基于移动应用的校园信息服务系统①

马建平, 奚亮亮

(浙江工业大学 计算机学院, 杭州 331023)

摘 要: 为了使 web 上的校园信息更加及时地服务于师生, 提出了基于移动应用的校园信息服务系统, 以此作为基于 web 的校园信息服务方式的补充, 设计了基于 XMPP 的通信协议, 实现了服务器后台抓取相关信息. 最后通过 android 平台仿真, 显示出这种基于移动应用的校园信息服务系统有一定的可行性和有效性.

关键词: 移动应用; 校园信息服务; XMPP; android

Campus Information Services System Based on Mobile Application

MA Jian-Ping, XI Liang-Liang

(Computer College, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: In order to make teachers and students enjoy more timely campus information service on the web, this paper proposed a campus information services system based on mobile application, as a supplementary of web-based campus information service system. Meanwhile, communication protocol based on XMPP is designed and relevant information is captured by the server. Finally, the Android platform-based simulation obviously shows some feasibility and effectiveness of this campus information service mode.

Key words: mobile application; campus information service; XMPP; android

基于 web[1]的校园信息服务已经日趋成熟, 院校的重要信息, 往往会第一时间在 web 上发布, 是获取校园动态的主要途径, 极大的便利了校园生活. 师生可以方便地从 web 上获取校园信息, 然而往往以下原因, 使其获取信息并不及时: (1)记忆这些 web 域名是有困难. (2)校院级信息往往是教学信息, 趣味性不大, 导致生活中不会时常访问. (3)所需信息分布在多个网页, 需要多次访问. (4)无法随时在 PC 前访问信息, 手机访问院校 web, 在界面呈现上并不友好.

针对以上分析, 本文提出了基于移动应用的校园信息服务系统, 以此作为 web 校园信息服务方式的补充, 使师生在移动客户端及时享受校园信息的一站式阅读.

1 系统概述

1.1 系统架构

系统的体系结构框图[2]如图 1 所示, 各个部分为:

移动客户端: 平台主要考虑各种主流的智能手机, iPad 等平板移动客户端.

服务器业务层: 数据交互组件, 信息抓取组件, 应用服务组件. 数据交互组件的功能是完成移动端和服务器的数据传输, 包括 socket 流的建立, 通信协议的建立, 基于通信协议的编码解码; 信息抓取组件在服务器抓取相关信息存入数据库, 以便将信息推送到移动端; 应用服务组件包括一系列登入成功后客户端能使用的功能模块.

服务器端数据库: 保存用户发布的信息, 服务器抓取的信息, 学院社团信息数据等.

数据交互: (1)客户端和服务器业务层, 在信息推送的时候, 服务器要向客户端发送数据, 所以本文用 Socket 实现 TCP/IP 协议的网络通信. 客户端用 Socket 类, 服务器端用 ServerSocket 类, 建立基于 XMPP 的通信协议; (2)信息抓取组件和数据交互组件, 信息抓取组件类通过关联数据交互组件类以便将抓取的信息打

① 基金项目:国家自然科学基金(61075118)

收稿时间:2012-03-26;收到修改稿时间:2012-04-25

包成符合通信协议的数据格式进行发送; (3)应用组件和信息抓取组件, 每个应用组件通过关联调用信息抓取组件类以抓取符合自己要求的信息; (4)服务器业务层和后台数据库, 通过 java.sql 包实现对数据库(本文选用 sql server 数据库)的连接, 创建数据表单, 实现表单的增删改查.

