College of Engineering,
Trivandrum

Computer Science & Engineering

Google Cloud Messaging

Submitted By:
Kevin Joseph
josephkevin1995@gmail.com
Roll No. 33

Supervisor:
Vipin Vasu
vipin@cet.ac.in

July 19, 2016

Contents

List of Figures

Abstract

Android traditionally kept data synchronization between android-device and server-side using a method of pullling. Each Android device has to poll the server for updated data, which leads to unneccessary network traffic and wastage of device battery. In order to overcome this weakness, a data pushing service, GCM was introduced. Push describes a style of communication where the request for a given transaction is initiated by the publisher or central server. Push messaging is a multi-channel mobile cloud communication platform that unifies push notifications, SMS and instant messaging.
GSM service allows sending data from the app engine or any other backend to android powered devices. GSM is a lightweight push notification based service notifying android applications about new data to be fetched from the server or sending messages containing 4KB of payload data. GCM manages all aspects of message queuing and delivery of message to target android application running on a target device. Applications on the target device need not be running to receive messages. This service will wake up the application as long as the application is set up with the proper broadcast receiver and permissions. The application might post a notification, display a custom user interface, or silently sync data

1 Introduction to Cloud

Cloud computing is a kind of Internet-based computing that provides shared processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) [5] [4] which can be rapidly provisioned and released with minimal management effort. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers.[3] It relies on sharing of resources to achieve coherence and economy of scale, similar to a utility (like the electricity grid) over a network. [6]

The cloud provides an appearance unlimited scalability and storage for less cost than in-house data centers. Since its origins cloud services have been providing greater and greater levels of abstraction, making life easier for users. With the advent of mobile and pervasive computing era, smartphones became ubiquitous, and wearable devices are getting traction. A significant portion of the applications for these devices rely on remote servers on the cloud. Considering the ever growing use of cloud on devices it becomes important to address the problem of making communications between the two more efficient. This is where services like Google Cloud Messaging(GCM) come in. GCM is a hosted push service that allows application administrators to send push to specific devices, groups or topics as per required without having to worry about details like queing and delivery of the messages. This report aims to present the workings and some of the network methodologies used by GCM to achieve the same, it also provides an analysis on it, C2DM and Firebase Cloud Messaging(FCM) in order to study the growth of the technology.

2 Cloud Computing

Cloud computing is a general term for the delivery of hosted services over the Internet. Cloud computing enables companies to consume compute resources as a utility – just like electricity – rather than having to build and maintain computing infrastructures in-house. Such a system provides features like scalability and elasticity providing new and interesting opportunities for many companies.It has proven particularly useful for small and medium enterprises

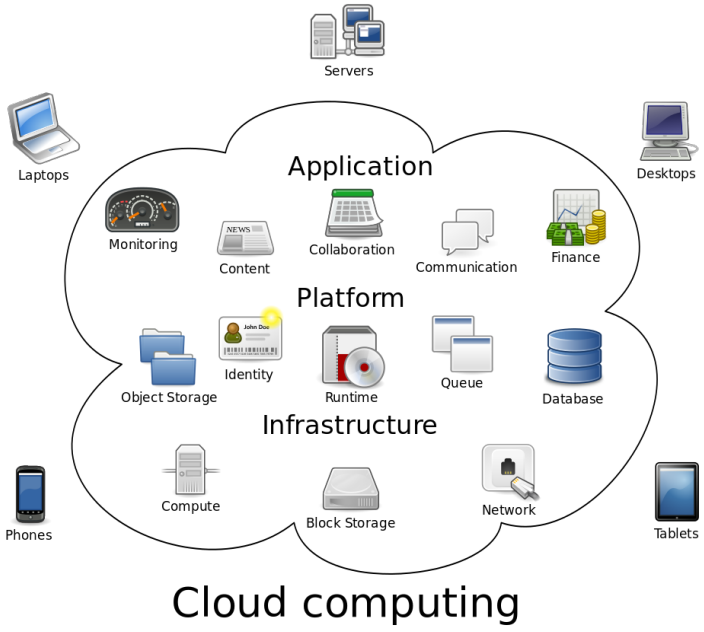

Figure 1: Cloud Computing
[6]

that have a large variation in their computing needs. However, not all businesses will benefit from moving their data centres to the cloud, e.g. when there are government regulations not allowing sen- sitive data to be stored with an external cloud provider.

