IEMS 5722
Mobile Network Programming and Distributed Server Architecture
2015-2016 Semester 2

Assignment 4: Using Google Cloud Messaging and Notifications

Due Date: 8th April, 2016

Notes:

i.) Read carefully the instructions and requirements to understand what you have to do

ii.) Follow the instructions in Section 4 to submit your files for marking

iii.) Late submissions will receive 30% mark penalty

1. Objectives

- To learn how to implement **push messages** in Android using Google Cloud Messaging
- To learn how to use **asynchronous tasks** and **message queues** on the server side

2. Instructions

In this assignment, you will further develop your mobile app and your server application, such that the server application will be able to push new messages to the Android app using **Google Cloud Messaging (GCM)**. Specifically, you will have to perform these tasks in this assignment:

- Register for a Google account, create a project, and set up Google Cloud Messaging API
- Extend your Android app to enable GCM, including getting a registration token from GCM service, submitting the token to the server application, and setting up the app to be able to receive GCM messages
- Extend your server application such that when a new message is submitted to a chat room, all other users (devices) should be notified by using GCM messages

2.1. Preparations

You should register for a Google account, create a project, and enable the Google Cloud Messaging API. You can follow what we discussed in Lecture 7. A detailed guideline can be found at this website: https://developers.google.com/cloud-messaging/android/start.

Once you have finished this step, you should have the following information at hand:

- A **google-services.json** file (to be added to your project)
- A **Sender ID** (to be hardcoded into your app)
- An **API Key** (to be used in your service application)

2.2. Extending the Android App

In order to receive GCM messages, you have to extend your existing app. A detailed guide can be found here: https://developers.google.com/cloud-messaging/android/client. In summary, you will have to complete these steps:

- Add “google-services” dependency in your app’s **`build.gradle`** file
- Edit the application’s **manifest** file
- Add code to check for Google Play Services in the main activity’s **`onCreate`** and **`onResume`** methods
- Create an **`IntentService`** that will query Google to obtain a token for GCM
- Add code to report the received token to your own server application
- Create a class that extends **`GcmListenerService`** to handle GCM messages pushed from the server
- Generate a **notification** when a push message is received. Use the **chat room name** as the title of the notification, and put the **message** in the short description field.

You can refer to the sample app at the following URL:
https://github.com/googlesamples/google-services/tree/master/android/gcm/app/src/main/java/gcm/play/android/samples/com/gcmquickstart

2.3. Extending the Server Application

Next, you should extend your server such that it can send out push notifications by sending requests to Google’s GCM server when necessary.

2.3.1. Storing the Token Submitted by the Client

Firstly, you should create a new table in the database, which will be used to store the tokens submitted by the app. You can create a simple table with the following schema, and insert a new record whenever a client has submitted a token.

```
CREATE TABLE push_tokens (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `user_id` int(11) NOT NULL,
    `token` VARCHAR(100) NOT NULL,
    PRIMARY KEY (`id`)
) DEFAULT CHARSET=utf8;
```

Next, you should create a new API to allow the client to report the token. Create an API as specified below.

`POST: /iems5722/submit_push_token`
Input parameters: `user_id, token`
Sample output : `{“status”: “OK”}`

2.3.2. Generating Push Messages Asynchronously

We discussed about asynchronous tasks and message queues in Lecture 9. In this assignment, you will use these to handle push notifications. This is because sending request to Google's GCM server may take some time, and therefore the operation should not be carried out within the HTTP request-response cycle.

In this assignment, you will have to extend your server-side system in a way illustrated in the following diagram.

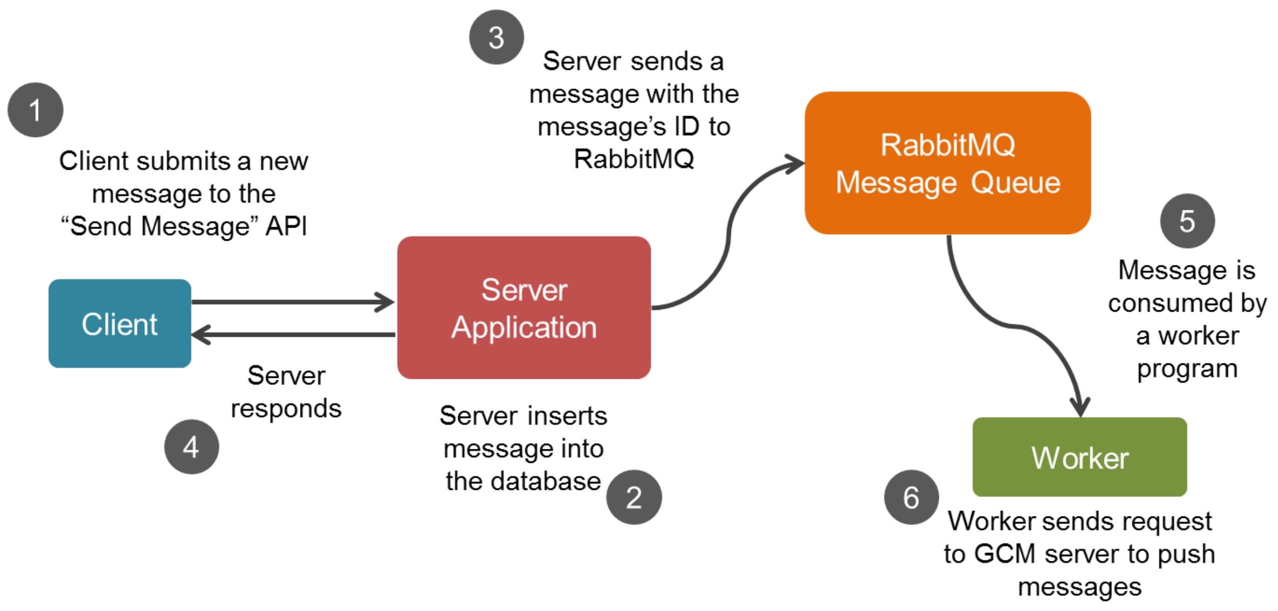
1
Client submits a new message to the "Send Message" API
Client
Server responds
4
Server Application
Server inserts message into the database
2
3
Server sends a message with the message's ID to RabbitMQ
RabbitMQ Message Queue
5
Message is consumed by a worker program
Worker
6
Worker sends request to GCM server to push messages

