# GOOGLE CLOUD MESSAGING (GCM): A LIGHT WEIGHT COMMUNICATION MECHANISM BETWEEN CLIENT AND SERVER ON ANDROID PLATFORM

Nilay Ganatra[1] and Rachana Patel[2]

[1]Department of Computer Applications, CHARUSAT, Changa,
nilayganatra.mca@charusat.ac.in
[2] Department of Computer Applications, CHARUSAT, Changa,
rachanapatel.mca@charusat.ac.in

***ABSTRACT***

*IN THIS PAPER, WE EXAMINE THE PUSH MESSAGING SERVICE GOOGLE CLOUD MESSAGING (GCM) FOR ANDROID PLATFORM. ANDROID PLATFORM TRADITIONALLY KEEP DATA SYNCHRONIZATION BETWEEN ANDROID DEVICE AND SERVER-SIDE USING METHOD OF PULLING. EACH ANDROID DEVICE HAS TO POLL SERVER FOR UPDATED DATA, WHICH LEADS TO UNNECESSARY NETWORK TRAFFIC AND WASTAGE OF MOBILE PHONE BATTERY. IN ORDER TO OVERCOME THIS WEAKNESS, DATA PUSHING SERVICE, GCM WAS INTRODUCED. PUSH, DESCRIBES A STYLE OF INTERNET-BASED COMMUNICATION WHERE THE REQUEST FOR A GIVEN TRANSACTION IS INITIATED BY THE PUBLISHER OR CENTRAL SERVER. PUSH MESSAGING IS A MULTI-CHANNEL MOBILE CLOUD COMMUNICATIONS PLATFORM THAT UNIFIES PUSH NOTIFICATIONS, SMS AND INSTANT MESSAGING. GCM SERVICE ALLOWS SENDING DATA FROM THE APP ENGINE OR OTHER BACKHANDS TO ANDROID POWERED DEVICE. GCM IS LIGHTWEIGHT PUSH NOTIFICATION BASED SERVICE NOTIFYING ANDROID APPLICATION ABOUT NEW DATA TO BE FETCHED FROM THE SERVER OR MESSAGING CONTAINING 4KB OF PAYLOAD DATA. GCM MANAGES ALL ASPECTS MESSAGES QUEUING AND DELIVERY OF MESSAGE TO TARGET ANDROID APPLICATION RUNNING ON TARGET DEVICE.*

***KEYWORDS***

*GCM, Push Notification, Multi-Channel, Mobile Cloud*

## 1. INTRODUCTION

Advancement in mobile phone technology is the high-speed network available to the public. With EDGE, 3G and 4G, users are almost always connected. Smart phone device combined with high speed network provide many new and electrifying innovation possibilities [1]. One of them is cloud computing. Combination of cloud computing, mobile computing and wireless networks bring rich computational resources to mobile users [3].The term cloud computing refers to the applications delivered over the Internet specifically and the hardware and systems software that is providing these services [4].

With the current development of mobile and inescapable computing era, smartphones became pervasive. A major portion of these applications depends on the cloud and Google Cloud Messaging is a very useful and popular service for client/server communication [5]. Google cloud messaging is an open service than enables developers to send messages between servers and client applications[2]. It provides facility of downstream messaging i.e. from server to client application, as well upstream messaging from client application to server. Today almost more than half of the smartphone users using Android OS based devices. With release of Android Wear, Android stretched its province to wearable devices like google glass, smart watch etc. Both, smart and wearable smart devices use GCM for notifications. Wearable devices includes computer and advanced electronic technologies. For example, Google glass forward most of its task to cloud-central elucidation called Mirror API, and Mirror API [6] uses GCM for client server communication.

Google cloud messaging is service for sending and receiving push notifications to and from and android applications. Before the push messaging support was added to the Android platform it was common to use a polling mechanism. This worked by making application itself would periodically poll your servers to check for new messages. You would need to implement everything from queuing messages to writing the polling code. Alerts are no good if they're delayed due to a low polling period but the more frequently you poll, the more the battery is going to die.

GCM is default push messaging service for the Android platform. GCM handles queuing of messaging and delivering those messages to the target application. GCM service is particularly useful whenever new data is available on server instead of making request to server on regular time interval. For example, email android application, it is not an effectual to have the application ping to server to check for new mails. Server should notify mobile device application about new mail.