The services provided via the cloud can be classified into 3 types:

2.1 Software As A Service(SAAS)

Software as a service (SaaS) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. SaaS removes the need for organizations to install and run applications on their own computers or in their own data centers. This eliminates the expense of hardware acquisition, provisioning and maintenance, as well as software licensing, installation and support.This kind of service provides the maximum level of abstraction where the user only has to provide an input and recieve output from an application provided by a

cloud provider.

2.2 Platform As A Service(PAAS)

In a PaaS model, a cloud provider delivers hardware and software tools, usually those needed for developing and hosting new applications. This kind of service works at a lower level abstraction than SAAS, here the provider provides the hardware and the software environment required for users to develop and host an application of their own.

2.3 Infrastructure As A Service(IAAS)

In an IaaS model, a third-party provider hosts hardware, software, servers, storage and other infrastructure components on behalf of its users. This works at the lowest level of abstraction where the provider only takes care of the hardware maintainance and the users have to create the required evironment and software as per their requirement.IaaS platforms offer highly scalable resources that can be adjusted on-demand. This makes IaaS well-suited for workloads that are temporary, experimental or change unexpected.

3 Communication Between Cloud And Device

Considering the increasing number of mobile devices and their dependence on the cloud it is important to make sure the modes of communication are efficient and ensure on time delivery. The methods of communication can be broadly classified into 2 types:

3.1 Pull Technology

Pulling is a style of network communication where the initial request for data originates from the client, and then is responded to by the server.Pull requests form the foundation of network computing, where many clients request data from centralised servers. Pull is used extensively on the Internet for HTTP page requests from websites.A push can also be simulated using multiple pulls within a short amount of time. For example, when pulling POP3 email messages from a server, a client can make regular pull requests every few

minutes. To the user, the email then appears to be pushed, as emails appear to arrive close to real-time. The tradeoff is this places a heavier load on both the server and network in order to function correctly.

3.2 Push Technology

Push, or server push, describes a style of Internet-based communication where the request for a given transaction is initiated by the publisher or central server.Push services are often based on information preferences expressed in advance. This is called a publish/subscribe model. A client "subscribes" to various information "channels" provided by a server; whenever new content is available on one of those channels, the server pushes that information out to the client.

A push can also be simulated using multiple pulls within a short amount of time. For example, when pulling POP3 email messages from a server, a client can make regular pull requests every few minutes. To the user, the email then appears to be pushed, as emails appear to arrive close to real-time. The tradeoff is this places a heavier load on both the server and network in order to function correctly.

Another way of simulating push using pull is long pollling. This is used particularly under circumstances where a real push is not possible, such as sites with security policies that require rejection of incoming HTTP/S requests. In this method the client sends a request to the server but the server need not reply instantly but may wait till it has relevant data and as soon as the server sends a reply the client sends another request. Long polling gets rid of the laatency seen in the previous method.

General Usage: Push technology is generally used in synchronous conferencing, instant messaging, email, etc. The SMTP mail protocol is a push protocol, the last step of mail where the client gets the mail from the mail server uses a pull protocol like POP3 or IMAP.

3.2.1 Hosted Push Services

Several companies provide push notification handling as a service. Here the application developer dosn't have to deal with details like queing of messages.Few of the major hosted push services are:

- Apple Push Notification Service

- Google Cloud Messaging
- Xtremepush

4 Cloud To Device Messaging(C2DM)

C2DM the predecessor of GCM was released in 2010 and was first featured in Android 2.2. Some of the features presented by C2DM were:
Features:

- When messages are received on the Android client, the system will wake up the application via an Intent broad- cast, and pass the message data
- Developers are encouraged to send short messages, essentially notifying the mobile application that updated information can be retrieved from the server
- Maxi- mum number of messages that can be sent is approxi- mately 200,000 per day. This limit could be increased if required but was not encouraged.

