Implementing a Solution to Communicate with APN Server for Sending Push Notifications

Petrika Manika
Department of Informatics,
University of Tirana
petrika.manika@fshn.edu.al

Elina Jaupllari
Ict Solutions
Tirane
elina.jaupllari@ictedu.info

Prof.Asoc.Dr. Endri Xhina
Department of Informatics,
University of Tirana
endri.xhina@fshn.edu.al

Abstract

Nowadays, getting information through notifications is becoming more and more used. Each of the mobile technology giants iOS and Android has created their own way to send push notifications. Mobile push notifications are an important feature of mobile applications and are widely used. Popular mobile applications such as Facebook, Hangouts, WeChat, and other applications like weather, transport service or traffic applications, have been built by implementing mobile push notification technologies. The purpose of this paper is to explain the construction of an architecture for sending a push notification from a server to an application in the iOS operating system and notification delivery management. The proposed system was implemented using Xamarin.iOS. The system provides a solution for sending a push notification automatically through the Apple Push Notification Service.

1. Introduction

In the last decades, human beings are living in the era of a technological revolution. The use of mobile devices is changing the daily lives of people. Mobile devices have become multifunctional and used both for communication, business or advertising, keeping and circulating a large information. The smartphone has become the main means of information, breaking the use of TV or newspapers, books, and so on. Nowadays, users encounter a large amount of information and a limited amount of time to get all this information. Often the amount of information is unmanageable, it is not possible to filter and select only information that is of interest to the user. Developers and researchers have found a way to manage this information, to attract user's attention, and simplify their use, by users. Push and locally scheduled notifications can provide users with timely information and the ability to initiate necessary actions as a response. Mobile push notifications are an important feature of mobile applications and are widely used because of the mobile IT services they offer [Rog08]. The most popular mobile applications such as Facebook, Hangouts, or instant messaging systems such as WeChat or other app are all implemented through mobile push notification. But many other applications such as weather, traffic and transport services have also been built using push notifications to send real-time information to users. [Hol10] Before this evolution, users should take out their mobile devices and check it from time to time for new emails. Urgent and important communication sometimes had to wait. There was a limitation of requests for email control because any email checking request was sent to the server, which then checked for new emails. All this process required time, and turned into a nightmare for the user.

The first presentation of a notifications started in 2003 with the Blackberry Push Email, the first email notification system. It was a popular device because it was ideal for moving businesses. They received "push" emails as soon as email arrived on the device. The announcement was the smallest information window on the mobile screen. The email was read and can be replayed immediately, in real time, saving time and money. This made communication much easier.

Then, in 2009 it was Apple who introduced its version of push notification and called APNS, the Apple Push Notification System. APNS is a notification service platform that allows third-party applications developers to send notifications to applications that are installed on apple devices. [Gus14]

In 2010 it was Google who introduced its Google Cloud Messaging service that enables third-party application developers to send information from their servers to the applications installed in android operating systems as well as in the google chrome browser. [Gus14]

In the following years, these services were improved by adding content, images, videos and interactive buttons that allowed communication with the application publisher or application navigation. [Warr14]
Once upon a time it was very difficult for the users to be informed in real time whether by email. Today's push notifications, stay alongside email as a legitimate way for users to communicate, solving this problem. Notifications have changed the way they interact with smartphones, releasing our inbox spaces, and at the same time developing and facilitating users to benefit from and to deliver services. Basically, a Push Notification is a short message or a notification that is sent through an installed application, to anyone who has installed the application and has received notifications. Mobile notifications are divided into two categories, local notification and remote notification or otherwise push notifications. Their change is that for local notifications, it configures the details of the notification and passes these details to the system, which then handles the notification, when the application is not in the foreground. Remote notifications are notifications that use one of the developer company servers to send notifications to user devices.
The purpose of this paper is to explain the construction of an architecture for sending a push notification from a server and managing the notification delivery. The rest of the article is structured as follows: Section 2 presents the construction of an architecture for sending a push notification. Section 3 explains how notification delivery is managed. Section 4 explains the tests performed and the implementation of this architecture in a concrete application. In the end, Section 5 concludes the paper presenting the lessons learned and work in the future.

