浙江科技学院学报,第 17 卷第 4 期,2005 年 12 月
Journal of Zhejiang University of Science and Technology
Vol. 17 No. 4, Dec. 2005

税务远程申报系统的设计与实现

钱亚冠[1],关晓惠[2]
(1. 浙江科技学院 理学院,浙江 杭州 310023;2. 浙江水利水电专科学校 计算机系,浙江 杭州 310018)

摘　要:介绍了税务远程申报系统 NTDS 的整体设计方案和关键技术的实现。着重分析了 XML 技术应用于异构数据源间的集成与交互;利用 PKI 不对称加密算法与 DES 对称加密算法相结合的方案,提高网络数据在远程传输过程中的加密效率,及利用 Java 多线程技术成功解决了税务远程申报系统在并发性方面的难题。给出了系统稳定运行后的相关测试数据和结论。

关键词:税务远程申报系统;XML;数据加密;RSA;DES;Java 线程
中图分类号: TP393.09　　**文献标识码**: A　　**文章编号**: 1671-8798(2005)04-0277-05

Design and implementation of a remote tax declare system

QIAN Ya-guan[1], GUAN Xiao-hui[2]
(1. School of Science, Zhangjiang University of Science and Technology, Hangzhou 310023, China;
2. Department of Computer Engineering, Zhejiang Water Conservancy and Hydropower College, Hangzhou 310018, China)

Abstract: This paper introduces the design and implementation of the key technologies based on remote declare system NTDS. Especially, it is dedicated to illustrate how to apply the key technology of XML to heterogeneous data source, take the combination of PKI and DES algorithm to improve the efficiency of encryption and make use of Java threads on the parallel system. Finally, it shows the testing result and expectation of NTDS.

Key words: remote tax declare system; XML; encryption; RSA; DES; Java threads

税务远程申报(RTDS)对于提高企业和税务部门的工作效率,加强税收的稽查力度都有重大的意义。目前,中小企业都已具备接入 Internet 的能力,充分利用互联网资源和计算机快速处理大规模数据的能力进行网上报税,成为一种可行的解决方案。笔者结合与宁波威尔公司合作开发的远程申报系统,详细阐述了系统中基于 XML 文件的数据交换、PKI 体系的加密技术及多线程服务器等关键技术在 RTDS 系统中的运用。

1　主要业务描述

企业在客户端输入申报数据,构建成 XML 文件,经压缩、加密上传至服务端的受理平台,受理平台进行身份认证后转发到处理平台,处理平台对申

收稿日期: 2005-04-18
作者简介: 钱亚冠(1976—　),男,浙江嵊州人,硕士,助教,主要从事计算机教学和信号与信息处理等相关研究。

报数据进行解密、解压，从原始的 XML 文件中抽取出具体的申报数据，进行税务审核，如果通过，则提交后台数据库。除了处理申报数据外，服务端还要处理来自客户端的各种查询请求。

2 设计目标和总体设计

2.1 系统设计的主要目标

(1)申报过程的高效性。要求申报受理操作能在 5 min 内响应，申报处理结果能在 24 h 内响应。

(2)系统运行的鲁棒性。做到 7 d×24 h 的连续无故障运行。

(3)安全性。安全性涵盖软件的各部分，包括用户权限管理，系统登录必须使用用户名和口令。采用 Internet 方式申报的必须使用 CA 证书。

(4)整个系统具有良好的可扩充性，可维护性。系统数据结构设计合理、稳定、灵活，从结构上保证系统的可扩充性适应系统实施规模的延伸，适应系统使用过程中的业务数据量的增加，随着技术的不断进步，可以方便地升级系统。

2.2 总体设计

整个系统采用客户/服务器架构。客户端为了兼容原来系统，仍然采用 PowerBuilder 开发，服务端采用 J2EE 架构，充分利用了分布式计算的鲁棒性。客户端与服务端通过 Socket 连接交换 XML 文件来实现远程数据交互。分析设计阶段利用 UML 工具进行面向对象的分析和设计。系统主要分为客户端和服务端，其中服务端又进一步分为受理平台、处理平台和后端数据库。服务端系统的用例图参见图 1。

