基于 Android C2DM 服务的云端推送研究与实现

邹 海,李 强,邱慧丽
(安徽大学 计算机科学与技术学院,安徽 合肥 230039)

摘 要:在基于 Android 平台上的应用中,保持其 Android 终端与服务器端的数据同步十分关键。传统的信息 pull 方法实现了保持 Android 终端与服务器端数据同步,它需要每个 Android 终端定时的去服务器上扫描,查看服务器是否发布或有更新数据,与最新的数据同步,需要客户端大量的网络流量和手机电量。针对 pull 方法的不足,文中实现了一种基于 Android 的 C2DM 服务来进行消息的云端推送的方法。运用该方法成功地让客户端接收到了服务器端发送过来的数据更新消息,大量地节省了 Android 客户端的网络流量和手机电量。

关键词:Android 平台;C2DM 服务;pull 方法;云端推送
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)07-0029-04

Research and Implementation of Cloud Pushing Based on Android C2DM Service

ZOU Hai, LI Qiang, QIU Hui-li
(School of Computer Science and Technology, Anhui University, Hefei 230039, China)

Abstract: It is very important to keep the data synchronization between the Android terminal and the server in the application which is based on Android platform. The traditional method of pulling implements keeping the data synchronization between the Android terminal and the server. Each Android terminal is needed to scan the server to see whether the data is published or updated by the server and to synchronize data. But it wastes a lot of network traffic and mobile phone battery. Against the traditional method's weekness, the method of cloud pushing based on the Android's C2DM service is implemented in this paper. The message of data updating is received from the server to the client by using this method. It saved much clients' network traffic and mobile phone battery.

Key words: Android platform; C2DM service; pull method; cloud pushing

0 引 言

近年来信息实时性日益受到重视,服务器端的最新信息如何及时地通知到各个 Android 设备终端成为研究的热点。传统的方法有 pull 方法,它是通过每个 Android 终端定时地隔一段时间去服务器上扫描,查看服务器是否发布或有更新数据,如果有数据更新,再将本机上的信息与服务器同步[1~3]。显然这种方法需要 Android 终端定时地不间断地去扫描服务器,查看数据是否更新,会消耗 Android 终端大量的网络流量和电量。

针对传统的 pull 方法存在的不足,文中研究和实现了基于 Google 发布的 C2DM 服务的消息 push 方法,它不需要 Android 终端定时去连接服务器。当服务器端有数据更新或有消息需要通知到各个 Android 客户端时,服务器端首先将消息发送到 Google 的 C2DM 服务器上,然后由 C2DM 服务器将消息 push 到各个 Android 终端,在 Android 终端收到服务器发送过来的消息后,再主动连接服务器,去服务器上同步最新的数据。在文中的最后实现了基于 C2DM 服务的消息的云端推送。

1 Android 平台系统框架及应用程序结构

Android 平台是一个完全开放性的平台,具有良好的开发和调试环境,并且支持各种可扩展的用户体验,提供了非常丰富的图形系统和强大的浏览器[4]。

1.1 Android 平台系统框架

Android 的系统架构一般从上而下分为四层,分别为应用层、应用框架层、系统运行库层、Linux 内核

收稿日期:2011-12-03;**修回日期**:2012-03-07
基金项目:国家科技重大专项资助项目(2008ZX05039-004)
作者简介:邹 海(1969-),男,博士,副教授,硕士研究生导师,主要研究领域有数据挖掘与信息检索、中间件理论与技术、工作流理论与技术、图像处理技术。

层[5~7],支持用户自由开发。

(1) 应用层:主要是用 java 语言编写的可以在 Android 虚拟机上运行的应用程序,这些应用程序也可以被开发人员用其他的程序替换,更加具有灵活性和个性化。如 E-mail 客户端、地图、浏览器等等。

(2) 应用框架层:Google 里核心应用所使用的 API 框架就在这一层。开发人员只要按照这些 API 框架的基本原则,就可以用它来开发自己的应用。

(3) 系统运行库层:该层也就是 C/C++库以及 Android 运行库层,Android 系统需要通过一些 C/C++库来支持在使用 Android 应用框架时所用的各个组件,来提供更好的服务机制。

(4) Linux 内核层:Android 是基于 Linux2.6 内核的,提供了一系列常用的服务,如内存管理、进程管理等等。

1.2 Android 应用程序的结构组成

Android 的每个应用程序都是由四个模块组成的:Activity、Intent、Content Provider、Service[8~11],但这四个模块并不是必须的,一个应用程序可以只有其中的部分模块。

