# 关于苹果 APNs 推送机制(3)

国内 90%以上的 iOS 开发者，对 APNs 的认识都是错的

字数 3671 阅读 6383 评论 14 喜欢 150

Collection/Bookmark/Share for width under 768px

前言： APNs 协议在近两年的 WWDC 上改过两次， 15 年 12 月 17 日更是推出了革命性的新特性。但在国内传播的博客、面试题里关于 APNs 的答案全都是旧的、错的。

对 APNs 的吐槽

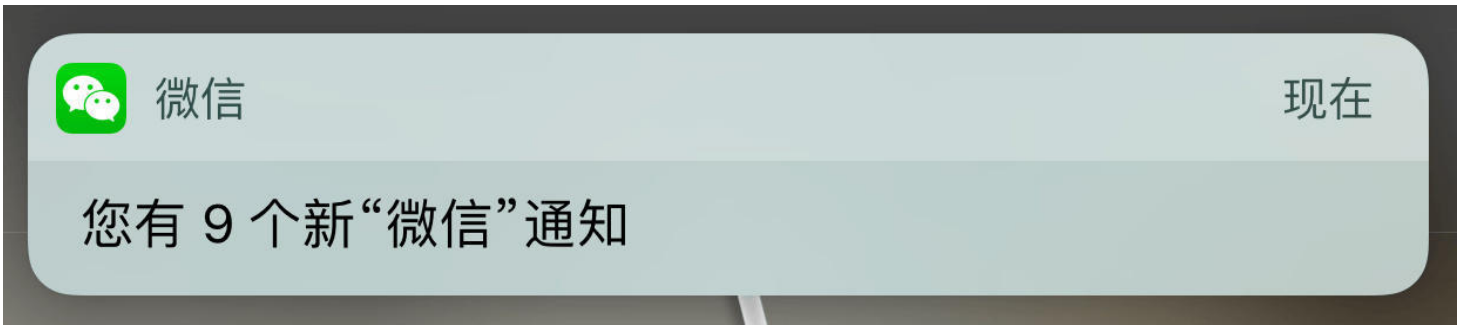
APNs 是 Apple Push Notification service 的简称（注意 APNs 的大小写，s 不需要大写）。

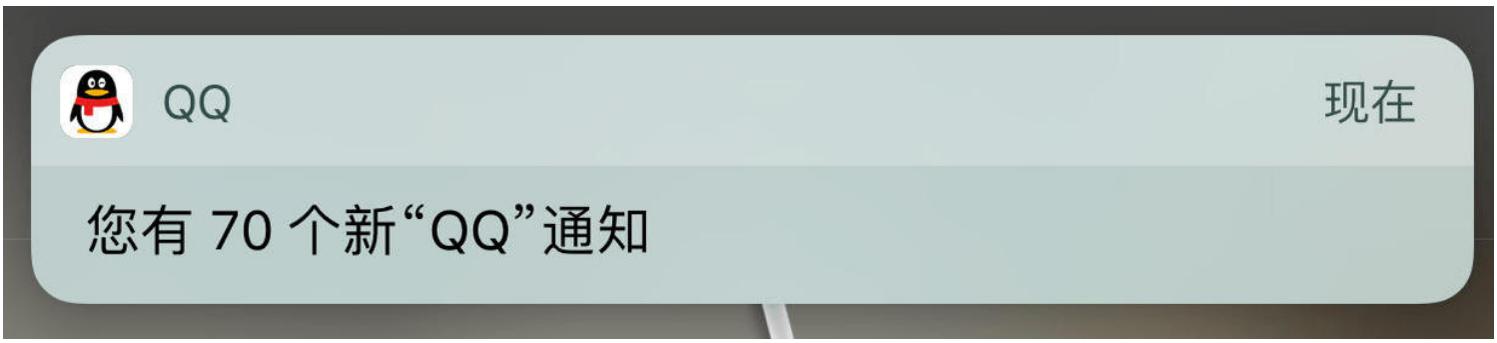
以下是我收集的一些关于 APNs 的吐槽，你先看下哪些吐槽比较“到位”：

-- 被吐槽的内容 吐槽

1 使用第三方 SDK 接入推送服务，SDK 提供商却告诉我，他们无法获知哪条消息成功发送给了 APNs，哪些失败了，而且即使 APNs 接收了，APNs 是否能保证投递成功，他们也无能为力。 我把消息交给你了，你告诉什么都保证不了？推送成功与否”基本靠猜“？

