Department of Information Engineering, CUHK
MScIE – 2nd Semester, 2015/16

IEMS 5722
Mobile Network Programming and Distributed Server Architecture

Lecture 7
Instant Messaging & Google Cloud Messaging

Lecturer: Albert C. M. Au Yeung

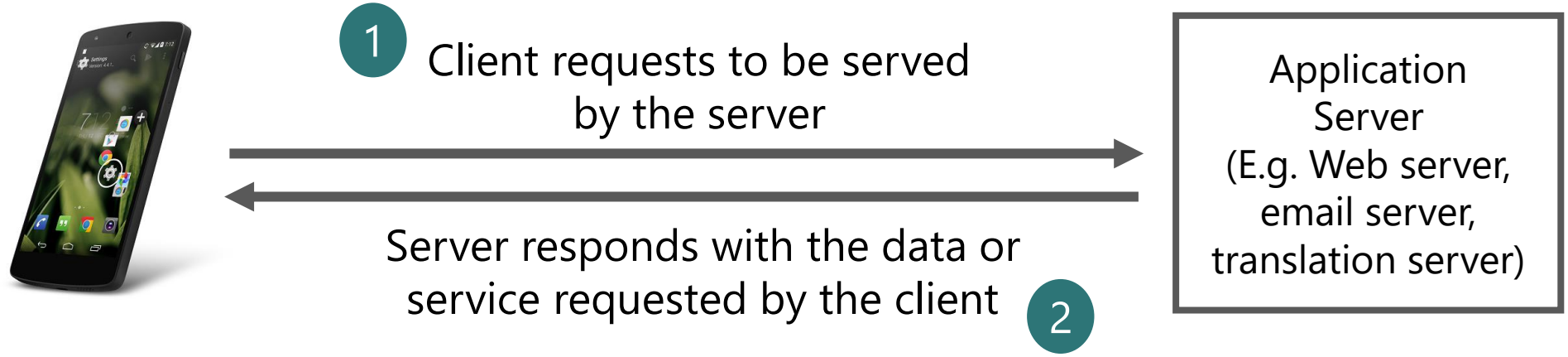
25th February, 2016

Push Technology

Pull and Push

All of the examples we have gone through in network programming so far can be regarded as using the "**pull**" method

- Communication is always initiated by the client
- Client "pulls" data or services from the server when necessary (e.g. when the user launches the app, or presses a button)



Pull and Push

HTTP is a pull-based protocol

- Users browse the Web and actively decide which Website to browse, which link to follow
- A effective and economical way (every user chooses what he needs)
- However, if some resources are regularly requested, the pull model can put heavy load on the server

Pull and Push

There are cases in which the server would like to initiate communication with the client(s)

- When new email arrives
- When a peer sends a message to the user through the server
- When the app/data needs to be updated

In these cases, the server needs to "**push**" data or services to the client

This kind of situations is getting more and more common as **smartphones** are getting more popular.

Implementing Push

The World Wide Web, and in particular the HTTP protocol, is designed for "pull", and additional engineering is required to implement push on the Web'

Some ways to "emulate" push on the Web

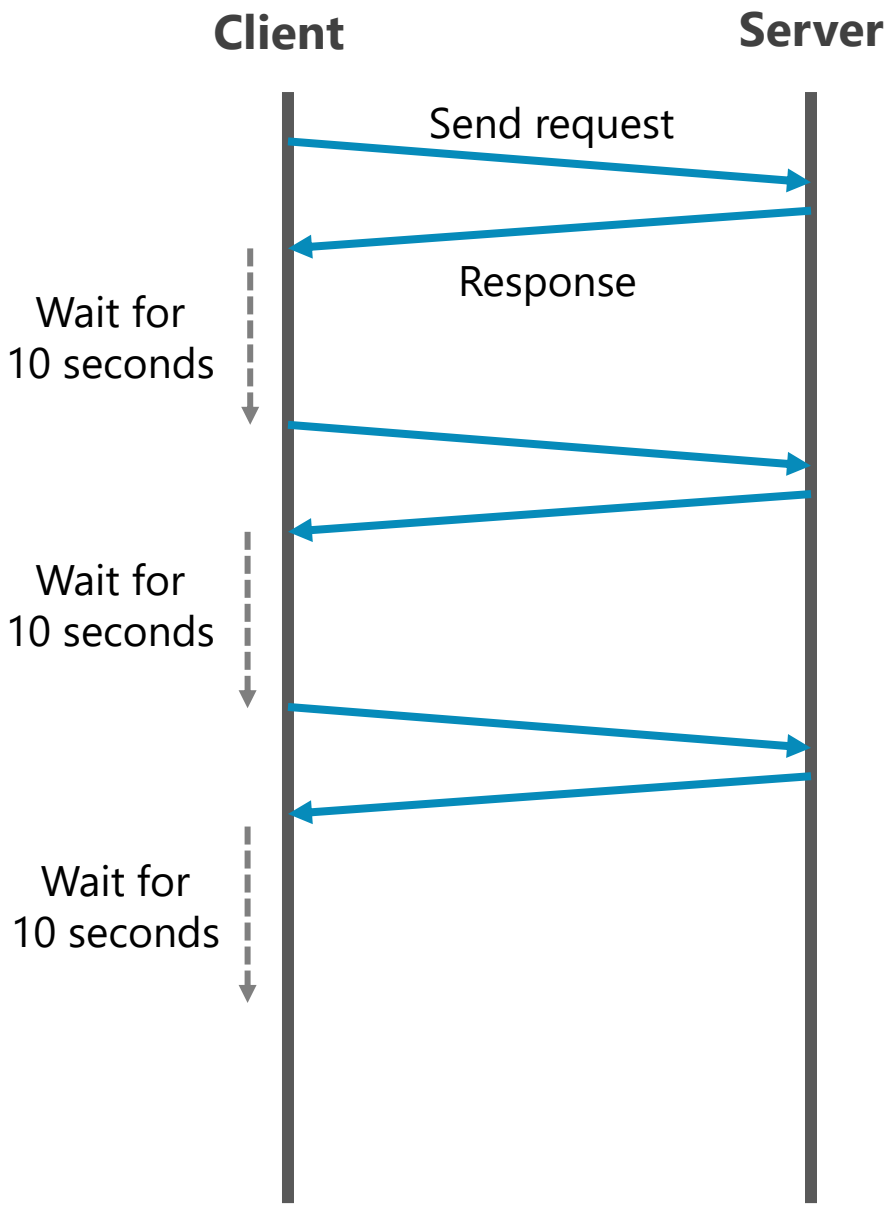
- Polling (Periodic pull)
- The Comet model
- BOSH
- WebSockets

Implementing Push – Polling

The client **polls** the server periodically to check if new messages or updates are available

Advantages:
1. Easy to implement
2. No extra development on the server-side

Disadvantages:
1. Unnecessary network traffic generated
2. Extra workload on the server



Implementing Push – Polling Examples

The Post-Office Protocol (POP) for email

- Email clients using the POP3 protocol make regular requests to the mail server to check for new emails

RSS Feed Readers

- RSS resources are served by HTTP, and thus are all pull-based
- RSS feed readers poll the RSS servers regularly and check for new updates of the feeds

Note: polling can place heavy workload on the server, the client and the network

Implementing Push – The Comet Model

"Comet" is a model for implementing Web applications that allow servers to push data to clients (browsers)

Implementations of Comet applications fall into two major categories

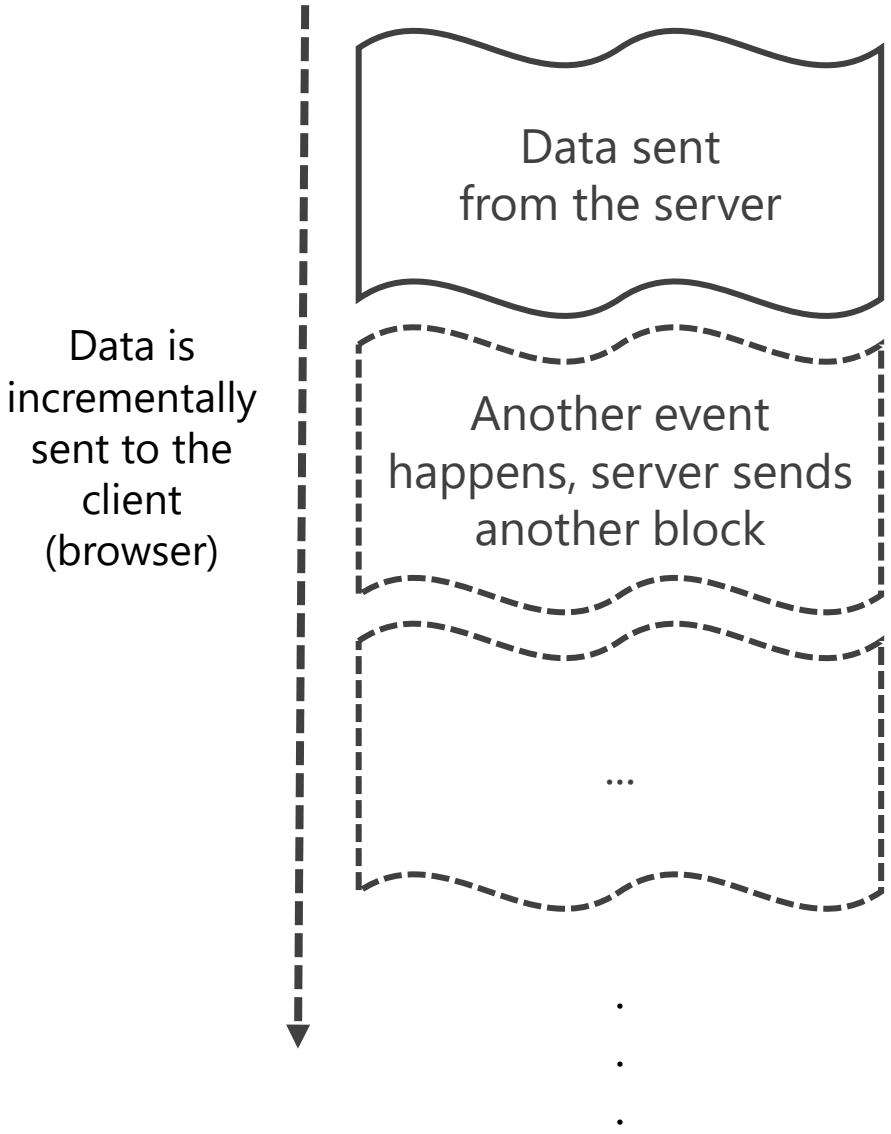
1. Streaming
2. Long-polling

Reference: http://en.wikipedia.org/wiki/Comet_(programming)

Implementing Push – The Comet Model

Streaming

- A persistent connection is established between the browser and the server
- Data is sent from the server a chunked block
- Events are incrementally sent to the browser (e.g. using <script> tags to execute JavaScript commands)

