大型养路机车监控系统通信服务器设计与实现①

Design and Realization of Online Monitoring System Communication Server in Huge Railway-Maintaining Trains

吴桂清　薛　进　李　海　(湖南大学 电气与信息工程学院 长沙 410082)

摘　要： 对铁路大型养路机车在线监控系统中的通信服务器作了详细介绍，对服务器各功能模块分别作了说明并给出了各模块的流程图；对服务器采用的关键技术作了详细描述并给出部分代码。

关键词： 服务器 监控系统 线程同步 重叠 I/O 模型

远程无线监控系统是安防系统的一个重要分支，由于其移动性强、安装灵活快捷、可用于恶劣环境等优点，使其广泛应用于公安、交通、金融、工业、电力、水利等领域。

本文基于铁路大型养路机车在线监控系统，对系统模型作了简要介绍，并对系统中通信服务器部分(作者所做的工作)的设计与实现作了详细说明。

1　系统模型介绍

本系统是基于 GPRS 网络的 C/S、B/S 混合模型监控系统，主要实现对广铁集团各种大型铁路养路机车的作业参数与工作状态的实时在线监控及历史数据的保存及查询。系统模型如图 1 所示。

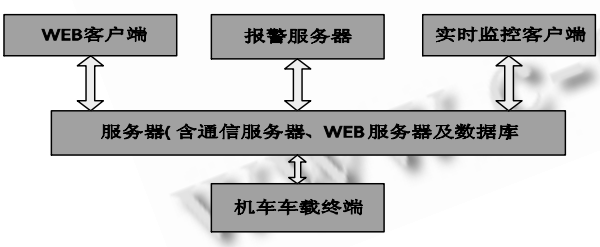

图 1　监控系统模型

机车车载终端通过 CAN 模块对机车各作业参数及工作状态数据进行实时采集，并通过 GPS 模块对 GPS 数据进行采集，然后通过 GRPS 模块将数据上传至通信服务器。通信服务对接收到的数据进行处理后将其通过网络发送到实时监控客户端处，同时同步将数据存入数据库。WEB 服务器根据 WEB 客户端要求从数据库中读取相应数据发送到 WEB 客户端。

2　通信服务器设计

通信服务器是系统的数据中转站，负责车载终端数据的及时上传及实时监控客户端配置命令的下传，并将数据写入数据库。通信服务器是一个基于多线程技术的多对多的服务器，用 VC++6.0 及 SQL Server 2000 实现。服务器结构如图 2 所示：

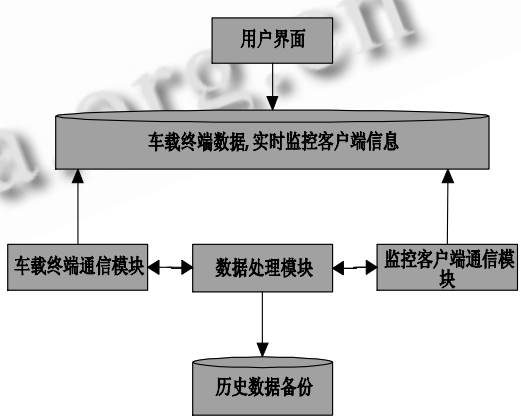

图 2　通信服务器结构

2.1 通信协议设计

由于对数据可靠性有较高的要求，故通信协议的设计在通信服务器的设计之中起着极为重要的作用。

① 基金项目:国家高技术研究发展计划(863)(2008AA04Z214)
收稿时间:2009-01-04

通信协议的设计必须充分考虑数据包的完整性检查，及数据内容的校验，以保证数据可靠性。基于此原因，通信协议数据包格式如下设计：

消息头	长度域	类型域	内容域	校验码	消息尾

其中，消息头采用字符串“$$$”，消息尾采用字符串“***”，长度域从类型域算起(含类型域)，至校验码(不含校验码)止，以字节为单位。类型域标志各种不同的消息类型，采用一个字节的硬编码，例如 0xB4 为实时数据消息，由机车终端发给通信服务器并由服务器转发给实时监控客户端。内容域则由不同类型的消息内容决定。校验码为长度域与校验码字段中间(不含长度域与校验码)以字节为单位求和的最后一个字节。