2




为什么我推了多条消息，APNs 就只给我最后一条？！

3 推送内容只能是 256 字节 这也太小了，根本不够用啊！

4　生产环境推送证书、测试环境推送证书、tvOS 推送证书、watchOS 推送证书、VOIP 推送证书。。　证书太多了，制作、切换证书太麻烦！

答案会穿插在下文中。

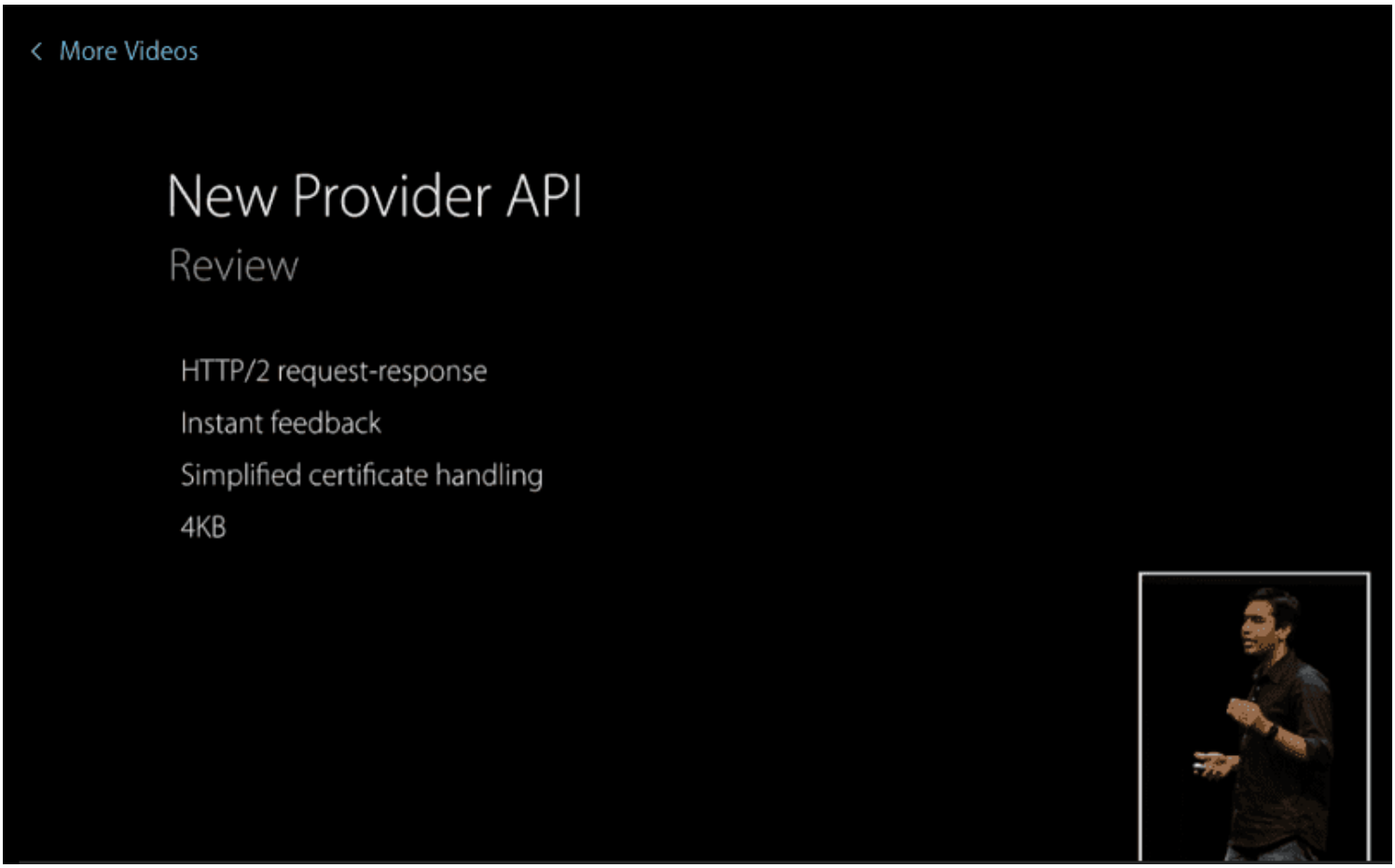
APNs 新闻一栏

时间　新闻　参考文档

2014 年 6 月 2014 年 6 月份 WWDC 搭载 iOS8 及以上系统的 iOS 设备，能够接收的最大 playload 大小提升到 2KB。低于 iOS8 的设备以及 OS X 设备维持 256 字节。 What's New in Notifications - WWDC 2014 - Session 713 - iOS


*enter image description here*

2015 年 6 月 2015 年 6 月份 WWDC 宣布将在不久的将来发布 “基于 HTTP/2 的全新 APNs 协议”，并在大会上发布了仅仅支持测试证书的版本。What's New in Notifications - WWDC 2015 - Session 720 - iOS, OS X