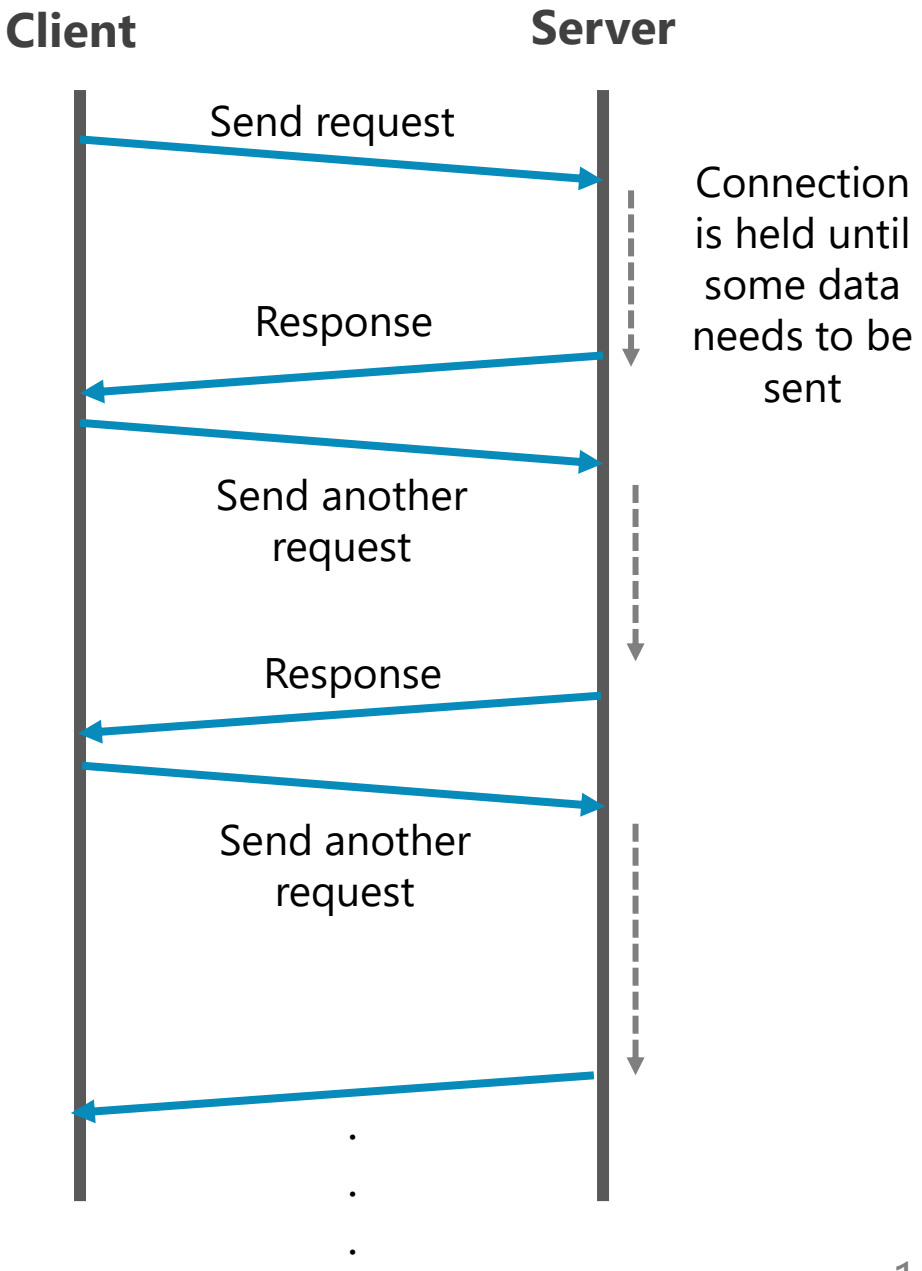


Reference:
http://en.wikipedia.org/wiki/Comet_(programming)
http://www.ibm.com/developerworks/library/wa-reverseajax1/

Implementing Push – The Comet Model

Long-polling

- A request is sent to from the client to the server
- The server holds the connection until some events happen, then response is sent back to the client
- The client, on receiving a response, issue another request immediately to the server
- (Usually implemented using Ajax)



Reference:
http://en.wikipedia.org/wiki/Comet_(programming)
http://www.ibm.com/developerworks/library/wa-reverseajax1/

Implementing Push – BOSH

BOSH stands for Bidirectional-streams over Synchronous HTTP

- It makes use of HTTP long-polling
- A single TCP connection is established to receive push data from server
- If no data needs to be pushed, server sends an empty <body/> message
- If client needs to send data to server, a second socket is used to send HTTP post requests
- The old and new connections will then switch roles (the new will be used for long-polling thereafter)

Reference: http://xmpp.org/extensions/xep-0124.html

WebSocket

- A protocol providing full-duplex communications channels between two computers over a TCP connection
- Designed to be implemented in Web browsers and Web servers
- Communications are done over TCP port 80 (can be used in secured computing environments)
- Socket.io – to be introduced later

HTTP is half-duplex: only one side can send data at a time, like walkie-talkie

Reference:
http://tools.ietf.org/html/rfc6455
http://www.websocket.org/

WebSocket

WebSocket is part of the HTML5 standard

- Supported in latest versions of major Web browsers
- Simple API in JavaScript
- Libraries also available on iOS and Android

```
var host = 'ws://localhost:8000/example';
var socket = new WebSocket(host);

socket.onopen = function() {
    console.log('socket opened');
    socket.send('Hello server!');
}
socket.onmessage = function(msg) {
    console.log(msg);
}
socket.onclose = function() {
    console.log('socket closed');
}
```

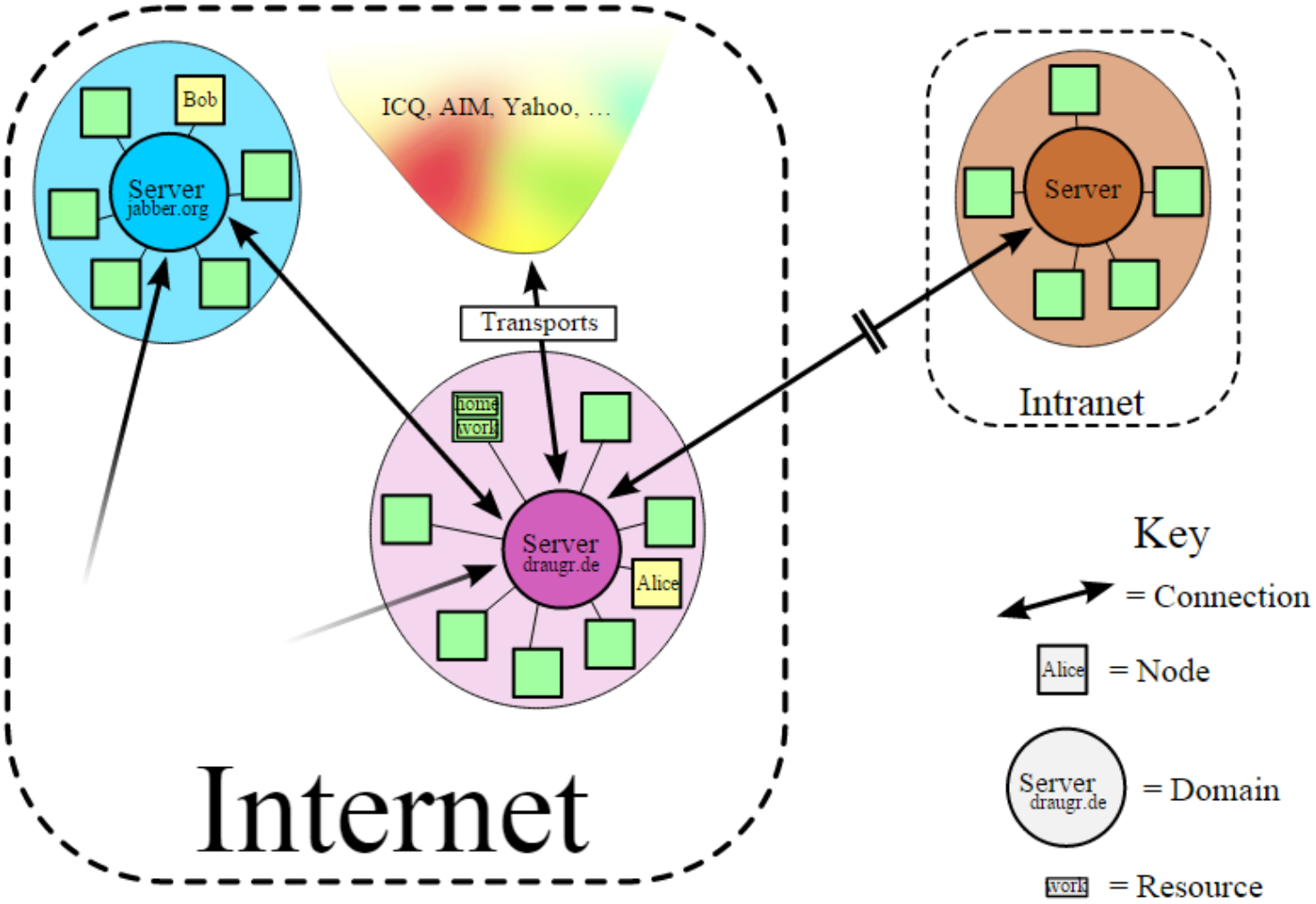
Try the game at
http://chrome.com/supersyncsports/

XMPP

XMPP stands for Extensible Messaging and Presence Protocol

- Originally named Jabber, an open source project for real-time and instant messaging
- Using a client-server architecture (non-P2P)
- Decentralized and federated model: no central server, but servers for different domains
- Each user connects to a public XMPP server, which will relay his messages to other users in other domains

XMPP



http://upload.wikimedia.org/wikipedia/commons/a/a8/JabberNetwork.svg

Mobile Push

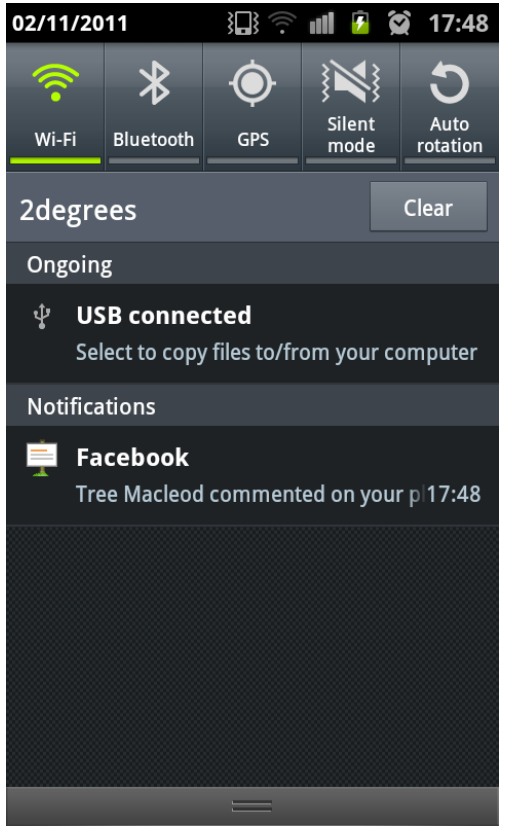
Push on Mobile Devices

"Push" is an important function on mobile devices, as it allows users to receive updates or messages without actively launching an app