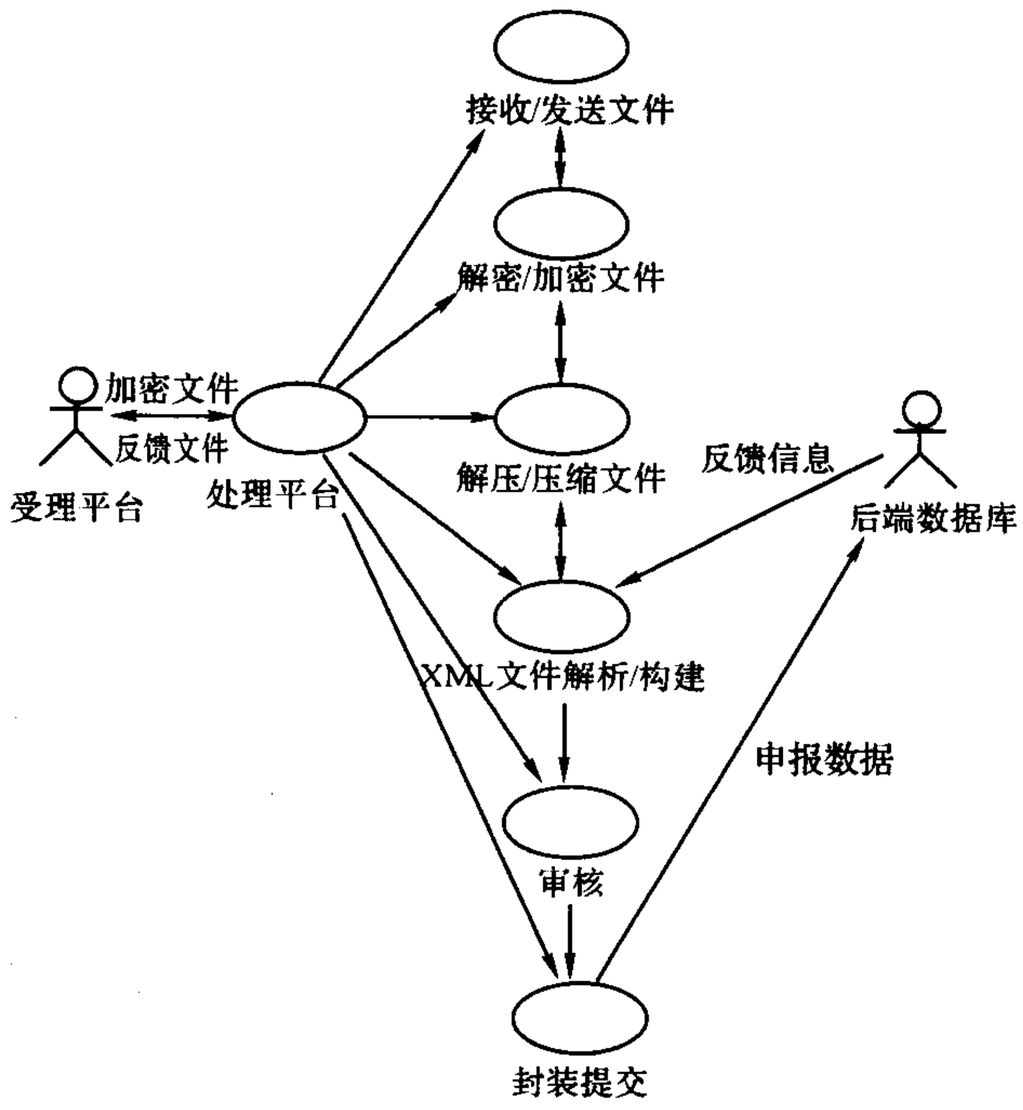

图 1 服务端系统用例图

3 系统网络拓扑结构和功能结构

3.1 硬件环境

客户端为普通的 PC 机，服务端使用 Sun Fire V480 服务器，可提供两个 1.05 GHz 的 UltraSPARC III 处理器，1 GB 的内存及两个 FC-AL 磁盘，100 M 以太网卡，具有优良的稳定性和出色的性能。受理平台和处理平台与处理平台和后端数据库间都有物理隔离器，当受理平台和处理平台在物理上连通时，处理平台和后端数据库在物理上断开；同样，当处理平台和后端数据库连通时，受理平台与处理平台物理断开，从而保证了后端数据库的安全。

