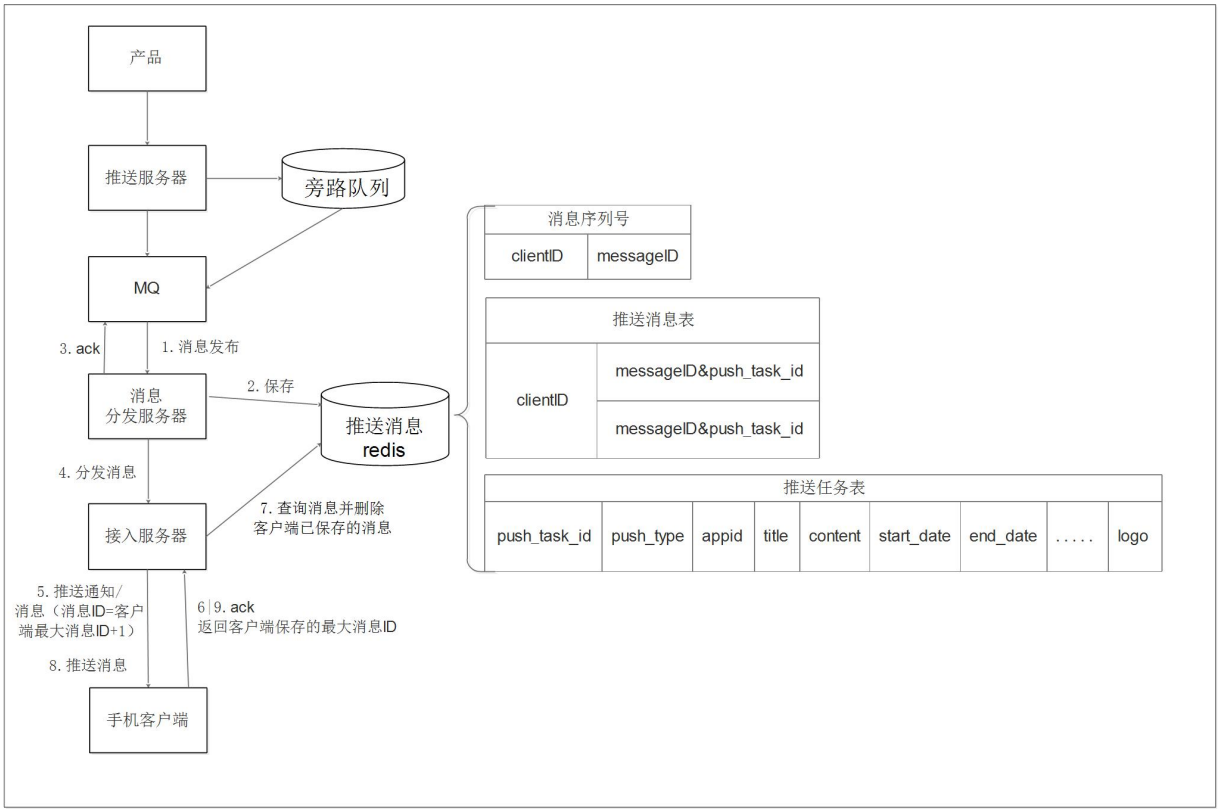


图 广播数据到达流程

1. 推送消息全量存储。
2. 针对每一个 clientID 有独立的 messageID 序列。
3. 消息分发服务器推送消息要确保按照 messageID 从小到大推送，每次有新消息时要检查是否有离线消息。

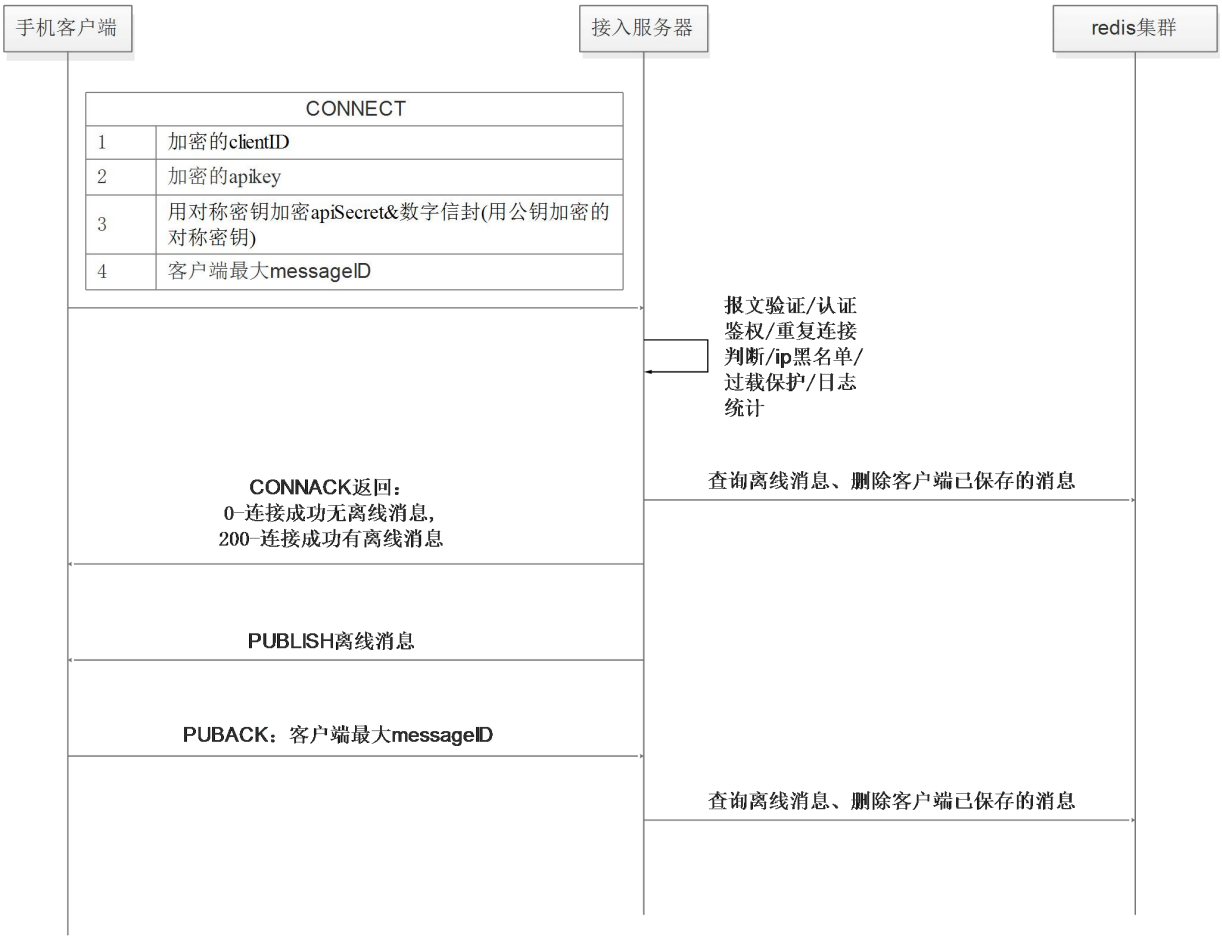
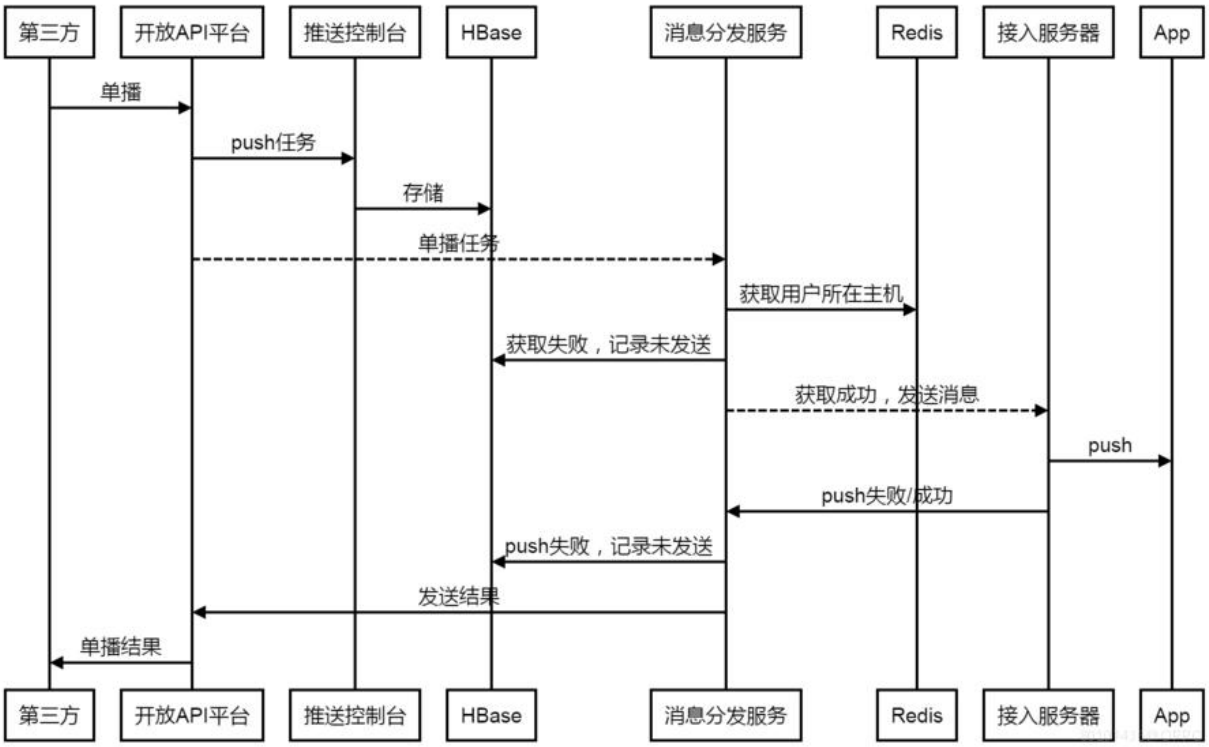


