ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 4, Issue 2 (Apr. - Jun. 2016)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2016)

Effective Server Load Optimizer on Push Notification For User Device Multiplicity

[I]Madhukar S, [II]Rajesh N
[I]PG Student, Dept. of IS&E, The National Institute of Engineering, Mysuru, India
[II]Assistant Professor, Dept. of IS&E, The National Institute of Engineering, Mysuru, India

Abstract
With expanding quantities of gadgets per client, information conveyance and disturbance issues emerge from sending push notice messages to all gadgets. The proposed framework exhibits a push notice administration approach that depends on a client driven perspective on cross-stage web advancements to convey warning messages to the right gadget at the ideal time. This builds the odds of a client to convenient react to a notice. Furthermore, the over-burden on the server side is lessened by minimizing copy notice messages sent to an individual client's variety of gadgets.We aim at alleviating the outlined problems through a new push service, termed Effective server load optimizer (ESLO). ESLO represents a new notification model with the main goal of the system to deliver notification events to the right device at the right time.

Keywords
Web service, Client-server systems, push notification, publisher, subscriber

I. Introduction

The quantity of keen versatile devices(e.g., portable workstations, tablets, cell phones) and settled gadgets (e.g., desktops, TV boxes) that the normal client switches between has expanded after some time. Every individual gadget have distinctive reason bringing about today's large number of devices. Thesedevices are generally connected with an individual client. The applications on these gadgets depend on push notices got over various sorts of systems, for example, cell systems remote neighborhood, or remote individual range systems.

Notice messages regularly speak to occasions of interest, which generally are characterized by the applications that show occasions (e.g., new email pointer, new news marker, change of on-line status of informal communication friends).Notification messages are utilized as status pointers, as well as progressively as method for starting client engagement. The current model sends push messages to all gadgets, whether the client effectively utilizes them or not, expanding the quantity of messages being transmitted to gadgets that don't require them.so the number notifications sent increases with number of devices per user,problem in analyzing activity of an individual application.

For the most part, push warnings can be intermittent or in view of remote occasions and ordinarily incorporate a particular message notwithstanding the notice of an application itself. Distinctive administration suppliers have risen that offer sending warning messages through their notice administrations, e.g., Amazon Simple Notification Service or Google's Cloud Messaging for Android (GCM), Responsor Appoxee(http://www.appoxee.com). Some of the administrations today require the applications to at first subscribe, taking after the distributer/endorser model.

As by and large, push warnings are being sent to each application or gadget that is subscribed, the push notice demonstrate that been utilized up to this point results as a part of a conveyance to all gadgets, whether the client effectively utilizes them or not, expanding the quantity of messages being transmitted to gadgets that don't require them.This additionally brings about a potential expanded level of client inconvenience, as alarms are being activated by a huge number of gadgets at somewhat moved time occurrences inside the client's region – the negative mental criticism may bring about clients abstaining from an establishment of these generally viewed as helpful applications.

The combination of device and provider heterogeneity, combined with the growing device multiplicity of individual users will likely increase future complexity and data delivery challenges. In this paper, we describe a user activity-based approach to the delivery of messages that is based on current web technologies and broadly applicable. The project describes a user activity-based approach in delivering the notifications only to the currently active device.

II. Related Work

In [1], The asynchrony, heterogeneity, and inherent loose coupling that characterize applications in a wide-area network promote event interaction as a natural design abstraction for a growing class of software systems. An emerging building block for such systems is an infrastructure called an event notification service [Rosenblum and Wolf 1997].Ubiquitous event notification service accessible from every site on a wide-area network and suitable for supporting highly distributed applications requiring component interactions ranging in granularity from fine to coarse. Conceptually, the service is implemented as a network of servers that provide access points to clients. Clients use the access points to advertise information about events and subsequently to publish multiple notifications of the kind previously advertised. Thus, an advertisement expresses the client's intent to publish a particular kind of notification. They also use the access points to subscribe for notifications of interest. The service uses the access points to then notify clients by delivering any notifications of interest. Clearly, an event notification service complements other general-purpose middleware services, such as point-to-point and multicast communication mechanisms, by offering a many-to-many communication and integration facility.

GCM [2] Google Cloud Messaging (GCM) is a popular service as a client/server communication solution for Android. Today more than half of the smartphones run Android OS. With the recent release of Android Wear, Android extended its domain to wearable devices. Both of these platforms use GCM for central notifications. For example, Google Glass forwards most tasks to a cloud-based solution called Mirror API, and Mirror uses GCM for client/server communications.GCM is a service which allows developers to send push messages to Android devices from the server. GCM handles the queuing of the messages as well as delivering those messages to the target applications on the devices. GCM is a free service by Google, and it has no quotas. It is the default push messaging solution for the Android platform.SOAP [3] headers tend to contain information about the data in the body, such as security and routing information. WSDL [4] serves two main purposes in real web services. It describes which interface (which RPC-style functions) the service exposes. It also describes the representation formats.The vision of UDDI [5] was one of multiple registries: a fully-replicated Internet-scale registry for businesses, and a private registry behind the firewall of every company that wanted to host one.

