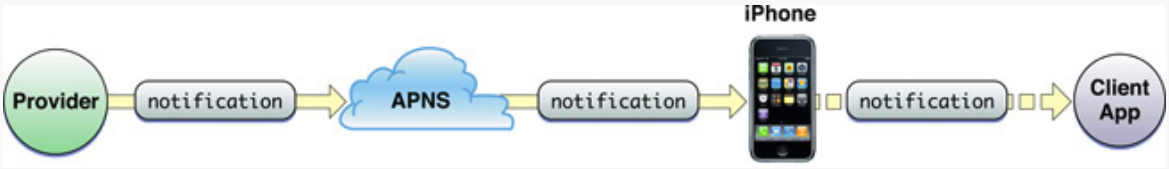
APNS的推送机制

首先我们看一下苹果官方给出的对ios推送机制的解释。如下图



Provider就是我们自己程序的后台服务器，APNS是Apple Push Notification Service的缩写，也就是苹果的推送服务器。

上图可以分为三个阶段：

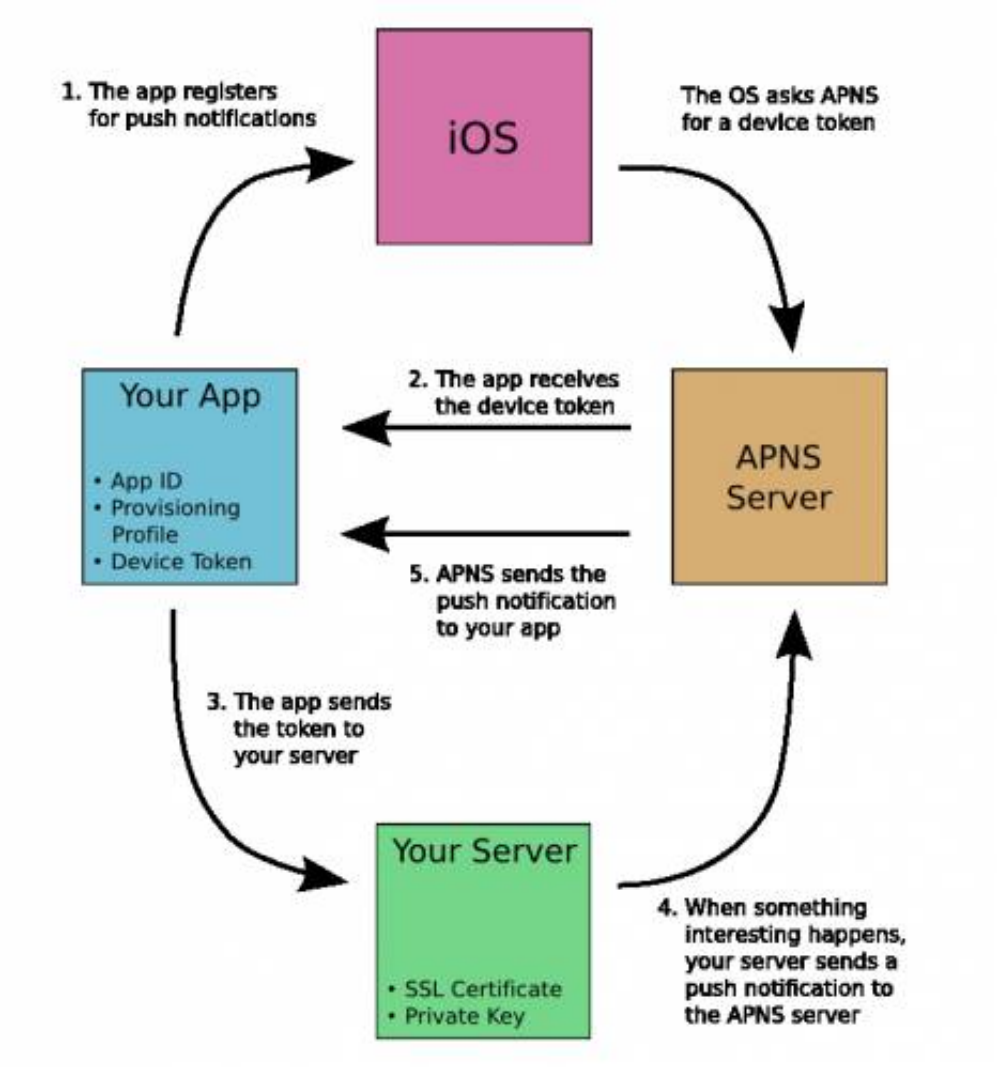
第一阶段：应用程序的服务器端把要发送的消息、目的iPhone的标识打包，发给APNS。