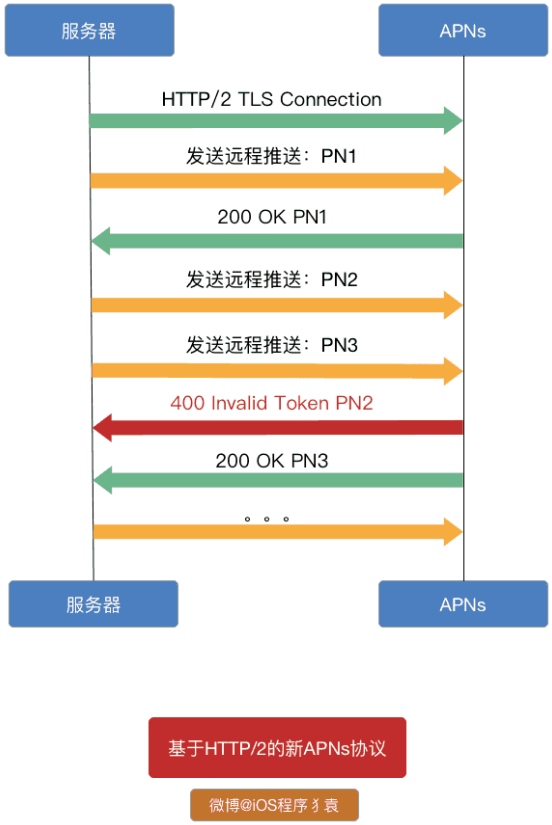



*enter image description here*

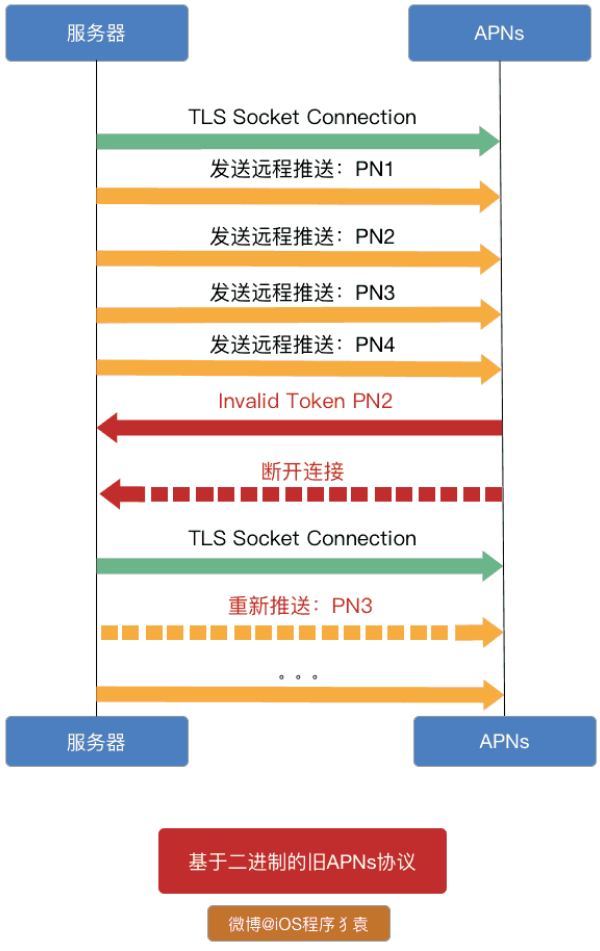
2015 年 12 月 17 日 2015 年 12 月 17 日起，发布 “基于 HTTP/2 的全新 APNs 协议”，iOS 系统以及 OS X 系统，统一将最大 playload 大小提升到 4KB。 Apple Push Notification Service Update 12-17 2015

新旧 APNs 协议工作示意图对比

基于 HTTP/2 的新 APNs 协议 基于二进制的旧 APNs 协议




*enter image description here*




*enter image description here*

接下来我们分别对新旧协议进行一下介绍：

反人类的旧 APNs 协议设计

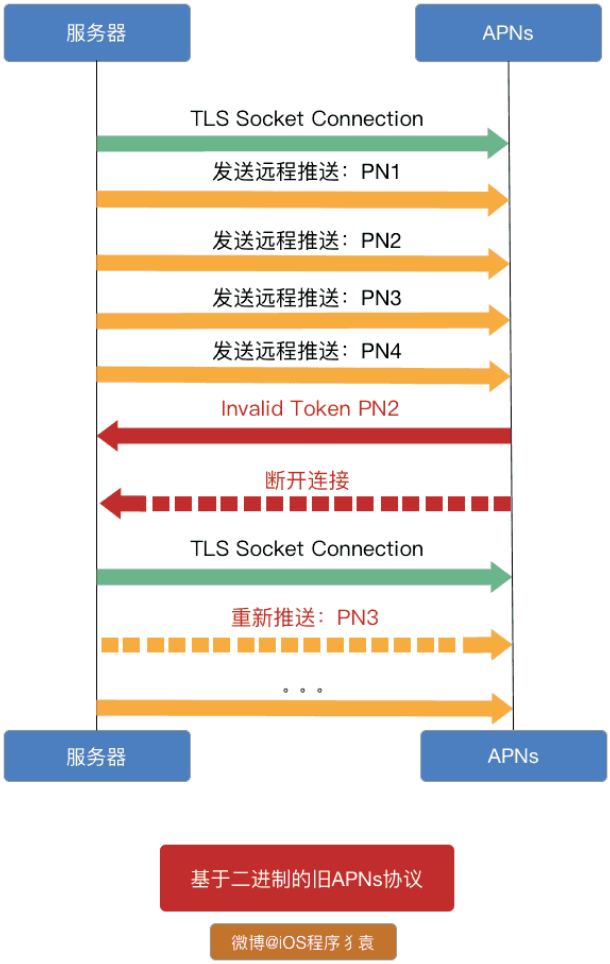
在介绍新版 APNs 前，让我们来吐槽下旧的基于二进制的 APNs 协议设计是多么反人类：

在理论上，推送分发的服务器要打开一个同 APNs 网关服务器的

连接，并保持这个连接。但在旧的协议下，APNs 服务却不保证 socket 能维持这个连接。如果通道上没有消息往来，空闲下来到话，socket 将被路由掐断。也就是说：APNs 连接说断就断，而你无能为力。有意思的是：在旧的协议下，如果服务器响应成功的话，你将不会收到任何回应，但是如果服务器响应失败（例如，使用了一个非法的 Push token），服务器将返回了一个错误编码，并关闭这个 socket。最重要的是，你必须重新发送使用这个无效 token 以后发送的所有推送（详情见示意图）。因此，你可能一直不能确定你的推送是否成功的被 APNs 服务器接收。

成功了不响应，失败了才响应，这个是最大的反人类。于是许多开发者想到了一个很 tricky 的办法:利用这个“漏洞”,比如在每发送 10 条后故意发送一个错误的 token，如果 APNs 有响应了，就可以确认 APNs 是处在可用状态的，进而确认这 10 条消息是发送成功的。如果没有响应就说明可能连接已经中断，那么这 10 条消息很可能是丢失的，然后做进一步的处理。但代价显而易见：将导致你们的推送系统性能低下。（本文中所说到“你们的推送系统”，如果是使用的第三方的 SDK 完成的推送服务，那么就是指 SDK 提供商所搭建的推送系统。如果是你们公司自己搭建的推送系统，那么就是指你们自己的推送系统。）苹果有一个名为"feedback"的服务，我们可以定时调用这个服务来获取 invalid tokens 的列表。这个服务你只要调用一次就可以获得所有的 invalid tokens 列表。所以，如果一个应用使用了很多不同公司的推送 SDK，他们将会争夺资源去轮询查找 invalid tokens 列表。invalid token 越多，你们的推送系统性能将越低。而且 APNs 只要一发生错误就关闭这个连接，然后重新连接。也就是“重启” socket 连接。

示意图：




*enter image description here*

图中的 PN2 去哪里了？它被放到了 feedback 列表里，等待下次你调用 feedback 服务，然后重发。

为什么 Apple 要在旧 APNs 中设计出“重启”的策略？

为了效率。

就像 PC 机出问题，我们总说“重启能解决 90%的问题”。

为了理解“重启”策略，我们可以类比下，熟悉 Erlang/OTP 的朋友可能知道，Erlang/OTP 在处理错误方面有独到之处：监督树（supervision trees）。大致来说，每一个 Erlang 进程都由一个监督进程发起并监视。当一个进程遇到了问题的时候，它就会退出。当进程退出的时候，其监督进程会将其重启。