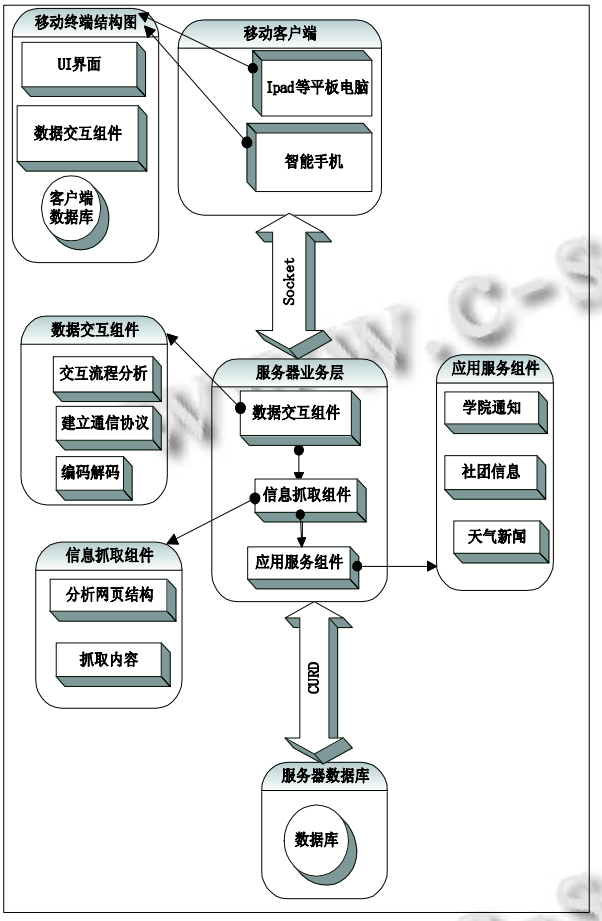

图 1　系统总体框图

整个系统的运转过程: (1)软件启动: 客户端通过 socket 流将用户名密码(可以是学号, 姓名)按照通信协议进行编码传送到服务器, 登录成功后服务器将学院通知, 社团通知等数据传到客户端, 客户端对传回的数据进行显示. (2)软件后台运行: socket 保持连接, 服务器端有信息更新(包括抓取到新信息和收到移动应用发送的新信息), 推送该信息到客户端. (3)客户端发布信息: 客户端用户在 UI 界面上输入要发布内容, 客户端数据交互组件将信息按通信协议编码, 通过 socket 将数据发送到服务器端, 服务器将数据解析后, 通对数据库进行操作, 推送信息到客户端.

1.2 应用服务模块

(1) 学院通知: 各个学院发布信息, 该模块信息推送到客户端.

(2) 社团信息: 各个社团可以在该平台发布社团的活动信息.

(3) 天气新闻: 抓取城市天气和重要新闻.

2 主要模块和关键技术

本文提出的基于移动应用的校园信息服务系统在性能上主要目标是信息获取的(1)即时性(2)全面性. 在以上系统框架中, 数据交互组件中基于 Socket 数据流通信决定信息的即时性, 信息抓取组件决定信息的全面性, 两者是主要模块.

2.1 数据交互组件

移动端和服务器的数据交互, 主要是基于 XMPP 的通信协议组件实现的. 首先实现基于 Socket 的客户端和服务器端得数据流连接. 其次我们要分析客户端和服务器的交互流程. 最后根据交互流程, 基于 XMPP 协议, 建立属于自己的通信协议.

2.1.1 建立 Socket 数据流连接

利用 java.net 包的 API 编写网络通信程序. 创建一个服务器的过程如下:

第一步,在指定的端口创建一个 java.net.Server Socket 对象, 代码如下:

```
ServerSocket server = new ServerSocket(9090);
```

第二步, 服务器等待客户端来连接, 代码如下:

```
Java.net.Socet server_client= server.accept();
```

第三步, 从 Socket 连接对象上调用方法得到输入输出流, 代码如下:

```
DataOutputStream  server_out=new
DataOutputStream(server_client.getOutputStream());
DataInputStream  server_ins = new
DataInputStream(server_client.getInputStream());
```

第四步, 使用输入/输出流对象进行通信数据的读写: 从输入流中读取数据, 向输出流中写入数据. 发送和读取的代码分别如下:

```
String  msg= “你好, 欢迎使用校讯通软件. ”;
server_out.writeUTF(msg);
String  readmsg=server_ins.readUTF();
```

类似的也可以建立客户端的 Socket 对象.

2.1.2 交互流程的描述

交互流程[3]描述如图 2 所示.

① 建立 Socket 连接.

② 客户端与服务器建立 TCP/IP 连接后, 发送一条地址信息请求登录消息.

③ 服务器发回是否登录成功的一条消息.

④ 服务器向客户端推送消息.

⑤ 客户端向服务器应答是否收到推送消息.

⑥ 客户端向服务器发送发布信息的消息.

⑦ 服务器应答客户端是否发布成功.

下线, 断开 Socket 连接.