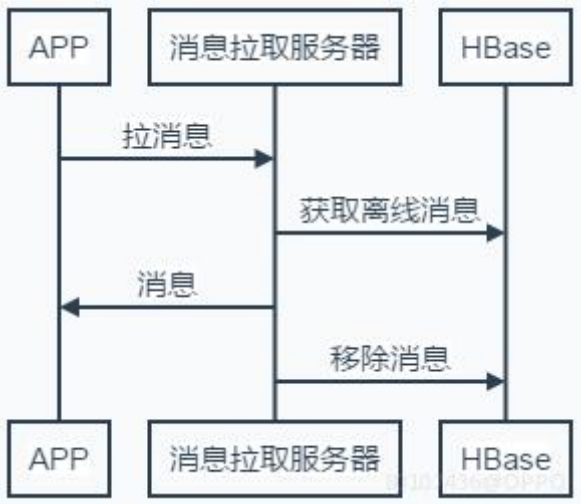
图 连接建立触发客户端拉取离线消息

1. 在客户端与接入服务器建立连接的时候检查是否有离线消息，需要客户端主动上报客户端保存的最大的消息 ID。
2. 接入服务器在收到连接请求后要检查是否有离线消息需要发送，如果有离线消息就返回状态码为 200,表示连接建立成功而且有离线消息。
3. 服务端主动下发离线消息，客户端接收消息后处理，返回已经保存的最大的消息 ID。
4. 服务端继续查询是否有离线消息同时删除客户端已经保存的离线消息，如果有消息就重复执行第三步，如果没有离线消息就结束。