III. System Architecture

Within our system, we denote publishers as providers of event notifications (messages) and subscribers as individual users, not devices or applications. In the ESLO environment, the Cloud Push Notification (CPN) provider initially manages subscribers as users while the individual user devices fall secondary under the user subscriptions. The provider generates the event and subsequently sends a message to the CPN provider requesting to push the message forward to subscribers. Based on the activation of the devices the notification is sent to those devices thus reaching the right device at the right time. Further the activation of device is identified by some process which sends the IMEI of the device whose device.

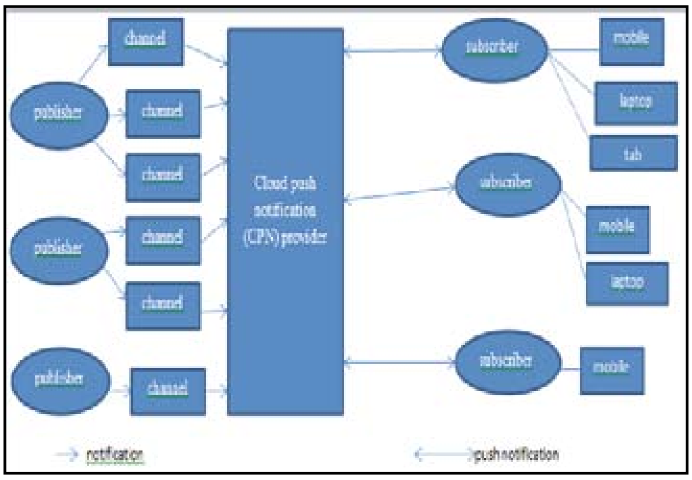


Fig.1: system architecture

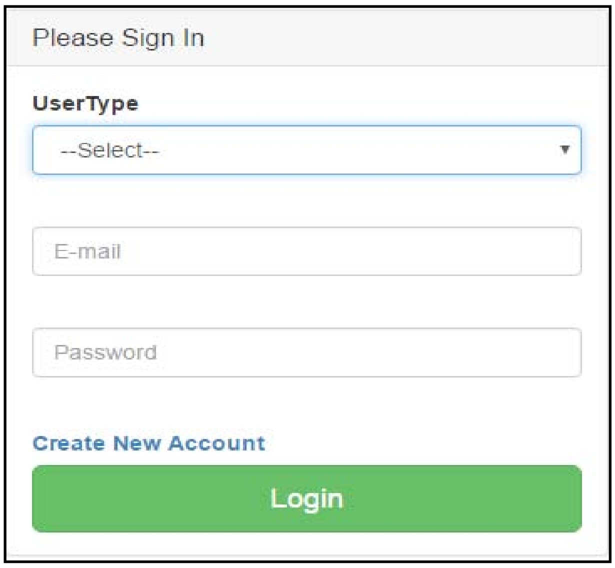
IV. Sample scenario

When the subscriber subscribes to a particular publisher after the registration process the user is supposed to give the ratings for the publisher according to their priority. Based on this priority the notifications are sent to the subscribers.Subscribers are the audiences interested in receiving notifications to specific events distributed over specific channels by a provider. The subscribers in the ESLO model are actual users that installed an application, which in turn is registered with the CPN provider to receive notification messages for an event. Subscribers have one or multiple devices; the devices are subordinates of a subscriber. This allows the CPN providers to collect finer granularity of activity information from the individual users rather than just in form of applications or devices. A device in in the ESLO model acts as a receiver for a subscriber and could be any physical or virtual device that has the desire to receive notification messages of an interest to the subscriber. Additionally, devices are responsible for the determination of the activity of a subscriber, i.e., whether they are in use, and forwarding the information to the CPN provider. Examples for the determination of current user activity are given by a user login or the device networking status.

Publisher and subscriber

In this module, the code has been written for the publisher. Initially either subscriber or the publishers have to get registered. In the registration process they are supposed to give the information about the user name, password, user type, email-id, mobile number, address. In the user type area there will be a drop down list which shows only two options to select according to the user. The options are publisher and the subscriber. After selecting the user type and giving further details as said above account will be created. The following diagram gives the snapshot of this registration process.

After the creation of the account the subscriber or the publisher should login by giving the user type, email id and password. This login will take into the home page. The following diagram shows the snapshot of the login page both for the publisher and subscriber.



