第 33 卷 第 3 期 计 算 机 工 程 2007 年 2 月
Vol.33 No.3 Computer Engineering February 2007
·工程应用技术与实现· 文章编号：1000—3428(2007)03—0228—03 文献标识码：A 中图分类号：TP393

# 面向 LonWorks 网络的 OPC Server 设计与实现

杨启亮，邢建春，王 平
(解放军理工大学工程兵工程学院国防工程智能化技术研究中心，南京 210007)

**摘 要：**OPC 是工业控制领域中应用程序互操作规范。针对自主研制的 LonWorks 网络，实现了其 OPC server，使得 OPC client 软件能方便地从本地或远程访问这种网络。在讨论了应用程序的实时数据库、图形用户接口、OPC 对象、I/O 读写等实现原理的基础上，研究了该 OPC server 的写命令处理和多线程同步等关键技术。

**关键词：**LonWorks；OPC server；COM/DCOM

# Design and Implementation of the OPC Server Oriented to One LonWorks Network

YANG Qiliang, XING Jianchun, WANG Ping
(Research Institute of Control and Measurement, Engineering Institute of Engineering Corps, PLA's University of Sci. & Tech., Nanjing 210007)

**【Abstract】** OPC(OLE for process control) is the co-operational specification between the applications in industrial control. One OPC DA server is implemented for the LonWorks network, which enables OPC clients to locally or remotely access this LonWorks network in an easy way. On the base of the discussion of the implementation principle of the server such as real time database, GUI, OPC objects and I/O read and write and so on, the key technologies such as processing write commands and synchronizing multi-threads are probed.

**【Key words】** LonWorks; OPC server; COM/DCOM

OPC(OLE for Process Control)是OLE技术在工业过程控制领域的应用[1,2]，是利用Microsoft的COM/DCOM技术实现的一套标准软件接口[3,4]。利用这些接口，客户程序可以按照统一的数据访问标准与不同厂商的软硬件产品进行通信，使得自动化系统集成变得非常简单。OPC规范最初仅是工业控制软件对现场设备的数据存取规范，经过几年的发展，现在包括了数据存取(Data Access)规范、报警和事件(Alarms and Events)规范、历史数据存取(Historical Data Access)规范、批量过程(Batch)规范和安全性(Security)规范，以及正在开发的OPC XML和OPC Data eXchange规范。

针对自主研发的LonWorks网络，开发出了该网络的OPC DA Server(OPC Server)。实现的OPC Server具有友好的用户界面、较好的数据交换能力、错误状态返回和错误日志记录、设备仿真、允许自定义设备的扫描周期、支持字符串类型等特点。LonWorks OPC Server使得自主研制的LonWorks网络能同任何的符合OPC规范的软件进行通信，大大拓展了这种LonWorks网络的应用范围。

## 1 LonWorks 网络概述

LonWorks 网络是在全球获得广泛应用的一种现场总线网络，自主研制的LonWorks网络基于通用的LonWorks技术，符合 LonTalk 协议。该网络包含一系列测控模块，支持模拟量输入、开关量输入输出、计数等信号形式。

