CLOUD TO DEVICE MESSAGING SERVICE (C2DM CLONE FOR WINDOWS OS)

Team Members:

Name	*RollNumber*	*EmailAddress*	*Contact*
Akanksha Upadhyay	MT2011007	akanksha.upadhyay@iiitb.org	8147449493
Suchitra Madathil	MT2011074	madathil.suchitra@iiitb.org	7760896127
Nikhil Dhawan	MT2011090	nikhil.dhawan@iiitb.org	7259197092
Raj Patel	MT2011100	rajkumar.patel@iiitb.org	9916624384
Pragya Tomar	MT2011105	pragyasingh.tomar@iiitb.org	9742370996

Team Leader: Raj Patel

Contents

1 Introduction

1.1 Background

This project deals with Cloud to Device Messaging(C2DM) on Windows mobile device. In the mobile domain it is known as *push notifications*.

Figure 1: Cloud to Device Push Notification

When applications are getting data from the internet, there are two ways to keep the data fresh. First is called polling. It means that the application initiates a connection to the server at set intervals to check if new data is available. The problem is determining the polling frequency. The alternative is called *pushing, or push notifications* [1]. Instead of the client initiating a connection, the server sends a data packet to the client whenever there is new data, then the client reacts to it. There is some overhead to maintain a persistent connection to the server, rolling out a push notification service using Cloud to device messaging. Note that the server doesn't actually send the new data, it only sends an alert that new data is available. This keeps the ping packet small so there is less overhead. Its up to the client application to receive the notification and then go to the server to retrieve the fresh data.

It requires a service named Windows Cloud to Device Messaging Framework which allows developers to send data to their applications running on Windows devices. Unlike most solutions which involve polling some server, this service is a *push* service (similar to SMS messages).

Developers should implement three components in order to use this service [2].

Client code: The client code which is responsible for registering the specific device and application with server's service and is responsible for handling new messages that arrive.
Web server: When the client device registers with server's service it receives a registration ID that allows the account holder to send messages to your application running on the user device. The device needs to send this ID to a web server that you can access.
Messages sending tool: This is a command line tool that is responsible for fetching the user ID from the web server and then sending the message to the device. This is done by sending a request to servers.

1.2 Problem Description

To create a clone of Android's C2DM service on mobile platform other than Android. We have chosen to develop basic version in Windows platform. Most mobile apps require data from the Internet. One approach for updating its data is that the applications periodically polls a server for new data (polling). If no new data is available this approach uses unnecessary network bandwidth and consumes the battery of the mobile phone. Even a single application polling every 5 minutes can cause a substantial drain on the battery - mostly wasted on checking for changes that don't exist. C2DM uses an alternative approach, that the server contacts the mobile application once new data is available (push). If the data does not change constantly, push is the preferred solution.

Since currently applications on other platforms rely on polling or individual persistent connections, a service similar to C2DM would reduce the most constrained resources on mobile phone i.e. the bandwidth and battery. This is implemented as a background service. It starts whenever the network is available, and maintains a connection with the server for new messages. To use it, an application on the Windows device registers with server, and gets back a registration ID [3] . It then sends that ID to the web application. To send a message to the device, the web application sends an authenticated message to the Windows push Application Programming Interface(API), which takes care of delivering it to the device. The push messages themselves consist mostly of a notification of new data, so it's up to the application to actually fetch the new data directly from the web application.

2 Gap Analysis

As of Android 2.2 it is possible to push information to an Android app. This service is called *Cloud to Device messaging* or in short C2DM. In a C2DM we have three involved parties. The Google's C2DM servers, application server to push messages to the C2DM servers and the Android application. The program on the application server can be written in any server side programming language, e.g. Java, PHP, Python, etc. When the application server needs to push a message to the Android application, it sends the message via an HTTP POST to Google's C2DM servers. The C2DM servers route the message to the device. If the device is not online, the message will be delivered once the device is available. Once the message is received, a Broadcast Intent is created. The mobile application has registered an Intent Receiver for this Broadcast. The application is started and processes the message via the defined Intent Receiver.

As far as C2DM in windows is concerned, there has not been any improvements in implementatios of this service. There is not much development on windows phone to use this facility. In order to create clone of this service, we need to generate messages from application server to Cloud servers and on Cloud server we need to create notifications to notify devices that data is available. Depending upon type of information, this may be image, plain data or formatted data.

Android maintains a persistent connection to the Android Marketplace. C2DM uses this existing connections to the Google servers. This connection is highly optimized to minimize bandwidth and battery consumption. C2DM is currently still in beta. C2DM applies a limit of approximately 200000 messages per sender per day and is currently free of charge. To create such common connection for windows platform, we will create a local server on one machine that will act as Cloud environment. C2DM messages are limited in size to 1024 bytes and are intended to inform the device about new data and not to transfer it. For our project, bandwidth limit and message size that can be pushed will depend upon the network capacity where the systems are connected. The typical workflow is that Google's C2DM servers notify the Android application that new data is available. Afterwards the Android application fetches the data from a different server.

