DOI:10.3969/j.issn.1001-4551.2016.12.018

面向 Android 的工业设备远程监控系统设计*

赵 炯,周 影
(同济大学 机械与能源工程学院,上海 201804)

摘要:针对当前工业设备远程监控系统便携性上的局限性,利用移动设备随时随地可使用的优势,设计和开发了一种面向 Android 移动终端的设备远程监控系统,对系统架构、数据通信、数据缓存、消息推送等关键技术进行了研究。根据设备数据的即时性和紧急性对设备数据进行分类,并针对不同数据类型的通信需求,设计了 Android 客户端与服务器的通信技术方案。为了保证了软件的可靠性与可移植性,基于 MVC(Model-View-Controller)思想,集成 Java Web 技术中的 Spring、Spring MVC 和 Mybatis 框架实现了服务器程序,实现了程序清晰的层次划分和各模块的解耦。系统测试结果表明,该系统能在客户端上实现警报推送、信息查询、实时数据查看等功能,并具有良好的可拓展性,系统可灵活架设在现有监控系统中,进一步拓展现有系统的功能。

关键词:移动互联网;远程监控;Android;数据推送
中图分类号:TP277 **文献标志码**:A **文章编号**:1001-4551(2016)12-1511-06

Remote monitoring system of industrial equipment for Android

ZHAO Jiong, ZHOU Ying
(School of Mechanical Engineering, Tongji University, Shanghai 201804, China)

Abstract: Aiming at overcoming the limitation of portability in current remote monitoring system for industrial equipment. A new-type monitoring system in needs of Android mobile terminals was designed and developed. Critical technologies in system like system architecture, data communication, data cache and message push were studied. Data of equipment were classified by instantaneity and urgency, and scheme of communication technology between Android client and server was designed according to the needs of different data types. To ensure the reliability and transferability, the server program was developed based on the idea of MVC(model-view-controller), and frameworks such as Spring, Spring MVC and Mybatis which were widely implemented in Java Web field were integrated into the program, thus making the program to be clearly stratified and least-coupled. The test results indicate that the system is able to realize monitoring functions such as data push, information query and real-time data transmission on Android client, and has good extensibility. The system can be neatly set up and further expand the function in existing monitoring systems.

Key words: Mobile Internet; Remote Monitor; Android; Data Push

0 引 言

传统的工业设备监控系统大多基于 PC 运行,工作人员必须在监控室内进行操作。移动互联网的发展为系统的研究提供了新思路。文献[1]指出,未来物联网中移动终端随时随地可用的特点,将加速物联网的发展,并使其延伸至工业领域。随身携带是移动终端区别于传统监控系统的重要特征,结合数据推送技术,可提高监控系统的自动化水平。

目前该领域已经有一些初步的相关研究出现,如基于移动流媒体技术的视频监控系统[2],实现了对生产现场的实时视频查看;利用手机作为智能家

收稿日期:2016-07-19
基金项目:上海市自然科学基金资助项目(13ZR1444600)

作者简介:赵 炯(1963-),男,江苏苏州人,博士,副教授,硕士生导师,主要从事计算机网络协议分析、Linux 操作系统和自动化系统中智能控制技术方面的研究. E-mail:Jiong.Zhao@tongji.edu.cn

居的远程操作端[3]，实现远程控制功能；还可对简单生产过程参数的检测，如观察农业生产的数据等[4]。结合移动端独有的特性，如数据推送、地理定位、社交媒介等功能丰富传统监控系统的功能，是今后研究的重点。

Android 操作系统作为主流移动操作系统，具有开源、开放的优点，本研究选择其作为移动客户端的实现平台。同时，本研究使用 Java Web 技术实现面向 Android 客户端的数据服务器，并使用 Spring、Spring MVC 与 Mybatis 框架编写服务器[5]。