用户程序对 LonWorks 模块的读取是通过调用自主开发的 LonWorks 网络驱动子程序来完成的。

LonWorks 网络驱动子程序是以 API 函数的形式提供的，这些函数都封装于 Lmdrv.dll 中。与访问 I/O 数据有关的函数主要有：

(1)LmPort，指定本机网卡的通信口地址；

(2)LmState，读取模块中输入输出开关量通道的状态；

(3)LmNum，读取模块中模拟量通道的数值；

(4)LmCmd，从上位机向模块传递各种命令，如开关量输出通道状态等。

## 2 LonWorks OPC Server 体系结构

LonWorks OPC Server 是完全基于 LonWorks 网络通信协议，在 LonWorks 网络驱动子程序的基础上开发的。

图 1 是 LonWorks OPC Server 的系统结构。软件主要包括以下模块。

(1)用户接口模块：此模块是同用户交互的图形接口，提供一个基于文档/视图技术的图形用户界面。

(2)OPC 数据访问接口模块：此模块用于实现符合 OPC DA2.0(OPC Data Access 2.0)规范的各种对象及相关接口。

(3)数据存储模块：该模块包括设备列表和实时数据库两个部分。设备列表存储用户对于设备的配置信息；实时数据库用于存储设备各通道实时值，供 I/O 读处理模块调用。

(4)扫描线程模块：该模块根据用户配置的设备列表，按照每个设备设定的周期扫描 LonWorks 网络，并将扫描结果存到实时数据库中。

(5)I/O 读处理模块：该模块循环处理客户端发来的读数据包，根据数据包的内容向实时数据库读取通道值，并置上时间和质量标签，然后返回给客户端。

**作者简介：**杨启亮(1975－)，男，硕士、讲师，主研方向：计算机分布式控制系统；邢建春，硕士、教授；王 平，硕士、副教授

**收稿日期：**2006-02-08 **E-mail：**yql@893.com.cn

(6)I/O 写处理模块：该模块解析客户端通过 OPC 数据接口发来的写数据包，将写数据包处理成 LonWorks 网络的命令串，并添加到写消息列表中。

(7)LonWorks 网络通信接口模块：该模块提供了一组 API 函数，这些函数都封装于 Lmdrv.dll 中，实现对现场 I/O 设备的读写。

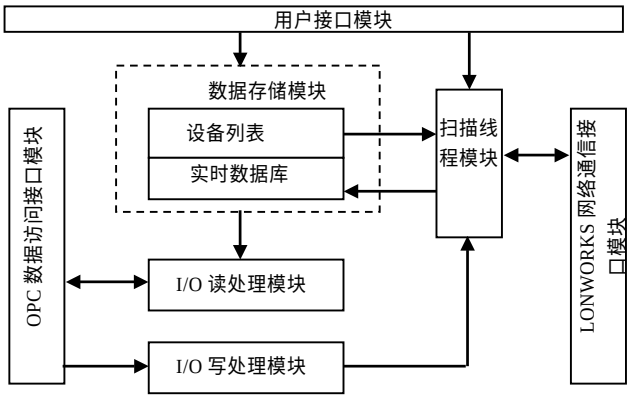


**图 1 LonWorks OPC Server 的软件结构**

## 3 LonWorks OPC Server 应用程序实现原理

### 3.1 实时数据库的构建

实时数据库是整个 LonWorks OPC Server 的核心，其实时地存储和刷新现场设备来的运行数据，同时保存在后台扫描线程，启动时从用户配置中加载最新的设备配置信息，包括设备名称、设备类型、设备地址等信息。扫描线程根据配置信息扫描 I/O 设备，同时将扫描结果实时存入实时库的相应位置。实时数据库驻留于内存中，有效保证 OPC Client 从实时库中读取运行数据的效率。实时数据库的逻辑结构如下。

```
//实时数据库
typedef struct DeviceRTValue
{
  CString deviceType;        //设备类型
  long      address;          //设备地址
  float     AIRTValue[80];   //模拟量实时值
  long      DIRTValue[240]; //开关量实时值
  WORD      ComIsErr;       //设备通信状态
  long      LastScanTime;   //设备上次扫描时间
  long      ScanPeriod;      //设备扫描周期
  BOOL      IsSim;           //是否仿真
} DEVICERTVALUE;
```

实时数据库对于整个应用程序而言是全局数据库，实现了其中的数据在应用程序中不同线程间的共享。

### 3.2 用户图形接口设计

普通的 OPC Server 分为进程外 Server(以 EXE 可执行文件的形式存在)和进程内 Server(以 DLL 动态链接库的形式存在)。进程外 Server 能为 Client 跨进程跨网络所访问，而进程内 Server 则不能。根据实际应用需要，LonWorks OPC Server 实现成进程外 Server，提供可交互的人机界面，具有独立的进程地址空间。主程序界面具有文档/视图框架，便于用户操作。

LonWorks OPC Server 采用设备、组和标签 3 个层次来管理数据。设备是指符合 LonWorks 通信协议的 I/O 控制模块，标签代表了 I/O 控制模块的 I/O 通道，组是 I/O 通道的一个逻辑划分。主界面采用 Windows 资源管理器风格用于分级显示设备、组和标签，用树状视图来显示设备和组，用列表视图显示标签及其属性。对于设备和标签都设计了相应的属性对话框来允许用户编辑和修改其属性值。图 2 为 LonWorks OPC Server 的用户主界面。