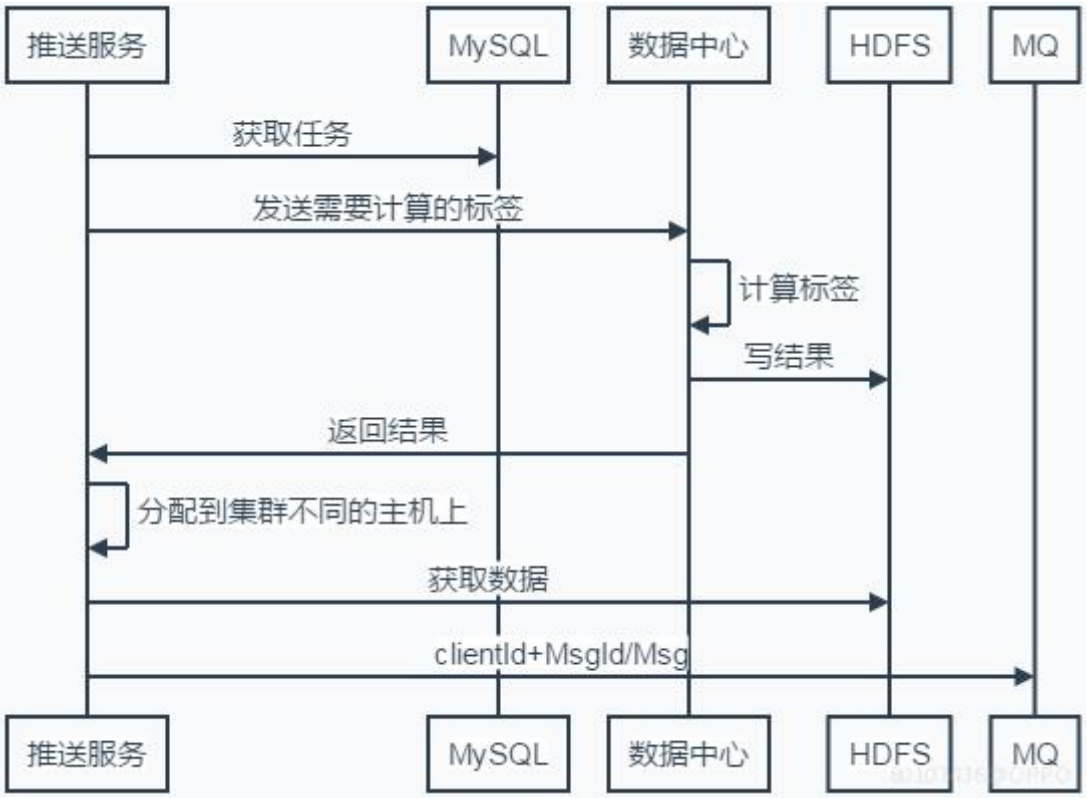
5.4 单播消息发送流程



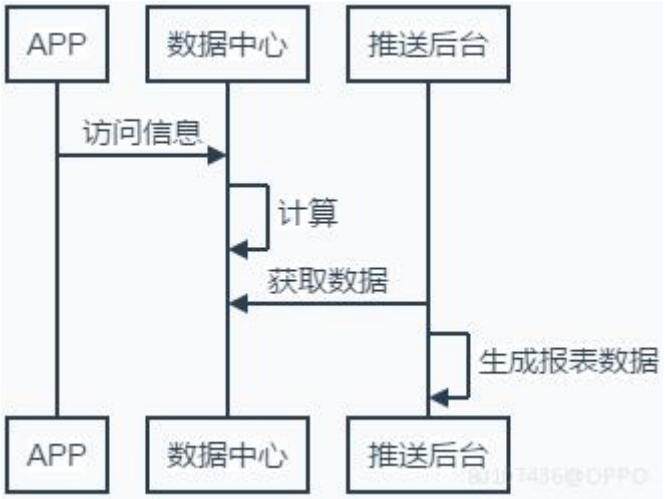
5.5 拉消息流程



5.6 批量任务分解流程

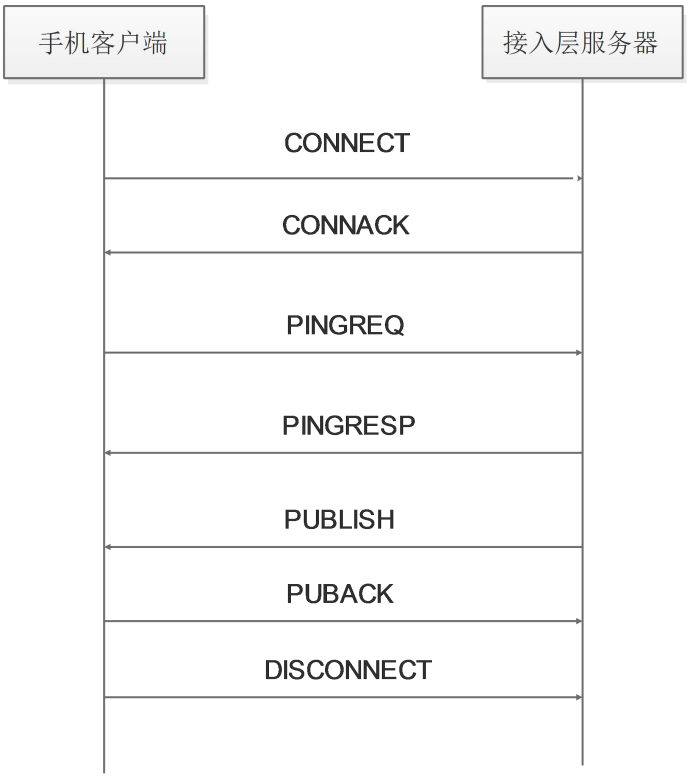


5.7 报表生成流程



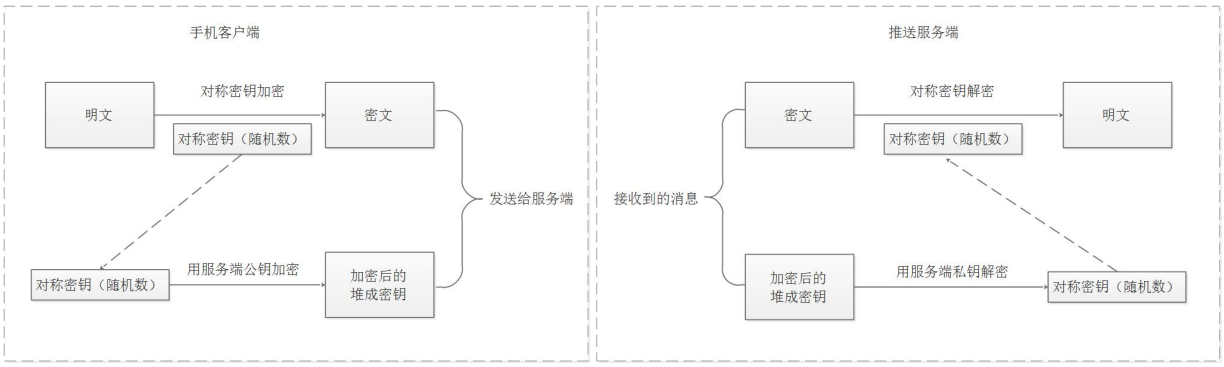
6 推送长连接接口协议

6.1 接口交互流程



6.2 数据加密

移动端与 OPPO 推送服务端交互数据安全性采用数字信封



1. 手机客户端生成一个随机数作为对称密钥。
2. 选用该随机数作为密钥对明文进行对称加密，生成密文。
3. 使用推送服务端公钥对随机数进行非对称加密，生成数字信封。
4. 发送的密文和信封给推送服务端。
5. 推送服务端用私钥对信封进行解密，得到对称密钥(随机数)。
6. 推送服务端使用该对称密钥(随机数)对密文合同进行对称解密，得到明文。

7. 采用数字信封技术需要考虑要更新保存在客户端的公钥证书。也可以考虑直接用 TLS/SSL 技术就可以避免这个问题，但是要测试一下性能。

6.2.1 接口协议

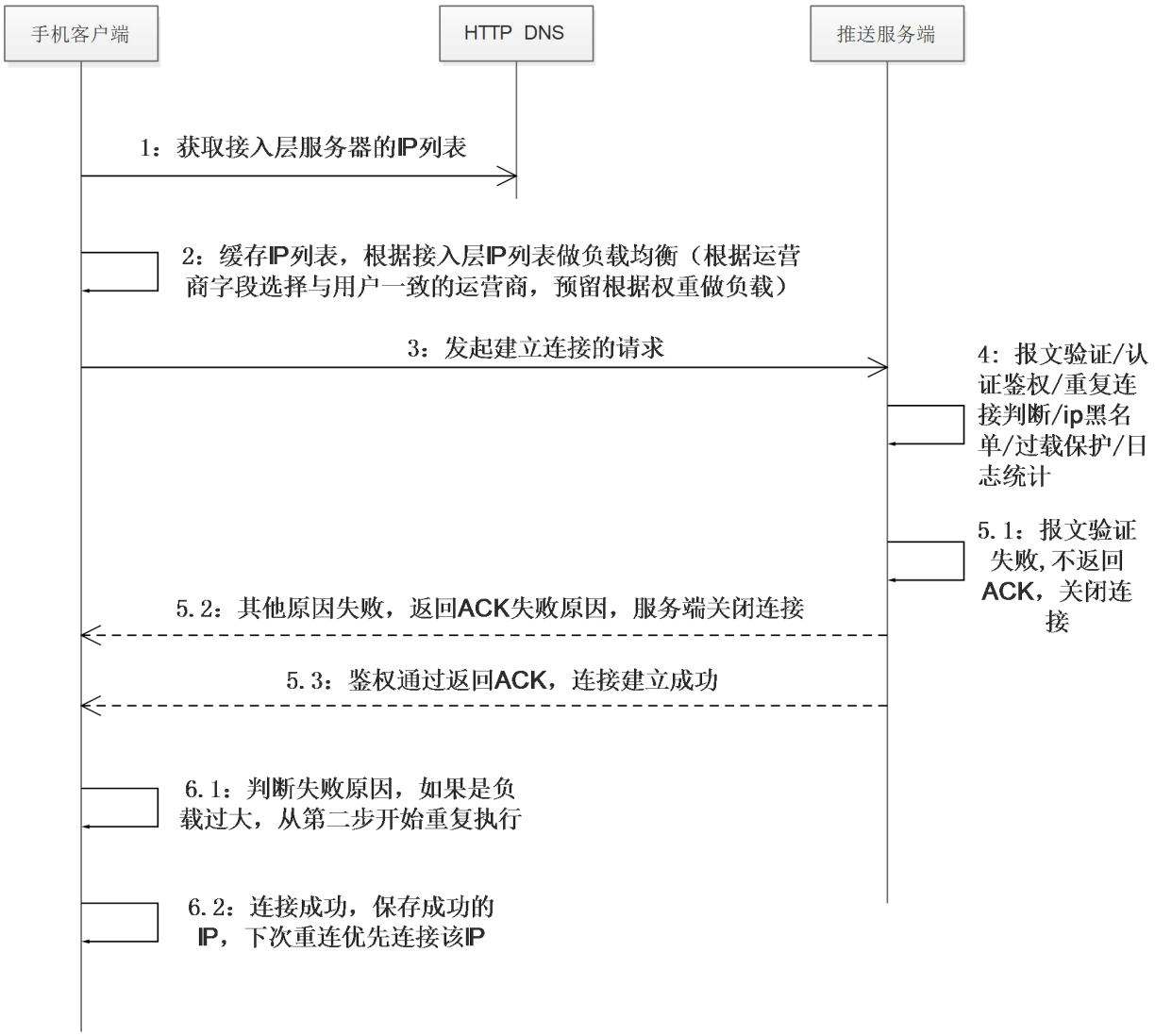
接口使用 tcp+mqtt 协议、消息内容体为二进制文本、UTF8 编码，数据采用加密传输，具体请参考安全设计。