2 Architecture for Sending a Push Notification

This section introduces the architecture of building, sending and receiving a notification, presenting the structure, key components, and functionalities.
Currently, the technological market offers various architectural solutions for sending and receiving push notifications. As noted above, Apple offers its iOS Operations System to its Apple Push Notification Service. Google provides its service initially through Google Cloud Messaging which is currently replaced through Firebase Cloud Messaging for iOS, Android and for web. Let's talk more specifically about the Apple Push Notification Service Architecture.
An APNs consists of three main elements:

a - Provider - A provider is a server, that you deploy and manage, and which is configured to work with APNs.
b - APNs -Apple Push Notification service (APNs) is the centerpiece of the remote notifications feature. It is a robust, secure, and highly efficient service for app developers to propagate information to iOS.[App]
c - Client Application- is the application on client mobile where are delivered the notifications.

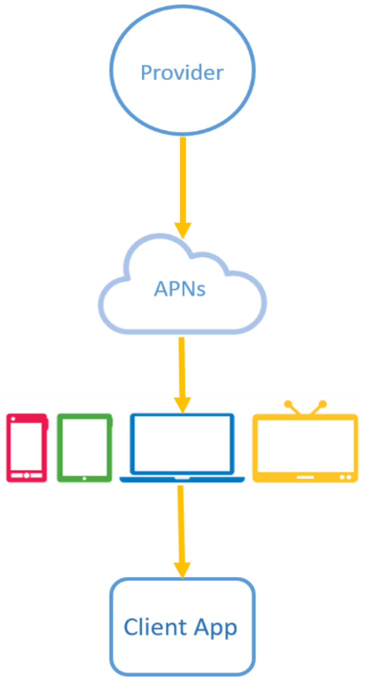


Figure 1: Architecture of delivering a remote notification from a provider [App]

2.1. Push Notification Path from Provider to Device

On initial lunch application app, the IOS system creates an accredited connection, encrypted, and continuous IP link, between the app and APNs. This connection allows the application to make the appropriate settings to enable it, to be notified, otherwise called the application when downloaded from the app store, requiring user approval to receive notification from this application and

performs the appropriate settings on the system [App].
- iOS requests a device token from Apple Push Notification Service (APNs).
-The application receives the token, which functions as the address to send a push notification to.
-The application sends the token of the device to your server.
-When prompted, the server will send a push notification with a device token to the APNs.
-APNs will send a push notification to the user's device.

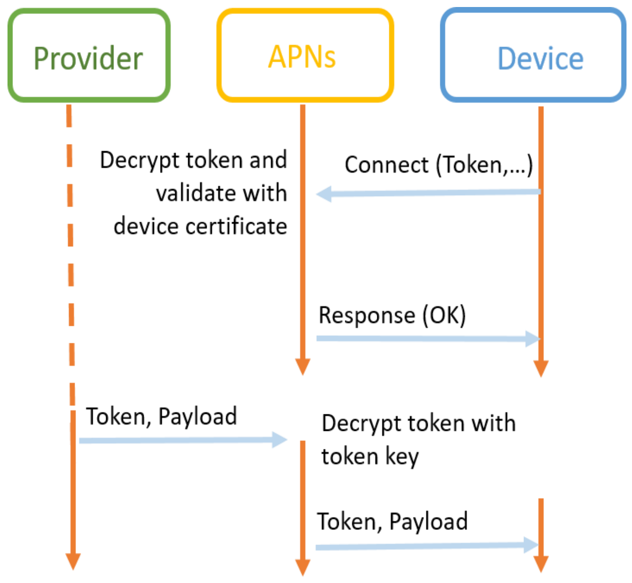

Provider
APNs
Device
Decrypt token and validate with device certificate
Connect (Token,...)
Response (OK)
Token, Payload
Decrypt token with token key
Token, Payload