**图 2 LonWorks OPC Server 的用户主界面**

图形用户接口允许用户执行添加、删除、编辑、复制等操作，将用户配置保存为文件，在程序启动时自动加载。

### 3.3 面向对象的 OPC 数据访问对象设计

LonWorks OPC Server 应用程序的 OPC 对象包括 Server 对象、Group 对象、Item 对象。Server 对象维护有关 Server 的信息并作为 OPC 组对象的包容器，可动态创建或释放 Group 对象；Group 对象除了维护其自身信息外，还提供了包容 Item 对象的机制，逻辑上管理 Item 对象。Item 对象则表示 OPC Server 的数据连接。从定制接口的角度看，Item 对象不可以由 OPC Client 直接操作的 COM 对象，所以其没有定义外部接口，所有对 Item 的操作都是通过包容该 Item 的 Group 对象进行的。Server 对象和 Group 对象是真正的 COM 对象，二者之间是聚合关系，即 Server 对象创建 Group 对象后，将 Group 对象的指针传给 Client，由 Client 直接操纵 Group 对象。图 3 示意了 OPC Server 中的对象与 OPC Client 的相互关系。

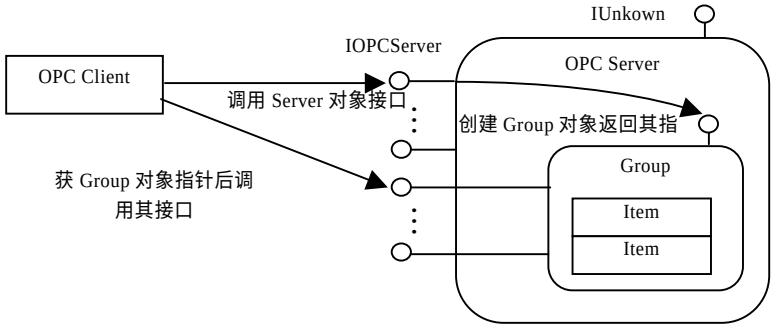


**图 3 OPC Server 中对象与 Client 的相互关系**

### 3.4 I/O 读写处理线程

对 I/O 读写的合理处理是提高 OPC Server 工作效率的关键。

LonWorks OPC Server 设计了一个 I/O 读写线程用于实现对 I/O 设备的读写。该线程在用户设备配置文档加载时创建。线程根据设备配置信息按照设定周期扫描相应 I/O 设备通道，并将扫描结果写到数据存储区的内存数据库中供 I/O 读处理模块读取。在扫描的过程中，线程同时检查写标志 WriteCmdIsComing，若 WriteCmdIsComing 为 True，则说明在写命令消息列表中存在写命令，线程按照取列表数据算法读取写消息列表中的写命令串，并逐一发送到 I/O 设备中。

该线程优先处理写，写命令处理完后再进入读循环，确保写命令得到及时响应。图 4 是 I/O 读写线程的程序流程。线程的主程序部分是一个死循环，该循环按照设备列表定时

扫描设备，当有写命令来临时优先处理写命令，写命令处理完后返回继续扫描设备。

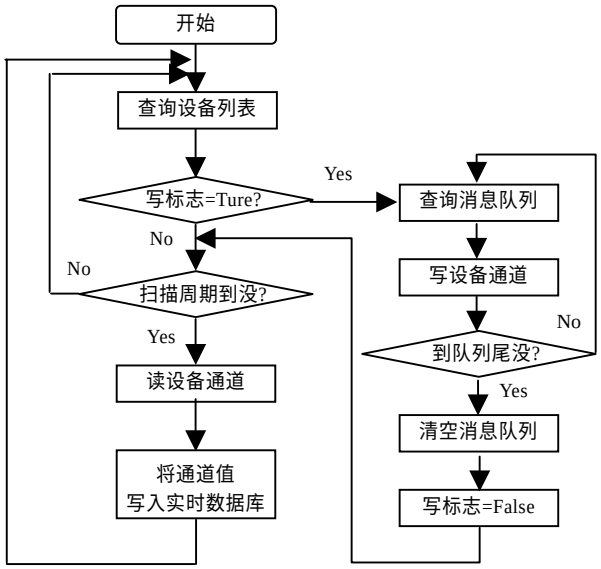


**图 4 I/O 读写线程的程序流程**

## 4 关键实现过程及技术

### 4.1 写命令处理

从 OPC Client 发送来的各种各样的控制命令：有的是单个命令，有的是批命令，有的是字符型命令，有的是数值型命令。对这些不同种类的命令的合理处理是整个 LonWorks OPC Server 实现的关键。本应用程序将写命令处理分为命令封装、命令暂存和命令转发 3 个阶段。