（这些监督进程由一个引导进程（bootstrap process）发起，当监督进程遇到错误的时候，引导进程会将其重启）

其思想是，快速的失败然后重启比去处理错误要快。像这样的错误处理看起来跟直觉相反 —— 当错误发生的时候通过放弃处理来获得可靠性。但是重启的确是解决暂时性错误的灵丹妙药。

这也可能让你想到 DNS 服务发展史：

DNS 在设计之初是基于 UDP 的，显然这样的设计不能满足当今社会的准确性的需求，于是涌现了如 DNSPod 这样的基于 HTTP 的 DNS 解析服务。但是当时为什么这样设计实际也很好理解，UDP 效率高，一来一回网络上传输的只有两个包，而 HTTP 则需要三次握手三个包，再一拆包，就需要四个包。这是受限于当时整个社会的带宽水平较低，而现在没人会感激 UDP 所节省的流量，所有人都在诟病 DNS 污染问题。这样你也许就理解了，为什么旧的 APNs 设计如此反人类。这个是必经阶段。

那么接下来就让我们看看 Apple 为解决这些问题而推出的基于 HTTP/2 的全新 APNs 协议。

基于 HTTP/2 的全新 APNs 协议

来看下新版的 APNs 的新特性：

Request 和 Response 支持 JSON 网络协议

APNs 支持状态码和返回 error 信息

APNs 推送成功时 Response 将返回状态码 200，远程通知是否发送成功再也不用靠猜了！

APNs 推送失败时，Response 将返回 JSON 格式的 Error 信息。

最大推送长度提升到 4096 字节（4Kb）

可以通过 “HTTP/2 PING ” 心跳包功能检测当前 APNs 连接是否可用，并能维持当前长连接。

支持为不同的推送类型定义 “topic” 主题

不同推送类型，只需要一种推送证书 Universal Push Notification Client SSL 证书。

示意图：


*enter image description here*

其中最大的变化就是基于了 HTTP/2 协议，采用了长连接设计，并提供 “HTTP/2 PING 心跳包功能检测、维持当前 APNs 连接，解决了老 APNs 无法维持连接的问题。

而且新增的状态码特性，也解决了这个问题：无法获知消息是否成功地从你们的推送系统投递到了 APNs 上。理论上，你们可以保证消息是 100%投递到了 APNs 的，因为你可以准确的知道哪条消息到达了 APNs，哪些没到。重发特定失败消息成为可能。

所以上文开头的吐槽中第一条，有一句是“不到位的”，因为现在 SDK 的提供商能够准确地告诉你哪些消息推送到 APNs 了，哪些没有。

顺便介绍下 HTTP/2：

HTTP/2 是 HTTP 协议发布后的首个更新，于 2015 年 2 月 17 日被批准。它采用了一系列优化技术来整体提升 HTTP 协议的传输性能，如异步连接复用、头压缩等等，可谓是当前互联网应用开发中，网络层次架构优化的首选方案之一。

Apple 对于 HTTP/2 的态度也非常积极，2015 年 5 月 HTTP/2 正式发表后不久，便在紧接着 6 月召开的 WWDC 2015 大会中，向全球开发者宣布，iOS 9 开始支持 HTTP/2。

而且如果我们要使用 HTTP/2，那么在网络库的选择上必然要使用 NSURLSession。

我们都知道 HTTP/2 是复用 TCP 管道连接的，而且 HTTP/2 也以高复用著称，这也使新的 APNs 协议更加高性能。（题外话：这点也同样体现在 NSURLSession 底层对于每个 session 是对多个 task 进行连接的复用。）

Universal Push Notification Client SSL 证书

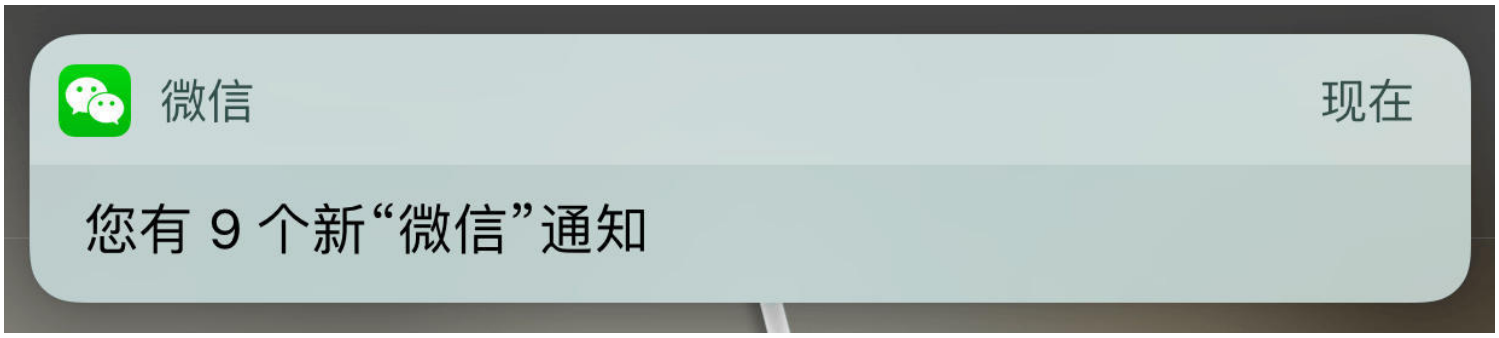
在开发中，往往一条内容，需要向多个终端进行推送，终端有：iOS、tvOS、 and OS X devices, 和借助 iOS 来实现推送的 Apple Watch。在以往的开发中，不同的推送，需要配置不同的推送证书：我们需要配置：dev 证书、prod 证书、VOIP 证书、等等。而从 2015 年 12 月 17 日起，只使用一种证书就可以了，不再需要那么多证书，这种证书就叫做 Universal Push Notification Client SSL 证书（下文统一简称：Universal 推送证书）。