本研究在说明系统的总体设计，介绍系统关键技术的基础上，最后给出系统的测试情况。

1 系统总体设计

1.1 系统拓扑结构

系统的拓扑结构如图 1 所示。

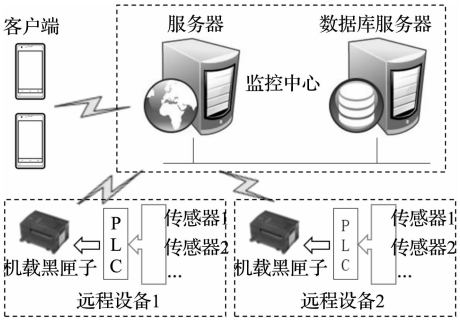

图 1 远程监控系统拓扑结构

在原有的系统中，每台设备安装有一台嵌入式终端，称为机载“黑匣子”，基于 Linux 系统实现，具有以太网和 3G 通信模块，能解决底层数据采集设备与监控中心之间通信协议与信道不一致的问题。底层的数据被 PLC 采集后通过以太网发送给黑匣子，黑匣子的通信程序将数据打包并经过 3G 模块利用 VPN 将数据发送到数据库服务器，数据库服务器上的数据解析程序将不断接受数据并存入数据库，数据库服务器仅能通过 VPN 被广域网访问，对所有数据进行统一的管理。另一台服务器面向外网，从数据库服务器中获得数据，并响应 Android 客户端的数据请求。

该系统在传统的远程监控系统的架构之上，增加了面向客户端的服务器程序，无需附加其他设备，具有很好的拓展性，能兼容以往的系统。

1.2 服务器架构

系统架构采用了 C/S 模式，即客户端/服务器模式，服务器直接将数据传输到客户端，不用考虑客户端的视图呈现，这种架构减少了 B/S 模式时向客户端传输 HTML 页面时的流量消耗。该系统中服务器程序包括两部分，一部分用于响应实时数据请求，与客户端的通信是基于连接的，作为一个单独的进程运行；另一部分响应用户的 HTTP 请求，发布并运行在在 Tomcat 容器中。

本研究基于 SSM 框架对系统进行层次划分，并将层次间的耦合降到最低。SSM 由 Spring、Spring MVC、MyBatis 3 个开源框架组成。其中，Spring 是一个轻量级的容器框架，旨在简化 JAVA 开发，其核心思想是通过依赖注入（dependency injection，DI）和面向切面编程（aspect oriented programming，AOP）降低耦合和对常规 JAVA 对象的侵入性，将其他框架整合为一个复杂的系统，并实现了对象生命周期的管理[6]。Spring MVC 是 Spring 的子框架，基于 MVC 模式。MyBatis 则是一个数据持久层框架，消除了 JDBC 繁琐的代码以及数据库结果集的检索，可使用简单的 XML 或注解用于配置原始映射，将接口和 Java 对象映射成数据库中的记录。SSM 框架集的应用使得服务器开发变得快捷可靠，具有优良的拓展性。

采用 SSM 框架后，服务器架构如图 2 所示。

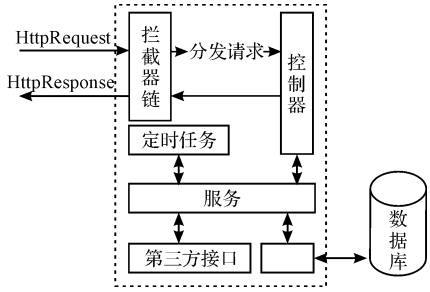

图 2 服务器程序层次结构

各组成部分说明如下：

（1）拦截器（Interceptor）。包括用户登陆拦截器、用户请求预处理拦截器等，它们组成了链式结构，前者用于检验用户是否登录，后者用于处理用户请求的首部信息、记录用户操作历史、向响应中写入 HTTP 首部信息。

（2）控制器（Controller）。控制器负责响应用户的 URL 请求。Spring MVC 的 DispatcherServlet 将用户的请求根据 URL 分配到不同的控制器进行处理，每个控制器可以响应多个不同的 URL 请求。

（3）定时任务（Schedule）。负责完成需要定期完成的任务，如定时扫描是否存在新的报警信息以及定时进行设备的故障诊断、评估等。

(4)服务(Service)。服务向上层模块提供了调用接口,本身则负责从下层调用 DAO 接口访问数据库和调用第三方接口。

(5)数据访问对象(Data Access Objects DAO)。DAO 提供了数据库访问的接口,Mybatis 通过接口与 XML 文件映射动态生成 DAO 的实例,完成数据库操作。