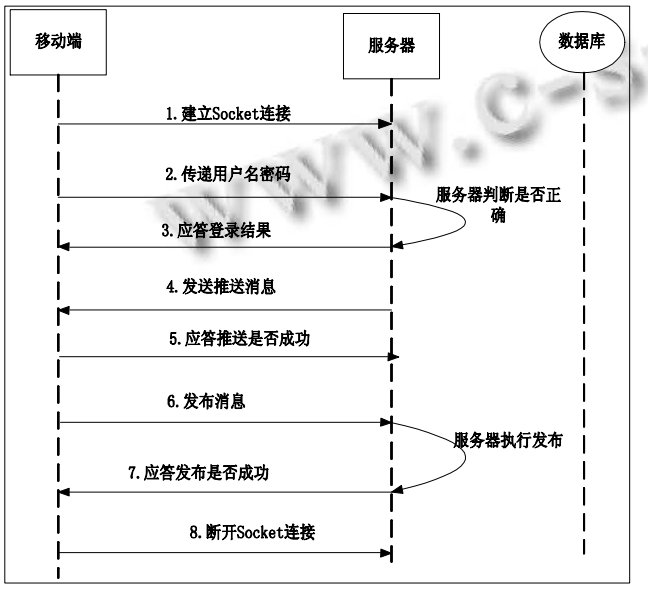

图 2 交互流程图示

2.1.3 XMPP 消息格式定义

1.登录请求

```
<information>
  <type> login </type>
  <location> 用户名,密码</location>
</information>
…
```

5.发布消息

```
<information>
  <type>release </type>
  <content> 消息内容</content>
</information>
```

6.发布消息应答

```
<information>
  <type> releaseResp </type>
  <state> 发布消息结果</state>
</information>
…
```

2.1.4 XML 格式消息的解析

移动客户端和服务端的数据都是 XML 格式的, 需要解析提取数据, 本文用 XmlPullParser[4]进行解析, 解析后生成一 个 Information 的自定义类对象.

假设手机端发布消息的时候传了这样一条 String 到服务器.String msg="<?Xml version=\"1.0\" encoding =\"UTF-8\"?><Message><information><type>release</type><author>计算机学院党支部</author><content>所有党员周五晚上 6:30 到郁 A201 开会</content></information></Message>".

下面是解析代码:

```
//eventType 为获得事件类型
eventType = xmlPullParser.getEventType();
//事件没有结束, 则不停的循环
while(eventType!=XmlPullParser.END_DOCUMENT){
//nodename 为获得节点的名称
String nodeName=xmlPullParser.getName();
 switch (eventType) {
  case XmlPullParser.START_DOCUMENT:
   informations=new ArrayList<Information>();
    break; //文档开始, 则用一个 list 保存对象
   case XmlPullParser.START_TAG:
   if("type".equals(nodeName)){
   //赋值 information 对象的 type 值
   information.setInF_Type(xmlPullParser.nextText());
   }else if("title".equals(nodeName)){
   ...//赋值 information 的 title 值
   }else if("content".equals(nodeName)){
   ...//赋值 information 的 content 值
   }
  break;
 case XmlPullParser.END_TAG://结束节点
      if("information".equals(nodeName)){
      informations.add(information); //放入列表
  }
  break;
   eventType=xmlPullParser.next();
   }
```

解析的结果是: information.title=党员会议

information.author=计算机学院党支部

information.content=所有党员周五晚上 6:30 到郁 A201 开会.

2.2 信息抓取组件

为了使移动应用能够对信息做到一站式阅读，服务器要在后台抓取经常被访问的信息. 本文以工大公告栏(如图 3)为例抓取相关信息.

图 3　工大公告栏信息网页

2.2.1 用 htmlparser 抓取链接

这里用开源软件 htmlparser[4-6]作为信息抓取工具. 首先要提取该公告的连接. 分析该网页的组织结构, 发现公告的链接都在其中的一个 table 标签里面. 这个 table 标签属性 with=”100%”，是区别于其它 table 的属性. 以这个属性可以把表提取出来. 将这个表提取出来后, 对表的内容再做一次抽取, 抽取出每个公告的链接和题目, 提取的方法是利用 htmlparser 里面的 LinkTag.class, 作为过滤标签. 过滤之后, 将链接和题目放入到 Map 中.