6.3 接口设计

下面的表格式 mqtt 协议的控制报文，我们根据推送系统的实际需求裁剪了一些报文，加下划线的报文是被裁剪的。

名字	值	报文流动方向	描述
~~Reserved~~	~~0~~	~~禁止~~	~~保留~~
CONNECT	1	客户端到服务端	客户端请求连接服务端
CONNACK	2	服务端到客户端	连接报文确认
PUBLISH	3	两个方向都允许	发布消息
PUBACK	4	两个方向都允许	QoS 1 消息发布收到确认
~~PUBREC~~	~~5~~	~~两个方向都允许~~	~~发布收到（保证交付第一步）~~
~~PUBREL~~	~~6~~	~~两个方向都允许~~	~~发布释放（保证交付第二步）~~
~~PUBCOMP~~	~~7~~	~~两个方向都允许~~	~~QoS 2 消息发布完成（保证交互第三步）~~
~~SUBSCRIBE~~	~~8~~	~~客户端到服务端~~	~~客户端订阅请求~~
~~SUBACK~~	~~9~~	~~服务端到客户端~~	~~订阅请求报文确认~~
~~UNSUBSCRIBE~~	~~10~~	~~客户端到服务端~~	~~客户端取消订阅请求~~
~~UNSUBACK~~	~~11~~	~~服务端到客户端~~	~~取消订阅报文确认~~
PINGREQ	12	客户端到服务端	心跳请求
PINGRESP	13	服务端到客户端	心跳响应
DISCONNECT	14	客户端到服务端	客户端断开连接
~~Reserved~~	~~15~~	~~禁止~~	~~保留~~

6.3.1 建立连接



1. 获取接入层 IP 列表的策略: 为空 或 建立连接失败次数/总 IP 列表数量>=05。
2. 如果选择的 IP 因为负载过大拒绝连接，要重新选择 IP，排除前一次失败的 IP。
3. 连接建立的超时时间建议 3 秒钟。
4. 根据 mqtt 协议不允许重复连接。

6.3.1.1 请求-CONNECT-连接服务端：

6.3.1.1.1 固定报头：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 报文类型 (1)				Reserved 保留位			
	0	0	0	1	0	0	0	0
byte 2…	剩余长度值							

6.3.1.1.2 可变报头：

	说明	7	6	5	4	3	2	1	0
协议名									
byte 1	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 2	长度 LSB (4)	0	0	0	0	0	1	0	0
byte 3	‘M’	0	1	0	0	1	1	0	1
byte 4	‘Q’	0	1	0	1	0	0	0	1
byte 5	‘T’	0	1	0	1	0	1	0	0
byte 6	‘T’	0	1	0	1	0	1	0	0

	说明	7	6	5	4	3	2	1	0
协议级别									
byte 7	Level(4)	0	0	0	0	0	1	0	0

连接标识位：

Bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
byte 8	1	1	0	0	0	0	0	0

我们只使用 User Name Flag 与 Password Flag，其他的标识位暂不使用。

保持连接：

Bit	7	6	5	4	3	2	1	0
byte 9	保持连接 Keep Alive MSB							
byte 10	保持连接 Keep Alive LSB							

6.3.1.1.3 消息内容体：

序号	字段	值	备注
1	客户端标识符	用对称密钥加密 clientID	UTF-8
2	用户名	用对称密钥加密 apiKey	UTF-8
3	密码	用对称密钥加密 apiSecret&数字信封(用公钥加密的对称密钥)	UTF-8

6.3.1.2 响应- CONNACK-确认连接请求：

6.3.1.2.1 固定报头：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (2)				Reserved 保留位			
	0	0	1	0	0	0	0	0
byte 2	剩余长度 (2)							

	0	0	0	0	0	0	1	0

剩余长度字段

表示可变报头的长度。对于 CONNACK 报文这个值等于 2。

6.3.1.2.2 可变报文头：

	描述	7	6	5	4	3	2	1	0
连接确认标志		Reserved 保留位							SP[1]
byte 1		0	0	0	0	0	0	0	X
连接返回码									
byte 2		X	X	X	X	X	X	X	X

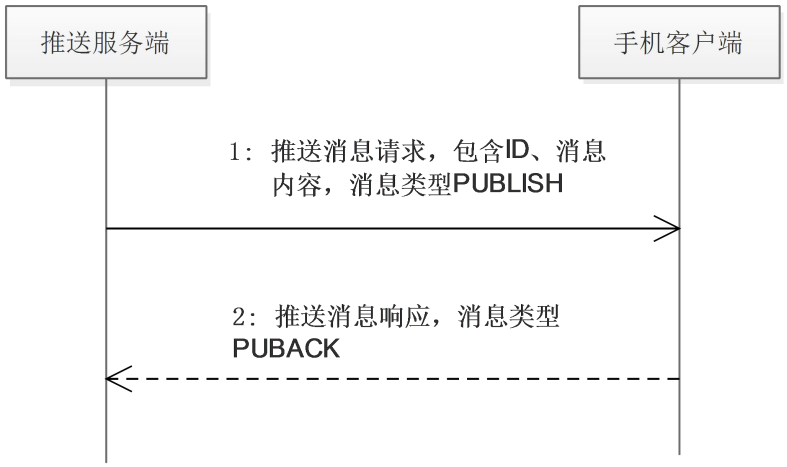
连接返回码：

值	返回码响应	描述
0	0x00 连接已接受	连接已被服务端接受
1	0x01 连接已拒绝，不支持的协议版本	服务端不支持客户端请求的 MQTT 协议级别
2	0x02 连接已拒绝，不合格的客户端标识符	客户端标识符是正确的 UTF-8 编码，但服务端不允许使用
3	0x03 连接已拒绝，服务端不可用	网络连接已建立，但 MQTT 服务不可用
4	0x04 连接已拒绝，无效的用户名或密码	用户名或密码的数据格式无效
5	0x05 连接已拒绝，未授权	客户端未被授权连接到此服务器
101	0x65 服务器忙，服务器过载保护	服务器的负载过多，无法接受更多的请求
102	0x66 重复连接，只允许一个连接	一个 clientID 只允许建立一个长连接
103	0x67 拒绝服务	客户端不合法，服务端拒绝服务
104	0x68 服务器错误	服务器发生错误
200	0xC8 连接已接受有离线消息	连接已接受有离线消息
6-100,104-199,201-255		保留

6.3.1.2.3 消息体：

无

6.3.2 推送消息



6.3.2.1 请求-PUBLISH-发布消息

PUBLISH 控制报文是指从客户端向服务端或者服务端向客户端传输一个应用消息。
推送系统 V1.0 只支持服务端向客户端传递一个应用消息

6.3.2.1.1 固定报头：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (3)				DUP	QoS 等级		RETAIN
	0	0	1	1	X	X	X	X
byte 2	剩余长度							

DUP：重发标志
QoS：服务质量等级
RETAIN：保留标志

QoS 值	Bit 2	Bit 1	描述
0	0	0	最多分发一次
1	0	1	至少分发一次
2	1	0	只分发一次
-	1	1	保留位

6.3.2.1.2 可变报头：

Field	Value
主题名	PUSH（业务名称），以后会有更多业务类型
报文标识符	PUSH 业务下的报文类型： 1：PushNotification 推送通知, 2：PushMessage 透传消息, 3：P2Pmessage 点对点消息

示例：

	描述	7	6	5	4	3	2	1	0
Topic Name 主题名									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	‘a’ (0x61)	0	1	1	0	0	0	0	1
byte 4	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 5	‘b’ (0x62)	0	1	1	0	0	0	1	0
报文标识符									
byte 6	报文标识符 MSB (0)	0	0	0	0	0	0	0	0
byte 7	报文标识符 LSB (10)	0	0	0	0	1	0	1	0