Disadvantages:

- Certain features, such as sending messages to multiple clients, is not supported. Messages could not be addressed to groups or topics.
- The development environment and API could be better. Seeing the possibility of improvment, another push notification library called Simple-C2DM was developed at the same time. Simple-C2DM had simpler APIs and was implemented with a higher level of abstraction.
- In certain cases quite a big increase in response times for some requests was seen.

5 Google Cloud Messaging(GCM)

Google Cloud Messaging (GCM) is a free service that enables developers to send messages between servers and client apps. This includes downstream messages from servers to client apps, and upstream messages from client apps to servers. The architecture of GCM includes 3 main components:

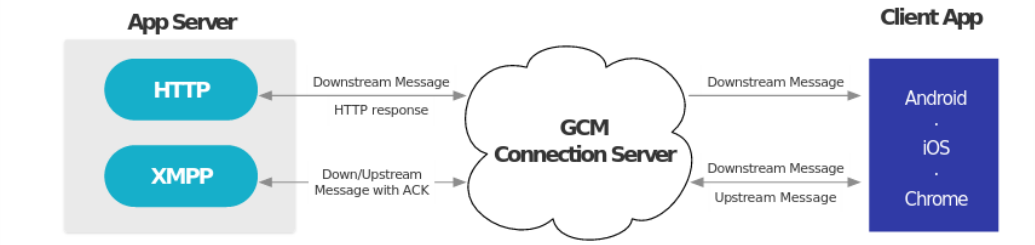

Figure 2: GCM Architecture

- **GCM Connection Servers:** These are Google servers that transmit the messages from users' application srver to the client application.
- **Client Application:** This includes all the instances of the GCM enabled application running on different devices.
- **Application Server:** This is the server handled by the application developer. The messages to the client applications originate from this server and are sent to the connection servers.

Another key concept of GCM is credentials, The IDs and tokens that are used in GCM to ensure that all parties have been authenticated, and that the message is going to the correct place.

Credentials:

- **Sender ID**: The sender ID is used in the registration process to identify an app server that is permitted to send messages to the client app.
- **API Key**: An API key saved on the app server that gives the app server authorized access to Google services. In HTTP, the API key is included in the header of POST requests that send messages. In XMPP, the API key is used in the SASL PLAIN authentication request as a password to authenticate the connection
- **Application ID**: The client app that is registering to receive messages.
- **Registration Token**: An ID issued by the GCM connection servers to the client app that allows it to receive messages.

5.1 GCM Lifecycle

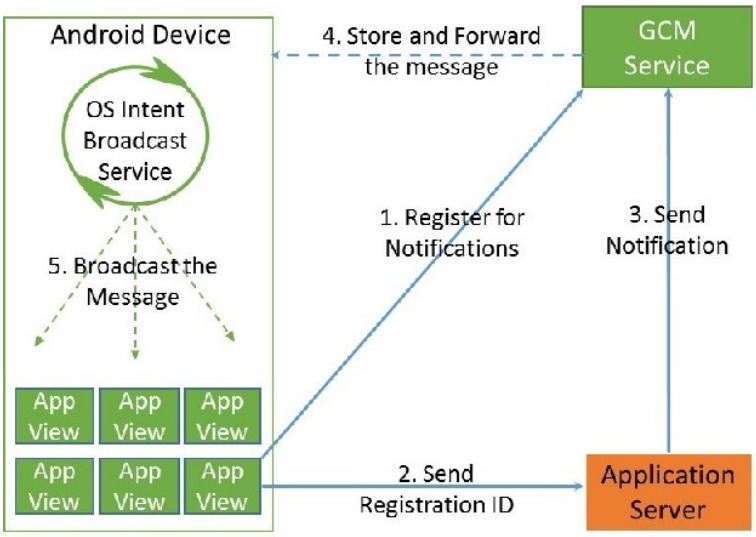

Figure 3: GCM Lifecycle

5.1.1 Lifecycle Steps:

- Registering Client Application
- Send and Recieve downstream messages
  - The app server sends a message to GCM connection servers.
  - The GCM connection server enqueues and stores the message if the device is offline.
  - When the device is online, the GCM connection server sends the message to the device.
  - On the device, the client app receives the message according to the platform-specific implementation.
- Send and receive upstream messages. This is available only if you use XMPP connection servers
  - On the device, the client app sends messages to the XMPP connection server.