(1) Activity:它是开发应用程序的最基本的模块,称之为"活动"。每一个 Activity 通常都是一个单独的屏幕,它都被实现成一个独立的类,并且都是继承于 Activity 这个基类。这个 Activity 类会显示由几个 Views 控件组成的接口,并对事件做出响应。一般情况下一个应用会包含多个屏幕。例如,一个短消息应用程序会有一个屏幕用于显示联系人的列表,另外一个屏幕用于写短消息,同时还会有用于进行系统设置的屏幕。每个这样的屏幕都是一个 Activity。一个新的屏幕打开后,前一个屏幕将会被暂停,并保存到历史栈中。用户可以返回到历史栈中的前一个屏幕。一个屏幕不再使用时,可以从历史栈中删除它。默认情况下 Android 将会保留从主屏幕到每一个应用的所有的运行屏幕。

(2) Intent:这个类的主要作用是实现 Activity 和 Activity 之间的切换。如果希望对一个外部的事件做出响应,可以使用 Intent Receiver。它在感兴趣的事件发生时,会调用 notificationManager 方法通知给用户,但不能生成新的界面。当一个 Intent Receiver 被触发时,系统会在需要的时候启动该应用。各种应用还可以通过调用 Content.broadcastIntent() 方法将自己的 Intent Receiver 广播给其它应用程序。

(3) Content Provider:这个类的主要作用是实现应用数据在其他应用中的共享,它实现了底层所提供的一组标准的 API,能让其他的应用保存或读取此内容提供的各种数据类型。

(4) Service:主要用来提供服务。它没有用户界面,生命周期很长。例如一个正从播放列表中播放歌曲的播放器。在这个应用中,会有很多个 Activity,让用户可以选择歌曲进行播放。然而,音乐播放这个功能并没有对应的 Activity,因为用户会认为屏幕导航到其它地方时音乐应该还是在播放的。这个例子中,播放器这个 Activity 会使用 Content.startService() 来启动一个 Service,让后台继续保持音乐的播放。同时系统也将保持这个 Service 一直执行,直到结束这个 Service。还可以通过使用 Content.bindService() 方法连接到一个现存的 Service 上。

2 C2DM 服务

目前,大多数的移动应用程序都需要从网上下载或更新数据。其更新数据的一种方法是,应用程序定时地从服务器上获取最新的数据。显然如果这段期间服务器上没有数据更新,那么它就会消耗不必要的网络宽带流量和手机电池等资源,即 pull 方法;另一种方法是,如果服务器上有数据更新,那么将信息推送到移动应用程序上,即 push 方法,如果数据不是经常变化,显然这种方法会更有效。

Google 公司在 Android2.2 版本中发布了"Cloud to Device Messaging"服务,简称"C2DM"。这种服务帮助开发人员从服务器上发送数据到 Android 设备上。它提供了一个简单、轻量级的机制,服务器可以用它来联系移动终端,将所需要获取或更新的数据推送到移动设备上去。

C2DM 服务包括三个部分:推送消息到 Android 设备上的应用服务器,Google 的 C2DM 服务器和 Android 应用程序[12]。在应用服务器上的程序可以用任何语言编写,如 Java,PHP 和 Python 等等。

C2DM 服务实际上就是一种 push 机制,当服务器端有变化(create/update/delete)时,服务器端会立刻通过 push 机制通知客户端,然后客户端会通过系统机制从服务器端获取服务器端的变化,同时也会把客户端的变化发送给服务器端。通过 push 机制,用户可以及时获得服务器端的改动,有更好的用户体验。它是在 XMPP 协议基础上的一套机制、一个标准。

3 基于 C2DM 服务实现云端推送

当开发需要和服务器上的资源进行交互和同步的应用时,需要获取服务器端的最新数据。一般有 pull(拉)方式和 push(推)的方式,但 push 方式只是费客户端的网络流量,而不会费服务器端的,所以 push 方式会更优越于 pull 方式。push 方法可以用基于 C2DM 的云端推送来获取服务器的最新数据。

首先,实现基于 Android C2DM 的云端推送有如下几个要求:

(1)需要 Android2.2 及以上的系统版本;

(2)使用 C2DM 功能的 Android 设备上要设置好 Google 账户;

(3)需要注册使用 C2DM 功能的用户邮箱账号。

实现云端推送的具体流程如图 1 所示:

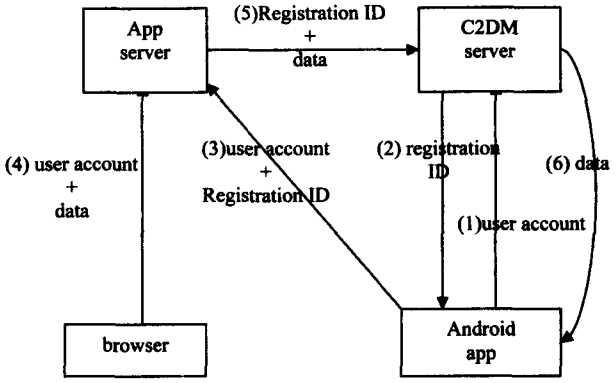

图 1　云端推送的流程

Android app,即 Android 终端,是推送消息的接收端,其上运行各类开发的客户端程序;App server 即第三方服务器,是程序员自己控制的服务器,它是推送消息的发送端,可以利用 C2DM 服务器将消息发送到 Android 终端设备上;C2DM server 是 Google 已经实现好的服务器,它接收第三方服务器发送过来的 id 和 data,并将其发送给各个与 id 对应的 Android 设备,具体步骤如下:

(1)注册:Android 移动设备需要把要使用 C2DM 服务的用户账号和 App 名称发送给 C2DM 服务器,允许接收 C2DM 推送过来的消息。

(2)C2DM 返回给 Android 移动设备一个 registration id,Android 设备需要保存这个 id。

(3)Android 移动设备需要把之前保存的 id 和使用 C2DM 服务的用户账号发送给自己的第三方服务器。

(4)第三方服务器需要获取数据。服务器会接收到浏览器发送的数据,这个数据也可以是本地产生的。服务器要获得 C2DM 服务的用户账号的客户端登录权限 Auth。

(5)服务器把数据和 registration id 放到一起,并且头部带上获取的 Auth,用 post 方式传递给 C2DM 服务器。

(6)C2DM 服务器将客户所需要的数据以 push 方式发送到对应的 Android 移动设备上,这样就成功地将消息基于 C2DM 服务器云端推送到 Android 设备上,Android 设备在程序中按照之前和服务器商量好的格式从相应的 key 中获取数据。

4　云端推送的实现

消息的云端推送实现包括服务器端和 Android 终端实现两部分。

4.1　服务器端的实现

服务器端的主要功能就是通过 C2DM 服务器向客户端发送要推送的数据。为了发送数据,服务器需要向 https://android.apis.google.com/c2dm/send 发送一个 post 请求,这个请求中要包含以下内容:

(1)registration_id:是客户端发送过来的 registration_id 值,必须包含。

(2)collapse_key:一个任意的字符串,用来表示一组相似的消息。设置这个值来避免 Android 终端上线时收到太多已经过时的消息,必须包含。

(3)data.<key>:所要推送的内容,以键值对的方式组织,可选的。

(4)delay_while_idle:若包含这项,说明 Android 终端 idle 时,C2DM 服务不会马上把消息推送到终端,可选的。

(5)Authorization:http 头部所要包含的信息,是为 senderId 申请的 C2DM 服务权限,也必须包含。

服务器端的任务就是构造一个这样的 post 请求,然后向 C2DM 服务器发送,在实现过程中,首先要获取 Auth 的权限 getAuthToken(String url, String params),在获取了 Auth 后,向 C2DM 服务器发送需要 Push 的数据 sendPushMessage(String url,String registration_id, String auth,String collapse_key,Map<String,String>);最后模拟 C2DMMessageServer 的 main 方法将消息推送到 Android 终端上。

程序的运行结果如图 2 所示:

```
The C2DMMessageServer is started...
auth responseCode = 200
获取auth: = DQAAALkAAACzd7tmYUgsWnUp6kBfI5edKmfJdOO5-y7ha-5qfqhcscNGRqBr
Please input the message to push:(Exit with q)
test c2dm server 2011-11-16
c2dm responseCode = 200
id=0:1321452679417942%1dc73bd900000031
Send Successfully
```

图 2　服务器端运行结果

Auth responseCode = 200 表示第三方服务器成功地获取到了 Auth,然后输入所要推送到 Android 客户端的消息内容,c2dm responseCode = 200 表示 C2DM 服务器响应成功,并且成功地获取了 id,将消息云端推送到了与 id 相匹配的 Android 终端上。