2.2.2 抓取内容

当获取公告的链接之后, 我们需要抓取该连接对应的具体内容. 如同 2.2.1 节内容, 分析公告内容网页的 html 结构. 结构显示告示内容都在 和<tr>中, 所以我们这里用 htmlparser 里面 Div 和 TableColumn 作为 filter, 并结合它们的属性作为过滤条件. 将 2.2.1 节所返回的 Map 中的链接值进行遍历, 每个遍历的过程都提取该链接对应的内容. 提取的内容包括 title,content,author,time 如图 4 所示:

图 4　公告抓取结果显示

将这些信息存储到服务器数据库，以便向移动端推送.

2.3 信息的发布，推送，存储，读取

信息的发布，推送，存储，读取是一环扣一环的过程. 移动端将要发送的信息按照规定的 XMPP 协议进行打包，通过 Socket 流将打包好的信息字符串传到信息的发布，推送，存储，读取是一环扣一环的过程。移动端将要发送的信息按照规定的 XMPP 协议进行打包，通过 Socket 流将打包好的信息字符串传到服务器端，服务器根据 XMPP 协议的方式进行解包得到一条信息对象，根据信息的内容，将信息通过 Socket 流推送到连接在服务器的移动端，服务器同时存储本条信息到数据库，移动端在收到服务器发送的信息后，同样进行解析，将解析后的信息通过 android 的控件显示出来.

2.3.1 信息的发布

客户端将要发送的信息按照 XML 格式进行打包，发送到服务器. 以下以发送一条通知为例.

当手机端点击发送后，主要执行以下代码:

```
//msg 是按消息格式定义的 String
client_out.writeUTF(msgString);
```

2.3.2 信息的推送

服务器解析出该 XML 信息的内容后，根据 type 判断出该信息为 release，需要向所有客户端进行推送，则向所有保存在 mClientList 表里面的客户线程发送信

息, 然后保存该条信息到数据库. 伪代码如下:

```
if(type.equals("release")) {
   信息的存储伪代码;
   //迭代 mClientList , 消息发送
   this.sendMessage(读到的字符串);
}
```

2.3.3 信息的存储

本文以 sql server 2005 建立了名为 xiaoxuntong 的数据库, 并建立了 Information 表.

服务器解析出该 XML 信息的内容后, 根据 type 判断出该 Information 为 release, 需要保存该条信息到数据库, 其中最为关键的 sql 语句为: String sql="insert into information("+"InF_Author],[InF_Title],[InF_Content],[InF_Time])values('"+"','"+information.getInF_Author()+"','"+information.getInF_Title()+"','"+information.getInF_Content()+"','"+information.getInF_Time()+"')";

2.3.4 信息的读取

在服务器向移动端发送基于 XMPP 消息后, 移动端收到基于 XMPP 的字符串, 对字符串进行解析, 生成 Information 对象, 通过 android[7,8]中的 Dialog, List View 控件等, 将 Information 对象的内容呈现给手机用户.

3 系统仿真

3.1 仿真环境

服务器配置软硬件环境包括: 配置联想 Z460, 时钟频率 2.53G, 物理内存 2.00G, 操作系统采用 win7; 服务器软件组合: android 模拟器+sqlserver; 客户端软硬件包括: android 模拟器;

3.2 移动应用系统的运作仿真

该部分模拟仿真了移动应用发布信息, 到服务器推送该信息到移动端的效果, 从而说明移动应用在信息服务方面的即时性, 便利性优势.

3.2.1 应用组界面显示

登录后, 界面效果显示如图 5.

3.2.3 手机端 B 在待机情况下收到服务器推送信息

图 6 为移动端在待机情况下收到推送信息的效果仿真, 手机会在状态栏和界面上都给予提示.

3.2.4 手机 B 查看信息

如图 7 所示, 进入移动客户端主界面, 可以浏览所有服务器推送的消息, 可点击查看详情.

图 5 登录后界面结果显示

图 6 移动端收到推送消息显示

图 7 移动端消息更新效果

4 结语

在手机越来越智能化的潮流下, 利用移动应用来改善我们生活与学习的研究也越来越具有重要意义. 本文提出了基于移动应用的校园信息服务系统, 相比于 web 形式需要用户主动去取信息, 该系统将信息送到用户手中, 使师生在信息获取上更加便利, 在此基础上还可以考虑用户信息的个性化定制, 满足不同学生的需求. 总的来说, 该服务形式可以很好地优化目前相对完善的 web 信息服务方式.