Figure 2: Remote Notification path from provider to device [App]

2.2. Provider Responsibilities

A provider is a server, that is deployed and managed from a company, and it can be configured to work with APNs. It has some responsibilities for participating with APNs.
● Receiving app-specific device token from APNs which allow a provider to know about each running instance of the app.
● Determining when remote notifications need to be sent to each device.
● Building and sending notification request to APNs.
Each remote notification request from provider must send:
1. A constructed JSON dictionary which contain the notification Payload
2. A globally-unique device token
3. HTTP/2 request to APNs, including cryptographic credentials in the form of a token or a certificate, over a persistent secure channel.
Then after notification request from provider the APNs delivers corresponding notifications to the intended devices on your behalf.

2.3 Security Architecture

APNs enforces end-to-end, cryptographic validation and authentication using two levels of trust: connection trust and device token trust.
A device token is an opaque NSData instance that contains a unique identifier assigned by Apple to a specific app on a specific device. Only APNs can decode and read the contents of a device token. Each app instance receives its unique device token when it registers with APNs, and must then forward the token to its provider. The provider must include the device token in each push notification request that targets the associated device; It ensures that notifications are routed only between the correct start (provider) and end (device) points. Connection trust works between providers and APNs, and between APNs and devices. We are mostly focused in Provider-to-APNs Connection Trust.
There are two schemes available for negotiating connection trust between your provider and Apple Push Notification service:
-Token-based provider connection trust:
A provider using the HTTP/2-based API can use JSON web tokens (JWT) to provide validation credentials for connection with APNs.
In this scheme, provider provision a public key to be retained by Apple, and a private key which must retain and protect. Providers then use their private key to generate and sign JWT provider authentication tokens. Each of push notification requests must include a provider authentication token.
-Certificate-based provider connection trust:
A provider can, alternatively, employ a unique provider certificate and private cryptographic key. The provider certificate, is provisioned by Apple when establish push service, and identifies bundle ID for apps. Depending on how you configure and provision the certificate, the trusted connection can

also be valid for delivery of remote notifications to other items associated with your app, including Apple Watch complications for your applications, and for voice-over-Internet Protocol (VoIP) status notifications. APNs delivers these notifications even when those items are running in the background. With certificate-based trust, APNs maintains a certificate revocation list; if a provider's certificate is on the revocation list, APNs can revoke provider trust.[App]

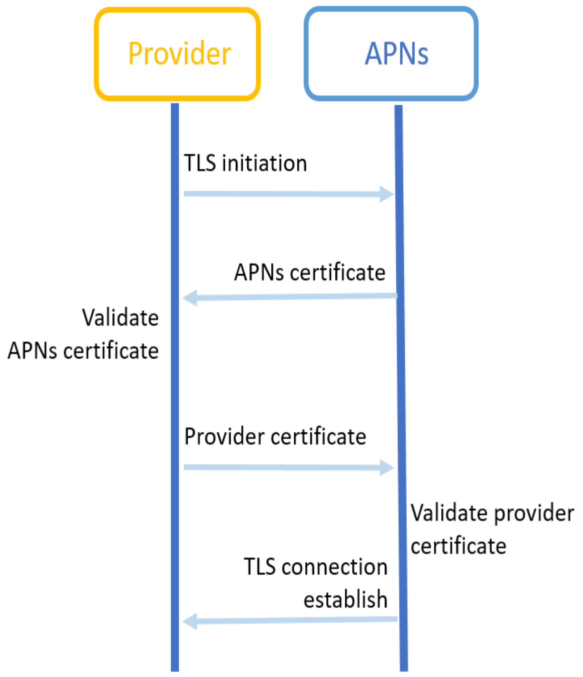


Figure 3: Illustrates the use of an Apple-issued SSL certificate to establish trust between a provider and APNs.