## 2. RELATED WORKS

A significant amount of work is found in this type of research. Here we have reviewed and used following references for this article.

Chetan D Wadate, Prashant T Suvare, Aniket S More and Rina Bora have published WI-FI based push notification in college campus noticeboard, using which they can update information regarding various campus activities like: meeting times, exam dates, class cancellations and other[7].

Jarle Hansen, Tor-Morten Grønli and Gheorghita Ghinea have compared various push notification technologies in the aspect of Stability, Response time and Energy consumption for Android platform, namely C2DM, XMPP, Xtify and Urban Arship [8].

Yavuz Selim Yilmaz, Bahadir Ismail Aydin and Murat Demirbas have evaluated GCM, and concluded GCM is not suitable for "must-deliver-to-all" app scenarios. They have identified GCM is good for the applications where random multicasting is sufficient [9].

Harminder Singh, Dr Sudesh Kumar, Harpreet Kaur have combined GCM service with location service and develop new service. They have mentioned advantages and limitations of this newly created service [10].

Naresh Kumar N and Prof. Mohan K have given overview of GCM architecture, implementation and advantages of GCM over C2DM [11].

## 3. ARCHITECTURAL OVERVIEW OF GCM:

Google Cloud Messaging for Android (GCM) is a service that allows you to send data from your server to the users' Android-powered device. GCM is lightweight message notifying your application about new data to be fetched from the server like new version of apps or something like that [12]. The GCM service serves all aspects of storing, queuing and delivery of messages to the target Android application running on particular device. It is a completely free service whatever you're messaging needs [13].

### 3.1. GCM Architecture

GCM architecture includes a Google connection server, an app server in your application that communicate with the connection server via HTTP or XMPP protocol and client application.

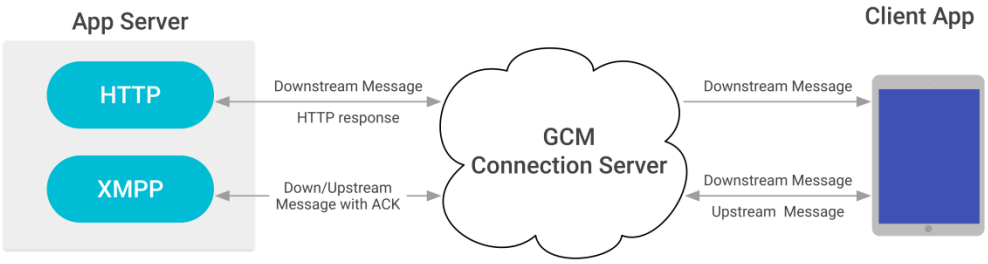


Figure 1. Architectural overview of GCM service

### 3.2. Working of GCM

1. Android device sends Sender id, application id to the GCM server for registration.
2. On successful registration, the GCM server gives a registration ID to android device.
3. Android device sends this registration ID to the local server.
4. The local server stores the registration ID in the database for later use.

    a) Whenever a notification is provided through the website, the server sends the message to the GCM server along with the registered ID.

    b) GCM server sends that message to particular device using that registration ID.

The Google's Cloud messaging platform as shown in the Figure 2 will act as the primary platform. The user will first be registering onto the GCM platform and will receive a token ID which will be stored on the server which identifies the user's phone based on that ID. The server is where the organization will be having a PHP based web client which is going to send the notifications to individuals with respect to their registration ID or in bulk however intended [13].

Figure 2. Shows the overall process of registration and sending a notification occurs. The user registers on the GCM in step 1 the GCM provides the registration ID in step 2. In step 3 & 4 the mobile stores the ID on the server and the step a. & b. are the phases where the server is sending the notification to the phone via the GCM architecture.

GCM is used as it is a client server architecture which is the most commonly and widely used architecture. GCM provides feature of folding messages into small parts; collapsible messages are a better choice for a mobile device performance point of view, because they put less burden on the device battery.

Scalability implies the ability for the architecture to grow and accommodate increasing numbers of users, applications, and systems. Scalability and Extensibility refer to an application's ability to inherently support changes to the hardware and software on which it depends. Using GCM we can send messages to a single or a group of android devices simultaneously in a single transmission.