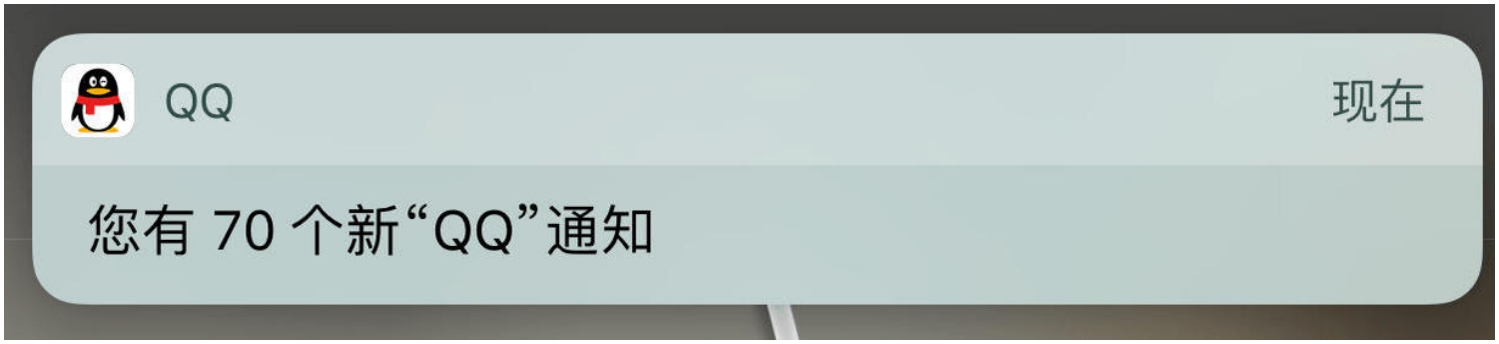
改进了，但仍需改进。还是有坑

APNs 的确改进来不少，但仍有需要改进对地方。还是有坑：

除了获取 TLS 证书比较复杂未解决外，还有一些坑：

文章开头提到过以下这种情况：




很遗憾的告诉你，你的吐槽是“到位的”：你以后还得忍受这种反人类的设计。

这中间发生了什么？

当 APNs 向你发送了多条推送，但是你的设备网络状况不好，在 APNs 那里下线了，这时 APNs 到你的手机的链路上有多条任务堆积，APNs 的处理方式是，只保留最后一条消息推送给你，然后告知你推送数。那么其他三条消息呢？会被 APNs 丢弃。

有一些 App 的 IM 功能没有维持长连接，是完全通过推送来实现到，通常情况下，这些 App 也已经考虑到了这种丢推送的情况，这些 App 的做法都是，每次收到推送之后，然后向自己的服务器查询当前用户的未读消息。但是 APNs 也同样无法保证这多条推送能至少有一条到达你的 App。很遗憾的告诉这些 App，这次的更新对你们所遭受对这些坑，没有改善。

为什么这么设计？APNs 的存储-转发能力太弱，大量的消息存储和转发将消耗 Apple 服务器的资源，可能是出于存储成本考虑，也可能是因为 Apple 转发能力太弱。总之结果就是 APNs 从来不保证消息的达到率。并且设备上线之后也不会向服务器上传信息。

所以上文开头的吐槽中第一条，也有一句是“到位的”，因为现在 SDK 的提供商依然无法保证，消息推到了 APNs，APNs 能推到 App 那里。

但 Google Cloud Messaging 就有这些特性。而且 GCM 现在也支持 iOS 设备了，那么 APNs 和 GCM 现在就形成了竞争关系。让我共同期待 APNs 在 2016 年 6 月的 WWDC 的能有新的改进吧。

对 App 开发的影响

想使用新协议，如果你用的第三方推送，这里最明显的操作，就是你必须更新到支持新协议的 SDK 版本。因为新协议需要 SDK 上传你 app 的 bundle id ,生成各个平台推送用的 topic。如果你们自己搭建的服务，则需要你自己上传。老协议不用上传。