6.3.2.1.3 消息内容：

消息内容是二进制字节，protobuf 协议。

PushNotification 推送通知：

字段	类型	描述	必选
push_task_id	int32	推送任务 ID	是
appid	int32	推送消息的应用 ID	是
show_mode	int32	展示方式 1：文本，2：图片暂时保留	是
title	string	标题	是
content	string	内容	是
click_action_type	int32	点击操作类型	是
click_action_url	string	点击操作地址	是
action_parameters	string	操作参数，以&分隔	否
show_start_date	string	展示开始日期，格式 YYYYMMDD	是
show_end_date	string	展示结束日，期格式 YYYYMMDD	是
balance_time	int32	散列时间（单位分钟）	是
show_time_ ranges	string	可展示时段 08:00-23:00,多个以&分隔	是
logo	string	通知栏图标，Cdn 地址，完整地址	可选

PushMessage 透传消息：

字段	类型	描述	必选
push_task_id	int32	推送任务 ID	是
appid	int32	推送消息的应用 ID	是
content	string	内容	是
start_time	int32	生效开始时间的秒数	是

end_time	int32	生效结束时间的秒数	是
balance_time	int32	散列时间（单位分钟）	是
time_ ranges	string	生效时段 08:00-23:00,多个以&分隔	是

P2PMessage 透传消息：

字段	类型	描述	必选
appid	int32	推送消息的应用 ID	是
mid	int32	消息 ID	是
content	string	内容	是
sender	string	发送者标识	是

6.3.2.2 响应-PUBACK-发布确认

6.3.2.2.1 固定报头：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (4)				保留位			
	0	1	0	0	0	0	0	0
byte 2	剩余长度(2)							
	0	0	0	0	0	0	1	0

剩余长度字段

表示可变报头的长度。对 PUBACK 报文这个值等于 2.

6.3.2.2.2 可变报头：

Field	Value
报文标识符	PUSH 业务下的报文类型： 1：PushNotification 推送通知, 2：PushMessage 透传消息, 3：P2Pmessage 点对点消息

报文标识符									
byte 1	报文标识符 MSB (0)	0	0	0	0	0	0	0	0
byte 2	报文标识符 LSB (10)	0	0	0	0	1	0	1	0

6.3.2.2.3 消息内容：

推送任务 ID

6.3.3 心跳信息

6.3.3.1 请求-PINGREQ-心跳请求

客户端发送 PINGREQ 报文给服务端的。用于：

1. 在没有任何其它控制报文从客户端发给服务的时，告知服务端客户端还活着。
2. 请求服务端发送 响应确认它还活着。
3. 使用网络以确认网络连接没有断开。
4. 服务端要记录最近心跳时间。

6.3.3.1.1 固定报头：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (12)				保留位			
	1	1	0	0	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

6.3.3.1.2 可变报头：

PINGREQ 报文没有可变报头。

6.3.3.1.3 消息内容：

PINGREQ 报文没有

6.3.3.2 响应-PINGRESP-心跳响应

服务端**必须**发送 PINGRESP 报文响应客户端的 PINGREQ 报文
服务端发送 PINGRESP 报文响应客户端的 PINGREQ 报文。表示服务端还活着。

6.3.3.2.1 固定报头：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (13)				保留位			
	1	1	0	1	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

6.3.3.2.2 可变报头：

PINGRESP 报文没有可变报头。

6.3.3.2.3 消息内容：

PINGRESP 报文没有消息内容

6.3.4 断开连接

6.3.4.1 请求-DISCONNECT-断开连接

DISCONNECT 报文是客户端发给服务端的最后一个控制报文。表示客户端正常断开连接。

6.3.4.1.1 固定报头：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT 控制报文类型 (14)				保留位			
	1	1	1	0	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