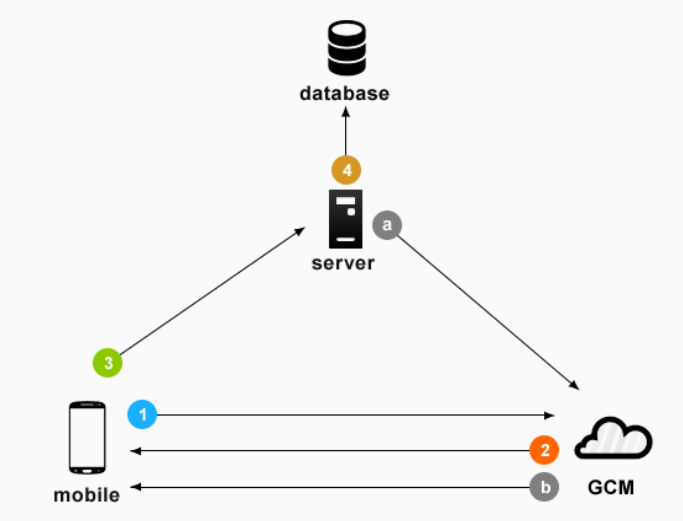


Figure 2. Working of Google Cloud Messaging

### 3.3. Android Device:

The Android application is the main interface via which the user is going to use to receive the push notifications. The important requirements for developing the application are listed below:

- **Client Application [14]:** This application is developed using android development studio in conjunction with SDK tools.
- **SDK tools [14]:** The Android SDK tools compile the code along with any data and resource files. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.
- **.apk [14]:** All the code in a single .apk file is considered to be one application and is the file that Android-powered devices use to install the application.
- **Android OS [14]:** Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smart phones and tablet computers. It enables replace and reuse of components.

### 3.4. Server Application and Database:

The server application and database are an essential part of the push notifications and the main activity of the server and database is to provide the admin the facility to send the message to the client and receive an acknowledgement from the client that the message has been received. The components which are required to setup a server are listed below:

- **PHP [13]-[15]:** PHP Hypertext Pre-processor", is an open-source, reflective programming language used mainly for developing server-side applications and dynamic web content. XAMPP is a free open source cross platform web server package consisting of Apache Http server, MySQL database and interpreters for scripts written in PHP and Perl programming language.

- **MySQL [16]:** MySQL is the most popular database system used with PHP. MySQL is open source RDBMS which manages the data contained within the databases. We have used the version MySQL 5.0.

### 3.5. Components and Credentials of GCM:

**Components:**

| Component | Description |
|---|---|
| GCM Connection Server | Google server sending messages between the app server and the client app |
| Client App | Client app with GCM-enabled communicate with your app server |
| App Server | An app server that you write as part of implementing GCM. This server sends data to a client app using GCM connection server. |

**Credentials:**

| Credential | Description |
|---|---|
| Sender ID | A unique numeric value generated when you configure your API project. This id is used in the registration process to authenticate the app server to send messages to the client app. |
| API Key | An API key saved on the app server that provides authorized access to Google service. You obtain the API key while configuring your API project. |
| Application ID | A client app that is registering to receive messages. For different platform you can obtain it in following way:<br>• Android: uses the package name<br>• iOS: use the app's bundle identifier<br>• Chrome: use the Chrome extension name |
| Registration Token | An ID given by GCM connection server to the client app that permits it to receive messages. |

### 3.6. Working Lifecycle of GCM [17]:

- **Register to use GCM:** An object of client app register to receive messages.
- **Downstream message communication:**
    - **Send a messages:** The app server sends messages to the client app:
        1. The app server sends a message to GCM connection server.
        2. If the device is offline, the GCM connection server queuing up and store the messages if device is not connected.

3. When device is connected, the GCM connection server sends the messages to the device.
4. On the device, the client app receives the message according to the platform-specific implementation.

- **Receive a message:** Client app receive a message from a GCM connection server according to platform-specific implementation of client app.

- **Upstream Message communication:**
  - **Send a message:** A client app sends messages to app server.
    1. On the device, platform specific implemented app sends messages to the XMPP connection server.
    2. The connection server queuing up and stores the messages if the server is disconnected.
    3. When app server is connected, the XMPP connection server will send the messages to the app server.
  - **Receive a message:** An app server receives a message from the connection server and does following:
    1. Parse the header information to verify client app sender information.
    2. Send "ack" to the connection server to acknowledge receiving the messages.
    3. Parse the messages payload, as defined by the client app.