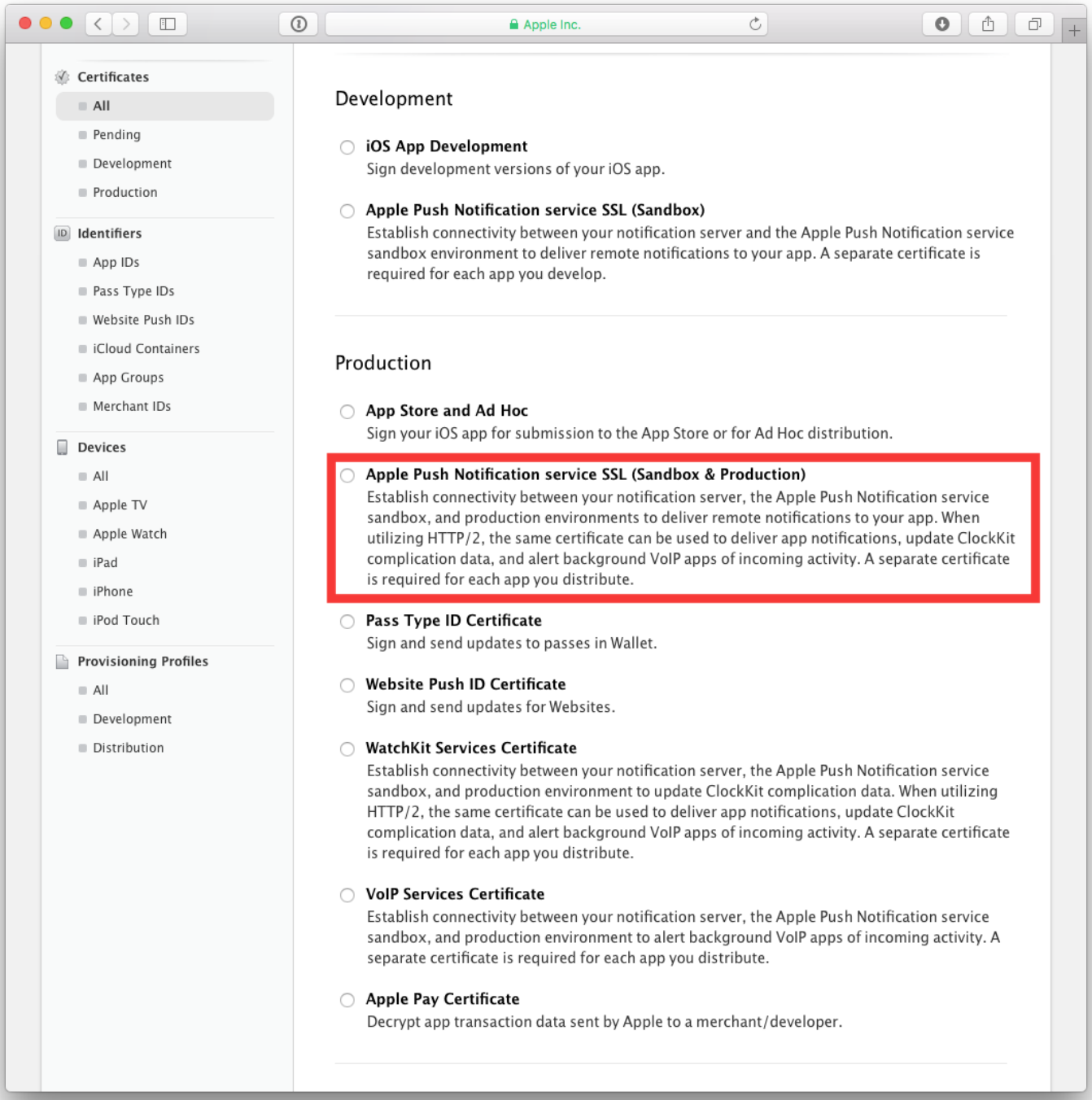
新 APNs 支持 iOS6 等全版本推送内容达 4096 字节，旧 APNs 是 14 年 6 月之前只支持 256 字节，在此之后支持 iOS8 以上 2048 字节。以前受限于推送字节，比如推文章 url，开发者选择超出 256 后推送 id，甚至不判断直接推 id，接收后再请求完整 url。一旦请求错误，推送内容可能丢失。现在可以避免了。

如何创建 Universal Push Notification Client SSL 证书

现在你知道什么是 Universal Push Notification Client SSL 证书了，那么如何创建它？

*what is Universal Push Notification Client SSL Certificate*

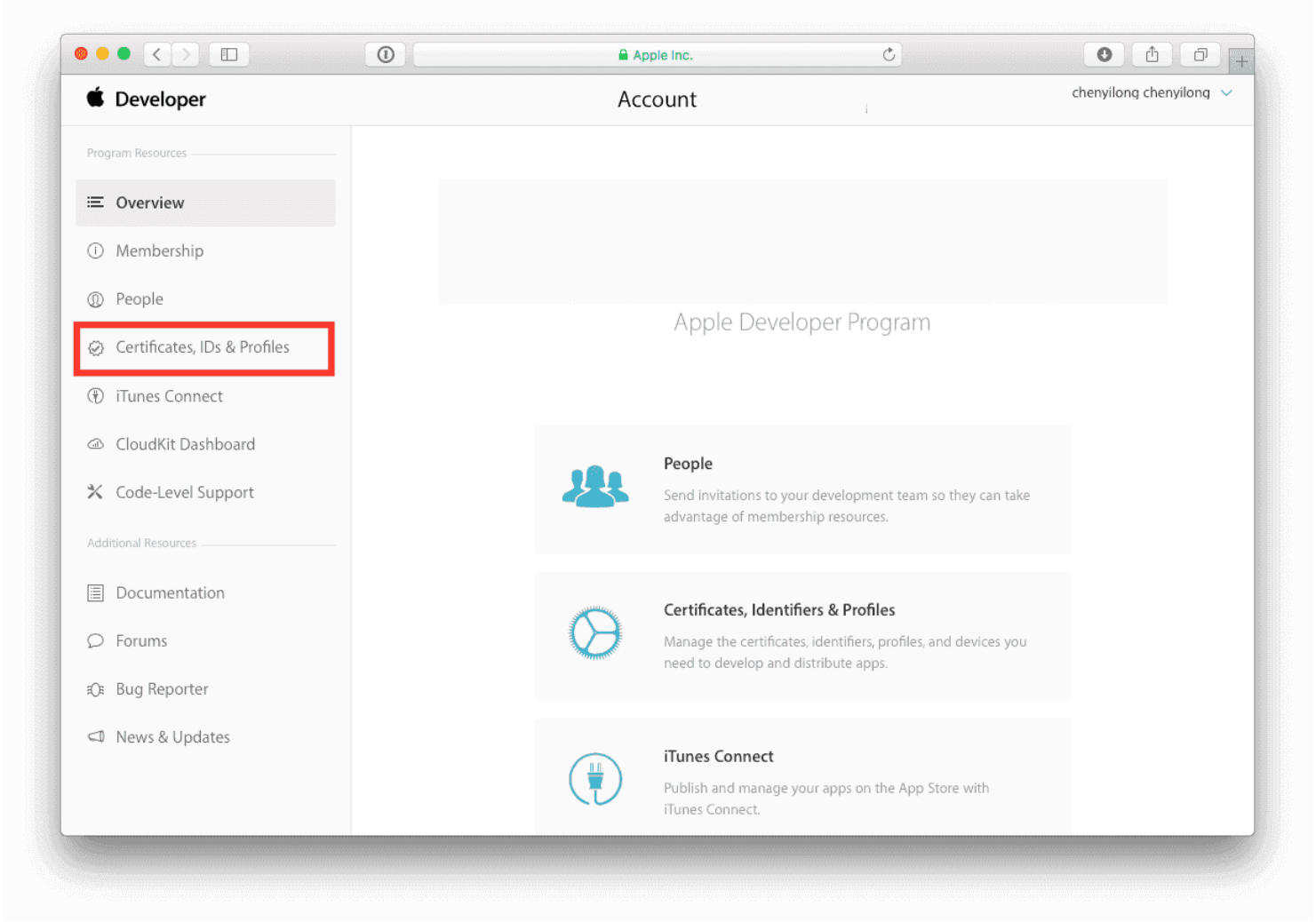
图中其他方式，就叫做非 Universal 方式（下文简称：非 Universal 推送证书）：