3.2 软件环境

客户端操作系统为 win98 或 win2000，SybaseAnywhere 数据库系统；服务端操作系统为 Win2000 Server 或 Linux，数据库管理系统为 Oracle9i 和应用服务器为 Weblogic7。开发工具使用 PowerBuilder8.0，JBuilder8.0，Visual C++6.0，RationalRose。

3.3 软件功能模块

系统主要划分模块见图 2。

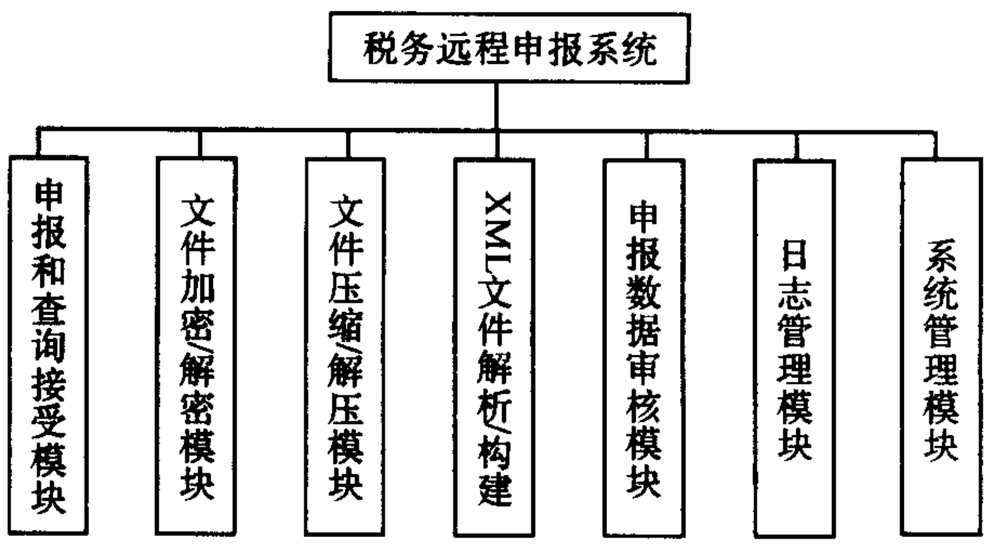

图 2 系统模块组织

4 系统实现的关键技术

4.1 基于 XML 格式的数据交换

XML(Extensible Markup Language)是一种可扩充的元标记语言，可以根据实际需要定义新的标记语言，并为这个标记语言规定它特有的一套标记[1]。XML 的一种重要应用是异种数据源的集成和交互，其自定义性和可扩充性足以表达各种类型的数据。在实际应用系统中，数据可能来自不同的 DBMS 或其他应用系统，它们有各自不同的复杂格

式,需有一种标准的语言进行交互,XML 可以很好地解决这个问题。

在本系统中,从客户端采集到的申报数据被格式化成事先规定好的 XML 文件,再上传至服务端的受理平台,对 XML 文件进行解析后,抽取出上传的各项申报数据进行审核,然后提交数据库。而对于申报结果和客户查询结果,也格式化成 XML 文件返回。客户端与后台数据库都针对 XML 文件交互,这样,即使后台数据库发生变化,也不需要修改众多的客户端程序,只需集中对服务端程序进行维护升级即可。

XML 文档的结构是一个树形等级结构。文档必须有一个唯一的根节点,根节点包含所有其他节点。下面给出一个申报数据 XML 文件样例:

```
<Declare>
    <User>
    <UserID>2003012235</UserID>
<Name>张三</Name>
<Sex>男</Sex>
    <Birthday>1960－05－08</ Birthday>
    <City>宁波</City>
<Phone>85689489</Phone>
    </User>
    <Data>
    …
    </Data>
</Declare>
```