6.3.4.1.2 可变报头：

DISCONNECT 报文没有可变报头。

6.3.4.1.3 消息内容：

DISCONNECT 报文没有

6.3.4.2 响应

客户端发送 DISCONNECT 报文之后：

- **必须**关闭网络连接 。
- **不能**通过那个网络连接再发送任何控制报文 。

服务端在收到 DISCONNECT 报文时：

- **必须**丢弃任何与当前连接关联的未发布的遗嘱消息，具体描述见 3.1.2.5 节 。
- **应该**关闭网络连接，如果客户端 还没有这么做。

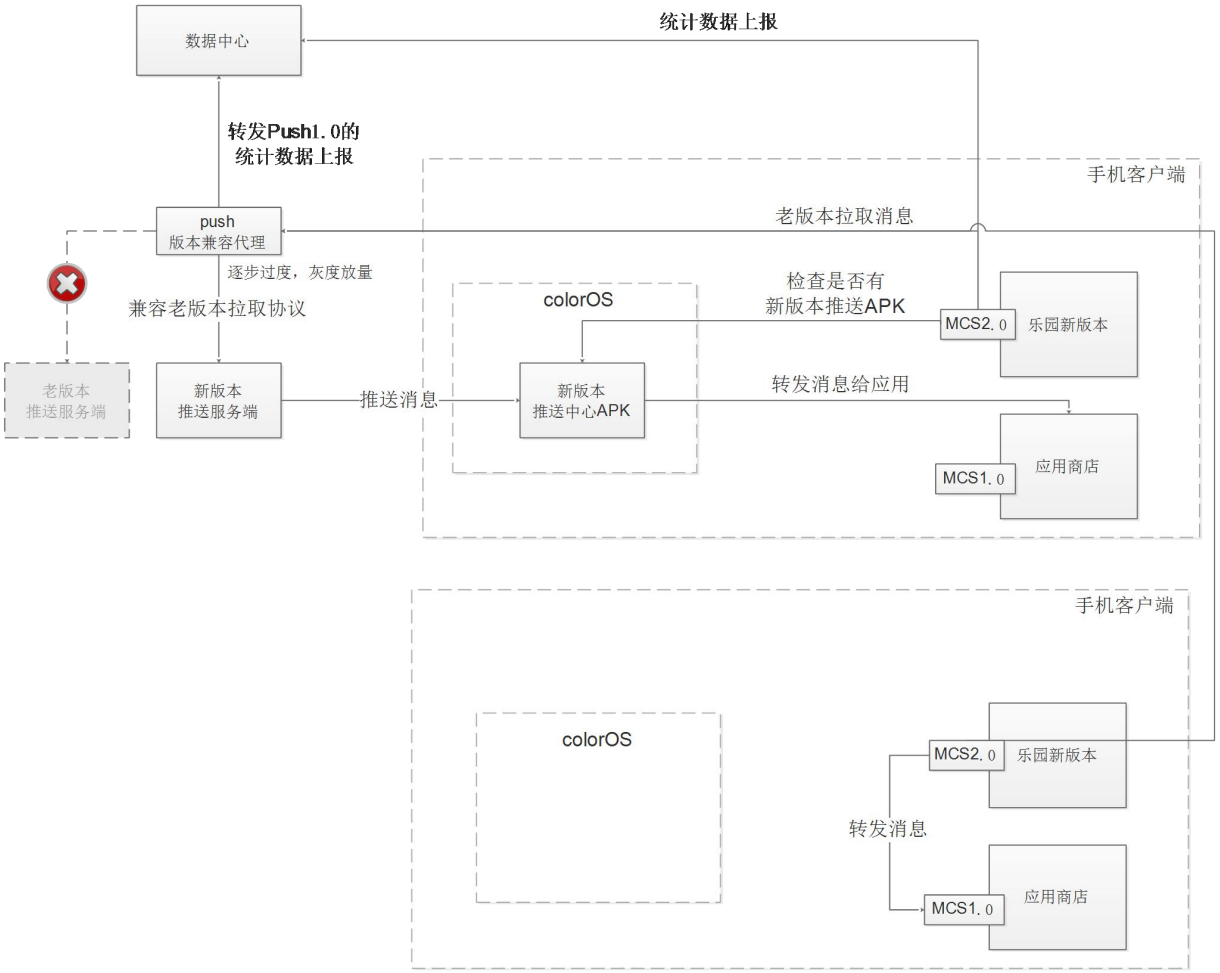
7 关键设计

关键设计主要通过推送版本兼容性机制、高可用性、扩展性、伸缩性、安全性四个方面设计。

7.1 推送版本兼容性机制

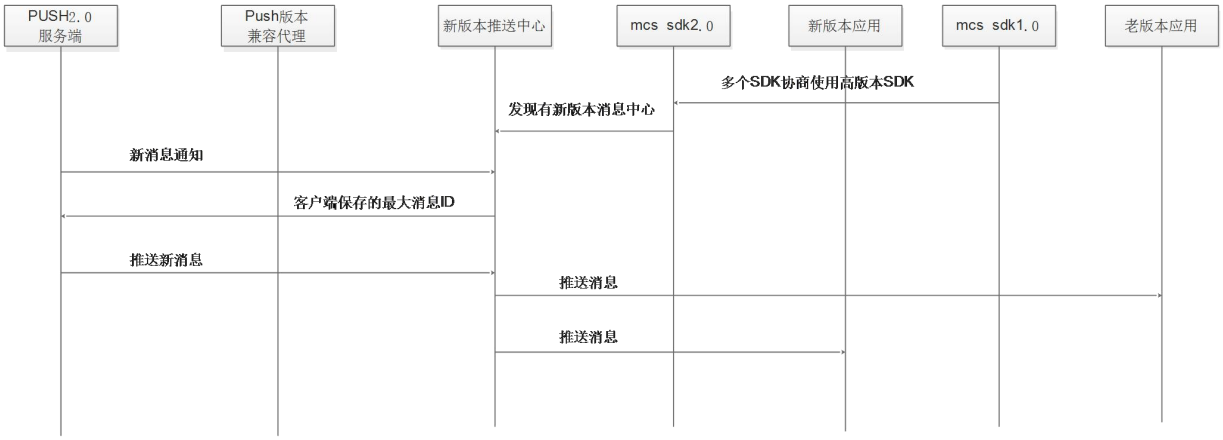
7.1.1 兼容方案目标：

业务侧运营人员只需要使用 PUSH2.0 的运营系统发布推送消息即可发送到全部设备上。



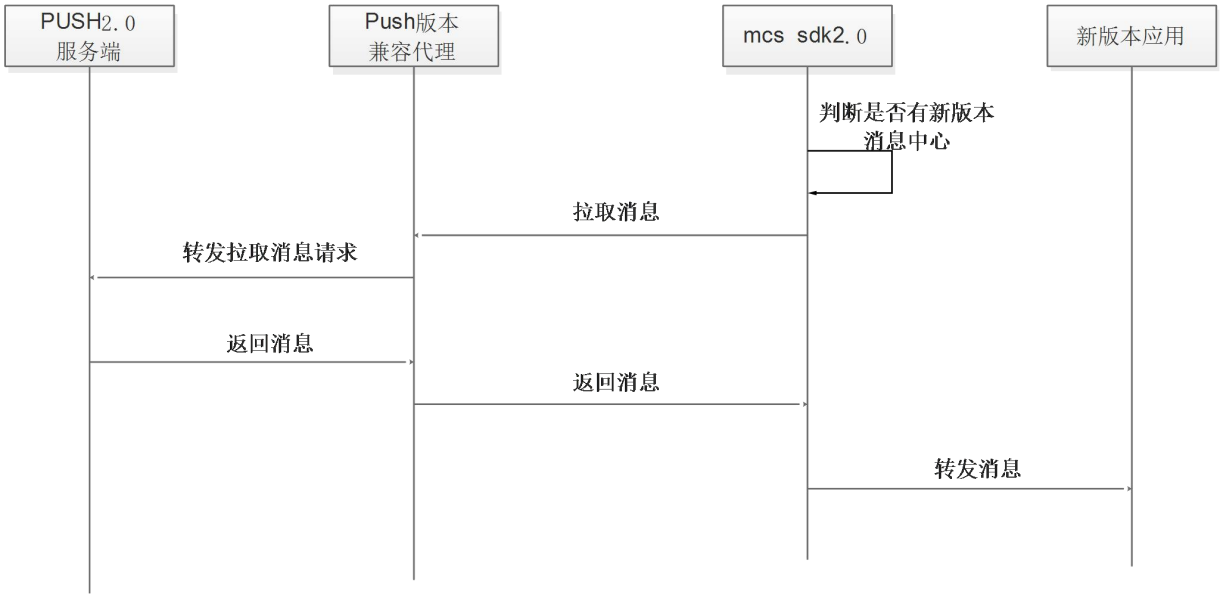
ROM 版本	应用版本	PUSH 协议版本	兼容性方案
新版本 ROM+新版本推送中心 APK	新版本应用+MCS2.0	PUSH2.0	无兼容问题
新版本 ROM+新版本推送中心 APK	老版本应用+MCS1.0	PUSH1.0	新推送中心+一个新应用多个老版本应用兼容方案
老版本 ROM	新版本应用+MCS2.0	PUSH2.0	老版本 ROM+新版本应用兼容方案
老版本 ROM	老版本应用+MCS1.0	PUSH1.0	老版本 ROM+老版本应用兼容方案