(下转第 33 页)

随机文档亦解释特别详细，在此不再赘述.

4.4 测试并启动 Squid

#/usr/local/squid/sbin/squid -z

这是初始化 Squid 的 cache 磁盘空间

#/usr/local/squid/sbin/squid -NCd1

这是测试 squid 配置文件是否有问题，如果没问题的话将显示:"Ready to serve requests", Squid 就可以正常运行了，在/etc/rc.local 里加入/usr/local/squid/sbin/squid－s，这样可以使得每次重启服务器，Squid 都能自动启动.

5 代理服务器集群功能测试

代理服务器集群的 LVS 调度服务器和 Squid Real Server 安装完成后可以做测试，在调度机上，用命令 ipvsadm 查看:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port
      Forward Weight ActiveConn InActConn
TCP 202.194.64.200:irdmi wlc persistent 600
  -> 192.168.1.2:irdmi    Route  1     1673    19874
  -> 192.168.1.3:irdmi    Route  1     1313     6620
  -> 192.168.1.4:irdmi    Route  1     1440    11010
```

显示的是目前 3 台 Real Server 的连接信息，我们可以做测试，拔掉其中一台 Real Server 的网线，或者用 kill 杀掉 Squid 进程，甚至关掉服务器，LVS 都能检测到，因为 LVS 检测的是 Real Server 的 TCP 端口（本项目是 TCP 端口 8000），任何导致 TCP 端口失效的故障（包括进程死掉或失效）LVS 都能检测到. LVS 检测到某台 Real Server 的故障后，将其 Weight 设置为 0，同时将不再向其转发请求，但当这台服务器的 Squid 进程回复正常以后，LVS 检测到以后，重新将其 Weight 设置为 1，重新向其转发需求，从而起到了负载分担和避免单点故障影响业务的状况.

6 结语

基于 LVS 和 Squid 的代理服务器集群系统目前在作者所在单位承担了大部分出口代理服务，我们为该系统配置了认证系统，利用 Squid 的认证功能和后台数据库记录，根据源 IP 地址决定是否为某个 IP 地址提供代理服务；为了提高 Squid 系统的工作效率和稳定性，对 Linux 操作系统和 Squid 的配置文件进行了许多优化. 由于 LVS 集群提供了极高的可靠性，所以极少发生因设备或系统故障导致的服务中断. 长期运行的结果显示，该系统提供优质的代理服务的同时，又极大地降低了系统维护的工作量.

参考文献

1 LVS 中文站点.http://zh.linuxvirtualserver.org

2 Linux Virtual Server Projeet. http://www.LinuxVirtua1Server.org

3 Squid 英文站点. http://www.squid- cache.org

4 姜文颖.网络中几种负载均衡实现技术的探讨.中国数据通信,2004,(1):61－62.

5 罗忠石,刘虹霞.用 Squid 架构本地信息代理服务器.计算机科学,2005,31(6):95－97,.

6 章文嵩.Linux 服务器集群系统.http://www.ibm.com/developerworks/ cn/ linux/ cluster/ lvs/ part1/ index.html

7 彭永华.Squid 权威指南.http://blog.s135.com/book/squid/

(上接第 23 页)

参考文献

1 王淑营,赵慧娟.基于 Web 的动态信息发布系统技术研究.计算机应用研究,2004,4:189–190.

2 王红萍,王黎明,梁民赞,等.基于 Facade 模式的移动定位服务系统平台的设计.计算机工程与科学,2010,32(7):151–153.

3 胡东锋.微博是这样炼成的.北京:人民邮电出版社,2010.46–50.

4 Haustein S, Slominski A. XMLpull Parsing. [2012-02-21]. http://www.xmlpull.org/

5 APACHE. htmlparser.[2012-03-24]. http://htmlparser. source-forge.net/.

6 罗刚,王振东.自己动手写网络爬虫.北京:清华大学出版社,2010.6–21.

7 Google. Androiddevelopers. [2012-03-24]. http://developer.android.com/.

8 杨丰盛.android 应用开发揭秘.北京:机械工业出版社,2010.28–109.