*what is not Universal Push Notification Client SSL Certificate*

这里也推荐使用 Universal 推送证书来进行推送服务。详细的创建步骤如下所示：

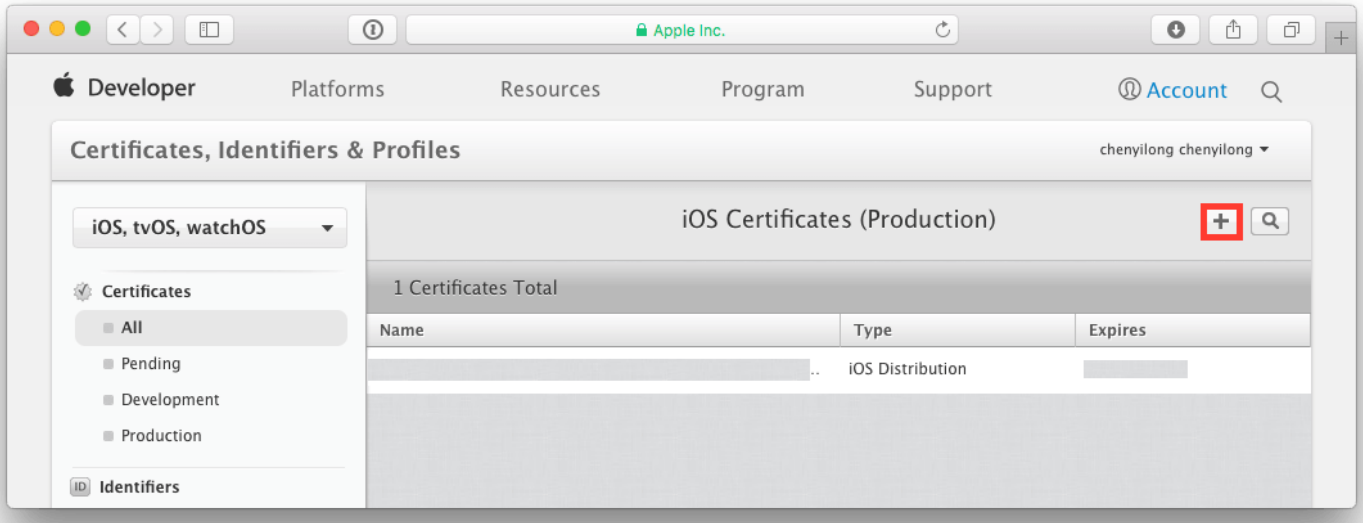
前往苹果开发者中心进行登录，并点击 “Certificates, Identifiers & Profiles”。