3 System Architecture

3.1 Architecture

This table summarizes the key terms and concepts involved in our project. It is divided into these categories:

Components: The physical entities that play a role in our project.

1. Mobile Device: The device that is running a Windows application of our project.
2. Third-Party Application Server: An application server that developers set up as part of implementing our project in their applications. The third-party application server sends data to Windows application on the device via our project server.
3. Our project Servers: Our project servers involved in taking messages from the third-party application server and sending them to the device.

Components: The IDs and tokens that are used in different stages of our project to ensure that all parties have been authenticated, and that the message is going to the correct place.

1. Sender ID: The device that is running a Windows application of our project.
2. Application ID: An application server that developers set up as part of implementing our project in their applications is uniquely identified by this ID.
3. Registration ID: Our project servers involved in taking messages from the third-party application server and sending them to the device is uniquely identified by this ID.
4. Sender Auth Token: A ClientLogin Auth token [4] that is saved on the third-party application server that gives the application server authorized access to Google services. The token is included in the header of *post* requests that send messages.

Architecture is explained for the corresponding diagram attached

Step 1: Mobile device has to register with C2DM service with user's email address and get an application ID/device ID per application/device.

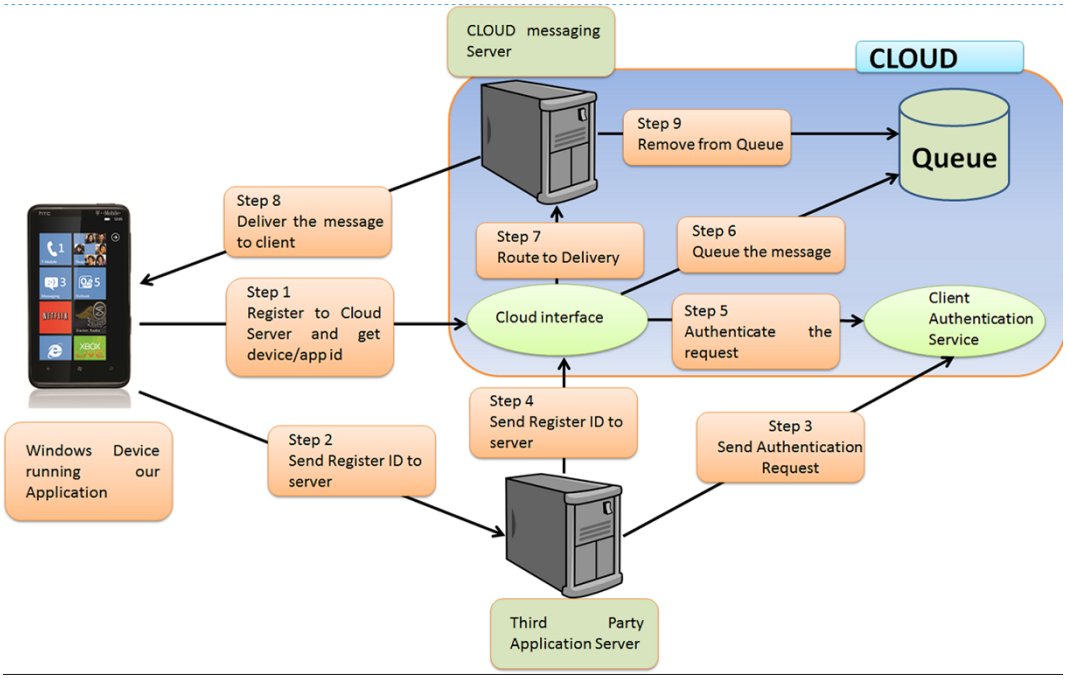

Figure 2: System Architecture

Step 2: Mobile device will send registration ID to third party application server.
Step 3: Application server sends authentication request with email address and password to Cloud server and receives authentication token.
Step 4: Application server posts messages to device with registration ID and authentication token.
Step 5: Cloud interface authenticate the request in authenticate service of Cloud.
Step 6: Cloud is one platform where many application developers can send their messages. So when a message is received, it is redirected to the queue of Cloud.
Step 7: Interface service will find the route of the client mobile device using client ID.
Step 8: Cloud server push the message to client based on the route found in previous step.
Step 9: After successfully sending the message to client,Cloud server will remove the message from the queue.

3.2 System Requirements

1. Hardware Requirements
    1. Mobile Phone with Windows OS.
    2. Two Laptops, one for Cloud server and other for application server.
2. Software Requirements
    1. Windows OS
    2. JDK 6 with Eclipse IDE for development.
    3. MySQL database system.
    4. Microsoft Visual Studio 2010.
    5. Microsoft Visual ServicePack1.
    6. Windows Mobile SDK.
    7. Windows Phone Emulator.
    8. CloudSim

4 Development Plan

4.1 Development Stages

Stage 1: Study existing product.

1. Working of Android applications.
2. Study of the open source code for Android C2DM and understand how it works.
3. Understand how message and push notification happens in C2DM messaging for an Android mobile device.

