International Journal of Advanced Science and Technology
Vol. 32, July, 2011

Designing the Multimedia Push Framework for Mobile Applications

Dongcheul Lee

Department of Electronics Computer Engineering,
Hanyang University, Seoul, Korea
jackdclee@hanyang.ac.kr

Abstract

Mobile applications mostly require multimedia data from a server through the mobile network. To update multimedia data in the application with new data in the server, push mechanism is usually used. Push frameworks provide simple mechanism that a server can notify mobile applications to contact the server to fetch new data on a server. Developers who want to use these frameworks should implement a push gateway which interconnects with the push framework. This paper designs a new multimedia push framework to integrate diverse existing push messaging gateways and to provide a store-and-forward function for mobile applications. By integrating the gateways, the server side developers do not have to implement many kinds of gateway functions. Also, by providing store-and-forward functionality, the mobile application developers do not have to worry about losing their notification message in the air area.

Keywords: *mobile application, push framework, framework design*

1. Introduction

Since smart phones were introduced to the world, many mobile applications have been developed [1]. These mobile applications mostly require data from the mobile network. There are two kinds of approaches to update data in the applications. One approach is polling, which the applications polls a server to check new data periodically [2]. Even though there is no new data on the server, this approach should check the server, which cause using unnecessary mobile network bandwidth and consuming the battery of the mobile devices. Also, if the polling period is extended to reduce these drawbacks, a mobile phone user should wait for the next polling period to get new data. Another approach is pushing, which the server transfers data to the mobile applications if new data is available on the server [3]. In this approach, the mobile applications do not consume unnecessary network bandwidth and a mobile phone user can get new data without waiting for the next polling period. Therefore, if the data does not updated periodically, pushing is the preferred solution.

To push new data to the mobile applications, a push massaging framework was introduced including Apple introduced Apple Push Notification Service (APNS) [4] and Google introduced Cloud to Device Messaging (C2DM) [5]. It is a service which helps mobile application developers send new data from a server to their mobile applications on iOS [6] or Android [7] devices. This service provides a simple mechanism that a server can notify mobile applications to contact the server to fetch new data on a server.

This paper proposes a new framework to integrate diverse existing push messaging gateways and to provide a store-and-forward function. By integrating diverse existing push messaging gateways, the server side developers do not have to implement many kinds of gateway functions. Once they implement an interface with the new framework, they can send push notification to diverse mobile applications by using APNS, C2DM, or 3rd party push notification framework. Also, by providing a store-and-forward function, the mobile application developers do not have to worry about losing their notification message in the air area. Since mobile devices can be in shadow area or can be turned off, the notification message can be lost if the framework does not provide the store-and-forward function.

2. Related Works

To push multimedia information to mobile devices, feature phones have used SMS and MMS [8]. Mobile network operators have offered these services to their subscribers so that they can exchange text messages, images, videos, and photos. However, smart phones need more extended push functions than SMS and MMS. They cannot send messages to a specific mobile application in a smart phone. To solve this problem, Apple introduced APNS and Google introduced C2DM.

2.1. APNS

APNS is a push notification service framework made by Apple for iPhone mobile applications. Because iOS has not supported background processes, mobile applications could not get messages by polling mechanism. Therefore, the applications that want to receive messages from the remote servers can use APNS framework. It is implemented in iOS 3.0 or above. Figure 1 shows the flow of the APNS messages. Service provider generates a notification request and sends it to the APNS gateway. The APNS gateway receives the request and forwards it to the APNS server. Since the APNS server has active sessions with the APNS clients, it can send its request to the iOS devices. Then iOS can wake up the specified application and the application can be active and process the request. Since there are air interface between the APNS server and the APNS client, the message can be dropped easily if the air condition is not good enough or if the device is turned off.