On the two popular mobile platforms, push is realised through their corresponding services:

iOS:
Apple Push Notification Service (APNS)

Android:
Google Cloud Messaging (GCM)

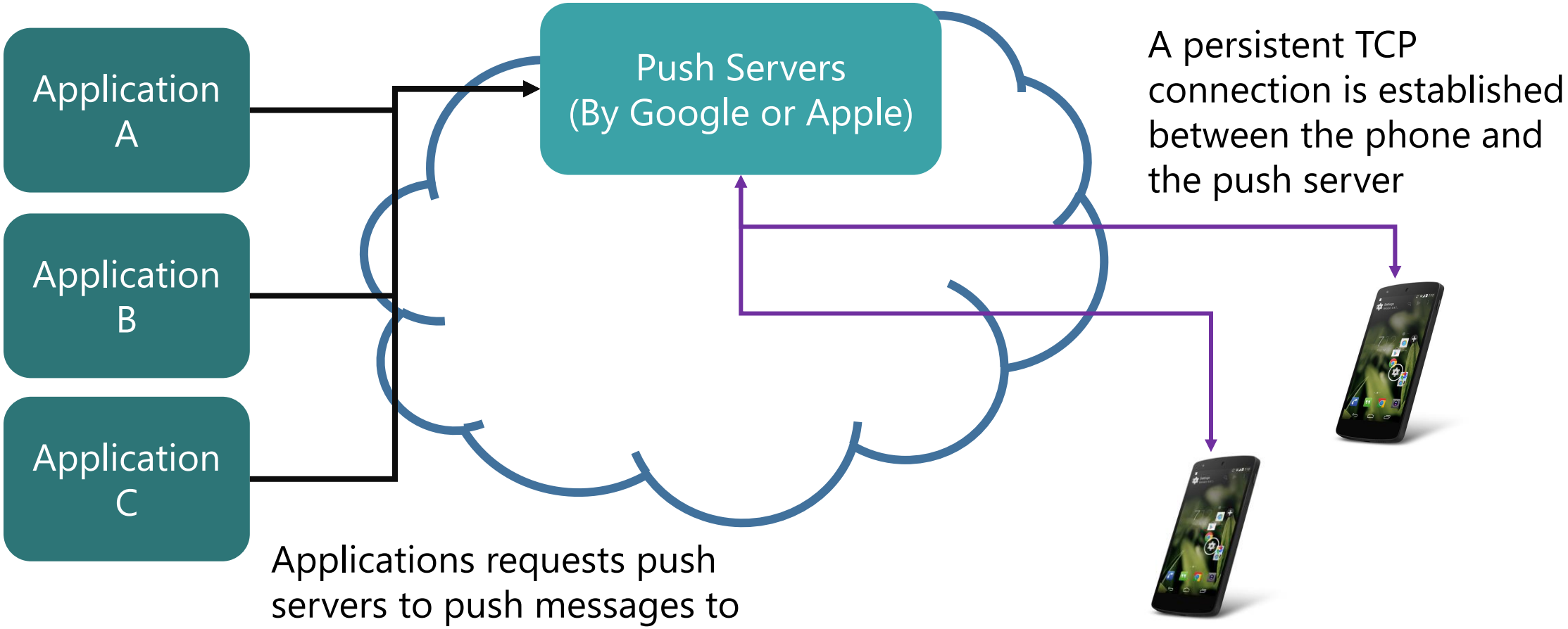


Android Notifications

iOS Notifications

Push on Mobile Devices

In general, push on mobile devices is implemented using the following architecture:



Push on Mobile Devices

- Push notification services provided by Google and Apple are standardized ways for pushing messages to apps
- However, it is not the only way:
  - It can be used as a form of signaling (request for updates, short notice, etc.)
  - Your app can still implement its own communication protocols for exchanging data with peers and servers

Reference:
https://code.facebook.com/posts/820258981365363/building-mobile-first-infrastructure-for-messenger/

Google Cloud Messaging (GCM)

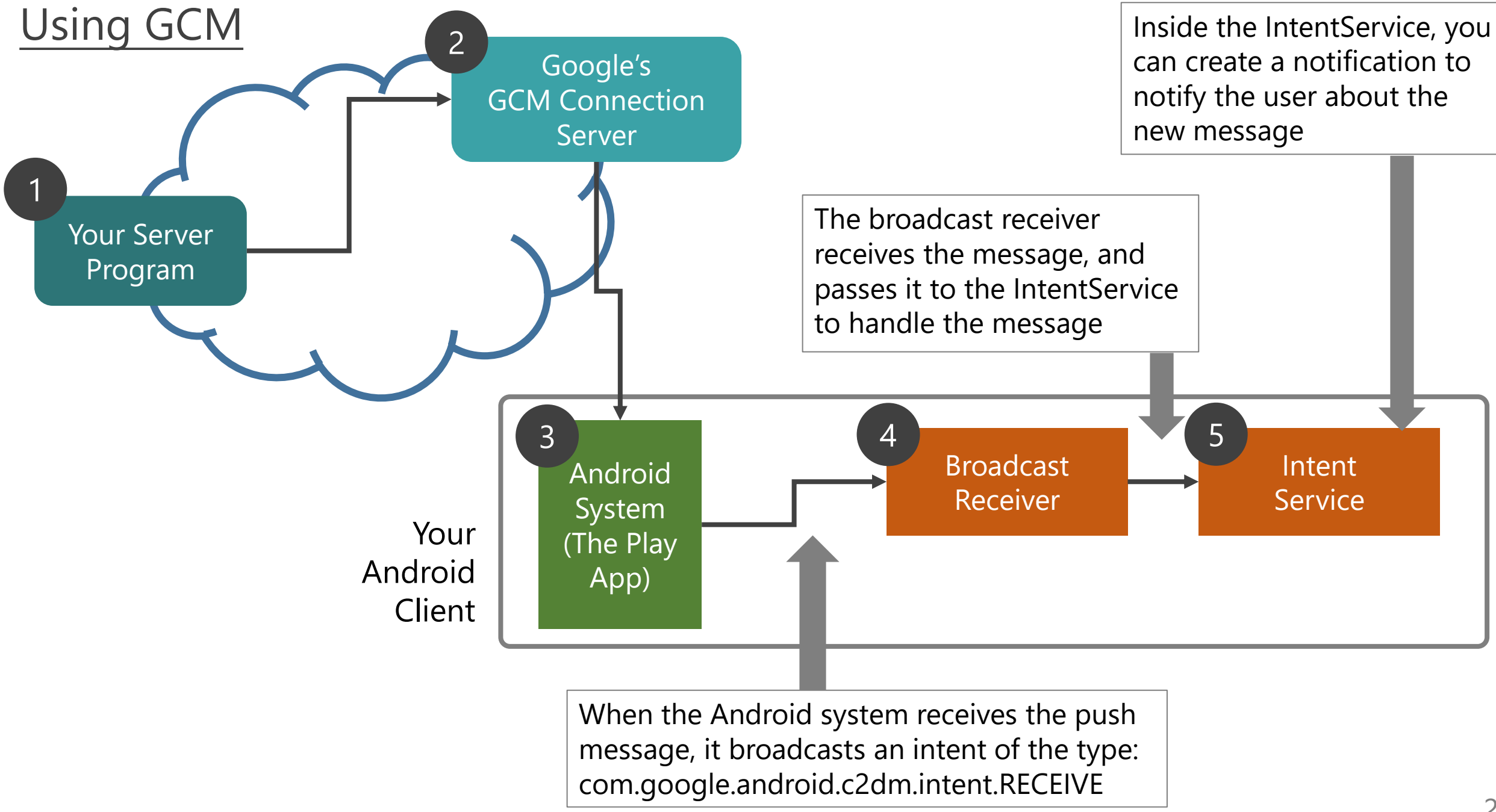
Google Cloud Messaging

Google provides a free service called GCM that allows you to develop a server program to push messages to your Android clients

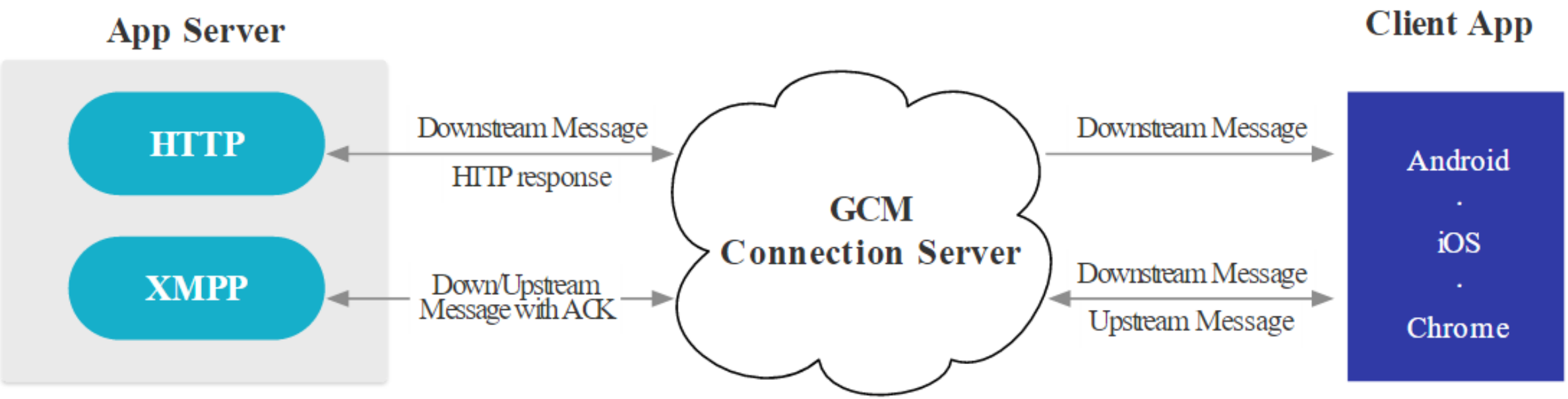
- Google's GCM servers handle all the queueing and delivering of messages
- Messages can contain up to 4KB of payload data
- The Android app does NOT have to be running to receive the message
- Available on Android 2.2 or above, devices must have Google Play Store installed