消息类型包括实时数据消息、配置数据传送周期消息等。其中最为重要的是实时数据消息。实时数据消息的内容域包括 12 个字节的 GPS 数据，以及三类数量不等的机车数据，包括模拟量，数字量，脉冲量。各类数据的数量由不同类型的机车决定。服务器接收到实时数据消息并作相应分析与校验之后将其以一定周期写入数据库。

2.2 车载终端通信模块设计

车载终端通信模块实现了服务器与车载终端的数据通信。本模块采用基于 TCP 协议的 socket 通信机制。TCP 是 TCP/IP 协议族的传输层协议，它在 IP 协议的基础上提供了基于数据流的面向连接的可靠的数据传输服务。

车载终端模块的程序流程图如图 3 所示。

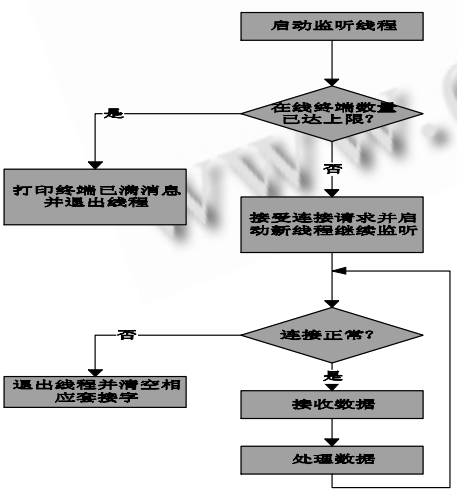

图 3　车载终端模块流程图

每台机车的车载终端有各自相应的终端编号，系统以此区别不同机车。此编号包含在终端上传的实时数据消息中。当服务器接收到新的连接请求时将启动新线程重新进行监听，并将当前线程作为发起连接请求的终端的工作线程，给此连接分配相应的套接字，并在此线程中进行数据的收发与处理。当又有新的终端连接请求到来时继续启动新线程进行监听，如此循环直到在线终端数量达到上限。

服务器在初始化工作中将建立一个终端套接字数组并全部置为 NULL，在和终端建立连接后将遍历此数组并从中找出第一个空的套接字将其分配给此连接，当终端退出之后将此套接字清空并重新放入套接字数组以供后续使用。

2.3 监控客户端通信模块设计

监控客户端通信模块实现了服务器与监控客户端的数据通信。监控客户端通信模块的设计与车载终端极为相似，也是采用面向连接的 TCP 协议，并采用多线程技术。

监控客户端模块的程序流程图如图 4 所示。

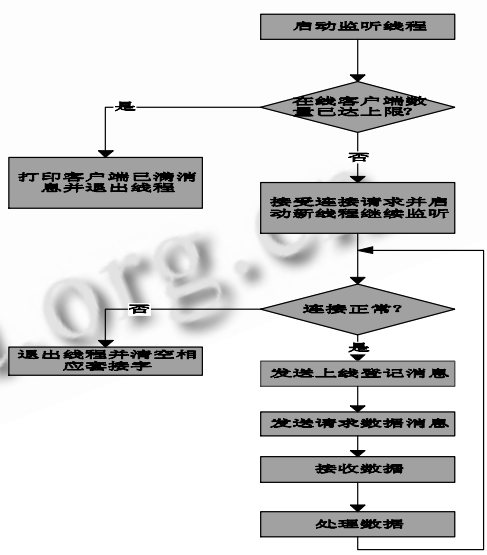

图 4　监控客户端模块流程图

系统用客户端的 IP 地址区分各不同客户端。服务器接收到客户端的上线登记消息以后将检查数据库中是否有此客户端，如果没有则将其登记到数据库中，如果有则将在线标志重置。服务器在接收到客户端的请求数据消息之后将此消息放入消息队列。数据处理模块从数据队列中提取消息并解析，然后根据消息内容给客户端发送所请求的机车数据。