由于申报数据只要从头至尾顺序抽取,不需要随机访问,因此,采用 XML 的简单 API(Simple API for XML,SAX)来实现 XML 文件的解析。与 DOM 相比,SAX 解析器提供更佳的性能优势,其最大的优点是内存消耗小。通过比较标签,解析抽取出的数据放到对应的 Java 实体 Bean 中:

```
public class UserBean {
private String UserID;
private String Name;
private String Sex;
private String BirthDay;
private String Phone;
public void setName(String s) {
        name = s;
}
public String getName() {
        return name;
}
…
}
```

利用此实体 Bean,通过 JDBC 接口实现与后台数据库的操作。

4.2　基于 PKI 体系的申报数据加密

由于税务申报数据属于敏感性数据,因此,在申报传输过程中需要有一套合理的安全方案。本解决方案采用了 PKI 体系,包括 X.509 证书的发放,密钥管理,数字签名,DES 对称加密和 RSA 不对称加密等一系列环节。

DES 加密算法是美国国家标准局(NBS)颁布的一种分组加密算法,由 56 bits 长度的密钥对 64 bits 分组进行加密,变换成 64 bits 的输出,其原理是混淆和散布[2]。就目前而言,56 位密钥的加密强度已证明在商业应用上是安全的。

公开密钥密码系统是由 Diffie,Hellman 和 Merkle 基于数论中的欧拉定理提出的,目的是解决两个最突出的难题:一是密钥分发;二是数字签名。基本原理是:每个用户保存一对密钥——公钥 PK 和私钥 SK,两者不能相互推导[2]。公钥用于加密,私钥用于数字签名和解密,又称不对称加密。本公钥系统用 RSA 算法实现。该算法的安全性基于大数因子分解,因此,选用 1 024 位的密钥长度目前证明是安全的。

DES 算法具有加密效率高的优点,但加密和解密使用同一算法和密钥,密钥和密文必须一同传输,密钥很容易中间被截获;而 RSA 算法的加密效率低,但它是不对称算法,私钥只用于解密,不需随同密文发送,因此,克服了 DES 算法的缺点。结合两者的优点,本加密方案采用 DES 加密算法对申报数据 XML 文件进行加密,而用 RSA 算法对 DES 密钥进行加密,接收方再用私钥对 DES 密钥解密,得到 DES 密钥,再对密文解密。这样既提高了加密效率,又解决了 DES 密钥在网络上传输不安全的问题,其原理如图 3。

服务端加密系统采用 JDK1.4 和 Bouncy Castle 的 JCE 软件包提供的 API 实现。JDK1.4 提供了进行加密解密的接口,具体算法实现的提供者可以选择。笔者选择 Sun 公司的实现软件包,对于 RSA 算法,采用 Bouncy Castle 的 JCE 软件包。加密和解密流程参见加解密时序图(图 4)。