## 4. IMPLEMENTATION & METHODS:

GCM basically provides three messaging techniques: Device Group Messaging, Downstream Messaging and Upstream Messaging.

### 4.1 Device Group Messaging [17]:

With this technique, application server is able to send a single message to multiple instances of an app running on different devices belonging to a group.

**Managing device groups:**

1. Get registration tokens for each device in the group.
2. Create the notification_key, which finds the device group by mapping a particular group.

The application server will send message to the notification_key, and GCM send the message to all the registered tokens of group.

### Creating device group:

The notofication_key_name is a uniue identifier given to a group.

```
https://android.googleapis.com/gcm/notification
Content-Type:application/json
Authorization:key=API_KEY
project_id:SENDER_ID

{
   "operation": "create",
   "notification_key_name": "appUser-Chris",
   "registration_ids": ["4", "8", "15", "16", "23", "42"]
}
```

Figure 3. Device Group

**Sending downstream messages to device group:**

HTTP POST Request

```
https://gcm-http.googleapis.com/gcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{
  "to": "aUniqueKey",
  "data": {
    "hello": "This is a GCM Device Group Message!",
   }
}
```

Figure 4. HTTP POST Request Format

**Suppose** notification_key has 2 registration token associated it, and message was successfully sent to both of them:

HTTP Response

```
{
  "success": 2,
  "failure": 0
}
```

Figure 5. HTTP RESPONSE Format

**Sending upstream messages to device groups:**
Client application can send messages upstream to device group by targeting messages to the appropriate notification key in the '*to'* field.

The following call to GCM sends upstream messages to a notification key.

```
GoogleCloudMessaging gcm = GoogleCloudMessaging.get(context);
String to = aUniqueKey; // the notification key
AtomicInteger msgId = new AtomicInteger();
String id = Integer.toString(msgId.incrementAndGet());
Bundle data = new Bundle();
data.putString("hello", "world");

gcm.send(to, id, data);
```

Figure 6. GCM Upstream Message

## 4.2 Downstream Messaging [17]:

In this, messages sent from the application server to directly client application on a device.
**Downstream messages from the server:**
The application server sets '*to*' with the receiving client applications' registration token.

HTTP POST Request

```
https://gcm-http.googleapis.com/gcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{ "data": {
    "score": "5x1",
    "time": "15:10"
  },
  "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1..."
}
```

Figure 7. HTTP POST Request

**Messages on an Android client application:**

To receive simple downstream messages, use a service that extendsGcmListenerService to handle messages received by GcmReceiver. GcmReceiver extends WakefulBroadcastReceiver, guaranteeing that the CPU is awake so that your listener service can complete its task.

By overriding the method GcmListenerService.onMessageReceived, you can perform actions based on the received message:

```
@Override
public void onMessageReceived(String from, Bundle data) {
    String message = data.getString("message");
    Log.d(TAG, "From: " + from);
    Log.d(TAG, "Message: " + message);

    if (from.startsWith("/topics/")) {
        // message received from some topic.
    } else {
        // normal downstream message.
    }

    // ...
}
```

Figure 8. Android App Message

## 4.3 Upstream Messaging [17]:

To initiate upstream messages, the client application sends a request containing the following:

- The address of the application server, i.e. SENDER_ID@gcm.googleapis.com.
- A message ID that should be unique per sender ID.
- The message data comprising the key/value pairs of the message's payload.

**Send an upstream message from an Android client application:**

```
public void onClick(final View view) {
    if (view == findViewById(R.id.send)) {
        new AsyncTask() {
            @Override
            protected String doInBackground(Void... params) {
                String msg = "";
                try {
                    Bundle data = new Bundle();
                    data.putString("my_message", "Hello World");
                    data.putString("my_action","SAY_HELLO");
                    String id = Integer.toString(msgId.incrementAndGet());
                    gcm.send(SENDER_ID + "@gcm.googleapis.com", id, data);
                    msg = "Sent message";
                } catch (IOException ex) {
                    msg = "Error :" + ex.getMessage();
                }
                return msg;
            }

            @Override
            protected void onPostExecute(String msg) {
                mDisplay.append(msg + "\n");
            }
        }.execute(null, null, null);
    } else if (view == findViewById(R.id.clear)) {
        mDisplay.setText("");
    }
```