(1)命令封装是指将 Client 发来的各种命令按照 LonWorks 网络协议封装为统一的格式。定义一个全局结构 WriteCmdPacket 来封装每个控制命令，该结构严格按照 LonWorks 通信协议实现，结构中各域的具体含义如下。

```
typedef struct WriteCmdPacket
{
  int    p;                    //LonWorks 模块地址
  char   szCmd[32];            //LonWorks 命令串
  int    cmdParaNum;           // LonWorks 命令所带参数的个数
  float A[2];                  // LonWorks 命令所带的参数
}WRITECMDPACKET;
```

写处理函数 WriteTag 根据从 Client 写来的首先判断 Tag 的寄存器类型，然后再根据不同的寄存器类型和 Tag 值生成不同种的 LonWorks 命令格式串，最后将命令信息封装到 WriteCmdPacket 结构中。

(2)命令暂存是将已经封装好的写命令包存入到列表中，等候专门的 I/O 处理线程来提取。应用程序定义了一个全局的命令列表，用于存储每次从 OPC Client 发送来的所有封装好的命令。该列表采用具有 WriteCmdPacket 类型的 MFC 标准模板库(STL)CList 模板类实现。命令暂存的过程实质是将每个 WriteCmdPacket 向 CList 中添加的过程。以下为封装好的 LonWorks 控制命令向 CList 列表实现添加的程序代码。

```
OneCmdPac.p =p;                                    //封装模块地址
    strcpy(OneCmdPac.szCmd,szCmd);          //封装命令字符串
    OneCmdPac.cmdParaNum=cmdParaNum;     //封装参数个数
    memcpy(OneCmdPac.A,xVal,sizeof(xVal));
  //封装参数数组
    POSITION    pos=theDoc->WriteCMDList.GetHeadPosition();
  //置指针到表头
  if( pos == NULL )
  {
   theDoc->WriteCMDList.AddHead( OneCmdPac );
  //添加命令包到表头
      }
      else
          theDoc->WriteCMDList.AddTail( OneCmdPac );
  //添加命令包到表尾
```

每次从 Client 来的所有控制命令都封装好并存入到命令列表后，将全局标志 WriteCmdIsComing 置为 True，命令列表中存在控制命令并已经添加完毕，去激活 I/O 处理线程中的写出代码，从而进入命令转发阶段。

(3)命令转发阶段命令列表中的控制命令被逐一取出，并通过专门的 LonWorks 命令发送 API——LmCmd 函数将命令转发到 LonWorks 网络中，进而驱动现场的实际设备。以下为命令转发阶段的程序代码，程序运用 MYCSLock 类来保护当前的命令列表，防止在取命令的过程中其它线程修改命令列表。

```
if (WriteCmdIsComing)
          //每次扫描时都要判断一次
{
MYCSLock wait(&ReadIOCS);                    //保护写命令列表
int cmdNum=theDoc->WriteCMDList.GetCount(); //命令数量
POSITION pos=theDoc->WriteCMDList.GetHeadPosition();
//移到列表头
for (j=0;j<cmdNum;j++)
{
  WriteCmdPacket    onePac=theDoc->WriteCMDList.GetAt(pos);
  //取出一个命令
e=LmCmd(&(onePac.p),onePac.szCmd,onePac.cmdParaNum,one
Pac.A); //发送到 LonWorks 网络
  theDoc->WriteCMDList.GetNext(pos);
}
theDoc->WriteCMDList.RemoveAll(); //清空当前列表
WriteCmdIsComing=FALSE;              //复位标志
}
```

### 4.2 多线程及其同步技术

LonWorks OPC Server 应用程序内含多个线程，如 OPC 主线程、Group 扫描线程、I/O 扫描线程等。为了防止线程相互间的数据访问冲突，必须要对这些线程进行同步。

线程同步主要有临界区(Critical Section)、互斥量(Mutex)、信号量(Semaphore)、事件(Event)和可等的计时器(Waitable Timer)等 5 种同步技术。在这 5 种同步对象中，除了临界区外其余都是内核对象。临界区不被操作系统的低级部件管理且不为句柄所操纵，因此它是最易于使用和理解的同步对象。

应用程序中采用基于临界区对象的线程同步机制。临界区一次只允许一个线程取得对某个数据区的访问权。一个临界区在同一时间内只能被一个线程所访问，其他线程必须在临界区中的线程离开后才能进入。使用临界区同步主要使用 3 个 API 函数：InitializeCriticalSection，EnterCriticalSection 与 LeaveCriticalSection。InitializeCriticalSection 用于初始化临界区，其必须在 EnterCriticalSection 调用前完成，而 EnterCriticalSection 使线程取得对临界区的所有权，LeaveCriticalSection 使线程放弃对临界区的所有权。

（下转第 242 页）