7.1.2 新推送中心+一个新应用多个老版本应用兼容方案

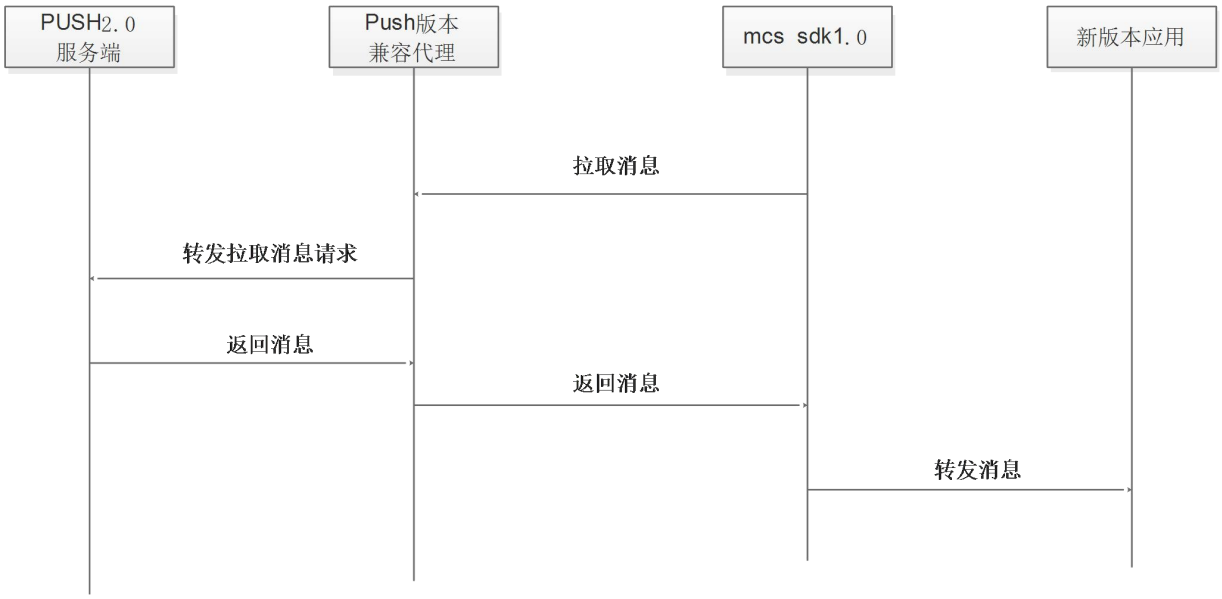


问题：可能有重复消息

7.1.3 老版本 ROM+新版本应用兼容方案



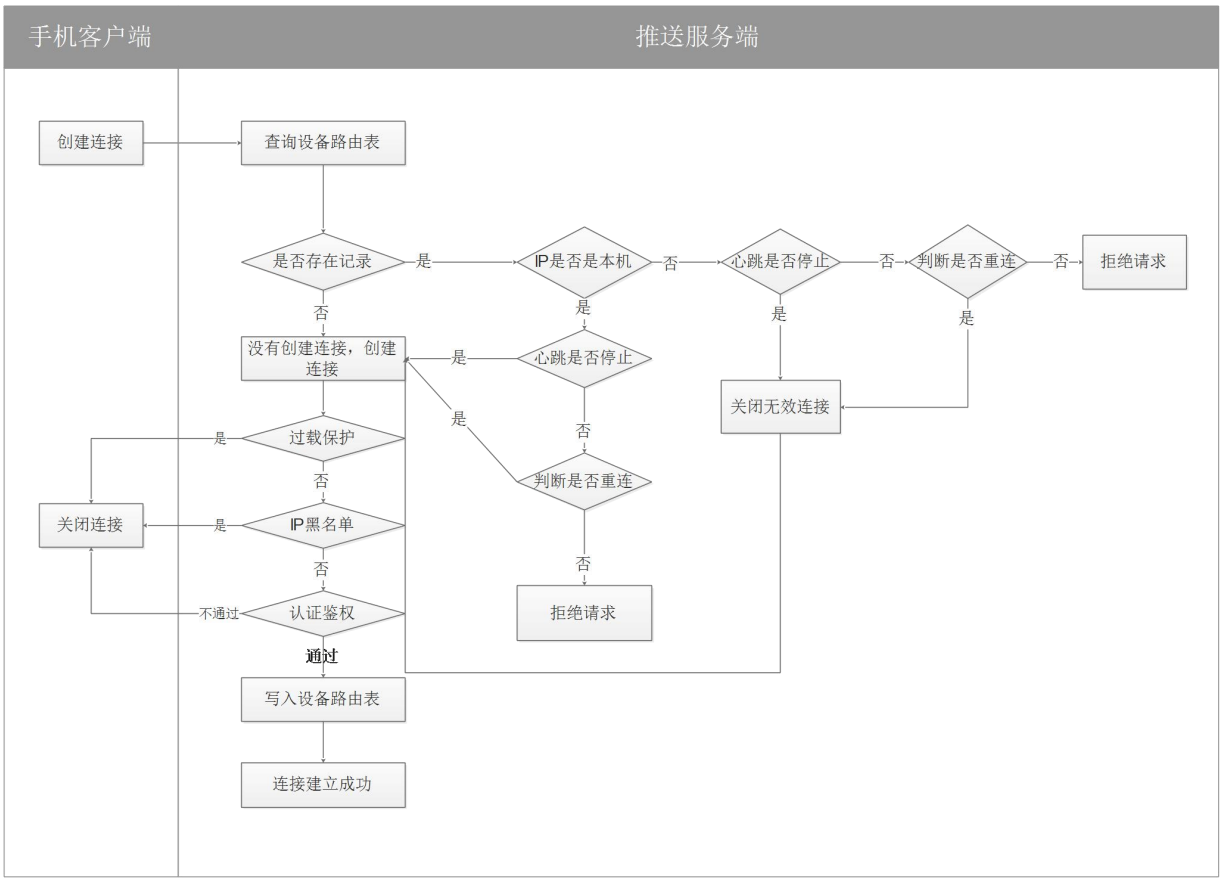
7.1.4 老版本 ROM+老版本应用兼容方案



问题：

1. 新旧版本的设备唯一标识不统一。

7.2 重复连接请求去重机制



7.3 高可用性设计

系统的高可用主要由从以下几个方面来考虑：

1. 服务的高可用
   大部分的服务都是无状态的，在服务器当机的情况下，很容易由集群内其他服务器来提供服务。部分有状态的服务，都提供了在当机情况下的异常处理。虽然可能照成部分消息的丢失或重发，但是整体还在可接受的范围内。
2. 存储的高可用
   系统中存储主要用到的中间件为 MySQL 和 Hbase。MySQL 整体的压力不大，本身不会参与到主流程中，做主备即可满足。Hbase 本身提供高可用的需求。
3. 限流
   系统会在两个入口进行限流：接入服务器和开放 API 平台。接入服务器在长连接数到达一定的数量后，拒绝其他的连接请求。开放 API 平台主要限制第三方平台发送的消息数量。
4. 限额
   开放 API 平台主要限制第三方平台发送的消息数量。
5. 系统监控
   系统的监控主要包括：JVM/中间件的监控、服务状态的监控以及消息执行路径监控、消息收发监

控。

JVM/中间件的监控和服务状态的监控通过 OMP 系统来实现。

消息执行路径监控计划统计接入调用链跟踪系统来进行监控。

消息收发监控：推送速率、成功率、平均推送时长、最长推送时长、最小推送时长等。

7.4 扩展性设计

本系统功能相对简单，不做过多的功能性上的扩展

7.5 伸缩性设计

系统主要的压力点包括：消息分发服务、接入服务器和开放 API 平台。

1. 接入服务器。接入服务器的接入主要由 HTTP DNS 的 hash 结果决定，在动态伸缩方面不存在太大的问题，二期会考虑用多 SET 的设计。
2. 消息分发服务。消息分发服务的输入分别为 MQ 和开放 API 平台。MQ 在感知服务上线或下线后，会开始或停止给该服务发送消息。开放 API 平台采用类是 dubbo 的方式调用消息分发服务，可以有效感知服务的上线下线。
3. 开放 API 平台。开放 API 平台对外为统一 URL 接入，有 nginx 进行反向代理，后端业务服务器伸缩较容易。

7.6 安全性设计

1. 通讯的安全性

   系统对外有三个接入点：接入服务器、拉消息服务器、开放 API 平台和推送服务前台。接入服务器和外部通讯采用 TLS 协议/数字信封加密，开放 API 平台、拉消息服务器接入采用 HTTPS，都足以保证通讯的安全性。推送服务前台录入的任务需要人为审核，保证错误任务被过滤掉。