第二阶段：APNS在自身的已注册Push服务的iPhone列表中，查找有相应标识的iPhone，并把消息发送到iPhone。

第三阶段：iPhone把发来的消息传递给相应的应用程序，并且按照设定弹出Push通知。

APNS推送通知的详细工作流程

下面这张图是说明APNS推送通知的详细工作流程：



根据图片我们可以概括一下：

1. 应用程序注册APNS消息推送。
2. iOS从APNS Server获取devicetoken，应用程序接收device token。
3. 应用程序将device token发送给程序的PUSH服务端程序。
4. 服务端程序向APNS服务发送消息。
5. APNS服务将消息发送给iPhone应用程序。

远程推送操作过程简介

1. App启动过程中，使用`UIApplication::registerForRemoteNotificationTypes`函数与苹果的APNS服务器通信，发出注册远程推送的申请。若注册成功，回调函数`application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken`

会被触发，App可以得到deviceToken，该token就是一个与设备相关的字符串.

2. App获取到DeviceToken后，将DeviceToken发送给自己的服务端。
3. 服务端拿到DeviceToken以后，使用证书文件，向苹果的APNS服务器发起一个SSL连接。连接成功之后，发送一段JSON串，该JSON串包含推送消息的类型及内容。
4. 苹果的APNS服务器得到JSON串以后，向App发送通知消息，使得App的回调函数`application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo`被调用，App从userInfo中即可得到推送消息的内容。

用到的证书文件及生成过程

1. certSigningRequest文件，该文件在MAC系统中生成，用于在Apple网站上申请推送证书文件。 生成过程： 打开应用程序中的“钥匙串访问”软件，从菜单中选择 “钥匙串访问”-》“证书助理”-》“从证书颁发机构请求证书”，邮箱和名称随便填写，然后选择保存到磁盘，就可以在本地生成一个CertificateSigningRequest.certSigningRequest文件。

2. 注册一个支持push的app id，后面会用到。 生成过程： 进入developer.apple.com，选择member center - Certificates, Identifiers & Profiles - Identifiers- App Ids，然后选择注册app id，设置appid名称，同时，app id suffix一栏必须选择explicit app id，然后设置bundle id，最后勾选 App Services中的 Push Notifications，这样就可以注册一个支持push的aphid。

3. 推送证书cer文件，该文件在developer.apple.com中生成，用于生成服务端需要的文件。生成过程： 进入developer.apple.com，选择member center - Certificates, Identifiers & Profiles - Certificates，然后选择创建certificate，类型分为Development和Product。这里以Development为例，选择Apple Push Notification service SSL (Sandbox) ，然后下一步，选择之前生成的支持push的AppId，然后下一步，提交之前创建的CSR文件，再下一步就可以生成cer文件，然后保存到本地。

4. 生成服务端使用的证书文件。如果是使用网上的mac 版PushMeBaby工具，在mac机器上进行推送消息的发送，那么有上面的cer文件就够了。如果是使用PHP、java/c#开发自己的服务端，那么还需要将上面的cer文件做一个转换，生成pem文件或者p12文件。

5. 生成Xcode使用的provisioning文件,该文件用于真机调试。 生成过程： 进入developer.apple.com，选择member center - Certificates, Identifiers & Profiles - Provisioning Profiles，然后选择创建Provisioning file，接着选择iOS App Development ，下一步选择AppId，选中之前建立的支持push的appid，接着下一步选择支持push的certificate，下一步勾选需要支持的device id，最后一步设置provisioning文件的文件名，这样provisioning文件就生成了。

客户端的开发

首先下载前面建立的cer文件和provisioning文件，双击，导入到xcode中，在build setting中code signing一栏里选择这两个文件的名称，这样就可以将支持push的app部署到真机中。然后是要处理推送消息。

客户端对推送消息的处理分两种情况：

1. 在App没有运行的情况下，系统收到推送消息，用户点击推送消息，启动App。此时，不会执行前面提到的`didReceiveRemoteNotification`函数，而是在App的applicationDidFinishLaunching函数中处理推送，通过以下代码可以获取推送消息中的数据：

```
NSDictionary *userInfo =[launchOptions objectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];
```

1. 当APP处于前台时，系统收到推送消息，此时系统不会弹出消息提示，会直接触发application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo函数，推送数据在userInfo字典中。

当App处于后台时，如果系统收到推送消息，当用户点击推送消息时，会执行application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo函数， 此时AppDelegate中函数执行的顺序为：`applicationWillEnterForeground`、`application:didReceiveRemoteNotification`、`applicationDidBecomeActive`