Abstract

In order to inform the remote users of the state of the power system and realize the remote and mobile supervising the system, we should expand the SCADA system to Internet. It is necessary to add remote supervision system to the existing system.

Fist, analyzing the demand of the remote supervision system, we know that the functions of the remote supervision system must have system state display, datagram of network display, event display, YX call and YC call, history report and remote user management. The system also requires usability, real time performance, reliability, data consistency and transportability. According this demand, we propose the design plan to the system. The double vision server is Web server and also is application server. The double vision client uses browser as the client application. The server on the on hand provides code and real time database to client, and the other hand, it also send the datagram it has received from network of supervising center LAN.

The design of the remote supervision system mainly solves two questions: the real process of data and reliable data display. With data processed in the local machine and real time network communication, we can get the real process of data. With CORBA middleware technology, we copy the data from the server to the client local machine. The network communication is based on UDP, which needs no building connection between the server and client. So system overhead is very little. The reliability of data display is acquired through the mechanism of counting datagram and retransmission. The counting and retransmission of network datagram is encapsulated in the module of network communication. This is transparent to the SCADA application. All what SCADA application to do is the invocation of the sending datagram and receiving datagram interface.

At last, we build a trial system and test the whole remote supervision system. The real time performance and reliability meet requirements.

Key words: CORBA; Java; SCADA; Remote Supervision System

第1章 绪论

1.1 前言

随着电气化铁道的迅猛发展，电力机车牵引逐渐取代内燃机车牵引而成为铁路的运输的主力。而牵引供电系统的高效、可靠性运行是电力牵引运输得以顺利进行的根本保证。微机监控技术为提高牵引供电系统调度管理综合自动化水平，为迅捷、快速分析、查找、切除故障点、提供系统智能化水平，确保系统安全、可靠、高效运行提供了有力的技术支持。

电气化铁道微机监控系统主要由三部分组成，即位于铁路局调度中心的调度端（监控主站），装设于铁路沿线牵引变电所（分区亭、开闭所等）的远方终端装置或被控站（RTU），以及远动通道。监控主站和被控站以远动通道为桥梁，共同实现对牵引供电设备遥控、遥信、遥测和遥调等功能。遥控就是指从被控站发出命令通过执行端实现对被控对象的远程操作。遥信是将被控站的牵引供电设备状态信号、事故报警信号等远距离传送至监控主站处理，如开关位置信号、事故、预告信号等。遥测是将被控牵引供电设备的某些运行参数远距离传递给监控主站处理（如有功）。下面简单介绍一下监控主站的功能和组成。

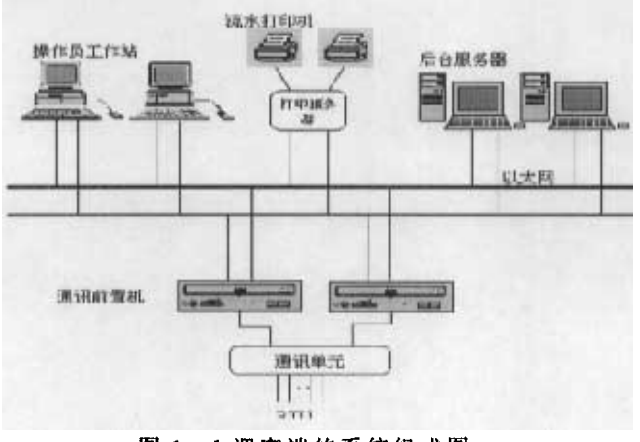

图 1－1 调度端的系统组成图

监控主站是微机监控系统的调度指挥中心，监控主站主要由后台服务器、操作员工作站、通讯前置机以及模拟屏、流水打印机组成。调度端的系统组成如图 1－1。监控主站以后台服务器为核心，通过调度中心的局域网，操作员工作站、后台服务器以及通讯前置机相互交换数据，对系统中各设备进行监控和管理。整个系统采用冗余配置，其主要目的是为了提供系统的可靠性。