*enter Certificates, Identifiers & Profiles*

选择在 Certificates 栏下的“All”。

点击下图中红色边框内的加号按钮。

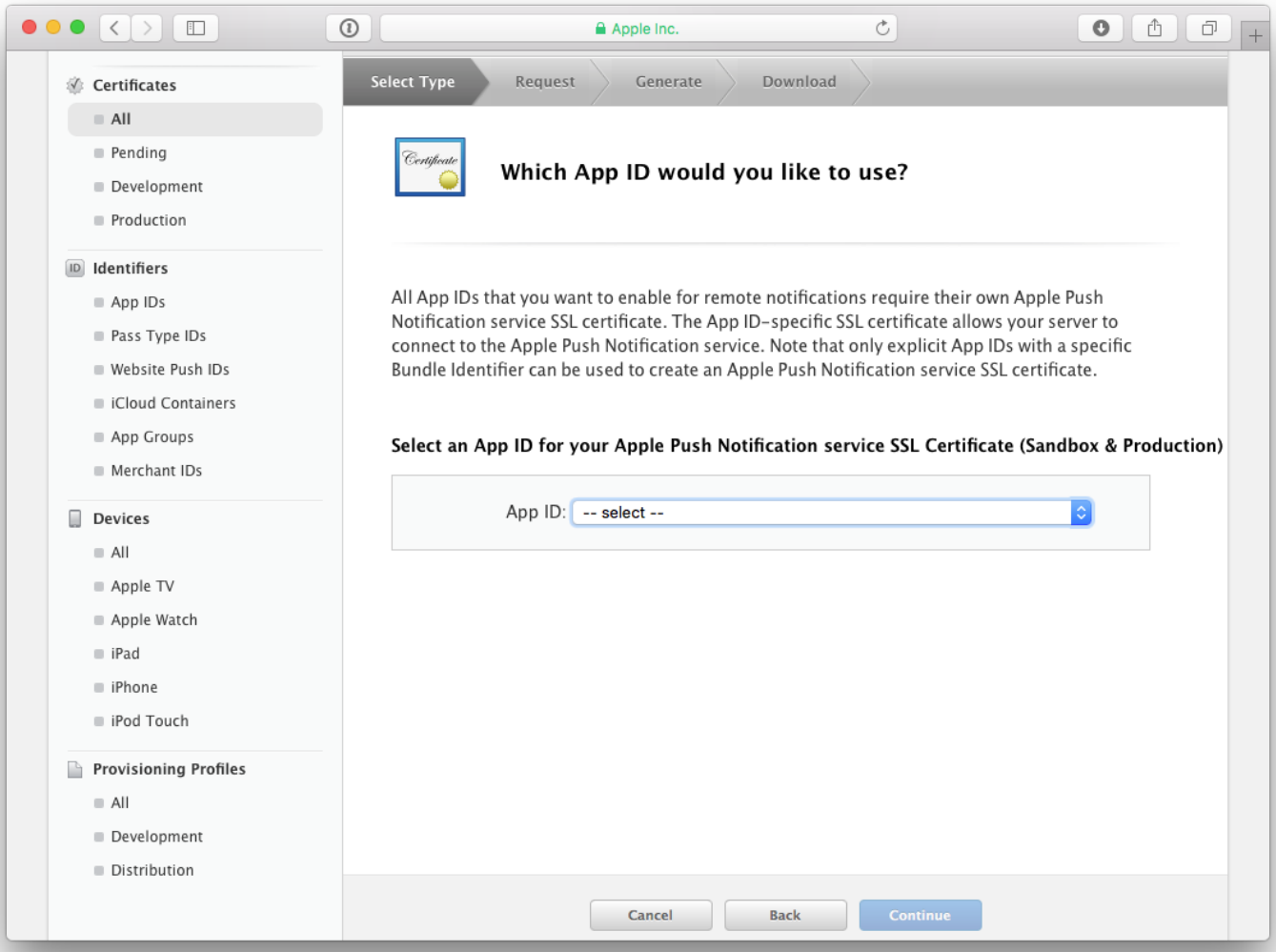



*Create SSL certificate*

选择 “Production” 栏下的 “Apple Push Notification service SSL (Sandbox & Production)” 勾选后，点击下一步。

*Select push certificate*

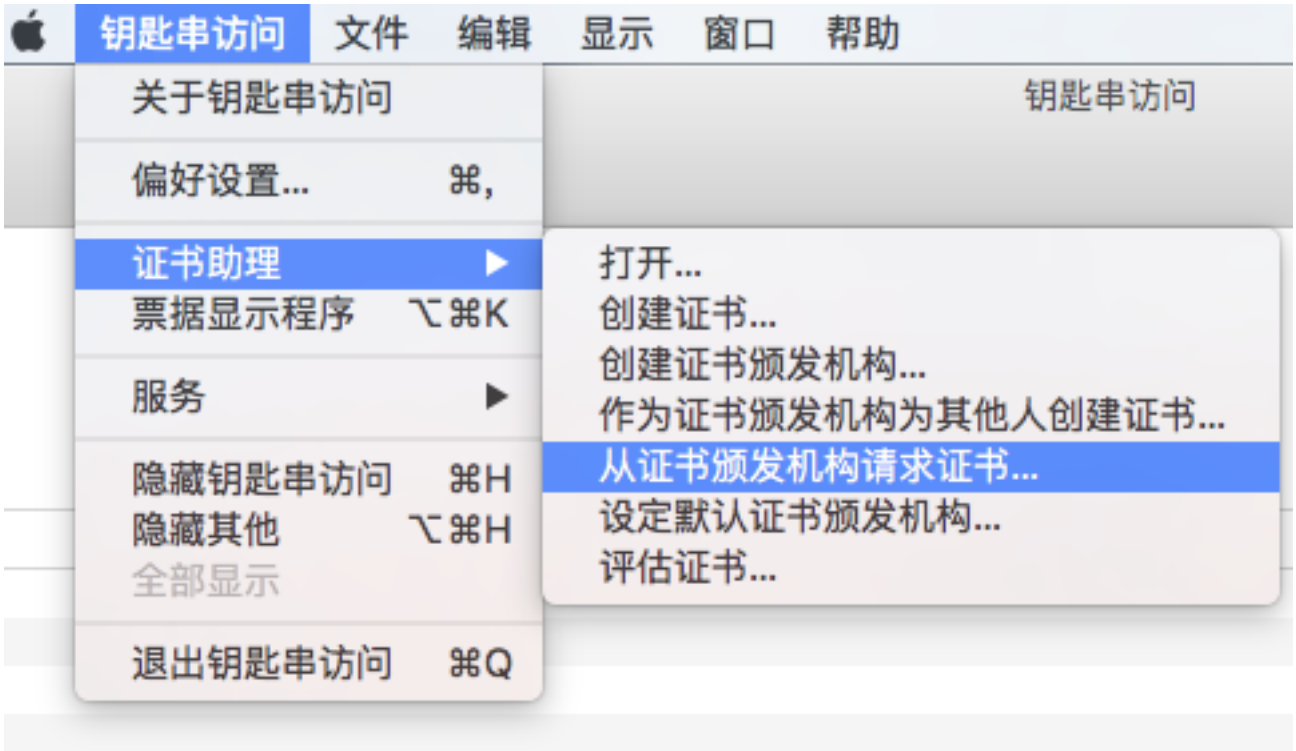
从 App ID 下拉菜单中选择你需要的 App ID ，点击下一步。




*select App ID*

这时会出现 About Creating a Certificate Signing Request (CSR)。


*guide to create a CSR* 根据它的说明创建 Certificate Signing Request。



*how to create a CSR*

点击下图中的 “Choose File” 按钮：

*upload CSR File*

上传刚刚生成的 .certSigningRequest 文件 生成 APNs Push Certificate。

下载证书。

双击打开证书，证书打开时会启动钥匙串访问工具。 在钥匙串访问工具中，你的证书会显示在 “证书” 中，注意选择左下角的 “证书” 和左上角 “登录”。


*confirm create cer success*

结束语

对于 APNs 而言，iOS9 的这一更新是有划时代意义的，请即刻敦促你们公司的服务端进行升级，或者使用支持新 APNs 协议的 SDK 进行推送服务。