Reference: https://developers.google.com/cloud-messaging/

Using GCM


1
Your Server Program
2
Google's GCM Connection Server
Inside the IntentService, you can create a notification to notify the user about the new message
The broadcast receiver receives the message, and passes it to the IntentService to handle the message
Your Android Client
3
Android System (The Play App)
4
Broadcast Receiver
5
Intent Service
When the Android system receives the push message, it broadcasts an intent of the type: com.google.android.c2dm.intent.RECEIVE

Using GCM



Reference: https://developers.google.com/cloud-messaging/gcm

Setup

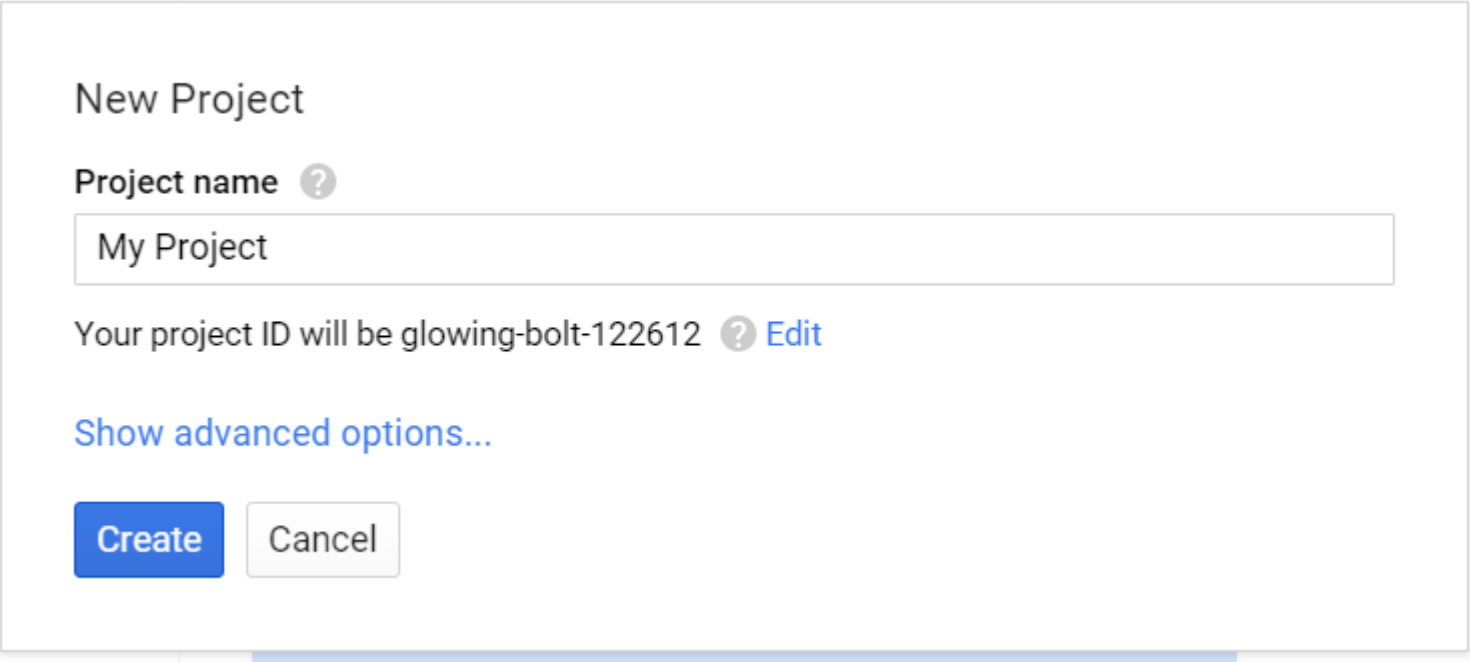
You have to go through the following steps in order to use GCM in your project

1. Create a Google account
2. Inside the Google Developer Console, create a project
3. Setting up for the client (app) and the server
   - Client: https://developers.google.com/cloud-messaging/android/client
   - Server: https://developers.google.com/cloud-messaging/downstream

Setting up a Project in Google

Once you have got a Google account, go to the Google Developer Console https://console.developers.google.com/

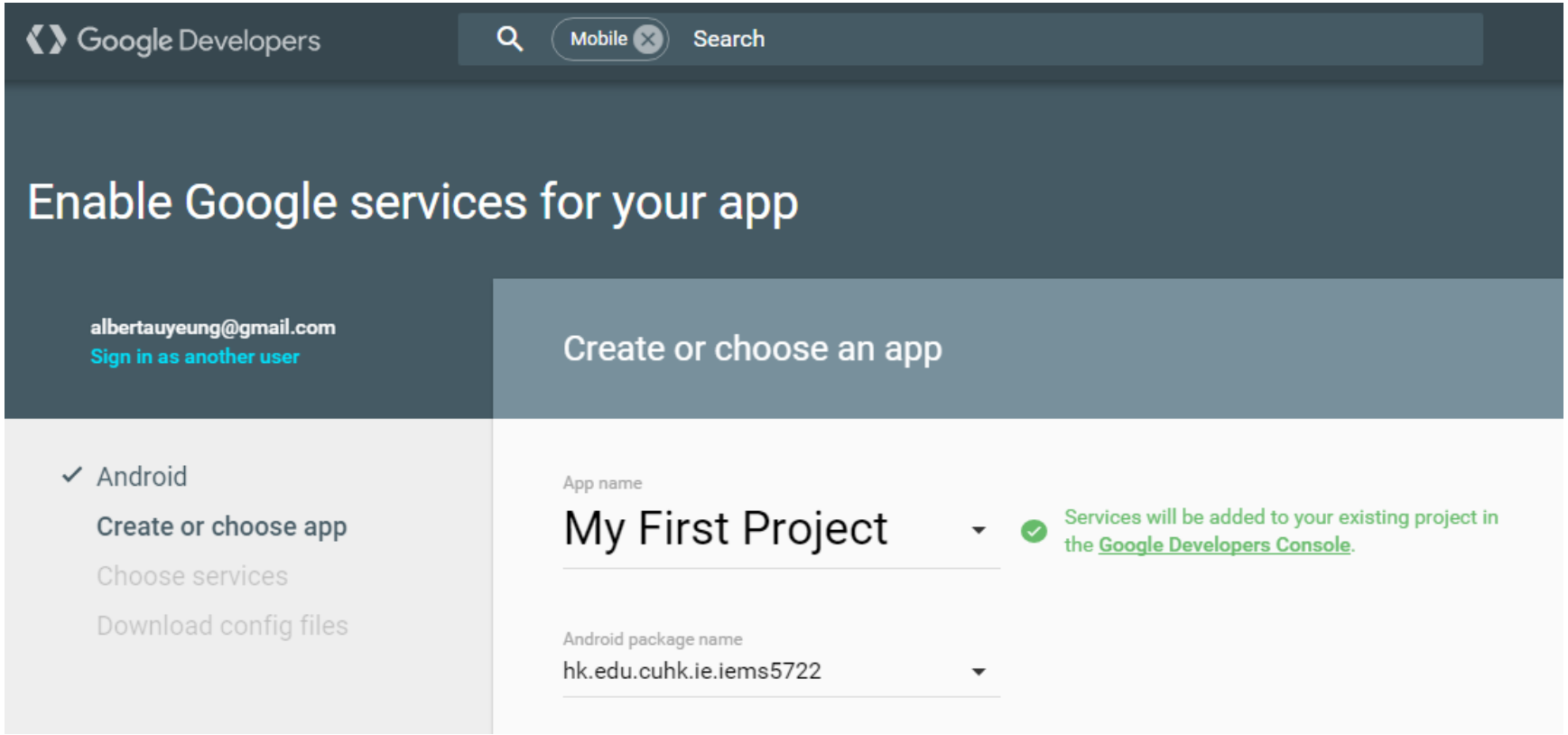
- Create a new project by providing a project name



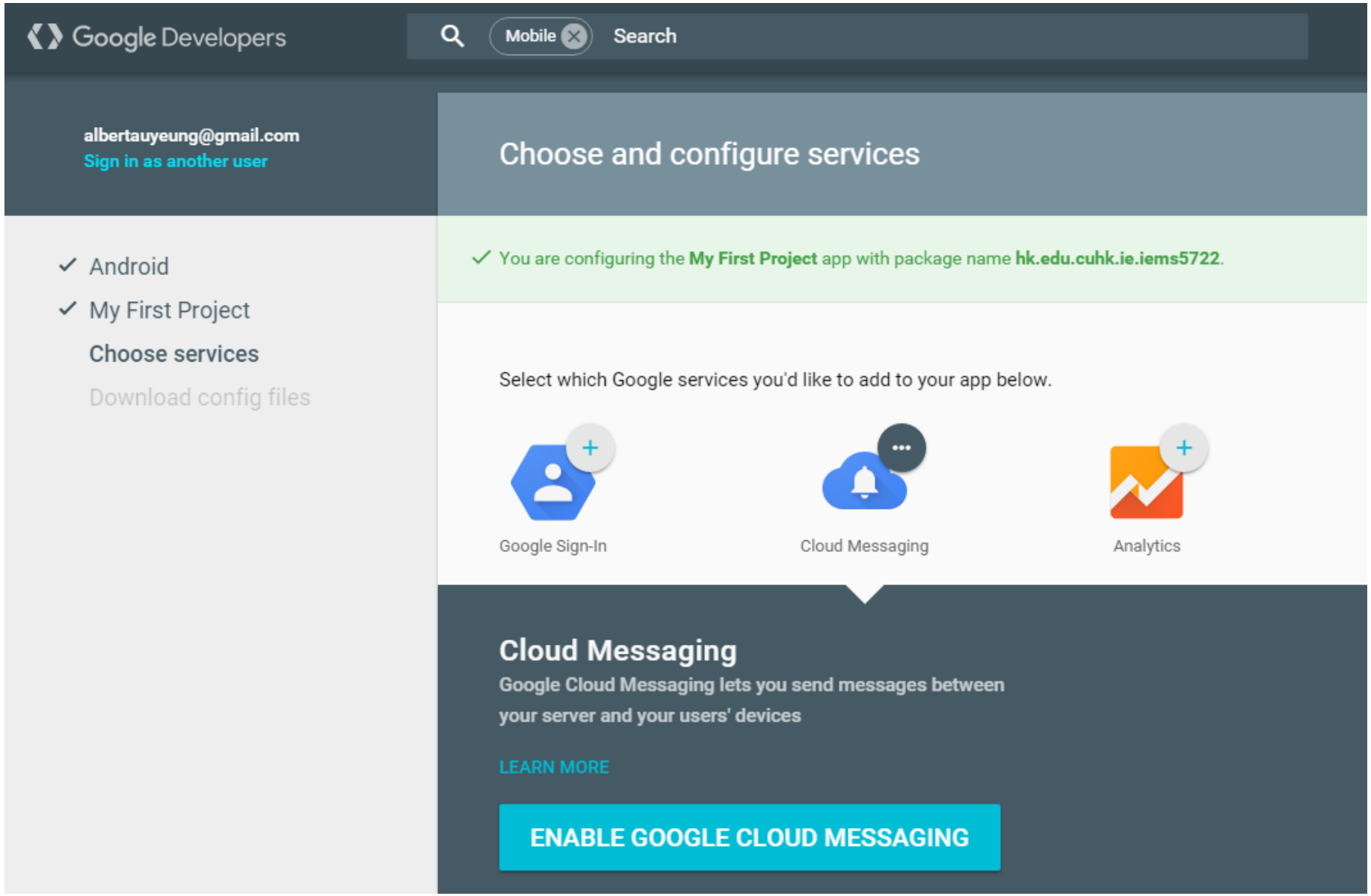
Setup

Once you have a project, you can use it to continue to set up GCM for your client and your server. Let's focus on the client side first.