Java 加密扩展简称 JCE,是 Sun 的加密服务软件,包含了加密和密匙生成等功能。JCE 没有规定具

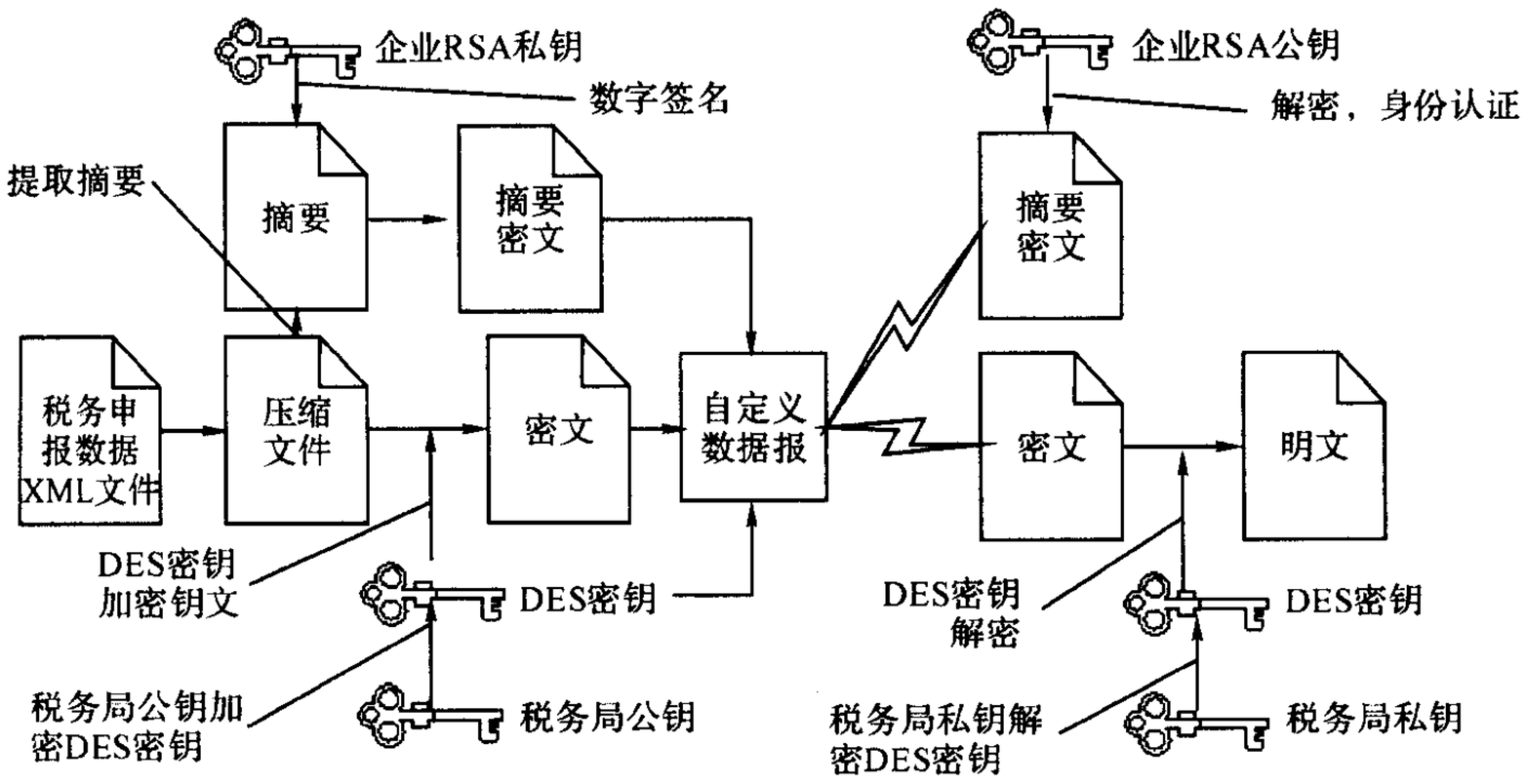

图 3 加解密方案原理

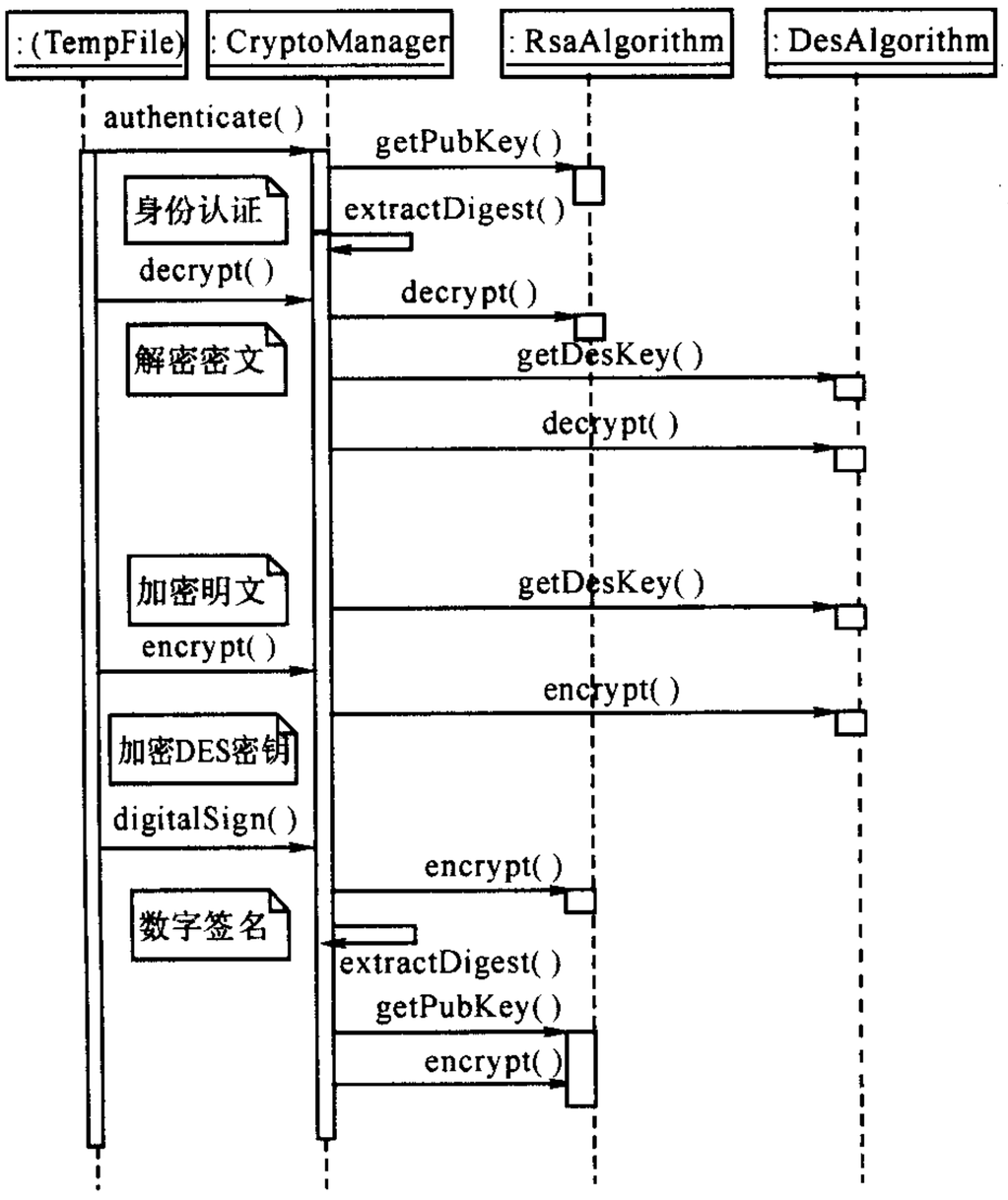

图 4 加解密时序图

体的加密算法，只提供一个框架，加密算法的具体实现可作为服务提供者加入。Java 加密体系结构引入了服务提供者的概念，它用于提供 JDK 安全 API 的密码子集的具体实现。应用程序可以只是简单地请求某类对象（如 Singature 对象）提供某种服务（如 DSA 签名算法），并从某一安装好的提供者上获取该对象的实现方法。载入提供者的方式通常是 getInstance ("PROVIDER")，如签名对象 Signature sig = Signature. getInstance("MD5withRSA")。

4.3 利用多线程机制解决多用户并发访问

税务服务端在申报期间，接受多用户并发访问是个关键问题。根据国家税务总局要求，最多要能接受 100 个用户同时完成申报。为此，笔者采用多线程方式解决这个问题。

