电信业务自动开停机系统中的实时文件传输和进程通信

曾广平　阳绿云　（长沙中南工业大学信息工程学院　410083）

摘要：实时文件传输和进程通信是在异构网环境中进行自动开停机系统开发的两个关键问题。本文分析了这两个问题，提出了在电信业务常见的网络环境下用非交互式 ftp 方法从不同数据源传输数据文件的分布式数据采集方法和利用 Socket 编程实现不同操作平台的进程通信的方法。

关键词：文件传输　进程通信　Socket 编程

一、前言

随着计算机网络应用的发展，网络构造越来越多样。基于不同平台的、结构不同的网络会同时并存于一个企业的各个应用系统之中，一个企业在以往各个时期建立起来的各种子网的拓扑结构或软、硬件平台往往不是单一的。为了保护先期设备和技术投资，在进行应用系统开发或技术更新研究时，必然会遇到各种网络之间的文件传输和数据交换，以及不同网络之间的进程通信问题。本文研究的对象是电信业务系统（见图 1）的自动开停机系统。

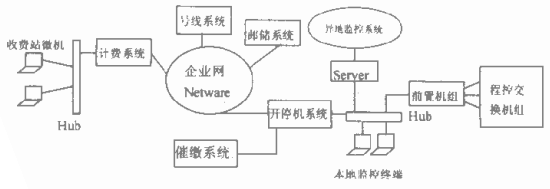

图 1　电信业务系统连接关系示意图

在上图中，新开发的是自动开停机系统。它与其他原有的计费系统、号线系统、邮储系统等一起构成一个电信业务系统。图中所示的与企业网相连接的多个业务子网的结构不是单一的，有的是总线型结构，有的是树型或星型结构；布线方式各不相同；操作平台也不相同，有使用专用服务器的，也有使用 PC 机作服务器的；有 Windows 平台的 PC 或 SUN 工作站，更多的是 Unix 平台的服务器或微机。本地监控系统也是多种平台的，但结构单一，为 Ethernet 结构。而自动开停机系统要在各个子网中协同工作，并完成它自己的专门功能。它要从连接在企业网周边的计费系统网络、号线系统网络、邮政储蓄系统网络中取得要加工的原始数据，按照约定的催缴条件生成催缴数据表提供给功能上与之相关的催缴系统；按照催缴结果和约定的停机和开机条件通过前置机系统向程控交换机发送开停机用户号的命令，并处理反馈的命令执行结果。因此，它与监控平台（前置机组）的关系是进程控制和数据交换的双重关系。开停机操作需要与企业网交互，也需要通过前置机与程控交换机通信。前者涉及实时文件传输方法，后者则要求处理好不同操作平台之间的进程通信。

二、用非交互式 ftp 控件实现数据文件的实时传输

在不同的网络间通过远程数据库的访问实现数据资源的提取与共享不是个新问题。特别是基于 Client/Server 模式的开发工具提供了较为完善的远程数据库访问方法。基于设备和数据的安全性的要求，结合实时性的要求，考虑到实际上不需要与庞大的数据库中的大多数数据表单打交道，我们实际选择了比较简便的方法，即在数据库中生成开停机系统所需要的数据的文本文件，由自动开停机系统定时到各个数据源提取的。为了达到实时性的要求，一种办法是当数据库中相关的数据发生刷新时，立即与开停机系统建立联系并发送数据文件到开停机主机系统。另一种办法是在开停机系统中定时轮流向数据源点发送请求，以便得到新的数据文件。一般的 ftp 工具只能由用户交互式地输入命令，才能建立会话，进行指定的数据传输。好在不管是何种平台，ftp 协议和操作是一样的。所以我们可以利用 windows 的控件 ftp 来实现非交互式的定时扫描的文件传输工具。其算法（用 Delphi3.0 实现）伪代码如下：

/ * 与数据源文件服务器连接并传输文件的过程 * /

```
.根据远程主机名设置
case 远程主机名 of
  计费系统服务器:
    {
      设置相应的用户名;
      设置相应的口令;
      设置远程主机名;
    };
    邮储系统服务器:{
      设置相应的用户名、口令、远程主机名属性
                    };
    号线系统服务器:{设置相应的用户名、口令、远程主机名属性
    }
    设定 ftp 端口号;
    建立 ftp 连接;
.取指定文件过程 / * fetchfile * /
    {
    .初始化;
    .设定欲取的文件名;
    .引用 GetFile 方法从文件服务器传输文件;
    }
```

这种方法用起来是比较灵活的，不管是用本机的消息触发连接和传输数据文件，还是利用数据源的更新事件来触发，对于数据源的保护都是可靠的。

三、用 TCP/IP 编程实现系统与 Unix 平台之间的进程通信

1. 进程通信原理

在基于 TCP/IP 协议的网络中，套接字(Socket)是网络通信的基本操作单元，他提供了不同主机间进程双向通信的端口。这些要通信的进程在通信之前各自建立一个 Socket，并通过对 Socket 的读/写操作实现网络通信。