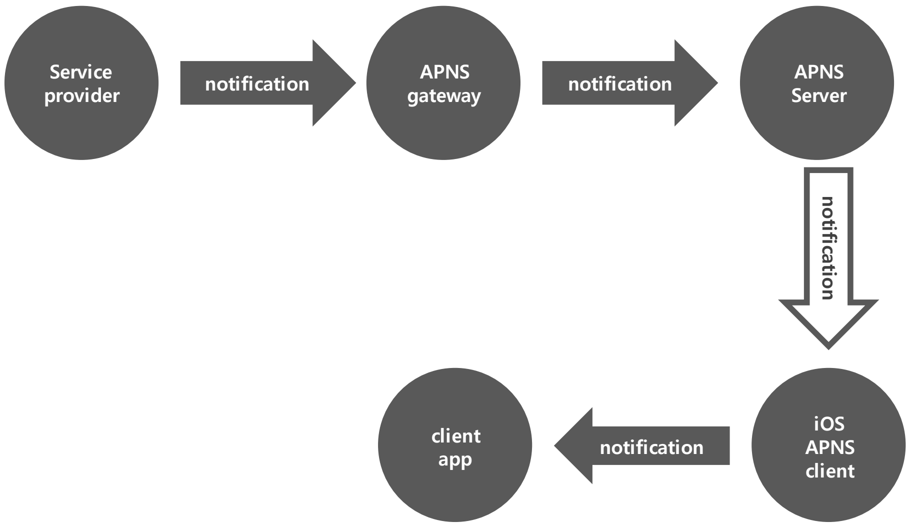

Figure 1. Flow of APNS Message

2.2. C2DM

C2DM is a push notification service framework made by Google for Android mobile applications. It is implemented in Android 2.2 (Froyo) or above. It can send messages to the mobile application which is registered at the Google Play [9]. For using the service, the users should login to their Android phones with their Google accounts. The Flow of the C2DM message is very similar with the flow of the APNS message. Service provider generates a notification request and sends it to the C2DM gateway. The C2DM gateway receives the request and forwards it to the C2DM server. Since the C2DM server has active sessions with the C2DM clients, it can send its request to the Android devices. Then Android can wake up the specified application and the application can be active and process the request. The detail operation mechanism of the C2DM is in Figure 2.

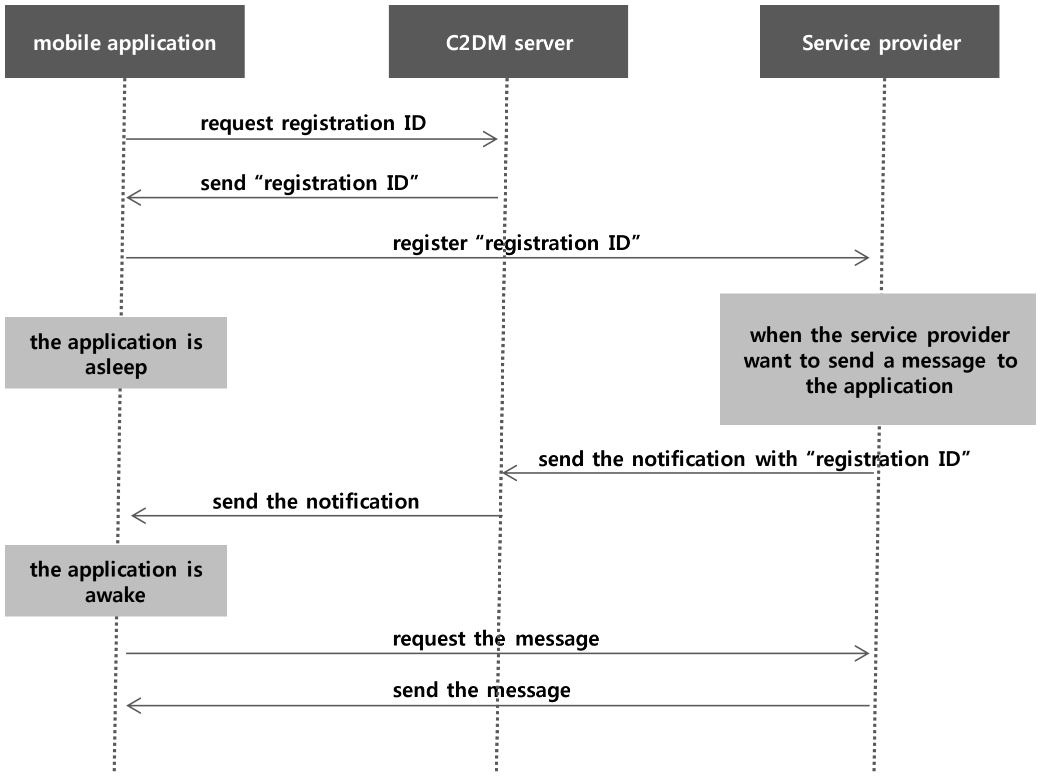

Figure 2. Sequence Diagram of using C2DM

3. Proposed Framework

Since there are many kinds of push notification frameworks including APNS, C2DM, and 3rd party push notification frameworks, service provider should have implemented each notification gateway respectively to send multimedia messages to their mobile applications. Also, since mobile devices can lose its connection with a mobile network when they enter the shadow area, the notification messages can also be dropped. Even though some notification frameworks provide retransmission function when the messages were dropped, their retransmission policies might not meet service provider's requirements.