  - The XMPP connection server enqueues and stores the message if the server is disconnected.
  - When the app server is re-connected, the XMPP connection server sends the message to the app server.

5.2 Registering Client Aplications

To verify that they can send and receive messages, client apps must register with GCM. In this process, the client obtains a unique registration token and passes it to the app server, which stores the token and sends an acknowledgement back to the client app. The registration token exchanged in this process is the same client app instance identifier that the app server uses to send messages to the particular client.

- The client app obtains a registration token using the Instance ID API. The call to this API must have the authorized entity set to your app server's sender ID, and the scope set to the appropriate value for GCM
- The client app passes the registration token to the app server.
- The app server saves the registration token and acknowledges to the client app that the process completed successfully

In case the server is unable to do its part of the handshaking, the application must retry sending the token or delete it.

5.2.1 Retry Using Exponential Backoff

In case the registration fails, the client app must use exponential backoff to retry *i.e.* the app must wait double the time it waited on the last try every time. The retries exponentially increase the waiting time up to a certain threshold. The idea is that if the server is down temporarily, it is not overwhelmed with requests hitting at the same time when it comes back up.

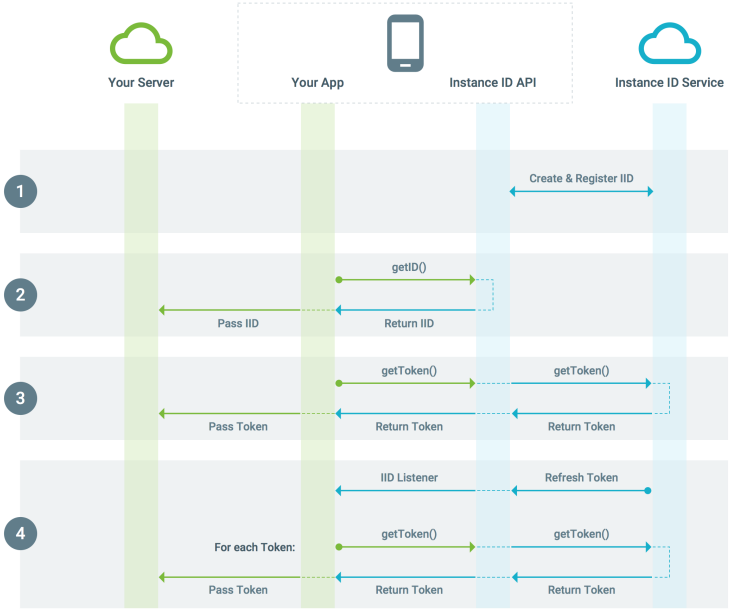

Figure 4: Instance ID lifecycle

5.2.2 Instance ID API(Android Impletation)

Get Instance ID

```
String iid = InstanceID.getInstance(context).getId();
```

Generate a Token

```
String authorizedEntity = PROJECT_ID; // Project id from Google
                                      //    Developer Console
String scope = "GCM"; // e.g. communicating using GCM, but you
                      //      can use any
                      // URL-safe characters up to a maximum
                      //of 1000, or
                      // you can also leave it blank.
String token = InstanceID.getInstance(context).getToken(authorizedEntity
```

Refresh Tokens

The Instance ID service periodically refeshes the tokens. It may also do so for the following reasons:

- security issues.
- Device information is no longer valid.
- The Instance ID service is otherwise affected.