简单地说，线程就是一个进程中的多个可并发的控制流。Java 是第一个内置支持线程操作的编程语言，因此，线程调度是由 JVM 来提供的，而不是 OS[3]。以线程代替进程作为处理器调度单位，一方面节省了资源的消耗，另一方面提高了系统的吞吐量[4]。本系统中，以一个线程来处理一个用户连接请求，当一个用户处于 I/O 阻塞（大量的数据库联机查询）等条件时，可以继续响应其他用户的网络连接请求。

考虑到系统中工作线程执行时间较长，因此，没有采用线程池（需要专门运行一个管理线程的程序[3]）的方式。分析如下：设维护 n 个线程的线程池本身的开销时间为 T_{pool}，线程创建时间为 T_s，执行时间 T_e，线程销毁时间 T_k，则在没有线程池的情况下系统有效执行时间比为 $\tau_1 = T_e/(T_s + T_e + T_k)$，而使用线程池时 $\tau_2 = T_e/(T_e + T_{pool})$，一般情况下，上述时间单位以 ms 计，$T_e < T_s + T_k$，此时 $\tau_1 \ll \tau_2$，因此，使用线程池有明显的优势。因为使用线程池可以避免频繁的创建和销毁线程，这两个操作将占用大量的系统资源，使响应时间变慢。本系统中的工作线程由于有大量的数据库查询和数据审核操作，执行时间 T_e 比较长，以秒计算，因此，$\tau_1 \approx \tau_2$，使用线程池在本系统中没有体现大的优势，反而需要有一个专门的进程管理线程池，可通过表 1 中的数据来分析比较。