In this paper, an integrated multimedia push framework is introduced. The proposed framework supports APNS, C2DM, and 3rd party push notification framework as well as always-on-based mobile applications. It also supports the store-and-forward function which stores notification messages when the connection with the client is lost and forwards the message when the connection is recovered. Figure 3 shows the architecture of the proposed multimedia push framework. If a service provider uses this framework, it does not have to implement all push notification gateways since the framework offers interfaces for sending the messages. The framework has the APNS gateway, the C2DM gateway, and 3rd party push notification gateway so that it can transform the messages from the service provider interface to the messages for each gateway. Also, the framework has storage for storing the lost messages and forwards them when the clients recover their connection with the framework.

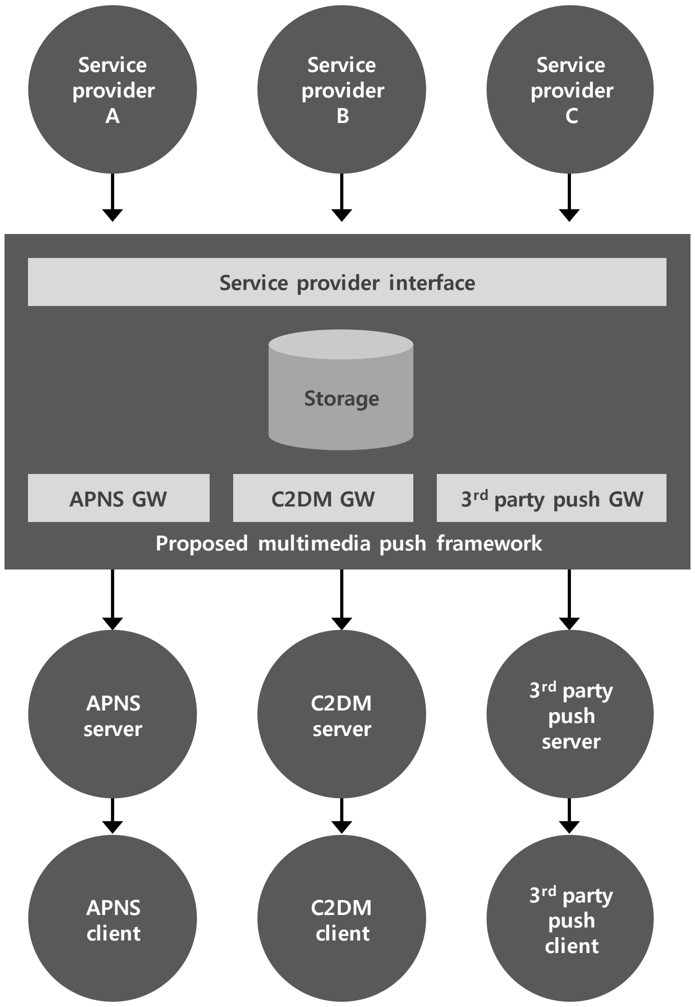

Figure 3. Architecture of the Proposed Multimedia Push Framework

When a service provider uses the service provider interface, the message structure should be like Figure 3. The command field indicates whether the provider requests push/store-and-forward service, push service, always-on/store-and-forward service, and always-on service. The detail description of the command code is in Table 1. The identifier field identifies each message. The expiry field indicates the expire timer of the message. The token length field indicates the length of the device token field. The device token field identifies the devices which have the push clients. The payload length field indicates the length of the payload field. The payload field has the content of the push message.