The app must implement a listener in order to handle these callbacks to refresh tokens.

```
public class MyInstanceIDService extends InstanceIDListenerService {
  public void onTokenRefresh() {
    refreshAllTokens();
  }

  private void refreshAllTokens() {
    // assuming you have defined TokenList as
    // some generalized store for your tokens
    ArrayList<TokenList> tokenList = TokensList.get();
    InstanceID iid = InstanceID.getInstance(this);
    for(tokenItem : tokenList) {
      tokenItem.token =
        iid.getToken(tokenItem.authorizedEntity,tokenItem.scope,tokenIte
      // send this tokenItem.token to your server
    }
  }
};
```

5.2.3 Uninstalled Client App Unregistration

The process of removing tokens of an uninstalled application instance is automated by GCM, but this does not happen instantly. It involves the following steps:

- The app server sends a message to GCM connection server addressed to an uninstalled instance.
- The GCM connection server sends the message to the GCM client on the device.
- The GCM client on the device receives the message and detects that the client app has been uninstalled.

- The GCM client on the device informs the GCM connection server that the client app was uninstalled.
- The GCM connection server marks the registration token for deletion.
- The app server sends a message to GCM for the same instance.
- The GCM returns a NotRegistered error message to the app server.
- The app server should delete the registration token.

5.3 GCM Connection Servers

The server side of GCM consists of two components:

- Connection Servers
- Application Servers

5.3.1 Application Servers

Before you can write client apps that use GCM, you must have an application server that meets the following criteria:

- Able to communicate with your client.
- Able to send properly formatted requests to the GCM connection server.
- Able to handle requests and resend them using exponential back-off.
- Able to securely store the API key and client registration tokens. Note: never include the API key in any client code.
- For XMPP, the server must be able to generate message IDs to uniquely identify each message it sends (GCM HTTP connection server generates message IDs and returns them in the response). XMPP message IDs should be unique per sender ID.

5.3.2 Connection Servers

Google provides two types of connection servers for GCM, for HTTP and XMPP protocols. The major difference between the two is that XMPP allows both downstream and upstream messages while HTTP allows only downstream messages.

HTTP Server

To send a message, the application server issues a POST request.A message request is made of 2 parts: HTTP header and HTTP body. The header must contain authorization and content-type. Example:

```
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{
  "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
  "data" : {
    ...
  },
}
```

Request Format

The possible types of messages are:

- **Send To Sync:** This is the smallest possible message

```
{ "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1..." }
```

- **Message with payload — notification message:**

```
{ "notification": {
                    "title": "Portugal vs. Denmark",
                    "text": "5 to 1"
                    },
"to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1..."
}
```

- **Message with payload — Data message:**

```
{ "data": {
                  "score": "5x1",
                  "time": "15:10"
```

```
                    },
"to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1..."
}
```

XMPP Server
The Google Cloud Messaging (GCM) Cloud Connection Server (CCS) is an XMPP endpoint that provides a persistent, asynchronous, bidirectional connection to Google servers. The connection can be used to send and receive messages between your server and your users' GCM-connected devices. The connection has two important requirements:

- You must initiate a Transport Layer Security (TLS) connection. Note that CCS doesn't currently support the STARTTLS extension.
- CCS requires a SASL PLAIN authentication mechanism using GCM sender ID and the API key as the password, where the sender ID and API key are the values you gathered when configuring your client app. See the client documentation for your platform for information on obtaining these credentials.

The possible types of messages are:

- **Send To Sync:** This is the smallest possible message

```
<message id="">
<gcm xmlns="google:mobile:data">
{
        "to":"REGISTRATION_ID",
}
</gcm>
</message>
```

- **Message with payload — notification message:**

```
<message id="">
<gcm xmlns="google:mobile:data">
{
        "to":"REGISTRATION_ID",
        "notification": {
                'title': 'Portugal',
                'text': 'test'
        },
        "time_to_live":"600"
```

```
}
</gcm>
</message>
```

- **Message with payload — Data message:**

```
<message id="">
<gcm xmlns="google:mobile:data">
{
        "to":"REGISTRATION_ID",
        "message_id":"m-1366082849205"
        "data":
        {
          "hello":"world",
        }
        "time_to_live":"600",
        "delay_while_idle": true/false,
        "delivery_receipt_requested": true/false
}
</gcm>
</message>
```

6 Performance Analysis

In GCM, push notifications to the same device are throttled using token bucket scheme.From the following graphs we can see that the bucket holds 20 tokens and is replenished at a rate of once every 180 seconds. Furthermore, the bucket appears to be completely replenished (randomly) every 0 to 90 minutes. When the bucket is out of tokens, push notifications are dropped instead of queued.