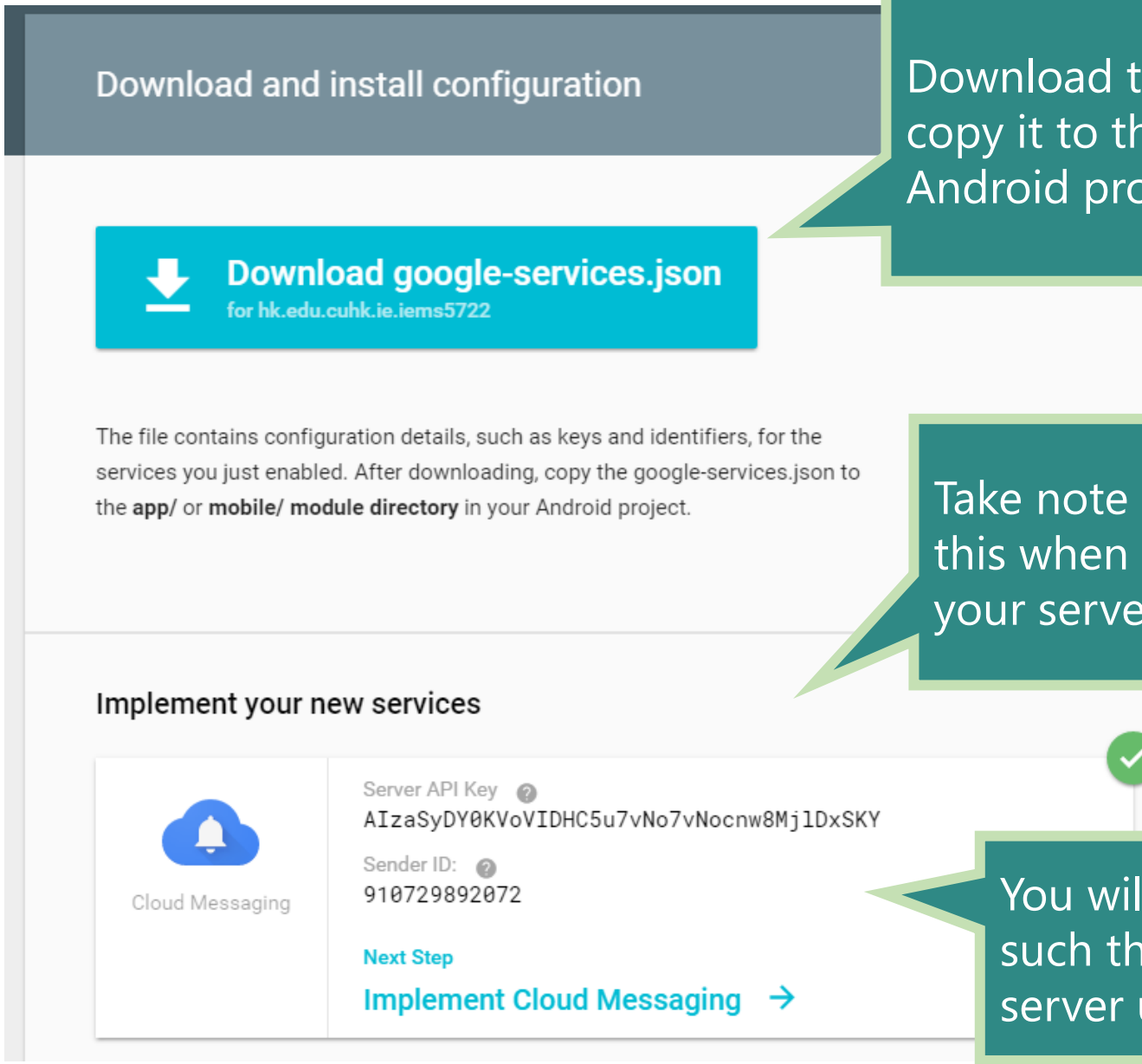
- Generate a configuration file at: https://developers.google.com/mobile/add?platform=android&cntapi=gcm



Setup


Google Developers
Mobile
Search
albertauyeung@gmail.com
Sign in as another user
Choose and configure services
Android
My First Project
Choose services
Download config files
You are configuring the My First Project app with package name hk.edu.cuhk.ie.iems5722.
Select which Google services you'd like to add to your app below.
Google Sign-In
Cloud Messaging
Analytics
Cloud Messaging
Google Cloud Messaging lets you send messages between your server and your users' devices
LEARN MORE
ENABLE GOOGLE CLOUD MESSAGING

Setup



Download this configuration file and copy it to the app/ directory of your Android project

Take note of this Server API Key, you will need this when you want to push a message from your server to the app.

You will need the Sender ID in your app, such that your app can receive GCM from a server using the API Key above.

Setup in the App Project

Next, you will have to purpose the following steps in your app

1. Add the configuration file and configure Gradle
2. Set up Google Play Services
3. Update your app's Manifest file
4. Implement the following:
   - Check for Google Play Services
   - Obtain a registration token from GCM server
   - Create a service that extends GcmListenerService to handle messages received from the server

Setup in the App Project (1)

Add the configuration file and configure Gradle

- Copy the download configuration file to the /app directory
- Add the dependency to your project-level **`build.gradle`** file

```
classpath 'com.google.gms:google-services:1.5.0-beta2'
```

- Add the dependency to your app-level **`build.gradle`** file:

```
apply plugin: 'com.google.gms.google-services'
```

Setup in the App Project (2)

Set up your app to use Google Play Services

- Add the following dependency to your app's build.gradle file:

```
dependencies {
  compile "com.google.android.gms:play-services:8.4.0"
}
```

Setup in the App Project (3) – Update Manifest File

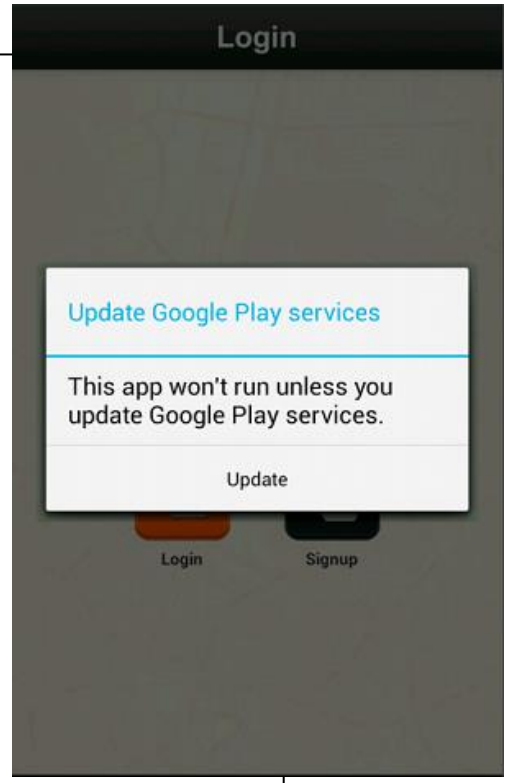
```
<permission android:name="<your-package-name>.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />
<uses-permission android:name="<your-package-name>.permission.C2D_MESSAGE" />

<application ...>
    <receiver
        android:name="com.google.android.gms.gcm.GcmReceiver"
        android:exported="true"
        android:permission="com.google.android.c2dm.permission.SEND" >
        <intent-filter>
            <action android:name="com.google.android.c2dm.intent.RECEIVE" />
            <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
            <category android:name="com.example.gcm" />
        </intent-filter>
    </receiver>
    <service
        android:name="com.example.MyGcmListenerService"
        android:exported="false" >
        <intent-filter>
            <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        </intent-filter>
    </service>
    <service
        android:name="com.example.MyInstanceIDListenerService"
        android:exported="false">
        <intent-filter>
            <action android:name="com.google.android.gms.iid.InstanceID" />
        </intent-filter>
    </service>
</application>
```

https://developers.google.com/cloud-messaging/android/client#manifest

Setup in the App Project (4)

Since GCM relies on the Google Play Services APK installed inside the Android device, you should check if this is available, and if not you should prompt the user to install or upgrade it

```
private boolean checkPlayServices() {
    int resultCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
    if (resultCode != ConnectionResult.SUCCESS) {
        if (GooglePlayServicesUtil.isUserRecoverableError(resultCode)) {
            GooglePlayServicesUtil.getErrorDialog(resultCode, this,
                    PLAY_SERVICES_RESOLUTION_REQUEST).show();
        } else {
            Log.i(TAG, "This device is not supported.");
            finish();
        }
        return false;
    }
    return true;
}
```



Setup in the App Project (4)

In order for your app to receive GCM, it has to register itself against the GCM service, this is done by using the Instance ID API provided by Android

You should create a service for this purpose:

