# GCM Service Driven Communication With An Android Application In Cloud Computing

[1] Naresh Kumar N (M.Tech), [2] Prof. Mohan K

[1] SIT, Mangalore, [2] Dept. of Computer Science SIT, Mangalore, India

**Abstract:** *In this paper, we incorporate a new form of cloud-based push service called Google Cloud Messaging that helps developers to send messages from servers to their Android Applications. The GCM service handles all aspects of queuing of messages and delivery to the target Android application running on the target device. The Google Cloud Messaging saves battery for android devices. Also, this paper introduces the multitenancy concept and use of location based services for android.*

***Keywords: Cloud Computing; Multi-tenancy; Google Cloud Messaging; Location Based Services***



## 1. Introduction

Cloud computing is the use of computing resources such as hardware and software that are delivered as a service over a network typically in the form of internet. Cloud computing entrusts remote services with a user's data, software and computation.

Smartphone's have evolved rapidly during the last three years. Now a day, the advances in processor, memory, flash storage, and mobile communication, and software, smart phones have enabled sophisticated applications for mobile users. The current leading brands for smart phones in the market are Google Android, Apple iPhone, Microsoft Window Mobile, Black- Berry RIM and all support applications such as multimedia playback, Internet browsing, email, voice mail, social networks and location-based Services. Still, the limited hardware resources and the constrained battery capacities have strongly impacted their user experiences.

Today, many Smartphone users take advantage of low-cost or free cloud-based services. The combination of smart phone and cloud-based service has worked quite successfully and has become very popular, as it essentially offloads computational workload and data storage from the user's smart phone. That way, an application could consume less power by having most of the application workload performed by a cloud-based service.

Depending on how a cloud infrastructure is exposed as a service to the user, there are so called service models which are commonly used to categorize a cloud-based service.

Software as a Service (SaaS) is a on-demand software in which the software and associated data are centrally hosted on the cloud. SaaS is typically accessed by users via browser and thin client.

Platform as a Service (PaaS) is a paradigm for delivering operating systems and associated services over the internet without downloads or installation.

Infrastructure as a Service (IaaS) is a type of service which delivers a computational infrastructure - typically in the form of a virtualization environment or a virtual machine.

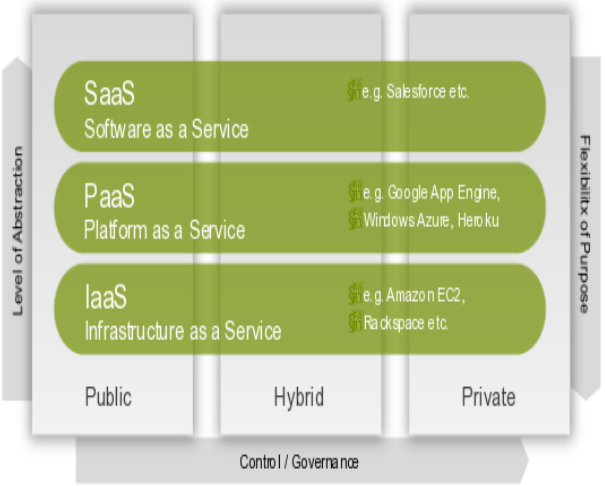


Fig. 1 Cloud computing service models

## 2. Multi-tenancy

The idea of multi-tenancy, or many tenants sharing resources, is fundamental to cloud computing. Service providers are able to build network infrastructures and data architectures that are computationally very efficient, highly scalable, and easily incremented to serve the many customers that share them. Multi-tenancy spans the layers at which services are provided. The vast majority of SaaS solutions are based on the multi-tenancy architecture.

Multi-tenancy is a software architecture where a single instance of the software runs on servers, serving multiple client organizations (tenants).

In IaaS, tenants share infrastructure resources like hardware, compute servers, and data storage devices. With SaaS, tenants are sourcing the same application, which means that data of multiple tenants is likely stored in the same database and may even share the same tables. When it comes to security, the risks with multi-tenancy must be addressed at all layers.

The concept of multi-tenancy comes in different flavours and depending on which flavour is implemented, the utilization rate of the hardware can be maximized.

The following variants of multi-tenancy can be distinguished:

1) Shared application, separate database.

2) Shared application, shared database, separate table.

3) Shared application, shared table (pure multi-tenancy).

## 3. Google Cloud Messaging

Google Cloud Messaging for Android (GCM) is a free service that helps developers to send data from servers to their Android applications on Android devices. This could be a lightweight message telling the Android application that there is new data to be fetched from the server or it could be a message containing up to 4kb of payload data. The GCM service handles all aspects of queuing of messages and delivery to the target Android application running on the target device.

Before GCM came into existence, Cloud to Device Messaging (C2DM) was used and developers can send data from servers to their applications on android devices.