2.4 数据处理模块设计

数据处理模块从消息队列中提取消息并解析，根据消息内容作出相应动作。消息的提取按如下步骤进行：首先从消息队列中查找出现的第一个消息头，如果到达队列尾仍未找到，则认为队列中的数据无效，将此数据段从队列中删除；如果找到消息头则继续查找相应的消息尾，如果未找到消息尾则不对队列作任何处理，如果找到消息尾则将此消息提取出来并解析。如此循环。

消息的解析分两步，首先对消息长度进行计算并与消息中的长度域进行比较。如果不等则认为消息错误，将此消息丢弃；如果相等则进行下一步：求校验和并与消息中的校验和进行比较。相等则认为消息有效，提取其类型域并作出相应动作。

数据处理模块流程图如图 5 所示。

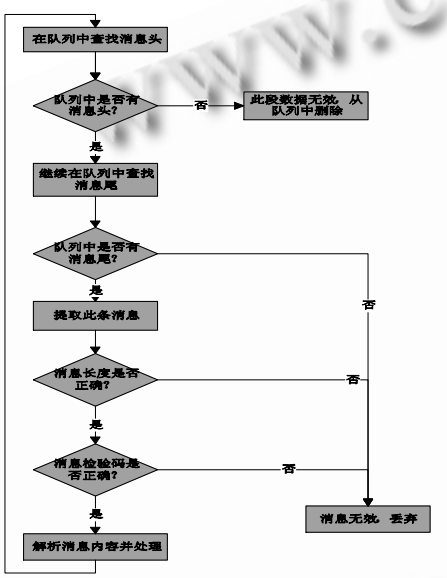

图 5　数据处理模块流程图

3　通信服务器的实现

通信服务器采用基于 MFC 的程序架构并采用多线程机制实现数据的实时并发处理。下面对通信服务器所采用的关键技术作详细说明。

3.1 多线程的同步

由于是多对多的通信方式，故数据必然会同时被多个线程访问，所以线程同步是服务器的技术关键之一。

MFC 为线程同步机制提供了良好的支持，共提供了几种线程同步类以供使用，如 CSemaphore,、Cmutex、CcriticalSection、 CEvent 等。线程同步类的使用有两种方式，一种是在对象外面使用。具体说来，就是对要被保护的数据进行操作之前先用线程同步类将此数据锁住，以免其他线程破坏此数据。操作完以后再将数据释放，以供其他线程使用。另一种方法是在类的内部嵌入一个同步对象，在必要的操作方法(成员函数)里进行数据的加锁与释放。这种方法较为直接。这样设计的类称之为线程安全类(Thread-Safe Class)。

本服务器综合使用了上述两种方法。在读写数据库以及写日志文件时使用了前一种方法；在数据队列类的设计中则使用了内嵌同步类的方法设计了线程安全的数据队列类。数据队列类的定义如下：

```
//Queue.h
#ifndef _QUEUE_H
#define _QUEUE_H
#include <afxtempl.h>
#include <afxmt.h>
typedef CList<BYTE,BYTE> MsgList;
class CQueue : public CList< BYTE, BYTE >
{
public:
    void AddToEnd(MsgList*);
    void AddToEnd( BYTE newByte );
    BYTE GetFromFront();
    int QGetCount();
    void PeekMsg(MsgList& msg);
private:
    CCriticalSection    m_CS;
};
#endif
```