Figure: System Architecture

In order to extend the system, you should first install a few software packages. Firstly, follow the instructions in the section "Using rabbitmq.com APT Repository" at http://www.rabbitmq.com/install-debian.html to install RabbitMQ server.

Next, you will need to install the pika Python library to submit messages to the message queues in RabbitMQ. Install this by using the following command:

```
$ sudo pip install pika
```

2.3.3. Modifying the server application

In your server application, you should have a function that is associated with the `/iems5722/send_message` API. In this function, you should add some codes to submit a message to the message queue in RabbitMQ. Specification, you should:

- Create a dictionary object in Python with the following two fields: "chatroom" and "message", the values of these two fields should be the name of the chat room to which the message is sent, and the message submitted by the user respectively

- Before your return the response to the client, submit the JSON representation of the dictionary object (convert it to string using **`json.dumps`**) as a message to a message queue named "**chat**"

For example, below is how you can convert a Python dictionary into a JSON string.

```
import json
data = {
    "chatroom": "General Chat Room",
    "message": "Hello from Albert"
}
json_string = json.dumps(data)
```

Once you have updated your server application, you can try to test whether the /iems5722/send_message API works as before. If you have submitted some messages, you can us the following command to check if some messages have been submitted to the message queue:

```
$ sudo rabbitmqctl list_queues
```

You should be able to the number of messages submitted to the queue so far.

2.3.4. Creating workers to consume messages

Next, you should create a worker program to consume messages in the message queue. Specifically, the worker should do the following:

- Implement a callback function that receive and process the message
- Extract the chatroom name and message from the JSON string in the message received
- Go into the database and retrieve all the tokens submitted by the clients
- Generate a request to Google's GCM server.

To implement this worker program, follow what we have discussed in Lecture 9, and the tutorial in the following link: http://www.rabbitmq.com/tutorials/tutorial-one-python.html.

In order to send requests to Google's GCM server, you can either use the **`requests`** module to generate HTTP requests by yourself, or you can use the **`python-gcm`** library to help you to do so. Refer to the guidelines in Lecture 7 for more details.

Once you have finished implementing the worker program, you can execute it and see if it has consumed the messages in RabbitMQ. To deploy it, you can use Supervisor. This is done by creating a configuration file and placing the file under **`/etc/supervisor/conf.d`**.

3. Requirements

You should have implemented the following features in this assignment:

- Extends the server application's `/iems5722/send_message` API to **submit a message to the message queue** when a user submits a new message to a chat room
- Creates a **worker program** using Python to consume the messages in the message queue, and send requests to Google's GCM server to generate push messages
- Extends your **Android app** and makes it able to **receive GCM messages**.
- Upon receiving a GCM message, the Android app should **create a notification** to notify the user of the new message received.

4. Submission

To facilitate marking of the assignment, you should strictly follow the instructions below. To submit your assignment, create a folder named **<your_student_id>_assgn4**. In the folder, you should include the following items:

- All source codes of your **Python server application**
- All source codes of your **worker program** for consuming messages from the message queue
- The **app/src/main** folder of your **Android app**

Compress this folder using ZIP, you should now have a file named **<your_student_id>_assgn4.zip**. Submit it in the CUHK eLearning System online: https://elearn.cuhk.edu.hk/.

5. Marking Criteria (Total 100 Marks)

- **(20 marks)** Correctly submitting message to the message queue in RabbitMQ (in the server application)
- **(20 marks)** Correctly implementing the worker program to consume message from the message queue
- **(20 marks)** In the worker program, tokens are retrieved from the database, and a request to Google's GCM server is generated to push the message to the devices with these tokens
- **(20 marks)** Correctly extending the Android app to receive GCM message
- **(20 marks)** Correctly displaying a notification in the Android app when a GCM message is received