The limitations of Cloud to Device Messaging (C2DM) are:

1) Each notification message size is limit to 1024 bytes.

2) Google limits the number of messages a sender sends in aggregate, and the number of messages a sender sends to a specific device.C2DM requires users to set up their Google account on their mobile devices.

3) C2DM makes no guarantees about the delivery or the order of messages.

For example, let's say if a user has a message that needed to be sent to 1,000,000 users, and the server could handle sending out about 500 messages per second. If user send each message with only a single recipient, it would take 1,000,000/500 = 2,000 seconds, or around half an hour. However, attaching 1,000 recipients to each message, the total time required to send a message out to 1,000,000 recipients becomes (1,000,000/1,000)/500 = 2 seconds. This is not only useful, but important for timely data, such as natural disaster alerts or sports scores, where a 30 minute interval might render the information useless.

The following are the primary characteristics of Google Cloud Messaging:

1) It allows third-party application servers to send messages to their Android applications.

2) An Android application on an Android device doesn't need to be running to receive messages. The system will wake up the Android application via Intent broadcast when the message arrives, as long as the application is set up with the proper broadcast receiver and permissions.

3) It does not provide any built-in user interface or other handling for message data. GCM simply passes raw message data received straight to the Android application, which has full control of how to handle it. For example, the application might post a notification, display a custom user interface, or silently sync data.

4) It requires devices running Android 2.2 or higher that also have the Google Play Store application installed, or an emulator running Android 2.2 with Google APIs. However, you are not limited to deploying your Android applications through Google Play Store.

*A. Architectural overview of GCM*

The architecture of GCM can be divided into two categories:

1) *Components*: The physical entities that role in GCM

a) Mobile Device: The device that is running an Android application that uses GCM. This must be a 2.2 Android device that has Google Play Store installed, and it must have at least one logged in Google account.

b) Third-party Application Server: An application server that developers set up as part of implementing GCM in their applications. The Third-party Application server sends data to an Android application on the device via the GCM server.

2) *Credentials*: The IDs and tokens that are used in different stages of GCM to ensure that all parties have been authenticated, and that message is going to the correct place.

a) Sender ID: The sender ID is used in the registration process to identify an Android application that is permitted to send messages to the device.

b) Application ID: The Android application that is registering to receive messages. The Android application is identified by the package name from the manifest. This ensures that the messages are targeted to the correct android application.

c) Registration ID: An ID issued by the GCM servers to the Android application that allows it to receive messages. Once the Android application has the registration ID, it sends it to the third-party application server, which uses it to identify each device that has registered to receive messages for a given Android application. The registration ID is tied to a particular Android application running on a particular device.

d) Google User Account: For GCM to work, the mobile device must include at least one Google account if the device is running a version lower than Android 4.0.4.

e) Sender Auth Token: An API key that is saved on the 3rd-party application server that gives the application server authorized access to Google services. The API key is included in the header of POST requests that send messages.

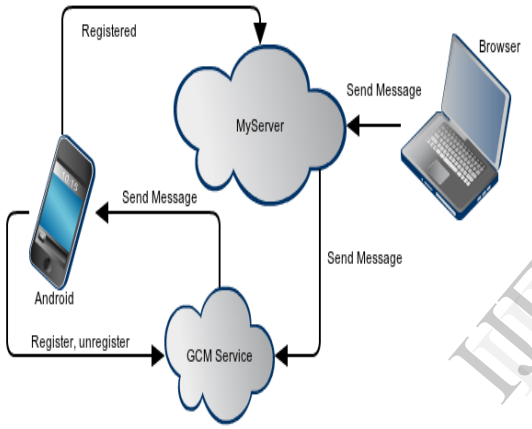


Fig. 2 GCM Architecture

B. *Working of GCM*

The application server sends a message to GCM servers. Google enqueues and stores the message in case the device is offline. When the device is online, Google sends the message to the device. On the device, the system broadcasts the message to the specified Android application via Intent broadcast with proper permissions, so that only the targeted Android application gets the message. This wakes up the Android application and the application does not need to be running before to receive the message. Finally, the Android application processes the message.

C. *Advantages of using GCM*

1) There are no sign-up forms to use. Developers simply obtain the API key from Google.

2) Google claims the new service has no quotas and increases battery efficiency.

3) GCM also supports multicast messages, multiple senders, and time-to-live messages.

4) Since the application no longer needs to poll the server for the updates, it saves both battery and data usage.

5) One of the most useful features in Google Cloud Messaging is support for up to 1,000 recipients for a single message.

D. Getting API key and Project ID to use GCM

The GCM have to be implemented in both server and client part. We need to get API key for web application to serve android application via the project ID.