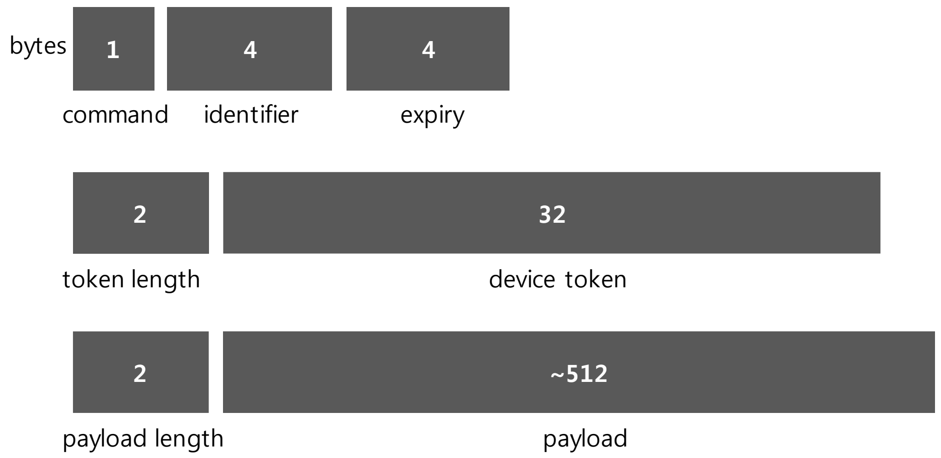

Figure 4. Message Structure of the Service Provider Interface

Table 1. The Description of Codes in the Command Field of the Message

command code	description
0	push/store-and-forward service; the mobile application can be awaken by the push client including APNS client, C2DM client, or 3rd party push client, and the store-and-forward service is requested
1	push service; the mobile application can be awaken by the push client including APNS client, C2DM client, or 3rd party push client
2	always-on/store-and-forward service; the mobile application always has active session with the framework and the store-and-forward service is requested
3	always-on service; the mobile application always has active session with the framework

The detail operation mechanism of the proposed framework is listed below.

(1) When the service provider interface receives the message, the framework identifies the command field.

(2) If the command field indicates push/store-and-forward service,

(2.1) If the message is a initial request,

(2.1.1) The framework stores the message in the storage.

(2.1.2) It sends the message to the push server through the push gateway.

(2.2) If the message is a subsequent request,

(2.2.1) If the framework has active session with the client,

i. It sends the message to the client without the push notification.

ii. If the transmission is failed, the message is stored and sent to push server.

(2.2.2) If the framework does not have active session with the client, the message is stored and sent to push server.

(2.3) The response of the request message should be always successful since the request is stored until the client gets the message successfully.

(2.4) When the push client awakes and requests to forward the message, the message in the storage is forwarded.

(3) If the command field indicates push service,

(3.1) The framework sends the message to the push server through the push gateway.

(3.2) If the push client does not wake up until the expire field, the response of the request message indicates failure.

(4) If the command field indicates always-on/store-and-forward service,

(4.1) The framework sends the message to the client without the push notification.

(4.2) If the transmission is failed,

(4.2.1) The framework stores the message in the storage.

(4.2.2) Wait until the framework has active session with the client, and then it forwards the message.

(4.3) The response of the request message should be always successful.

(5) If the command field indicates always-on service,

(5.1) The framework sends the message to the client without the push notification.

(5.2) If the transmission is failed, the response of the request message indicates failure.

(6) Go to step 1

4. Conclusions

This paper designed new multimedia push framework which integrated existing push framework gateways and provided a store-and-forward function. By using this framework, mobile application developers do not have to implement diverse push framework gateways, which can reduce complexity of interworking function. Also, the developers who use the proposed framework do not have to concern about losing a push notification message in the air since the framework stores the lost message and forward it when the mobile device can connect with a mobile network.

References

[1] H. Y. Ho and L. Y. Syu, “Uses and gratifications of mobile application users”, Proceedings of the International Conference on Electronics and Information Engineering, **(2010).**

[2] Q. H. Mahmoud and P. Popowicz, “Toward a Framework for the Discovery and Acquisition of Mobile Applications”, Proceedings of the Ninth International Conference on Mobile Business and Ninth Global Mobility Roundtable, **(2010).**

[3] I. Podnar, M. Hauswirth and M. Jazayeri, “Mobile push: delivering content to mobile users”, Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops, **(2002).**

[4] Apple Inc., “Apple Push Notification Service”, http://developer.apple.com/library/mac/ #documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushServi ce.html, **(2009).**

[5] J. Hansen, T. M. Gronli and G. Ghinea, “Cloud to Device Push Messaging on Android: A Case Study”, Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, **(2012).**

[6] C. Fernandes, Y. N. Kok and H. K. Boon, “Development of a convenient wireless control of an autonomous vehicle using apple iOS SDK”, Proceedings of the IEEE Region 10 Conference on TENCON, **(2011).**

[7] Y. H. Cheng, W. K. Kuo and S. L. Su, “An Android system design and implementation for Telematics services”, Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems, **(2010).**

[8] A. Setyono, M. J. Alam and R. A. Saqour, “Design and study for the algorithm of multimedia messaging service (MMS) framework in message delivery”, Proceedings of the International Conference on Electrical Engineering and Informatics, **(2009).**

[9] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero and P.G. Bringas, “On the automatic categorisation of android applications”, Proceedings of the Consumer Communications and Networking Conference, **(2012).**