Figure 9. Android Upstream Message

**Receive XMPP message on the application server:**

```
<message id="">
  <gcm xmlns="google:mobile:data">
  {
      "category":"com.example.yourapp", // to know which app sent it
      "data":
      {
          "hello":"world",
      },
      "message_id":"m-123",
      "from":"REGID"
  }
  </gcm>
</message>
```

Figure 10. XMPP Message on Server

## 3. CONCLUSIONS

Google cloud messaging provides great convenience and flexible way for upstream and downstream messaging between client and server. In this paper, we have discussed architecture, different ways of GCM implementation and how third party application server sends messages to registered Android device via GCM. GCM solved many obstacles between client and server synchronization by push messaging technique. There are some limitations of GCM system like message delivery is unpredictable. The output of the system can't be expected with certainty. GCM works efficiently only when device has stable internet connection.

## REFERENCES


[1] Wikipedia, "Push technology".
Available: http://en.wikipedia.org/wiki/Push_technology

[2] Google Cloud Platform, "Cloud Messaging Support".
Available: https://cloud.google.com/tools/android-studio/messaging/

[3] Jarle Hansen, Tor-Morten Grønli, Gheorghita Ghinea, "Towards Cloud to Device Push Messaging on Android: Technologies, Possibilities and Challenges", Int. J. Communications, Network and System Sciences, 2012, 5, 839-849

[4] Wikipedia, "Mobile cloud computing".
Available: http://en.wikipedia.org/wiki/Mobile_cloud_computing

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

[6] Android Developers, "Google Glass-Mirror API."
Available: https://developers.google.com/glass/develop/mirror/index

[7] Chetan D Wadate, Prashant T Suvare, Aniket S More and Rina Bora- "A Survey of Automatic Wi-Fi based Push Notification in College Campus using Cloud", International Journal of Computer Applications (0975 – 8887), International Conference on Advances in Science and Technology (ICAST-2014).

[8] Jarle Hansen, Tor-Morten Grønli and Gheorghita Ghinea- "Towards Cloud to Device Push Messaging on Android: Technologies, Possibilities and Challenges", Int. J. Communications, Network and System Sciences (839-849), December-2012.

[9] Yavuz Selim Yilmaz, Bahadir Ismail Aydin and Murat Demirbas- "Google Cloud Messaging (GCM): An Evaluation", Globecom 2014 -Symposium on Selected Areas in Communications: GC14 SAC Internet of Things.

[10] Harminder Singh, Dr Sudesh Kumar, Harpreet Kaur- "Location Based System Using Google Cloud Messaging", National Conference on Innovative Trends in Computer Science Engineering (ITCSE-2015), April-2015.
[11] Naresh Kumar N and Prof. Mohan K-"GCM Service Driven Communication With An Android Application In Cloud Computing", International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 5, May – 2013.
[12] Google Play, "View & diagnose Google Cloud Messaging (GCM) statistics" Available: https://support.google.com/googleplay/android-developer /answer/2663268? hl=en
[13] AndroidHive, "Android Push Notifications using Google Cloud Messaging" Available: http://www.androidhive.info/
[14] Tutorialspoint: "Android-Architecture" Available: http://tutorialspoint.com/android/android_architecture
[15] PHP, "PHP Manual" Available: http://www.php.net/manual/
[16] MySQL, " MySQL Documentation" Available: http://www.mysql.com
[17] Google Developers, "Cloud Messaging", Available: https://developers.google.com/cloud-messaging

**Authors:**

**Nilay Ganatra** received Bachelor's degree in Computer Science B.Sc. (Computer Science) from Sardar Patel University, Gujarat,India and Master's Degree in Computer Applications (M.C.A) from Gujarat University, Gujarat, India. He is with MCA Department at Smt Chandaben Mohanbhai Patel Institute of Computer Applications, Charotar University of Science and Technology (CHARUSAT), Changa, Gujarat,India. His research interests include Wireless Networks and Mobile Computing.

**Rachana Patel** received Bachelor's degree in Computer Application B.C.A (Computer Science) from Dharmsinh Desai University, Gujarat, India and Master's Degree in Computer Applications (M.C.A) from Gujarat University, Gujarat, India. She is with MCA Department at Smt Chandaben Mohanbhai Patel Institute of Computer Applications, Charotar University of Science and Technology (CHARUSAT), Changa, Gujarat,India. His research interests include Machine Learning and Wireless Networks.