2. 限流

   系统在接入服务器和开放 API 平台两个点进行限流。保证系统只处理能力范围内的数据，超过能力范围内的数据会丢弃。

7.7 包名/子系统命名规范

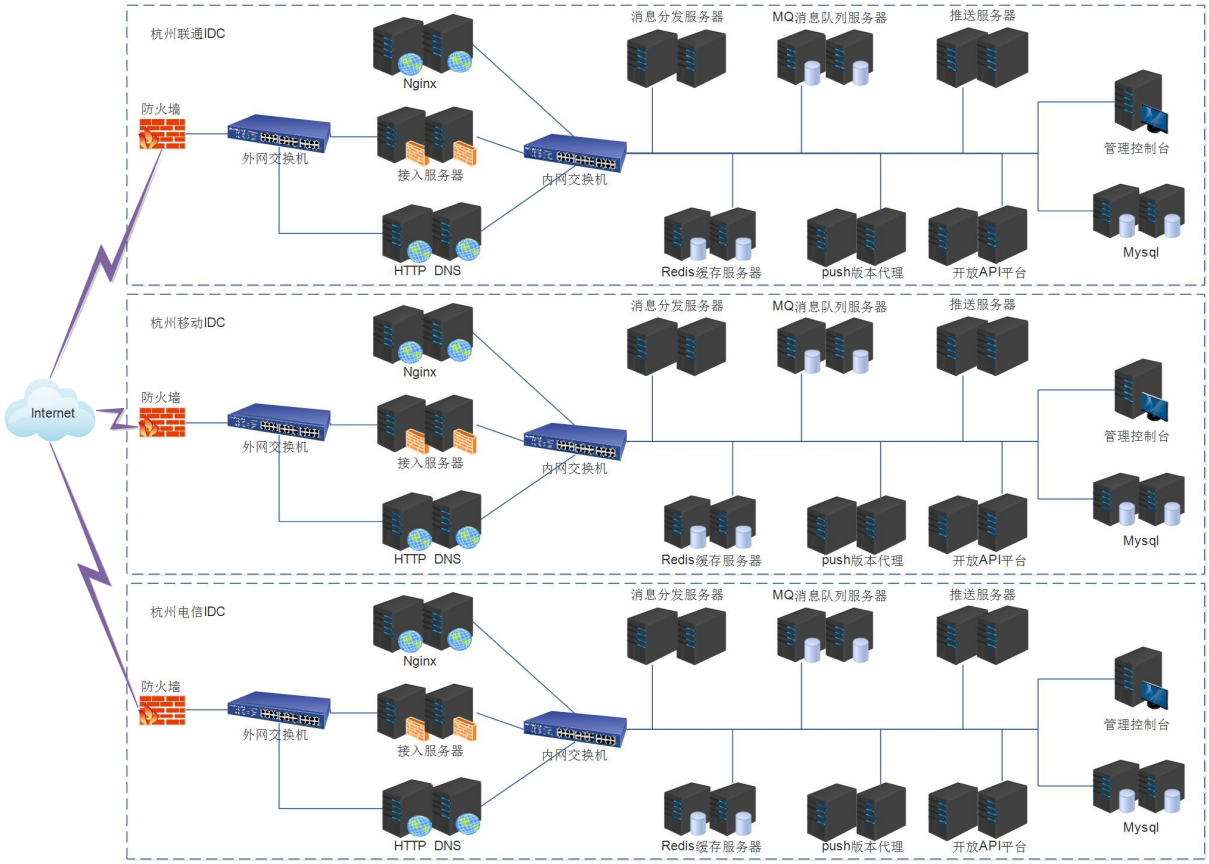
com.oppo.push.{子系统名称}.api	服务接口 API
com.oppo.push.{子系统名称}. controller	控制器
com.oppo.push.{子系统名称}.dao	数据访问层
com.oppo.push.{子系统名称}.dao.mapper	Mybatis 的映射文件（放在 resources 目录中）
com.oppo.push.{子系统名称}.service	业务服务层

子系统名称：

connector	接入服务器
dispatcher	消息分发服务
pusher	推送服务

open	开放 API
console	管理控制台

8 部署要求



8.2 设备说明

一期独立主机设备：总计 18 台

设备名称	物理机数量	实例数	部署说明	配置要求
接入服务	6	12	高 IO，高内存消耗，单机保持 1000 万长连接	CPU：2*6 core，如 E5-2620 系列 内存：32G（不要大于 128GB） 存储：2*300G 10K SAS(系统) 2T 10K SAS(数据) 网卡：1*10000Mbps+1*1000Mbps(运维管理用)
消息分发服务器	3	6	消息分发服务器，实时分发消息，高 IO	CPU：2*6 core，如 E5-2620 系列

				内存：32G 存储：2*300G 10K SAS(系统) 6*600G 10 SAS(数据) 网卡： 1*10000Mbps+1*1000Mbps(运维管理用)
推送服务器	3	6	高 IO	CPU：2*4 core，如 E5-2600 系列 内存：32G 存储：2*300G 10K SAS(系统) 6*600G 10 SAS(数据) 网卡： 1*10000Mbps+1*1000Mbps(运维管理用)
redis 集群	3	6	保存设备路由信息与全量消息表，混合部署，一个集群 6 个实例。高 IO、高内存	CPU：2*6 core，如 E5-2600 系列 内存：64G 存储：2TB SATA(数据) 网卡： 1*10000Mbps+1*1000Mbps(运维管理用)
MQ 集群	3	6	保存待消费的消息，高 IO	CPU：2*4 core，如 E5-2600 系列 内存：32G 存储：6*900GSAS(数据) 网卡： 1*10000Mbps+1*1000Mbps(运维管理用)

一期云主机设备：

设备名称	数量	部署说明	配置要求
管理控制台	2	推送管理后台	CPU：2*4 core，如 E5-2400 系列 内存：8G 以上 存储：2*300G 10K SAS(数据)

			网卡：2*1000Mbps
数据库服务器	2	主从双机热备	CPU：2*6 core，如 E5-2600 系列 内存：12G 以上 存储：4*600G 10K SAS(数据) 网卡：2*1000bps
Nginx	3	拉取消息方式的负载均衡服务器，可以复用老版本的 push 服务器	CPU：2*4 core，如 E5-2600 系列 内存：12G 以上 存储：300G 10K SAS(系统) 600G 10K SAS(数据) 网卡：4*1000Mbps
Push 版本兼容服务代理	3	1 实现老版本的拉取协议 2 实现新版本的消息拉取功能	CPU：2*4 core，如 E5-2600 系列 内存：24G 以上 存储：2*300G 10K SAS(系统) 6*600G 10 SAS(数据) 网卡：2*1000Mbps
Http DNS	2	双机热备	CPU：2*4 core，如 E5-2600 系列 内存：12G 以上 存储：300G 10K SAS(系统) 600G 10K SAS(数据) 网卡：4*1000Mbps

8.3 性能计算

推送总量数据			
设备总数量	接入服务器数量	单机连接数	推送消息总量（每用户 10 条）
5000 万	50	100 万	5 亿

接入服务器单机数据

数据项	性能
单设备连接次数（网络不稳定断连重连）	30 次/天
100 万设备连接总次数	3000 万/天
单设备心跳次数	288 次/天
100 万设备心跳总次数	2 亿 8800 万/天
单用户推送消息数量	10/天
100 万设备推送消息总数量	1000 万/天
推送速度	6 万/秒
单机 1000 万消息下发总时间	166 秒

8.4 存储计算

接入层单机存储日志					
日志名称	条数	大小（条）	新增大小（天）	保存时间	总大小（10）
连接建立	100 万*30 次	0.2k	5.72GB	10 天	58GB
心跳	2 亿 8800 万/天	50B	13.4GB	10 天	134GB
推送消息	5000 万	0.5K	23.84GB	10 天	238.4GB
其他					50GB
合计					480.4GB

消息收发服务器单机存储日志					
日志名称	条数	大小（条）	新增大小（天）	保存时间	总大小（月）
消息发送	5000 万	0.5K	23.84GB	10 天	238.4GB
消息接收	5000 万	50B	2.4GB	10 天	24GB
其他					50GB
合计					312.4GB

Redis				
日志名称	数量	大小（条）	总大小	集群大小（内存）
设备路由表	5000 万	60B	2861MB	4GB
全量消息	5000 万	250B	11920MB	16GB
其他				5GB
合计				25GB

MQ 存储					
日志名称	条数	大小（条）	每天新增大小	保存时间	集群大小（磁盘）
消息量	5 亿	40B	18GB	10	180GB
其他					50GB
合计					230GB

8.5 网络计算

接入层单机网络流量						
日志名称	条数	上行大小（条）	上行 总大小（天）	下行大小 （条）	下行 总大（天）	总大小（天）
连接建立	100 万*30 次	150B	4.2GB	4B	114MB	4.314GB
心跳	2 亿 8800 万/天	2B	275MB	2B	275MB	550MB
推送消息	1300 万	8B	99MB	0.6K	7.44GB	7.539GB
合计			4.57GB		7.82	12.4GB
接入层网络流量总计(单机流量*50)						
合计			228.5GB		391GB	620GB

接入层流量峰值总计				
推送消息量峰值	上行大小（条）	上行总量	下行大小（条）	下行总量
20 万/秒	8B	1.53MB	0.6K	117.2MB

8.6 内存计算

服务器（单机）	JVM 实例	单 JVM 内存	操作系统内存	其他	总计
接入服务器	2	12GB	2GB	38GB	64GB
消息分发服务器	4	6GB	2GB	38GB	64GB
推送服务器	2	4GB	2GB	22GB	32GB
MQ	2	4GB	2GB	22GB	32GB