As shown in Figure 3, certificate-based provider-to-APNs trust works as follows:

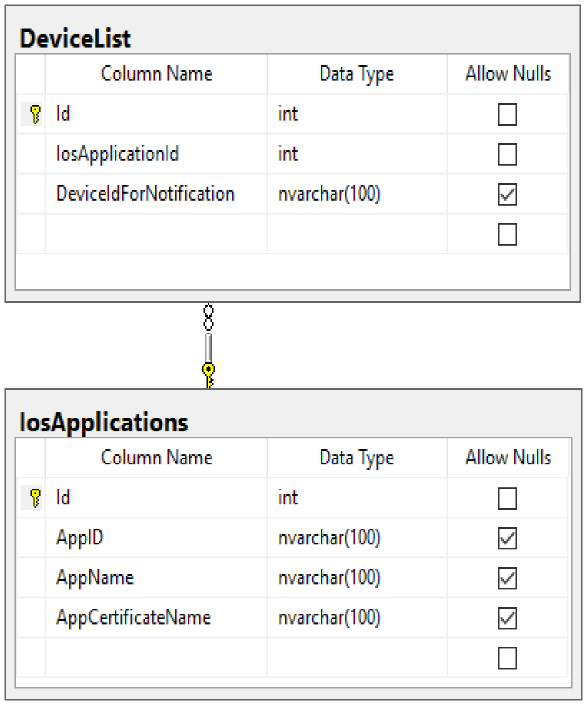
1. Provider asks for a secure connection with APNs using transport layer security (TLS)
2. APNs then gives the provider an APNs certificate, which then is validated from the provider.
3. Provider must then send its Apple-provisioned provider certificate back to APNs.
4. APNs validates provider certificate, thereby confirming that the connection request originated from a legitimate provider, and establishes a TLS connection.

At this point, connection trust is established and provider server is enabled to send certificate-based remote push notification requests to APNs.

3. Management of Notification Delivery

Managing the delivery of a notification through the Apple Push Notification System requires the configuration and construction of an appropriate architecture. This architecture is comprised of a provider with a notification and window service database that enables communication with the APNs and the Application where the notifications are received.

Initially, to send a notification to a mobile application with iOS, a database should be built to save the various application notifications. For each of the notifications in the database, it should be kept, if the notification is sent to the user or not. This can be accomplished through a column where sent alerts can be saved with a value of 1 and uninvited ones can be saved with a value of 0.



DeviceList

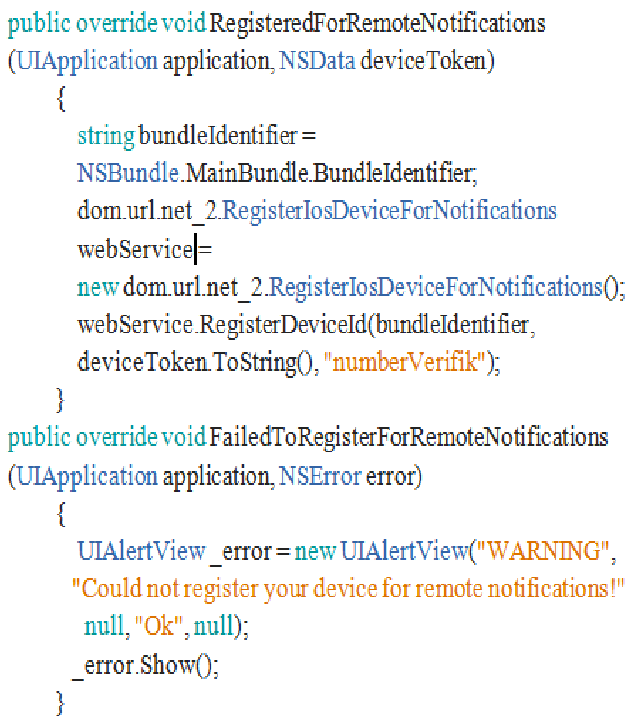
Column Name	Data Type	Allow Nulls
Id	int	☐
IosApplicationId	int	☐
DeviceIdForNotification	nvarchar(100)	☑
		☐

IosApplications

Column Name	Data Type	Allow Nulls
Id	int	☐
AppID	nvarchar(100)	☑
AppName	nvarchar(100)	☑
AppCertificateName	nvarchar(100)	☑
		☐

Figure 4: Database Diagram for Device List Identifier and applications of request.