监控主站调度管理自动化系统数据来源于操作员工作站上的调度员的操作命令及被控站（RTU）采集到的被控对象的有关数据上送信息。前者的远动操作命令称为下行命令，后者的上送有效信息称为上行信息。整个系统围绕下行命令和上行信息展开处理工作。操作员工作站作为主要的人机交互界面，接受和初步处理调度操作命令，是下行命令的第一受理者；通讯前置机通过远动通道查询获取被控站有关信息，预处理后通过网络传送给主机处理，因此可以说通讯前置机是上行信息的第一接待站。同时也可以形象地说，通讯前置机是下行命令地出口，操作员工作站是上行信息的终点站。整个系统的信息处理如下图。

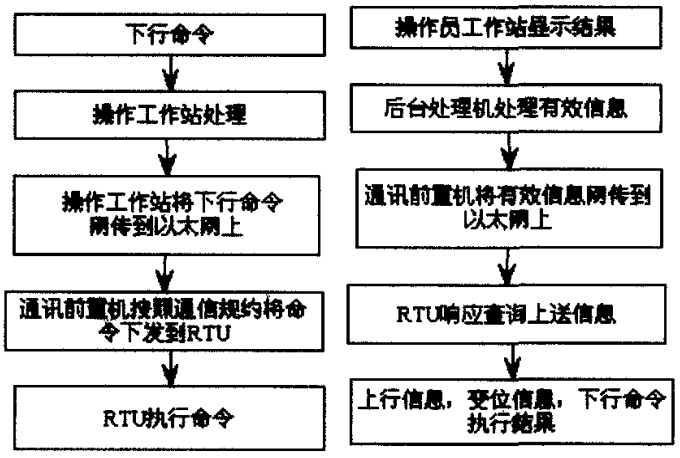

图 1－2 下行命令及上行信息处理流程图

后台服务器主要用于数据和网络服务及定时任务管理，进行数据的后台处理，管理实时数据和部分历史数据，负责网上节点资源的

分配、管理和信息交换，进行网络信息汇总、组织和分派，为操作员工作站和通讯前置机提供初加工数据。

操作员工作站是实施调度作业的人机界面，并集中反映调度员的意图和效果，监视牵引供电设备运行状态。每个调度台配备有两台操作员工作站，互为备用，远动操作尤其是遥控（包括单控、程控）操作时互锁。操作员工作站由高可靠性的工业控制用PC、大屏幕彩色显示器以及操作系统和相应的软件构成。它的作用主要是与后台服务器和通讯前置机交换操作命令、上行信息以及设备工作状态；进行远动操作（包括单控、程控、遥信全召、遥测全召、故障信号复归等操作）、本地操作（包括各种画面的调出与切换、统计报表、电量曲线的选择显示、记录以及手动置位等）和命令管理（主要对远动命令进行记忆、超时监视并给出相应提示和处理）；对上行信息的处理（对上行远动信息进行分解、对网络协议及RTU通讯规约进行转换处理、实时信息的存储和显示）。

通讯前置机和远动通道线路是连接监控主站和被控站的桥梁，是监控主站与被控站的信息纽带。每个监控主站配备有两套通讯前置机，互为备用。通讯前置机也是由高可靠性的工业控制用PC、彩色显示器、操作系统和相应的软件以及Modem构成。它主要的作用是与操作员工作站、后台服务器交换下行命令和RTU上行信息及通讯设备状态信息等；对所辖的RTU进行轮询（Polling），查问有无遥信、遥测变位信息需要上送；对RTU上送的信息进行筛选、RTU通讯规约的转换、信息的转发等；对RTU进行模拟，用于对主站进行调试、演示及用户培训等。

为了便于远程用户及时的了解整个系统的运行情况，将现有的系统扩展到Internet上，实现系统的远程监视、移动监视，在上述系统的基础上增加一个复示系统是必要的。

1.2 复示系统的组成及作用

复示系统从系统的角度来看分为两个部分：复示服务器和远程的复示终端。复示服务器的作用：首先在WWW服务器上发布客户端程序，使得复示终端能够不需要任何配置，直接通过浏览器来下载