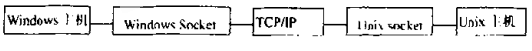

图 2　进程通信实体连接图

在本文研究的实例中，进程通信实体连接图如下(见图 2)：

在 Unix 主机上有一个 TCP 并发服务器，它守候着来自监控系统所有终端的 Socket(套接字)请求并建立连接，唤醒相应的服务进程为客户进程服务。这里要注意的是 Socket 编程在不同的操作系统中有所不同，主要是 Socket 函数在不同系统中的定义形式不一样。在 Windows95 编程环境中的 Windows Sockets API 函数集其实是基于 Unix Socket 函数族的。在 Delphi3.0 中它们被嵌入到可视化编程部件 Clientsocket 和 ServerSocket 之中；但这些编程接口的基础——TCP/IP 协议是一致的，无论是 Dos、Windows3.2、Windows95、Windows NT、Unix 或是其他操作平台，他们之间都可以以 TCP/IP 进行通信。Windows 客户端 ClientSocket 与 Unix 服务器端 Socket 的通信过程如图 3 所示：

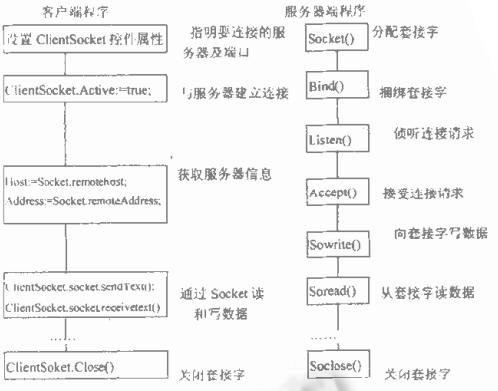

图 3　windows sockets 与 Unix Socket 通信流程图

在 Unix 服务器上有一个守候进程，它在初始化端口之后调用 Accept()过程来接受一个新的连接请求，当接收到一个连接请求时进行可能的连接，分配相应的套接字，并唤醒服务进程 server 来为客户方服务；server 进程的功能是在双方建立 Socket 连接之后，根据来自客户程序的服务请求和附带的信息进行处理，这种处理往往又是唤醒其他的服务进程来实现的。server 还负责将服务器方进程的信息通过 Socket 返回给客户方。数据的传送是通过 Socket 的 read()函数和 write()函数来实现的。在完成所有要求的处理后，根据客户方发出的"命令"来调用 close()函数关闭此次连接。

2. 跨平台进程通信的算法

在客户方的 Windows 下的应用系统，我们用 Delphi3.0 设计界面并实现，界面包括 Winsock 控件。其中进行通信的关键算法伪代码如下：

```
    /* 建立 socket 连接并激活相应的程控交换机端口
*/
    {
    初始化;
    提取电话号码特征字;
    建立 socket 连接;
    Delay(2000);//延迟一段时间
    If 成功建立 socket 连接
      {
      case 端口名特征字 of
      'b':
        {
        发送字符串'21b 2400 s1240 o';
        Delay(4000); //延迟 4 秒时间;
        if 收到端口占用标志{
          显示端口被占用信息;
          关闭此次连接;
          }
          else 收到端口激活标志 {
          sendtext('mm;');//发送进一步命令
          延迟几秒钟时间;
          if 收到命令提示符
            {
            置激活端口成功标志
            };
          else 置激活端口失败标志;
      //其他端口情况与上面类似,限于篇幅,略去。
      //连接失败,显示有关信息
        else {
          显示激活交换机失败信息;
          进行相关设置;
          }
/* 与交换机端口通信的过程 */
if 端口激活成功 {
    通过 Socket 连接往服务器方发送第一条命令;
    Delay(3000); //延迟一段时间以适应交换机的响应
速度
    继续发送命令;
    Delay(5000); //延迟一段时间;
    }
    else
    {
    继续发送命令;
    Delay(5000); //延迟一段时间
    };
    //通过 Socket. receivetext 接受数据,根据接受到的
数据决定客户方的数据库操作
    if 收到期望的字串
    {
    关闭此次连接;
    修改主数据表的字段;
    在发送命令表单中删除这条命令记录;
    }//if
  扫描至下一条命令;
  }//if 端口激活成功
}. //发命令过程
```

四、结论

利用 ftp 控件建立 ftp 会话,可以控制系统要求的数据文件的传输;通过 Socket 与 WinSockets 之间的通信,我们在客户方发出命令——服务请求,让服务器守候进程按请求唤醒有关服务进程完成指定的操作,而且可以监视来自守候进程回送的信息。整个系统按照设计的流程工作,达到了预期的目的。如果在程控交换机的响应速度上能有较大的提高,例如分配专门的交换机端口供系统使用,则系统的实时性将会有较大的改善。就系统本身而言,如采用基于 C/S 模型的数据库共享技术来实现数据共享,可以提高系统的性能。

参考文献

[1] 袁隽,谷保山,丛扬编著. NOVELL 网络开发环境的构造与 TCP/IP 编程指南. 北京:学苑出版社,1994. 144154

[2] 刘欢. 用 Java 实现 Windows 系统与 UNIX 进程间通信. 电脑编程技巧与维护,1998(4),4951

[3] 孙玉芳主审,吴健、朱光远、查良钿译校. UNIX 系统 V/386 第 4 版 网络程序员指南,电子工业出版

(来稿时间:1999 年 3 月)