为此，采用服务端程序启动时就生成若干线程，在整个服务程序生命周期中一直存在，即不需要频繁地创建和销毁，也不需要线程池来维护，简化的实现代码如下：

表 1　三种线程方案的比较

	描述	创建时间 T_s/ms	执行时间 T_e/ms	销毁时间 T_k/ms	系统管理开销/ms	有效执行时间比
临时创建线程	有请求时创建线程,任务完成后立即销毁	0.2	0.1	0.3	无	τ=0.1/(0.2+0.1+0.3)=16.67%
线程池	动态维护线程池中的线程数量	在系统空闲时创建,可忽略	0.1 1	在系统空闲时销毁,可忽略	0.2 0.2	τ=0.1/(0.1 + 0.05)=66.6% τ=1/(1+0.2)=83.33%
预先创建固定数量线程	线程数量固定	系统启动时创建,可忽略	1	系统关闭时销毁,可忽略	无	$\tau\approx$100%

注:表中时间为概率意义下的平均时间

```
public class DeclareServer{    //申报服务器类
    public static final int PORT=8888;//服务器端口号
    public static final int THREADS=100;//初始线程数
    public DeclareServer(int port, int threads){
        ServerSocket serverSocket;
        ServerSocket = new serverSocket(port);
        for(int i = 0; i < threads; i++){
new Thread(new Handler(serverSock, i)).start();……①
        }
    }
    public Handler extends Thread{
        ServerSocket serverSocket;
        int threadNum;
        public Handler(ServerSocket s, int i){
            super();
            serverSocket = s;
            threadNum = i;
        }
        public void run(){
            while(true){……②
              snchronized(servrSocket){
                Socket clientSocket = serverSocket.accept();……③
              }
            //线程中要完成的具体工作
            ……
            }
      }
}
```

在上述实现代码中:

①服务器启动时就产生 i 个服务线程,这样做是为了加快以后的客户端连接速度,同时,可避免重复创建 Handler 对象,避免 Java 的垃圾回收带来的消耗。

②每个线程在完成工作后并不销毁,而是继续等待服务下一个连接。

③在 ServerSocket 类中,accept()方法不是串行化的,因此,在这里必须作为临界区管理。原因是当一个线程在侦听连接时,其他 $n-1$ 个线程必须要处于 wait 状态,当这个线程侦听到新连接后,它就退出临界区,由余下的线程竞争得到接下去的侦听机会。而当没有客户的连接请求时,只有一个线程在侦听。和通常一个主线程在侦听连接,侦听到后产生一个新的工作线程的做法相比较,上述方案的优点在于侦听到连接后不需要重新生成一个新线程,和线程池比较,可省去维护线程池本身的开销。

5　系统测试结果

笔者在 Internet 环境下对系统进行了压力测试,分别测试模拟 10、30、50、100、200 个并发用户,系统连接响应时间分别为:0.3,1.2,2.8,7,13s。从目前的测试数据来看,系统在少于 100 个客户程序同时发起请求时,服务器程序能及时响应,超过 100 个连接时,响应时间有稍微的延迟。总体上来说,本系统能很好符合 100 个并发用户实时响应的设计要求。

参考文献:

[1]　孙一中. XML 理论和应用基础[M]. 北京:北京邮电大学出版社,2000.

[2]　William Stallings. 密码编码学与网络安全:原理与实践[M]. 第三版. 刘玉珍,王丽娜,傅建明,等译. 北京:电子工业出版社,2004.

[3]　Abraham Silberschatz. Operating System Concepts (Sixth Edition)[M]. Hoboken, New Jersey: John Wiley & Sons, Inc,2001.

[4]　徐迎晓. Java Socket、线程及 RMI 开发框架[J]. 计算机应用,2001,(12):75－76.