1.3　客户端程序架构

根据大型设备远程监控的基本要求,结合移动端特征,面向移动端服务的系统,需要满足以下基本要求。

(1)快速预览。企业往往存在多台设备分布在各地,通过地图可以直观展示设备的地理位置信息,地图上还可展示各个设备的实时状态,有助于方便快速查找异常设备。

(2)查询设备信息。能够通过互联网访问系统数据库当中的设备、零部件、传感器、相关人员、供应商等信息。

(3)查看实时数据。系统能够高效可靠地进行实时数据通信,通过图表和数值形式直观展现当前设备的运行参数和状态信息。

(4)查看历史数据。能够根据时间区间选择并查看历史运行的状态,并能生成报表文件。

(5)数据推送。移动应用能克服地理与时间的限制,使得工作人员能够随时随地获取系统数据,同时系统也能随时将系统的状态的变化、预警等信息即时反馈到相关人员手中。

根据上述需求,本研究设计的客户端界面跳转流程如图 3 所示。

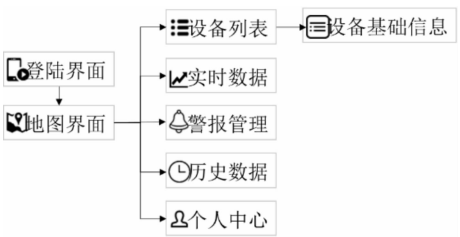

图 3　客户端界面跳转流程

系统采用的通信手段与传输的数据类型有关。本研究将系统数据分为 3 类,第 1 类属于设备的基础信息,变动性小,采用 HTTP 访问,并采取持久化缓存机制;第 2 类数据不断更新,需要实时获取;而第 3 类数据为故障诊断系统生成的数据,随时可能变动,并需要服务器主动发送到客户端,涉及到数据推送技术。设备数据分类如表 1 所示。

表 1　设备数据分类

类别	数据类型
第 1 类	设备信息、零部件信息
第 2 类	设备状态、传感器实时数据
第 3 类	故障信息、报警信息、预测评估信息

笔者根据上述需求可制定系统通信方案,如图 4 所示。

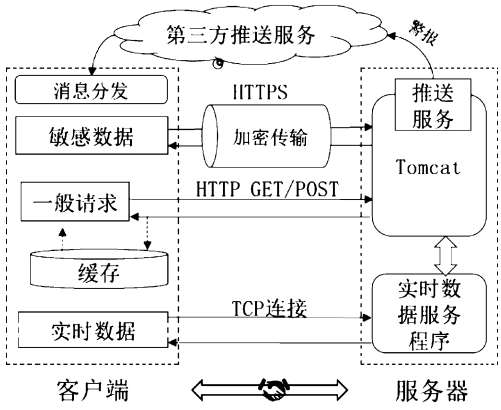

图 4　C/S 的通信方案

其中,HTTP 请求主要用于向第 1 类数据请求,并增加了客户端缓存机制减少通信量;对于有安全性要求的通信,利用 HTTPS 协议,通过 SSL 进行数据传输;对于实时数据,则使用了 TCP 长连接,并在应用层制定了协议;此外,利用第三方推送云平台实现了数据推送功能。

2　系统的实现

2.1　数据的序列化与反序列化

数据在服务器与 Android 中都以 Java 对象的形式存在,但在通过网络传输时要经过相应的转换,这涉及到数据的序列化与反序列化。序列化是将对象状态转换为持久化的或可传输的格式的过程。与序列化相对的就是反序列化,它将数据流转换为对象。JAVA 语言提供了 Serializable 接口,可自动实现对象二进制序列化,具有很高的效率,但是这种依赖于 Java 语言的机制并非最佳选择。目前,JSON 和 XML 等平台无关的交换格式应用广泛。

JSON(JavaScript Object Notation)是一种轻量级数据交换格式,采用键值对文本存储数据,使用“[ ]”表示数组,“{ }”表示对象,可以表达各种复杂的数据结构。文献[7]对 JSON、XML 的开销、传输时间以及反序列化的效率进行了量化比较,结果表明,相比于 XML 格式,JSON 格式的数据大大降低了反序列化时的冗余度,同时,使用 JSON 格式可减少网络流量。

在该系统的开发中,本研究使用了统一的 JSON 格式作为数据传输格式,其结构为:

{"code": 0, "status": "ok", "data": { }}。其中 code 为状态码,表示了本次请求的状态,code 为 0 代表成功,非 0 则代表异常。status 字段作为辅助字段,说明了状态码的含义,这些状态包括:成功、密码错误、登陆异常、未登录、无权限、参数错误、服务器内部错误、验证码错误、新版本存在等。data 字段中保存了本次传输的数据,这些数据同样以 JSON 格式嵌套其中。

Android SDK 中提供了 JSON 的序列化工具类 org.json. JSONObject,可将 JSON 字符串反序列化为 JSONObject 对象,可通过 getX(String key)方法获得 key 对应值,其中:X 代表返回值类型,如 Long、String 等。

但该方法有局限性,即不能将 JSON 字符串完全反序列化为实体对象,开发过程中必须准确地按照字段的名称和字段的类型获取键值,降低了开发效率与可移植性。针对该问题,开源工具 FastJSON 提供了更为便捷的办法。FastJSON 可将 JSON 反序列化成对象。以如下实体模型类为例:

```
class User {
    public Integer userId;
    public String name;
}
```

以下 JAVA 代码说明了两种反序列化工具使用的对比。

```
//定义 json 字符串
String json = "{"userId":1;"name":"visitor"}";
//使用原生 json 解析 json 字符串
JSONObject jsonObject = new JSONObject(json);
int userId = jsonObject. getInt("userId");
String name = jsonObject. getString("name");
//使用 FastJson 进行反序列化
User user = JSONObject. parseObject(json,User. class);
```

可见,使用 FastJson 后,可直接使用 User 的实例进行操作,大大提升了程序的可移植性和可靠性,简化开发复杂度。

2.2 持久化缓存

为了减少在网络中传输的重复数据量,应采用数据持久化缓存策略,Android 中的缓存的策略有 SQLite 数据库、缓存文件、SharedPreferences 等[8]。

SharedPreferences 以键值对的形式保存数据,用于少量数据的保存,不适用于网络数据缓存。SQLite 是一种轻量级的关系型数据库,基于文件操作,在程序中运行时没有专门的服务进程,效率很高,但其本质上仍然是对文件操作,在数据查询过程中,需要大量文件读写操作。

本研究采用了文件缓存机制,原因如下:一是缓存数据具有时效限制,单靠 SQLite 难以实现;二是可减少代码量,在读取网络数据和本地缓存使用同样的解析代码,避免在程序中加入大量 SQLite 操作代码。程序根据是否存在可用的缓存数据决定是否重新请求数据。在该系统中,设备本身的基础信息变动不大,缓存的有效期限比较长;而对于某些具有变动性或生命周期极短的数据则需要设置较短的缓存时间或不缓存。

缓存使用程序的默认缓存目录,该目录在无 Root 权限的情况下仅本应用程序可以访问,保证了用户数据的私密性。在程序内,不同的用户具有独立的缓存,缓存以请求的 URL 作为键值,将通过该 URL 得到的响应实体作为缓存内容。缓存数据均设置有效期。

本研究采用开源轻量级缓存框架 ACache 作为缓存处理工具,程序在获取网络数据时,采用的缓存机制如图 5 所示。

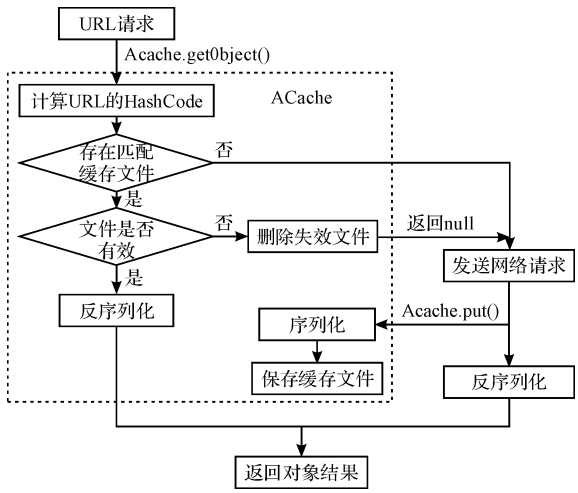

图 5 程序缓存机制

2.3 实时数据传输

客户端和服务器使用 TCP 连接交换 JSON 字符串实现了实时数据传输,客户端与服务器的数据字段如表 2、表 3 所示。

表 2 客户端请求字段

字段	含义
request_id	请求标识
op_code	客户端的操作类型
parameter	请求携带的参数
sensor_id	请求的传感器 id 列表
interval	刷新时间间隔
用户会话 ID	session_id