举 QGetCount()函数为例，此函数定义如下：

```
int CQueue::QGetCount()
{
    m_CS.Lock();
    int k = GetCount();
    m_CS.Unlock();
    return k;
}
```

m_CS.Lock()、m_CS.Unlock()即是在操作前后对队列的加锁与释放。

3.2 重叠 I/O 模型

比起阻塞、select、WSAAsyncSelect 以及 WSA EventSelect 等模型，重叠 I/O 模型使应用程序能达

到更佳的系统性能。

重叠 I/O 模型和这 4 种模型不同的是，使用重叠模型的应用程序通知缓冲区收发系统直接使用数据，也就是说，如果应用程序投递了一个 10KB 大小的缓冲区来接收数据，且数据已经到达套接字，则该数据将直接被拷贝到投递的缓冲区。

而这 4 种模型中，数据到达并拷贝到单套接字接收缓冲区中，此时应用程序会被告知可以读入的容量。当应用程序调用接收函数之后，数据才从单套接字缓冲区拷贝到应用程序的缓冲区，差别就体现出来了。

管理重叠 I/O 请求的完成情况有两种方法：

① 事件对象通知(event object notification)

② 完成例程(completion routines)

本服务器采用事件对象通知方法并将重叠模型的具体操作封装在 CGPRS_ServerView 类的 RecvData（）和 SendData（）两个成员函数之中。

以下以 SendData（）函数为例具体讲解重叠模型的使用方法。

首先是初始化工作，包括变量的定义，事件对象的创建等：

```
    DWORD
RecvBytes=0,Flags=0,BytesTransferred=0,Index;
    WSAEVENT Event;
    WSAOVERLAPPED Overlapped;
    Event=WSACreateEvent();
    ZeroMemory(&Overlapped,
sizeof(WSAOVERLAPPED));
    Overlapped.hEvent = Event;
    WSABUF DataBuff;
    DataBuff.len=buffsize;
    DataBuff.buf=pData;
    //pData 为指向 char 型的指针;
```

其次，以 WSAOVERLAPPED 结构为参数，在套接字上投递 WSASend 请求：

```
    if (WSASend(sendsock, &DataBuff, 1, &RecvBytes,Flags,&Overlapped,NULL)=SOCKET_ERROR)
    {
        if(WSAGetLastError()!=WSA_IO_PENDING)
        {
    this->pMainFrame->SetStatusBarText((CString&)"发送出错");
    WSACloseEvent(Event);
                return false;
        }
    }
```

接下来，用 WSAWaitForMultipleEvents 函数等待重叠操作返回的结果并在返回之后使用 WSAResetEvent 函数重设当前这个用完的事件对象：

```
    Index=WSAWaitForMultipleEvents(1,&Event,
FALSE, WSA_INFINITE, FALSE);
    WSAResetEvent(Event);
```

最后，使用 WSAGetOverlappedResult 函数取得重叠调用的返回状态：

```
    WSAGetOverlappedResult(sendsock,
    &Overlapped,&BytesTransferred,FALSE,&Flags);
    if (BytesTransferred = 0)
    {
        WSACloseEvent(Event);
        //发送出错
        return false;
    }
    return true;
```

此时如果 BytesTransferred 不为 0 则表示数据发送成功。

4　总结

本文介绍了铁路大型养路机车在线监控系统中通信服务器的设计与实现，对服务器各个模块作了详细介绍，并对服务器使用的一些关键技术作了详细说明且给出了部分代码。通信服务器设计的机车车载终端数目上限为 200 台，实时监控客户端数目上限为 50 个；实际应用环境为 100 多台机车及 40 多个客户端。在实际的应用中表明通信服务器的设计完全能够满足实际需要，具有良好的性能和高度的可靠性。

参考文献

1 原仓周.大规模车辆监控通信服务器的设计与实现.北京航空航天大学学报, 2004,30(3):232—235.

2 潘越清.浅淡无线监控系统.科技信息, 2006,(7):17.

3 Jeffrey Richter. Windows 核心编程(第五版).北京:人民邮电出版社, 2008.