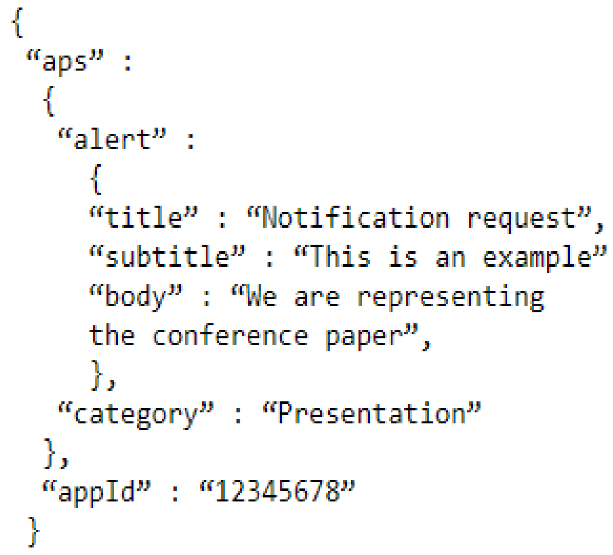
Also, a server must create a model for storing devices that want to receive notification from the application which I will explain below.
Secondly, in the iOS mobile application, users should be offered the option of choosing a popup if they want to receive notifications from this application or not. At the moment the user accepts the application to send notification to that device, APNs generates a unique location for this device and sends the device token that is being registered. This device token as mentioned above is stored in the database of recorded devices for receiving notifications. In this database, apart from the device token, there is also information about the application from which this device comes from.



```
public override void RegisteredForRemoteNotifications
(UIApplication application, NSData deviceToken)
      {
        string bundleIdentifier =
        NSBundle.MainBundle.BundleIdentifier;
        dom.url.net_2.RegisterIosDeviceForNotifications
        webService=
        new dom.url.net_2.RegisterIosDeviceForNotifications();
        webService.RegisterDeviceId(bundleIdentifier,
        deviceToken.ToString(), "numberVerifik");
      }
public override void FailedToRegisterForRemoteNotifications
(UIApplication application, NSError error)
      {
        UIAlertView _error = new UIAlertView("WARNING",
       "Could not register your device for remote notifications!"
         null, "Ok", null);
       _error.Show();
      }
```

Example 1: Register for notification C# code in Xamarin.iOS.

In the server there is also a service window which controls in a given time frame which is determined by the developer himself, whether there are new releases or not. Once the window service finds a new notification, it creates a JSON format with the data of this notice and sends this notification to the APNs, at this moment a connection is established between the provider and APNs that we explained in the above section. Window Service sends APNs apart from the notification and the list of Devices that should receive the notification. Then it is APNs that manages the sending of notifications even if any of the application is not in the foreground or is uninstalled from mobile.



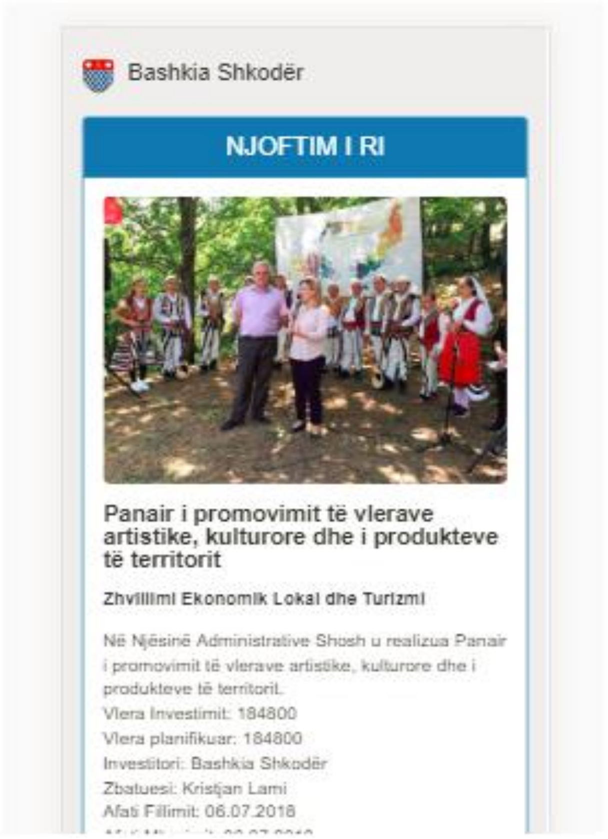
```
{
 "aps" :
  {
   "alert" :
     {
     "title" : "Notification request",
     "subtitle" : "This is an example"
     "body" : "We are representing
     the conference paper",
     },
   "category" : "Presentation"
  },
  "appId" : "12345678"
 }
```