注:op_code 标识的操作类型包括用户验证、请求传感器数据、更改刷新频率、切换传感器列表、暂停、重新开始、心跳信号等;sensor_id、interval、session_id 等皆保存在 parameter 字段中

表 3　服务器响应字段

字段	含义
request_id	请求标识
timestamp	时间戳
code	用户请求的状态码
data	实时数据
sensor_id	传感器 id
value	传感器数据

注:request_id 表示这是属于用户哪次请求的响应;sensor_id 与 value 保存在 data 中

实时传输基于 TCP 连接,在应用层采用了如下交互协议:

(1)客户端建立 TCP 连接。

(2)验证请求合法性。客户端将会话的 ID 发送给服务器,实时数据服务进程先向 Tomcat 服务器发送一次 HTTP 请求验证用户合法性,对于不合法的请求,服务器返回错误码,并主动关闭连接释放资源。

(3)开始数据请求。客户端在 parameter 字段中告知服务器需要查看的传感器的 ID 以及刷新时间间隔,实时数据服务程序查询将实时数据、时间戳发送给客户端。此后,服务器按照用户指定时间间隔或默认间隔定时发送数据包。

(4)切换传感器列表。客户端切换传感器列表时,将重新生成随机的 request_id,再次向服务器发送一次请求。

(5)客户端在实时数据暂停传输时向服务器定时发送心跳包以防止客户端地址被运营商回收,这个时间间隔不大于 10 min。

(6)客户端关闭 TCP 连接,通信结束。

2.4　数据推送

远程监控中心常常有一些数据需要主动发送到客户端,如某些零部件发生故障时向用户主动发送故障报警,或者在对设备的运行状态进行故障诊断预测后,向用户发送报警信息,以及时要求用户采取措施进行预防和维护。这需要数据推送技术来实现。

目前 Android 数据推送技术较多。Google 官方提供了 GCM(Google cloud messaging)服务,它采用统一的应用推送服务进程,可减少手机内维持的后台进程数目,节省系统资源,但国内该服务限制颇多。轮询(Polling)技术定时向服务器发送 HTTP 请求,这种技术实现简单,但是不断建立和断开连接的开销很大。更优的方法是建立 TCP 连接,基于此方法的协议主要有 MQTT[9](message queuing telemetry transport,消息队列遥测传输)和 XMPP(extensible messaging and presence protocol)。

系统也可采用第三方的推送服务,如极光推送、百度推送等。使用时须在程序的 AndroidManifest. xml 文件中注册相应组件。以百度推送为例,使用时需注册百度推送的 BroadcastReceiver 与 Service 组件。百度推送的后台通信服务运行在一个独立于该应用程序进程的名为 bdservice_v1 的进程中,保证了单一终端上使用百度推送的应用共享一个服务进程和一条 TCP 连接,有效降低资源消耗,并保证推送服务和应用程序的独立性,关闭应用进程时,推送服务不受影响。

百度推送的后台服务进程在接收到新的推送消息时,会发出一个广播事件,因此需要在程序中创建一个 BroadcastReceiver 组件作为该广播事件的接收者。百度提供了继承自 BroadcastReceiver 的抽象类 PushMessageReceiver,并提供了处理推送消息的回调接口 onNotificationClicked、onNotificationArrived、onMessage 供开发者重写。这些接口用于处理两种不同的消息类型:透传消息和通知。透传消息到达时不会在通知栏展示,而是静默回调 onMessage 方法;通知到达时会调用 onNotificationArrived 方法并将消息显示在通知栏中,用户点击系统通知栏的消息时,onNotificationClicked 方法会被回调,重写该方法,可利用 Android 的 Intent 机制进入到程序完成相关处理。

3　程序测试

3.1　服务器运行环境

该系统实验环境下的监控对象为分布在各地的大型港口设备,每台设备上均安装了大量传感器。在该系统之前,已有数据采集、通信、数据库、数据库管理系统等基础。该系统在原有的服务器主机上,安装了 Java 运行环境与 Tomcat 服务器。服务器端程序开发完成后打包为 war 文件,发布在 Tomcat 容器中。该系统服务器操作系统为 Windows Server 2012,Tomcat 运行在 Java 8 环境中,采用的数据库为 SQL Server 2012。实际结果表明,服务器能长时间稳定运行。