Deliverable:- Analysis of Android deployment of C2DM and identification of requirements for Windows clone.

Stage 2: Develop Windows mobile application.

1. Using Microsoft Visual Studio 2010 and Windows Mobile SDK which is a Microsoft product, we setup development environment for Windows mobile 7 application.
2. Create Windows application which supports push notification.Windows mobile 7 application will be developed using CSharp.

Deliverable:- Windows application which supports push notification.

Stage 3: Setup application server

1. Setup development architecture using Java and Eclipse IDE.
2. Create application which connect Windows mobile device to the application server and test application.
Deliverable:- Messaging from application server to mobile device.

Stage 4: Setup virtual Cloud and server on that.

1. Using CloudSim which is an open-source Cloud computing frame-work, we setup our Cloud.
2. Setup server in Cloud environment using Java.
3. Test connectivity between application server and Cloud server.
Deliverable:- Messaging from application server to Cloud server.

Stage 5: Test complete application.

1. Enabling Cloud: Need to register to send or receive messages.
2. Sending a message: Our application server sends messages to device.
3. Receiving a message: Windows application receives message from main Cloud.
Deliverable:- Final Application.

4.2 Working

Processes involved are:

Enabling c2dm: Need to register inorder to send or receive messages.
Sending a message: Our application server sends messages to device.
Receiving a message: Android application receives message from Cloud server.

Enabling C2DM

These are set of steps that occurs when an application running on a mobile device registers to receive messages:
1. The first time the application needs to use the messaging service, it triggers a registration Intent to a C2DM server.
This registration Intent (com.google.android.c2dm.intent.REGISTER) [4] includes the sender ID (to authorize to send messages to the application), email address, and the application ID.
2. If the registration is successful, the C2DM server broadcasts a REGISTRATION Intent which gives the application a registration ID.
3. To complete the registration, the application sends the registration ID to the application server. The application server typically stores the registration ID in a

database.
The registration ID stays until the application by itself unregisters or until Google refreshes the registration ID for your application.

Sending a Message

Application server needs following to send a message:
1. The application has a registration ID that allows it to receive messages for a particular device.
2. The third-party application server has stored the registration ID.
The ClientLogin token authorizes the application server to send messages to a particular mobile application. An application server has one ClientLogin token for a particular third party application, and multiple registration IDs. Each registration ID represents a particular device that has registered to use the messaging service for a particular third party application.

Here is the sequence of events that occurs when the application server sends a message:
1. The application server sends a message to C2DM server.
2. Google enqueues and stores the message in case the device is inactive.
3. When the device is online, Google sends the message to the device.
4. On the device, the system broadcasts the message to the specified application via Intent broadcast with proper permissions, so that only the targeted application gets the message. This wakes the application up. The application does not need to be running beforehand to receive the message.
5. The application processes the message.
An application can unregister C2DM if it no longer wants to receive messages.

Receiving a Message

This is the set of events that occurs when an application running on a mobile device receives a message:
1.The system receives the incoming message and extracts the raw key/value pairs from the message payload.
2.The system passes the key/value pairs to the targeted application in a com.google.android.c2dm.intent.RECEIVE [4] Intent as a set of extras.
3.The Android application extracts the raw data from the RECEIVE Intent by key and processes the data.

5 Timeline

Milestone	Estimated Completion	Date
Phase 1	Initial project goals document draft and choosing team leader.	17-Jan-2012
Phase 2	Address review comments and finalize project goals document. Setting up of SVN repositories.	25-Jan-2012
Phase 3	Analyze the installation and working of C2DM on Android. Brief presentation of the project architecture and plans. Corresponds to Development Plan Stage 1	27-Jan-2012
Phase 4	Messaging from application server to mobile device in Windows platform. Corresponds to Development Plan Stage 2.	7-Feb-2012
Phase 5	Set a virtual Cloud environment and server on that. Corresponds to Development Plan Stage 3.	1-Mar-2012
Phase 6	Messaging from application server to Cloud server. Corresponds to Development Plan Stage 4.	25-Mar-2012
Phase 7	Testing and Review. Corresponds to Development Plan Stage 5.	9-Apr-2012
Phase 8	Final Release. Submission of final documents and reports.	21-Apr-2012

References

[1] Sachin.Agarwal, "Toward a push-scalable global internet," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, april 2011, pp. 786 –791.

[2] Erel.Uziel, "Anywhere software," http://www.basic4ppc.com/forum/basic4android-getting-started-tutorials/10542-android-push-notification-c2dm-framework-tutorial.html, 2005, online accessed 12-Jan-2012.

[3] Isaac.Liu, "Simple android c2dm," http://www.eecs.berkeley.edu/~liuisaac/site/blog/201109/simple-android-c2dm.html, 2011, online accessed 08-Jan-2012.

[4] Google.Code, "Android cloud to device messaging framework," code.google.com/android/c2dm/, 2011, online accessed 10-Jan-2012.