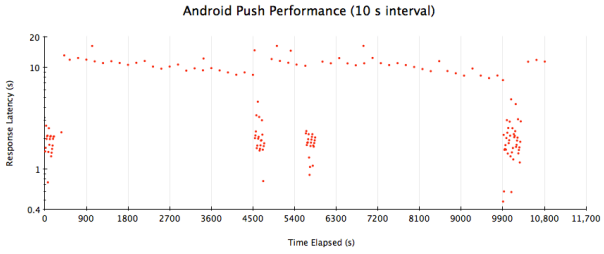

Figure 5: GCM Performance(10s interval)

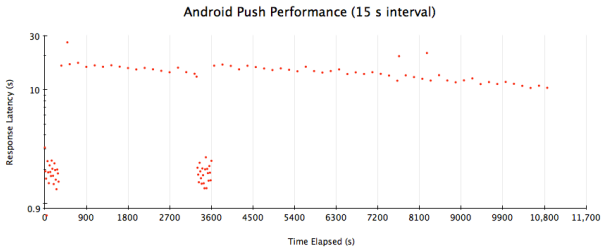

Figure 6: GCM Performance(15s interval)

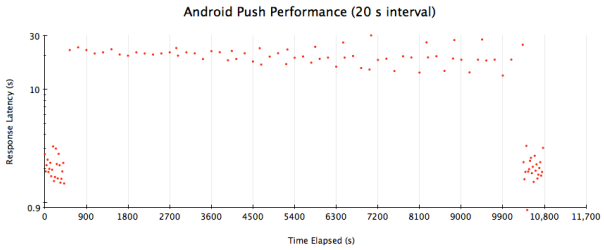

Figure 7: GCM Performance(20s interval)

7 Firebase Cloud Messaging(FCM)

Figure 8: FCM architecture

Firebase Cloud Platform inherits GCM's core infrastructure but simplifies the

client development. Developers no longer needs to write their own registration or subscription retry logic. FCM includes a web console called Firebase Notifications which allows users to send notifications without an App server.
Firebase covers all that mobile developers need to build, maintain and scale a mobile app on all platforms (even on iOS): from Storage and databases to innovative tools like Remote Config and Test Lab.
Firebase also presents some performance improvements, it has an average response latency on 250ms and 95% delivery rate which is a huge improvement when compared to 10s and 40% on GCM.

8 Conclusion

A study on GCM reveals that it is not suitable for time sensitive and/or "must-deliver-to-all" app scenarios since it does not guarantee a timely message arrival even when a reliable connection to Google's GCM servers on the client device exists. GCM is suitable for the application scenarios where random multicasting is sufficient, such as crowdsourcing systems. But with the arival of FCM it will be possible to use it for time sensitive and "must-deliver-to-all" app scenarios.

References

[1] G. G. Jarle Hansen, Tor-Morten Grønli, "Towards cloud to device push messaging on android: Technologies, possibilities and challenges," *Int. J. Communications, Network and System Sciences*, May 2012.

[2] A. M. D. Yavuz Selim, Yilmaz Bahadir Ismail, "Google cloud messaging (gcm): An evaluation," *Globecom 2014 - Symposium on Selected Areas in Communications: GC14 SAC Internet of Things.*

[3] P. M. K. Naresh Kumar N (M.Tech), "Gcm service driven communication with an android application in cloud computing," *International Journal of Engineering Research and Technology*, May 2013.

[4] P. B.Dhivya, G.Lakshmiprabha, "Gcm service driven communication with an android application in cloud computing," *International Journal of Emerging Technology and Innovative Engineering*, March 2015.

[5] Q. Hassan, "Demystifying cloud computing," *The Journal of Defense Software Engineering*, December 2014.

[6] "Cloud computing: Wiki." https://en.wikipedia.org/wiki/Cloud_computing.

[7] “Google cloud messaging.” https://developers.google.com/cloud-messaging/.