Example 2: A remote notification payload for showing an alert in iOS mobile devices.

4. Implementation of Apple Push notification in iOS Application (Testing and result)

The presented system is implemented in the mobile app "Buxheti.info/Shkoder". This application is currently on the iOS platform. For the purpose of the law "On Public Information" and with the desire of the municipality to be transparent and closer to the citizen, the Municipality of Shkodra has implemented the Budget Transparency Application.
The application enables citizens to get acquainted online with investments and expenses incurred by the municipality. Budget transparency has been designed and structured as an instrument for monitoring municipal public spending by citizens and civil society organizations. This increases the public's access to information and sensitizes them to

increase participation in the process of drafting the strategic financial documents of the municipality. The implementation of the push notification was seen as a good opportunity for informing the citizens about the expenses incurred by the municipality.
To implement the notifications, a server was used where the database and a windows service were located. For each device that downloaded the application, a device token was stored in the database that also served as the device identifier for the notification delivery. Also, notifications were stored in the database and specified whether they were sent or not. The server was running a service window every 30 minutes and checked if there was any new notification or not. As soon as it found new releases it was sent to the APNs and notification was received on the devices that were already registered.



Panair i promovimit të vlerave artistike, kulturore dhe i produkteve të territorit

Zhvillimi Ekonomik Lokal dhe Turizmi

Në Njësinë Administrative Shosh u realizua Panair i promovimit të vlerave artistike, kulturore dhe i produkteve të territorit.
Vlera Investimit: 184800
Vlera planifikuar: 184800
Investitori: Bashkia Shkodër
Zbatuesi: Kristjan Lami
Afati Fillimit: 06.07.2018

Figure 5: Example of a notification sent in an iOS device.

5. Conclusion

This paper presented a proposal architecture for iOS Push notification system using APNs. To develop a whole notification system, it is really hard, and requires lot of time and money, it also needs to manage invalidated tokens, security, applications and users. Needs to use different technologies and a good development team. As the conclusion, using third party push notification service provider reduce the pressure and also make the notification system easy and fast. It costs less and is more safe and secure.

References

[Rog08] R. Roger.; V.Wright, “*Assessing technology’s role in communication between parents and middle schools*”, Electronic Journal for the Integration of technology in education, 2008, Vol 7, pp. 36-58

[Hol10] P.Holleis, M.Wagner, S.Böhm, & j.Koolwaaij, "*Studying mobile context-aware social services in the wild*", Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries,2010, pp. 207-216 http://dx.doi.org/10.1145/1868914.1868941

[Gus14] M. Gusev and S. Ristory, "*Alert notification as service*", Croatia, 2014. http://dx.doi.org/10.1109/mipro.2014.6859584

[Warr14] I.Warren, A.Meads,S. Srirama., T.Weerasinghe & Paniagua, "Push Notification Mechanisms for Pervasive Smartphone Applications ",Pervasive Computing, IEEE, 2014,13(2), pp. 61-71. http://dx.doi.org/10.1109/MPRV.2014.34

[App] Local and Remote Notification Programming Guide.