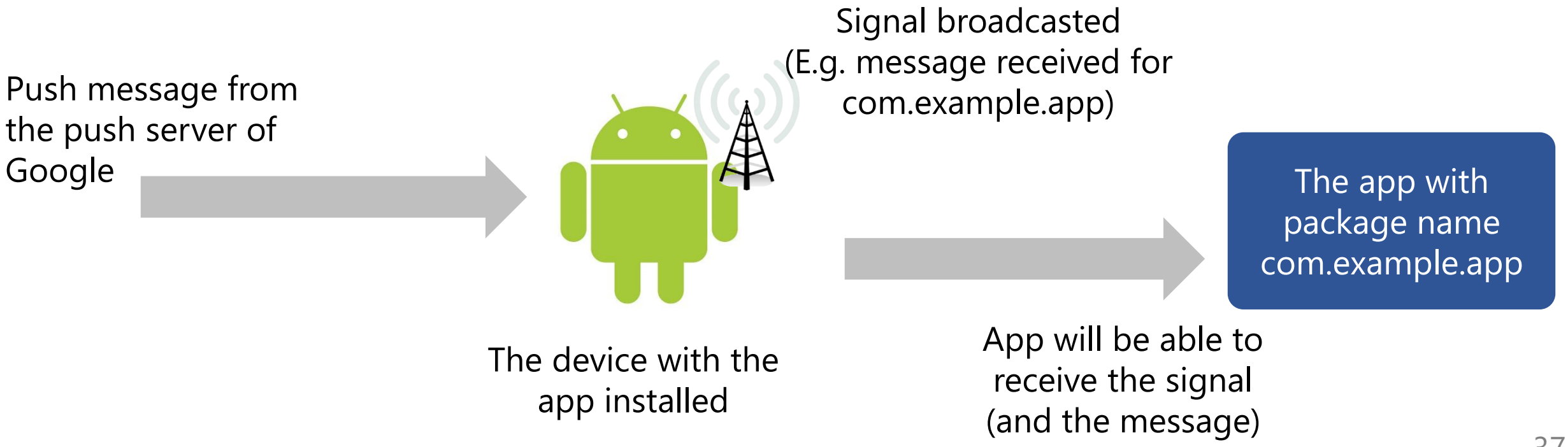
```
public class RegistrationIntentService extends IntentService {
    @Override
    public void onHandleIntent(Intent intent) {
        // ...
        InstanceID instanceID = InstanceID.getInstance(this);
        String token = instanceID.getToken(getString(R.string.gcm_defaultSenderId),
                GoogleCloudMessaging.INSTANCE_ID_SCOPE, null);
        // ...
    }
}
```

Setup in the App Project (4)

Once you received the token, you should send it to your server application and let it store it (using HTTP request to an API you implemented in your server)

Implementing GCM Client (4)

- When an Android device receives a push notification, it will broadcast a signal to all apps installed in the device
- If you app has registered for that signal, and if the package name matches, your app will receive that broadcast



Implementing GCM Client (4)

- You will need to implement a service that extends GcmListenerService to handle messages sent from the server

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

    // ...
}
```

- From here, you can then create a notification to notify the user, or further download new data from the server to update the status of the app

Implementing GCM Client

- You can find a complete example here: https://github.com/googlesamples/google-services/tree/master/android/gcm

Other Similar Solutions

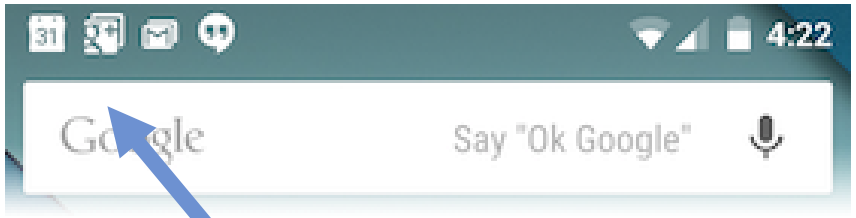
Other third-party cloud messaging SDKs and APIs

- Xiaomi
  http://dev.xiaomi.com/doc/?page_id=1670
- QCloud (by Tencent)
  http://www.qcloud.com/product/dove.html
- Amazon's SNS (Simple Notification Service)
  http://aws.amazon.com/sns/

Notifications in Android

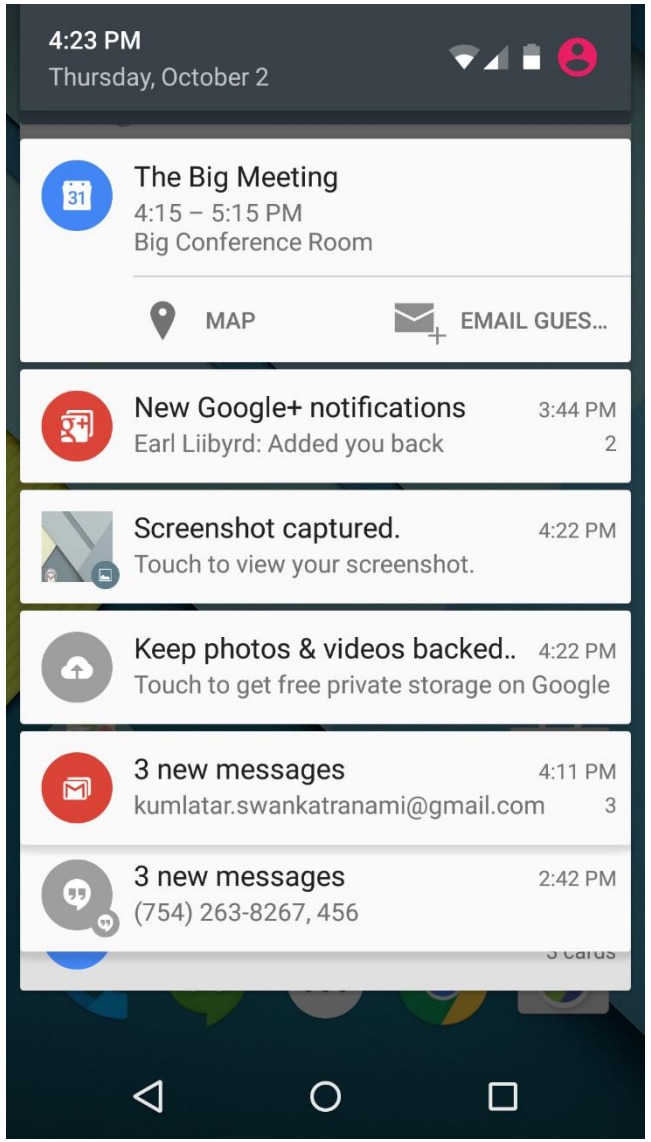
Notifications

Notifications are ways to notify the user in the notification bar about updates of your app



Notification Area

Notification Drawer



Notifications

A notification in Android contains a few important components

- An icon (usually the app icon, but can be customise for each notification)
- A title
- A message or short description
- Timestamp (automatically generated)

For notifications in Android 5.0, see
http://developer.android.com/design/patterns/notifications.html

Creating Notifications

The following block of codes can be used to create a notification

```
NotificationCompat.Builder mBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");

...

NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

mNotificationManager.notify(mId, mBuilder.build());
```

Notification Actions

You can also control what happens when the user clicks on the notification

```
NotificationCompat.Builder mBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");

Intent resultIntent = new Intent(this, ResultActivity.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(ResultActivity.class);
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
        stackBuilder.getPendingIntent(
            0,
            PendingIntent.FLAG_UPDATE_CURRENT
        );

mBuilder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

mNotificationManager.notify(mId, mBuilder.build());
```

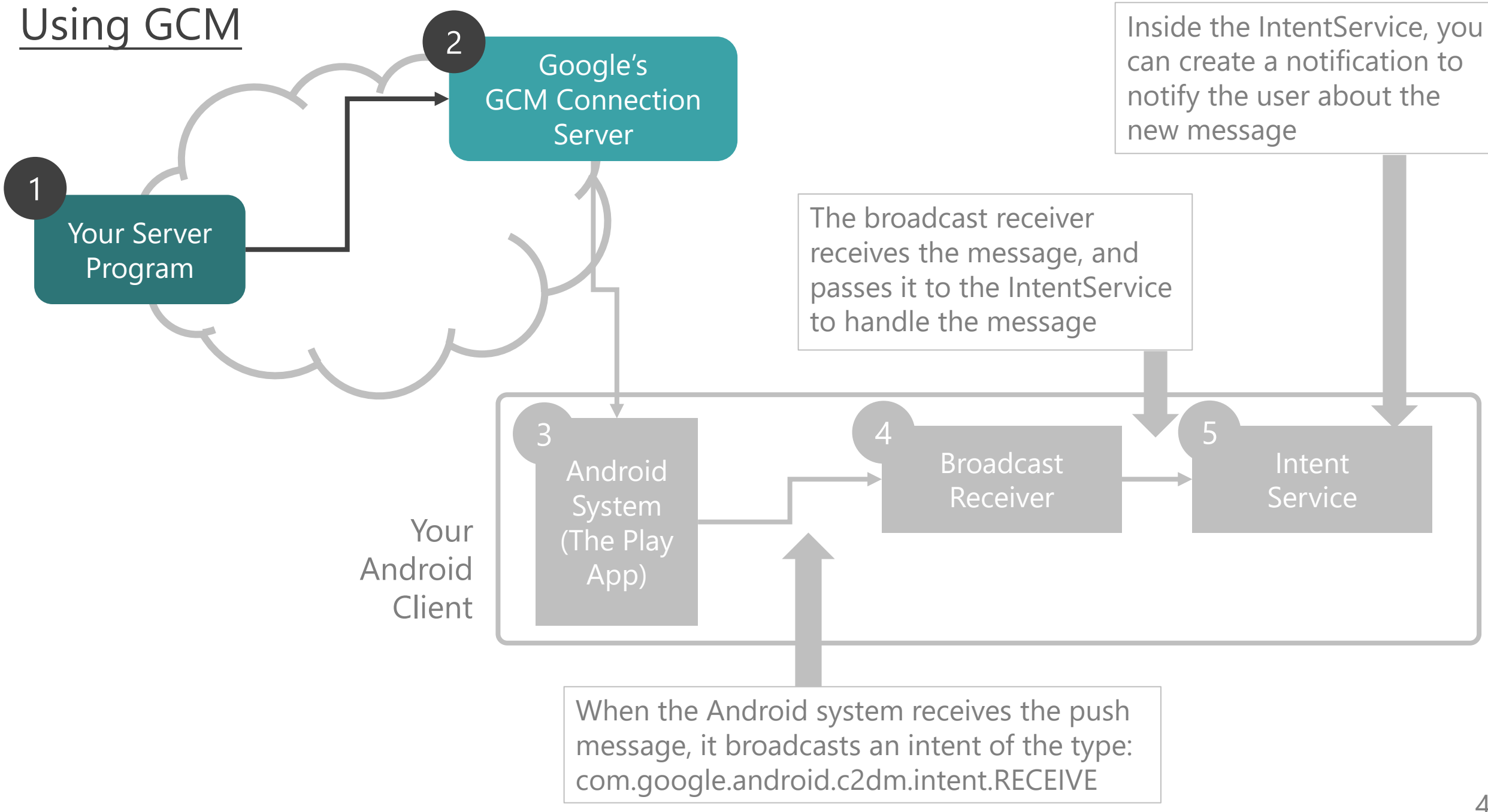
Notification Actions

Notifications can be accompanied by sound, vibration or LED flash (if supported by the device)

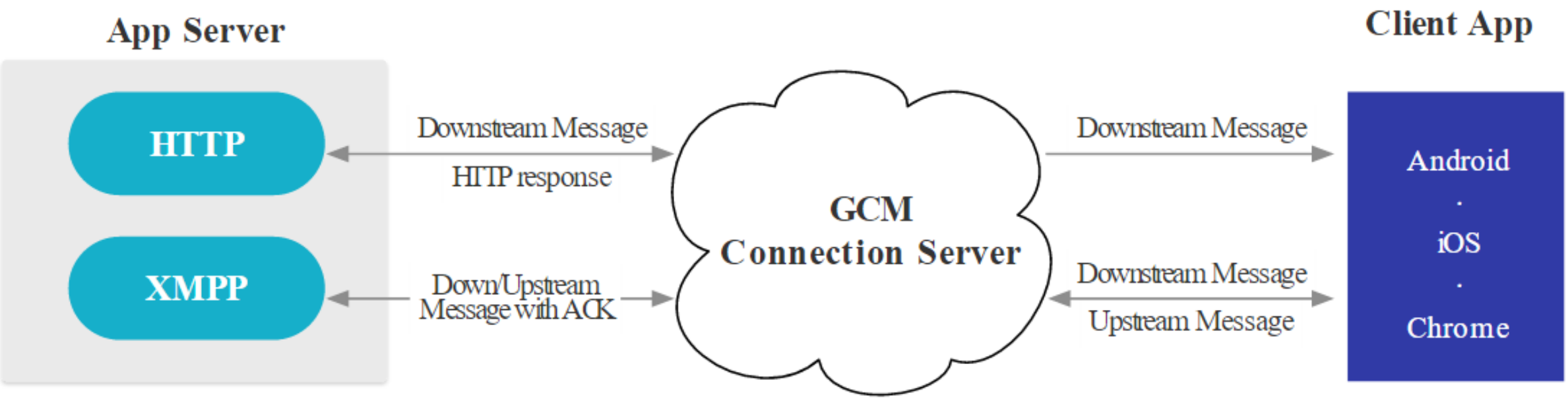
```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
builder.setLights(Color.BLUE, 1000, 300);