4.2　Andoid 终端的实现

Android 终端的实现,主要包含两个步骤:注册

C2DM 服务和接收推送消息。

●Android 终端需要向 C2DM 服务器注册 C2DM 服务,使程序允许接收推送消息,过程如下:

(1)Android 终端程序要向 C2DM 服务器启动注册需要的 registration intent。

(2)若注册成功,C2DM 服务器广播一个 com. google. android. c2dm. intent. registration intent,Android 终端程序要响应并接收这个 intent,并且从中获取注册成功所返回的 registration id。

(3)Android 终端需要把获得的 registration id 值发送给第三方服务器,并且第三方服务器需要把 registration id 值保存到数据库中。

这个过程的实现中,要新建一个 C2DMRegistration 类来实现 C2DM 注册的相关功能。

●Android 终端还需要接收 C2DM 服务推送过来的消息,过程如下:

(1)Android 系统获取 C2DM 服务器推送过来的信息,并且从信息中提取键值对数据。

(2)Android 系统向对应的 Android 终端发送 com. gogle. android. c2dm. intent. receive intent。

(3)Android 终端响应 receive intent,从中提取出键值对数据,最后根据之前和发送数据的第三方服务器商量好的键值,提取相应的数据。

在这个实现过程中,要新建一个处理 C2DM 数据的 C2DMReceiver 类;完成了 java 代码部分后,最后在 Manifest. xml 文件中声明和 C2DM 相关的权限信息。

Android 终端的运行结果如图 3 所示:

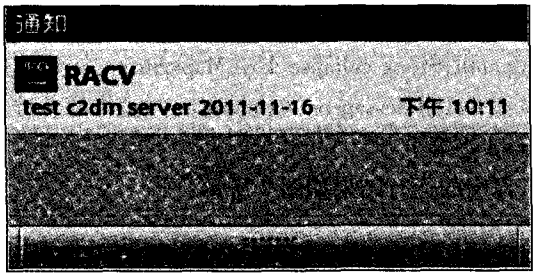

图 3 Android 终端运行的结果

5 结束语

Android 平台的应用程序开发已经日益广泛,通常服务器上有数据更新的时候,客户端是定时地去服务器上获取最新的数据,即 pull 方法。该方法无疑会浪费客户端的流量和手机的电量。

文中采用 push 方法,服务器端基于 Google 的 C2DM 服务,一旦有最新数据或者消息,会通过 C2DM 服务及时地发送到 Android 终端。用 C2DM 服务来云端推送,流量的开销极小,同时它是内建在 Android 的同步机制中的,因此不会带来额外的电力开销。

另外由于 Android 天生的 Intent 机制,C2DM 不需要应用程序的聆听,系统收到消息后会自动启动应用程序,大大降低了对性能、耗电的要求,对用户也更加方便。

参考文献:

[1] 华 春,胡 明. Push 和 pull 两种调度机制的仿真分析与研究[J]. 光通信技术,2008,32(10):57-59.

[2] Clos C. A Study of Non-blocking Switching Networks[J]. The Bell System Technical Journal,1953,32(5):406-424.

[3] Thompson T. The Android Mobile Phone Platform[J]. The World of Software Development,2008,33(9):40-47.

[4] 宋小倩,周东升. 基于 Android 平台的应用开发研究[J]. 软件导刊,2011,10(2):104-106.

[5] 公 磊,周 聪. 基于 Android 的移动终端应用程序开发与研究[J]. 计算机与现代化,2008(8):85-89.

[6] Palenchar J. Android to Set Wireless Markets Free;Supporters [J]. This Week in Consumer Electronics,2007,22(24):6-16.

[7] Darcey L. Android Wireless Application Development[M]. [s. l.]:Addison-Wesley Professional,2009.

[8] 刘 峰,王晔晗,汤步洲,等. 基于 Android 的智能中文输入法[J]. 计算机工程,2011,37(7):225-227.

[9] 闵现畅,黄理灿. 基于 Android 平台的 Web 服务技术研究[J]. 工业控制计算机,2011,24(4):92-94.

[10] 马 越. Android 的架构与应用[D]. 北京:中国地质大学,2008.

[11] 张艳芳,周 聪. 基于 Android 平台的移动终端设备的面积测量应用开发[J]. 计算机与现代化,2009(12):143-145.

[12] 张 京,刘甫迎. 基于 Android 云计算消息框架(C2DM)的 FoxNews_MID 手持移动系统的研究[J]. 计算机科学,2011,38(10A):461-463.