Please Sign In

UserType

--Select--

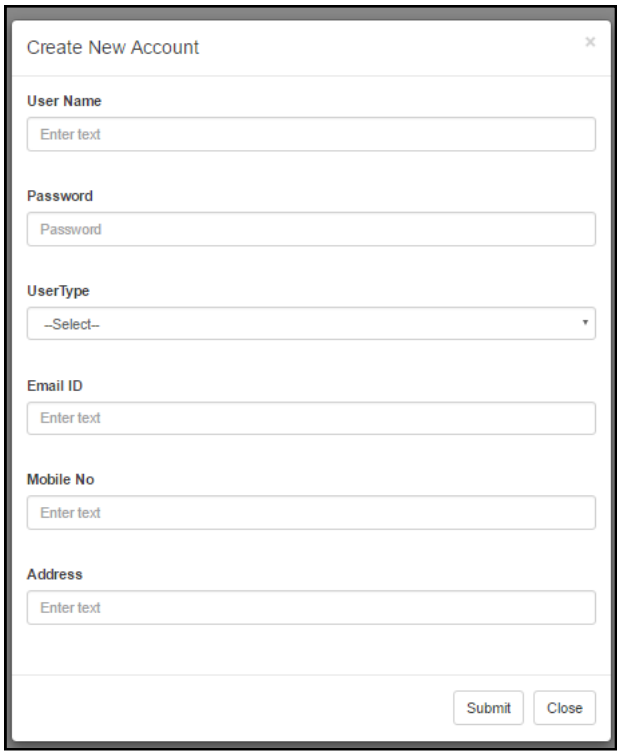
E-mail

Password

Create New Account

Login

Fig.3: Publisher and Subscriber login



Create New Account ×

User Name

Enter text

Password

Password

UserType

--Select--

Email ID

Enter text

Mobile No

Enter text

Address

Enter text

Submit Close

Fig.2: Registration

The publishers here can see their published contents. This area gives the total number of published contents by him. There will be few categories under which the publishers can publish the contents. The categories are like sports, business, technology, education. Similarly the subscribers can get registered to heir interested category. The following diagram shows the snapshot of selecting the category.

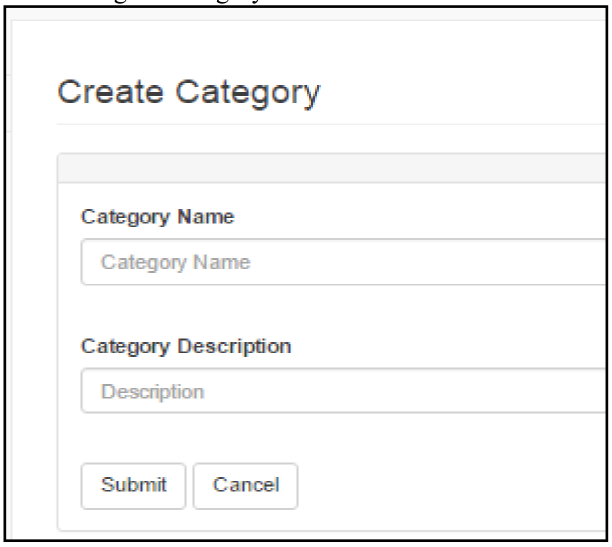


Fig.4: Creating the category

For the publisher to publish the content, there will be an option publish content. The view publish content options provides option to view the published contents and the publisher has options to change the contents and again re publish. From the subscriber point of view ones the subscriber subscribes to the publisher under some category. After the subscription the subscribers will give the ratings to the publishers out of 10. This rating will be stored in database with publisher id, subscriber id and rating. This rating will be further used in the next modules to determine the importance of the notification required. 1 being the highest and 10 being the lowest.The subscribers as they subscribe to any of the categories they will find the notification by selecting the particular category they need.