Notification notification = builder.build();
notification.defaults = Notification.DEFAULT_SOUND | Notification.DEFAULT_VIBRATE;
```

Sending GCM from Your Server

Using GCM


1
Your Server Program
2
Google's GCM Connection Server
Inside the IntentService, you can create a notification to notify the user about the new message
The broadcast receiver receives the message, and passes it to the IntentService to handle the message
Your Android Client
3
Android System (The Play App)
4
Broadcast Receiver
5
Intent Service
When the Android system receives the push message, it broadcasts an intent of the type: com.google.android.c2dm.intent.RECEIVE

Using GCM

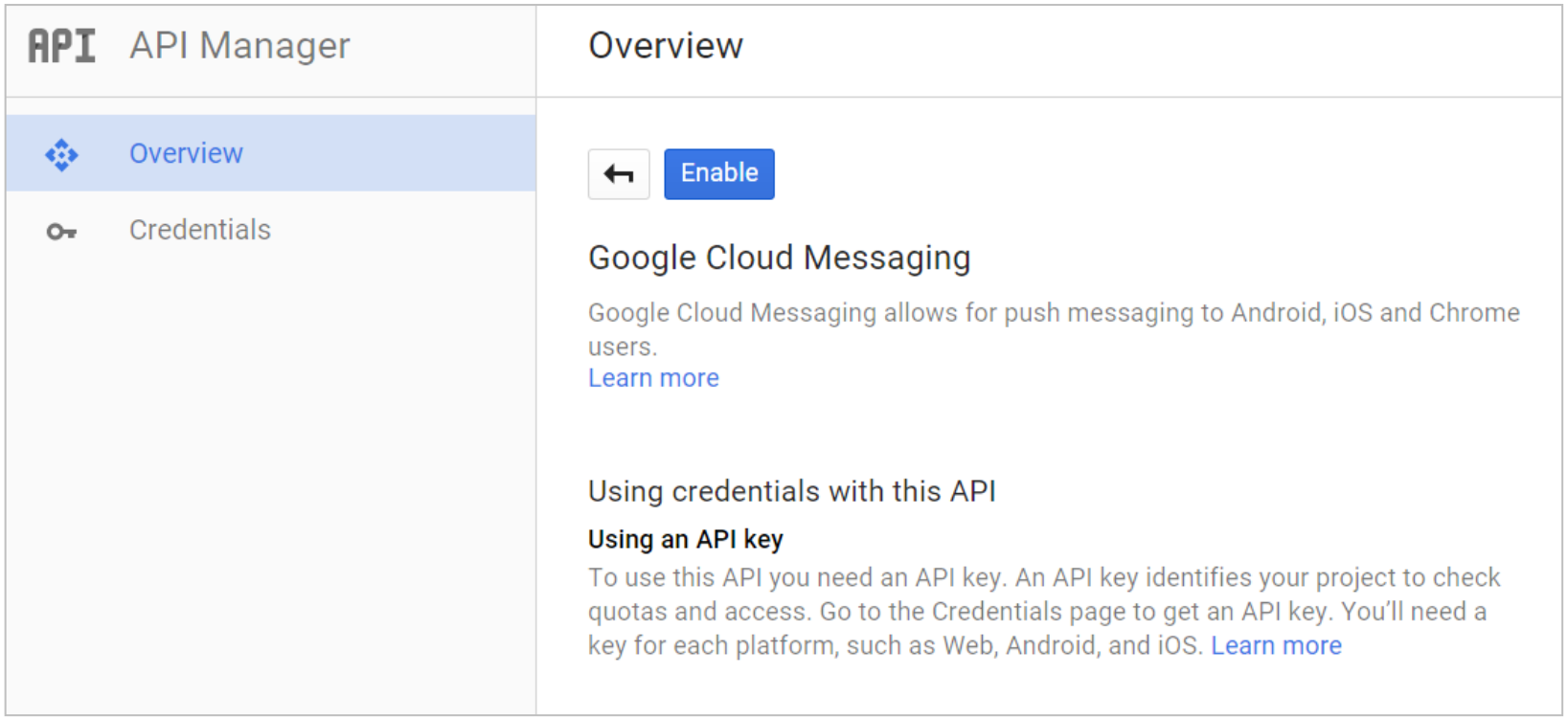


Reference: https://developers.google.com/cloud-messaging/gcm

Enabling GCM API

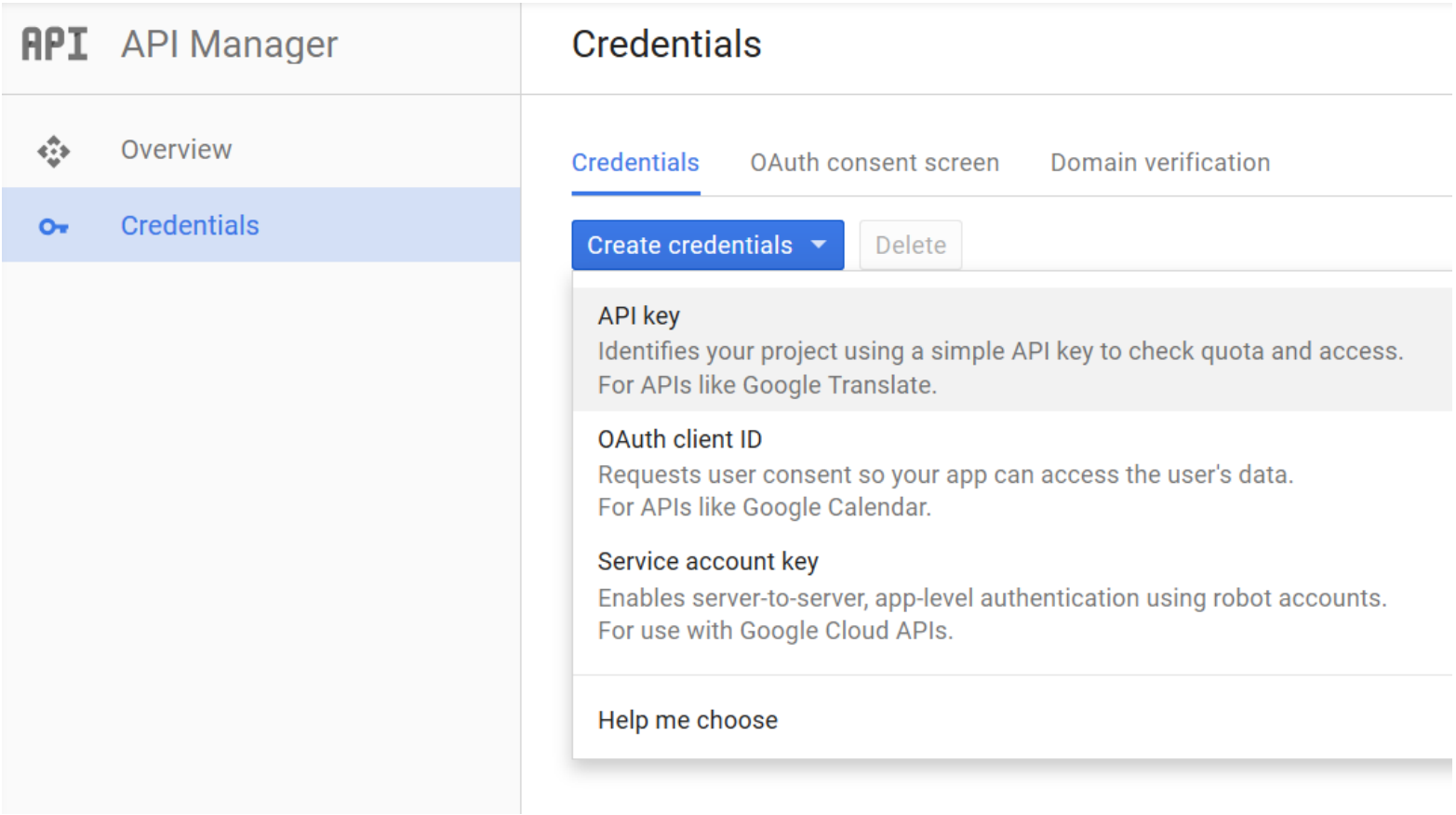
Once you have got a Google account, go to the Google Developer Console
https://console.developers.google.com/

- Create a new project by providing a project name
- Go to the menu on the left, select API Manager
- Search for Google Cloud Messaging, and enable this API



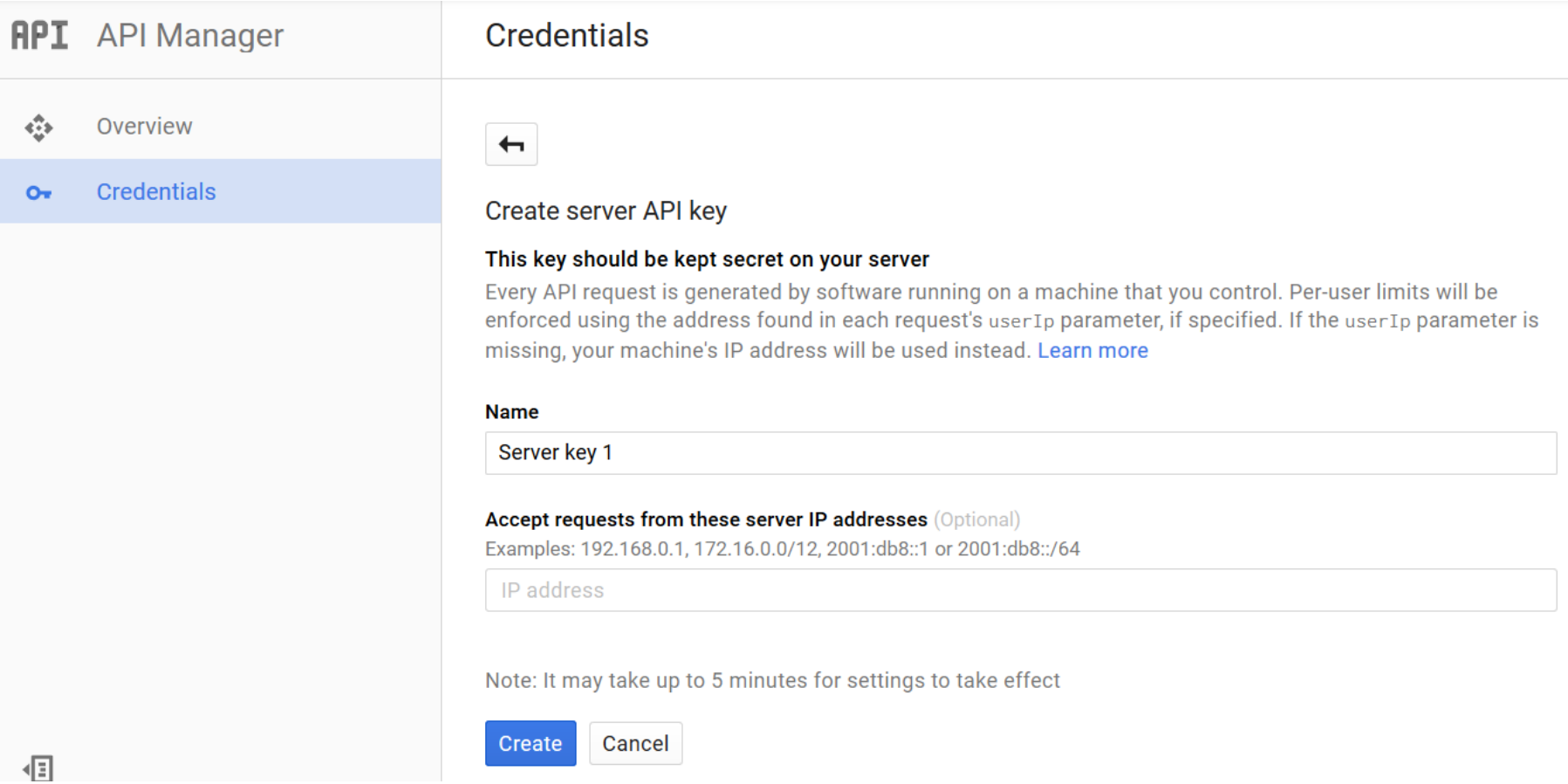
Setting up a Project in Google

- Skip this step if you already have the configuration file done
- Go to ‘Credentials’, and create an “API Key” (Server Key)



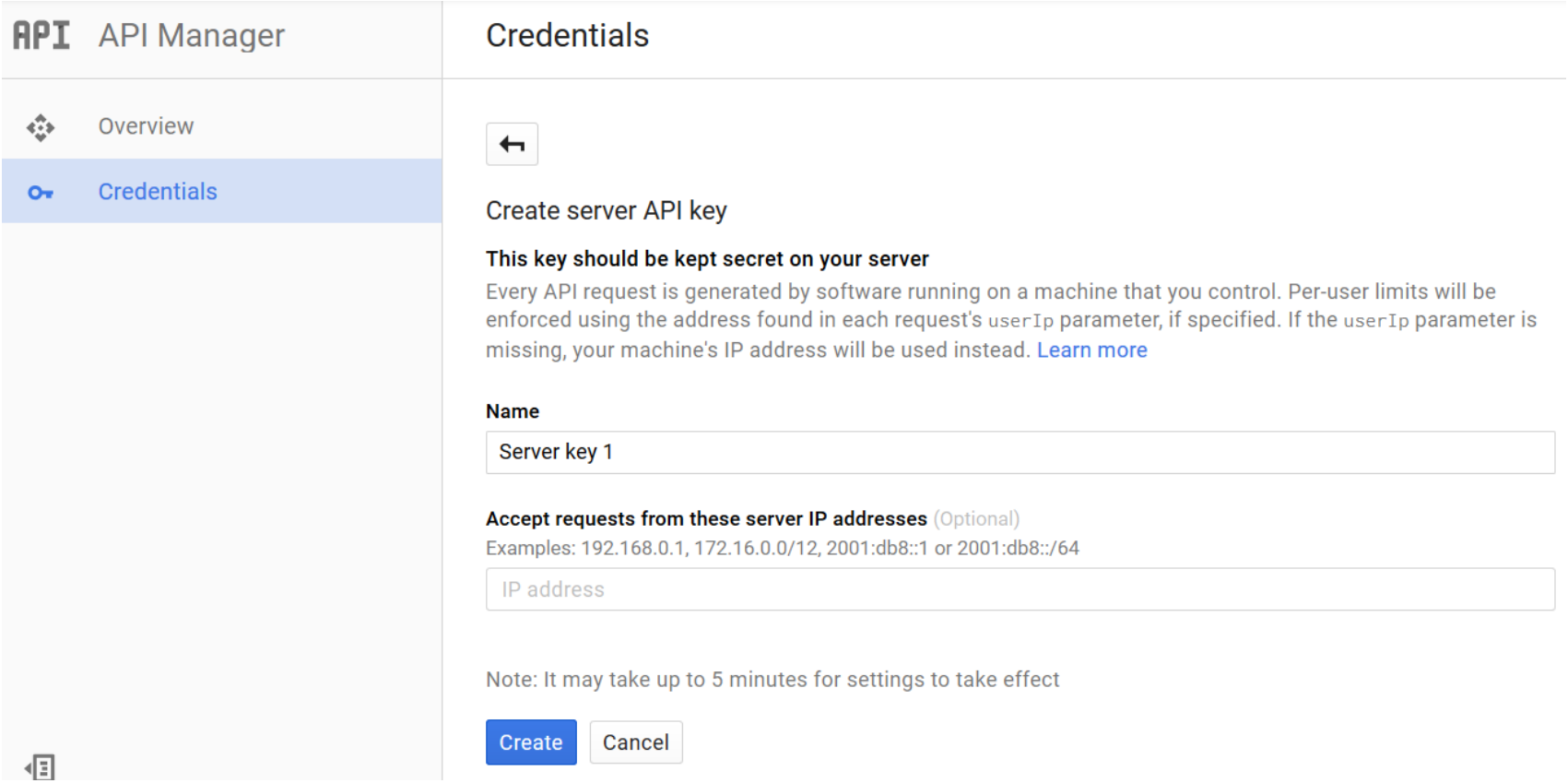
Setting up a Project in Google

- Give the key a name, and input the server IP address (the server from which you will send out GCM, optional)



Setting up a Project in Google

- Give the key a name, and input the server IP address (the server from which you will send out GCM, optional)
- Take note of the key generated for you



GCM – The Mechanism

- With the key generated, you can send out “downstream messages” to clients (Android, iOS or Chrome) from your server
- This is done by sending a POST HTTP request to Google’s GCM HTTP Connection Server, which will handle the queueing and delivery of your message
- URL of the API: https://gcm-http.googleapis.com/gcm/send

GCM – The Mechanism

- An example of a request sent to the GCM HTTP Connection Server at https://gcm-http.googleapis.com/gcm/send

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

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

- You can either construct and submit HTTP requests by yourself, or use a the Python GCM module

Using the Python GCM Module

- In your Python server program, you can make use of the requests module to help you send HTTP requests
- http://docs.python-requests.org/en/master/