The following are the steps to get API key and project ID:

1) Go to Google API console and login with Google account.
2) After logging to console, project ID will be generated.
3) Turn on the Google Cloud Messaging.
4) Go to API access and create server key to use it in web application.

Below figures 3.1 and 3.2 pictorially represents the above procedure.

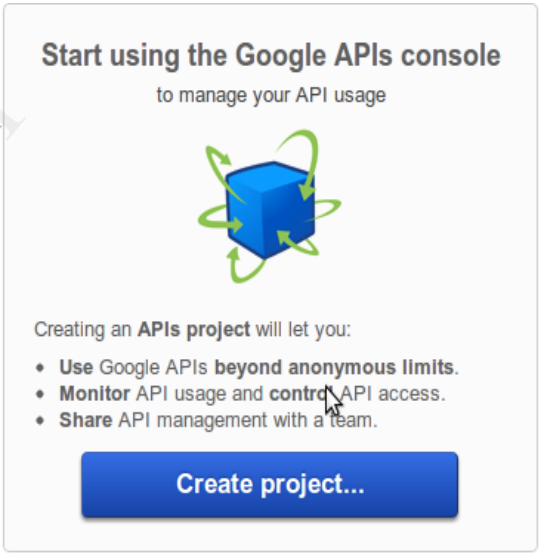


Figure 3.1 creating project ID


Figure 3.2 creating server API key

### 4. Location Based Services

The appearance of different technologies such as wireless networks, Internet, Geographical Information Systems (GIS), Global Positioning Systems (GPS) have introduced a new type of information technology called Location Based Services.

Location based services (LBS) is defined as the ability to locate a mobile user geographically and deliver services to the user based on his location.

A positioning component is usually needed in LBS application to determine the location of user's mobile device. Most of the current LBS services do not require users to input location manually, like giving zip code or street name. Instead, user location can be obtained by using some positioning technologies such as satellite positioning, cellular network positioning, WLAN stations.


Fig. 2 Components of location based services

Service providers maintain service servers which offer different kinds of location based services to users and are responsible for processing service requests and sending back request results. Servers calculate positions, search for a route, or search specific information based on user position. Service providers usually do not store and maintain all the information requested by users. Instead, content providers are responsible for collecting and storing geographic data, location based information and other related data.

Android devices may have one or more of these services available to them. Maps and location based services use latitude and longitude to pinpoint geographic locations. Android provides a Geocoder that supports forward and reverse geocoding.

Geocoding lets user translate between street addresses and longitude/latitude map coordinates. This can give user a recognizable context for the locations and coordinates used in location based services and map based activities.

The Geocoder class provides access to two geocoding functions:

1) Forward Geocoding: Finds the latitude and longitude of an address.
2) Reverse Geocoding: Finds the street address for given latitude and longitude.

## 5. Implementation

The Web application can be developed with java as programming language using eclipse IDE. The developed web application can be deployed using Google App Engine and once the web application is deployed, it will be running in the Google infrastructure. It provides access to databases via Java Data Objects and Java Persistence API.

When we use Google App Engine, it's not possible to access to a traditional relational database such as Oracle, MySql or SQLite. Instead of accessing traditional relational database, we can access to the Google Data Store that is more of a hierarchical object oriented storage approach. The Google Data Store is based on Google technology such as Google Bigtable.

The Bigtable and the Google Data Store achieves efficient application scalability within the Google cloud given the dynamic and distributed nature of the Google production infrastructure.

To develop Android applications that use GCM, we must have an application server.

The two primary steps involved in writing a client Android application are:

1) Creating a manifest that contains the permissions, the Android application needs to use GCM.

2) To use GCM in an application implementation, it must include:

a) Code to start and stop the registration service.

b) Receivers for the com.google.android.c2dm.intent. RECEIVE and com.google.android.c2dm.intent. REGISTRATION intents.

## 6. Conclusion

Push notifications services have provided great convenience and flexibility for applications to receive light messages or notifications from application servers. In this paper, we have incorporated architecture of Google Cloud Messaging service for the android platform and how third party application server sends messages to registered android through Google Cloud Messaging. Also, we described about Google App Engine to deploy web application and the concept of pure multi-tenancy.

## References

[1] V.L.DIVYA, Mobile Application with cloud computing. : IJSRP, Volume 2, Issue 4, April 2012.

[2] C.P.Bezemer and A.Zaidman Multi-Tenant SaaS Applications:Maintenance Dream or Nightmare? Antwerp, Belgium : IWPSE-EVOL'10, 2010.

[3] Application Fundamentals Android Developers. http://developer.android.com/guide/topics/fundamentals.html

[4] Android Cloud to Device Messaging Framework-Google Developers. http://developers.google.com/android/c2dm/index.html

[5] Google Cloud Messaging for Android Developers http://developer.android.com/google/gcm/index.html

[6] Cloud computing –Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Cloud_computing