3.2　客户端界面

移动互联网时代,界面友好性是程序的重要评价标准,但移动客户端在工业领域的应用尚在起步阶段,目前产品界面与交互设计方面有待研究,本研究在客户端的界面编写中,遵循了 Google 官方推荐的 Material Design 规范[10],以提高程序的美观性。

用户登陆后将进入首页,首页的地图利用不同颜色的图标对不同地点设备的状态进行标注。点击图标,将会出现的弹层如图 6(a)所示,展示设备的信息

并定时刷新设备的重要传感器的实时数据。从左上方的导航键切换页面,还可实现对设备各种信息的查询,如图 6(b)所示。客户端界面如图 6 所示。

(a) 首页与设备实时数据 (b) 设备信息

图 6 客户端界面

3.3 客户端报警推送

服务器通过定时任务检查新报警,报警数据将被推送到客户端,显示在通知栏内,如图 7 所示。

图 7 推送警报在通知栏的显示

实验表明,警报推送信息到达客户端的时延在 2 s 以内,基本满足警报推送的实时性要求。

4 结束语

移动智能终端在工业领域的应用,能促进工业领域的商业模式和生产模式转变,具有重要意义。本研究基于 Android 系统实现了实时数据、数据推送、信息查询等功能,拓展了传统远程监控系统的功能;在服务器的开发中使用了多种开源框架,提高了系统可移植性。该系统还适用于其他应用场合,如智能家居和各种工业监控领域。

本研究更进一步的发展方向是实现远程控制,通过网关向局域网内部控制网络发送控制指令完成操作请求;同时,也可利用故障诊断与预测等技术,进一步使系统自动化、智能化。此外,由目前系统还只适用于 Android 系统,具有一定局限性,还可采用基于 HTML5 的跨平台 Hybrid 开发技术使本研究拓展到其他移动操作平台[11]。

参考文献(References):

[1] GUBBI J, BUYYA R, MARUSIC S, et al. Internet of Things (IoT): A vision, architectural elements, and future directions[J]. **Future Generation Computer Systems**, 2013,29(7):1645-1660.

[2] 曹晓芳,王 超,李 杰. 一种基于 Android 智能手机的远程视频监控的设计[J]. 电子器件,2011,34(6):709-712.

[3] 杨 威,高文华. 基于 Android 的智能家居终端设计与研究[J]. 计算机技术与发展,2013,23(7):245-248.

[4] 李 慧,刘星桥,李 景,等. 基于物联网 Android 平台的水产养殖远程监控系统[J]. 农业工程学报,2013,29(13):175-181.

[5] 徐 雯,高建华. 基于 Spring MVC 及 MyBatis 的 Web 应用框架研究[J]. 微型电脑应用,2012,28(7):1-4.

[6] WALLS C. Spring in Action[M]. 3rd ed. Greenwich: Manning,2011.

[7] 高 静,段会川. JSON 数据传输效率研究[J]. 计算机工程与设计,2011,32(7):2267-2270.

[8] 倪红军. 基于 Android 系统的数据存储访问机制研究[J]. 计算机技术与发展,2013,23(6):90-93.

[9] 许金喜,张新有. Android 平台基于 MQTT 协议的推送机制[J]. 计算机系统应用,2015,24(1):185-190.

[10] GOOGLE, Material Design [EB/OL]. [2016-07-02]. http://material.google.com.

[11] ZIBULA A, MAJCHRZAK T A. Cross-Platform Development Using HTML5, jQuery Mobile, and PhoneGap: Realizing a Smart Meter Application[J]. **Lecture Notes in Business Information Processing**,2013(140):16-33.

[**编辑**:周昱晨]

本文引用格式:

赵 炯,周 影. 面向 Android 的工业设备远程监控系统设计[J]. 机电工程,2016,33(12):1511 - 1516.

ZHAO Jiong, ZHOU Ying. Remote monitoring system of industrial equipment for Android[J]. Journal of Mechanical & Electrical Engineering, 2016,33(12):1511 - 1516.

《机电工程》杂志:http://www.meem.com.cn