```
$ sudo pip install requests
```

- A simple HTTP library for Python
- Example:

```
>>> import requests
>>> r = requests.get(“http://www.cuhk.edu.hk/”)
>>> content = r.text
>>> ...
```

Using the Python GCM Module

- Sending a post request to the GCM server

```
import requests

api_key = ‘AIzaSyZ-1u...0GBYzPu7Udno5aA’
url = ‘https://gcm-http.googleapis.com/gcm/send’

headers = { ‘Authorization': ‘key=’ + api_key }

client_token = ‘bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...’
payload = { ‘to’: client_token, ‘data’: {...} }

r = requests.get(url, headers=headers, data=payload)

if r.status_code == 200:
    print “Request sent to GCM server successfully!”
```

Python GCM Module

Instead of constructing the HTTP request by yourself, you can send GCM request more conveniently using the python-gcm module

- https://github.com/geeknam/python-gcm
- Install the module by the following command:

```
pip install python-gcm
```

Python GCM Module

An example of using the Python GCM module

```
from gcm import GCM

gcm = GCM(API_KEY)

# Downstream message using JSON request
data = {'param1': 'value1', 'param2': 'value2'}

# Specify one or more tokens (one for each client)
reg_ids = ['token1', 'token2', 'token3']

response = gcm.json_request(registration_ids=reg_ids, data=data)
```

Project Arrangement

Project Arrangement

- By this time, you should have formed your project group
- Next, think about the topic of your project, and what you plan to do
- **Some guidelines**
  - Instead of functions, think about what problems do you want to solve
  - Your app should make use of networking functions (e.g. send and receive data, files, broadcasting or streaming, multi-player games, local area connectivity, etc.)
  - A 3-student group should do more work than a 2-student group

Project Arrangement

You should now prepare a simple **document** with the following content

- What problem(s) is your app going to solve?
- What functions or features will be found in your app
- What are the tasks required on 1) the client side, and 2) on the server side
- What difficulties do you foresee?
- What kind of help do you need?

(No specific format required)

Project Arrangement

Project Consultation

- Refer to the following link for the time of consultation of your group https://docs.google.com/spreadsheets/d/1jAhNrbC_qlPqwF-n4QqHDxhxHc52NQ_OPBRG1fkO7to/edit?usp=sharing
- Send me a email with your group ID to confirm that you can attend the consultation, or ask for a reschedule if you really cannot make it
- You should bring along the document you prepared to the consultation

Project Arrangement

Assessment Criteria (Tentative)

- Networking functionality (30%)
- Client side system design and complexity (20%)
- Server side system design and complexity (20%)
- Error handling and robustness (10%)
- User-friendliness & UI/UX design (10%)
- Presentation & report (10%)

End of Lecture 7