V. GCM Module

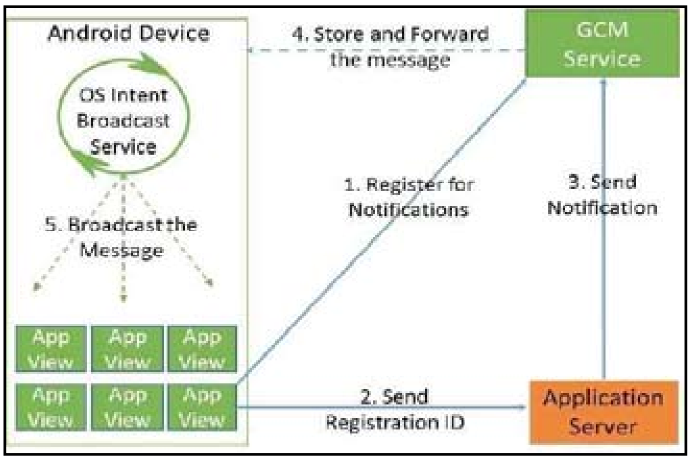


Fig.5: Functional diagram of GCM

This project makes uses of GCM. As GCM is provided by the Google we are supposed to have a Gmail account. In order to make use of this GCM we need to go to Console.developers.google.com website. When a new project is created a unique id for the project will be generated. In the project we are supposed to give the type of service that we want. So here we need to enable the Google cloud message service. In this overall process the credentials should be fetched from the Google. The API key will be used by the person who is sending the notification. So we finally have the project id and the server key. To receive the notification we need an android application. Using android programming we should use the project id and request to register the device. In this android application there will be an option to register. Ones we click register inside there will be a code which uses this project number and sends to the servers that this device to be registered. The server in turn will be proving a unique Id to that device. This unique id given by the Google to the device will also be maintained in Google. The android application will send this received device id to the publisher who wants to send the notification through email. The publisher or the admin will be having the records of all such device id received. When the message is sent the google will be responsible to send the message. Google now delivers the message to each device as the google will maintain the queue. The existing GCM send the message to all the devices whether the device is currently being used or not. This leads to the wastage of bytes, more network traffic and server side load.
Interaction with the GCM module is the interaction of the GCM module with the developed android application that gives the status of the device. As said above google will be maintaining the device id to which the notification has to be sent. The interaction with the gcmand the android will make it sure that the massage or the notification that has to be sent to the active device is sent. The active of the device is done by the android application that has been developed and sending thee notification to the active device is done by the gcm module.

VI. Implementation

The purpose of system implementation is making the new system available to a prepared set of users and positioning on-going support and maintenance of the system within the performing organization.

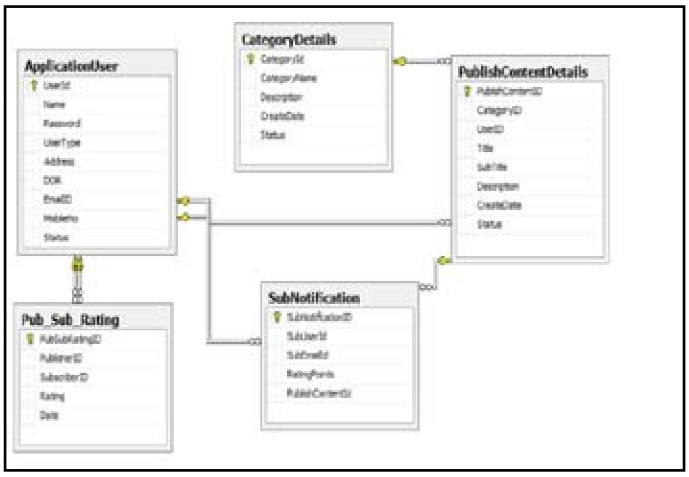


Fig.6 : Database implementation for publisher and subscriber

The following are important tables in the publisher and subscriber modules.

- ApplicationUser (primary key will ne UserId)
- Category (primary key will be CatId)
- PublishContentDetails (primary key is PublishContentId, foreign keys are CatId and userid)

- PublisherSubscriberRating (primary key id PubRating, foreign keys are PubId and SubId)
- SubscriberNotification (primary key is SubNotId, foreign key is SubId)

The Fig.6: shows the database diagram that has been generated from the SQL server according to the tables created. This diagram depicts the relation between the tables and also specifies the primary keys of each table. The primary keys of the tables are followed by the key symbol

VII. Conclusion

With increasing numbers of devices per user, data delivery and annoyance problems arise from sending push notification messages to all devices. The proposed system presents a push notification service approach that is based on a user-centric viewpoint on cross-platform web technologies to deliver notification messages to the right device at the right time. This increases the chances of a user to timely respond to a notification. Additionally, the overload on the server side is reduced by minimizing duplicate notification messages sent to an individual user's multiplicity of devices.

References

[1] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," ACMTransactions on Computer Systems (TOCS), vol. 19, no. 3, pp. 332–383, Aug. 2001.

[2] S. Vinoski, "Web services notifications," Internet Computing, IEEE, vol. 8, no. 2, pp. 86–90, Mar. 2004.

[3] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon, Eds., "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," W3C, Apr. 2007.

[4] F. Curbera, M. Duftler, R. Khalaf, and W. Nagy, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," IEEEInternet Computing, vol. 6, no. 2, pp. 86-93, Mar. 2002.

[5] L. Richardson and S. Ruby, RESTful web services. O'Reilly Media, Sebastopol, CA, USA, May 2007