服务器上的代码然后运行；其次，服务器系统应为客户机准备系统初始化的数据像画面显示数据以及实时数据库等；最后，服务器还要维持和操作员工作站、通讯前置机、后台服务器以及和客户机的网络通讯，及时地把网络上的有效上行信息转发到复示终端上。复示终端的作用主要是及时的处理从服务器传过来的上行信息，在画面上加以显示。它首先从服务器下载服务器代码，然后再从服务器获得系统初始化的数据，最后实时接收服务器上传的信息，来刷新画面。加上复示系统以后，整个微机监控系统的结构图如图3－3所示。总的来说，复示系统是对整个系统功能的一个很好的扩展。

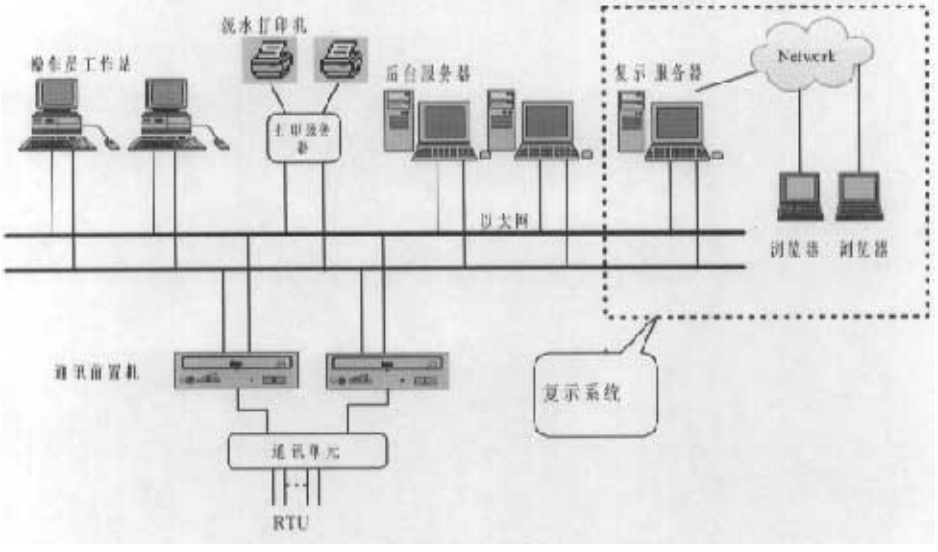

图3－3 加上复示系统后的系统图

复示客户端和复示服务器之间有两种连接方式，一种是同通过调度中心内部的局域网访问，实现系统的本地监视；另一种方式是远程用户利用拨号网络拨号到调度中心的局域网络内实现对系统的远程监视和移动监视。这两种方式都是通过访问复示服务器来实现对系统的监视，后一种方式采用的次数要多一些。

1.3 复示系统特点

复试系统最基本的特点是实时性和可靠性。复示系统具有实时性

的特点，是因为它所反映的系统的状态应是当前电力系统的实时状态。当电力系统的状态发生改变时，复示服务器和复示终端也要做相应的改变，且时延不应太大，只有这样，复示系统所反映的状态才有实际意义。

复示系统也应满足可靠性的要求，复示系统所显示的系统信息必须真实可靠。这就要求，复示终端本地的实时数据库必须可靠、复示终端和复示服务器之间的网络通讯要可靠。

1.4 复示系统的意义

复示系统和操作员工作站有些相似，但又不尽相同。它主要的功能是及时反映当前系统的运行情况及查看整个系统的历史记录和统计报表情况（包括日报、月报、年报、跳闸报、越限报、故测报等）。它不要求对系统进行远动操作和本地操作。远程用户通过拨号网络或通过 Internet 访问复示服务器，就能够得到当前电力系统的运行情况、以及历史记录。这样就可以实现远程监视、移动监视。根据复示终端当前所反映的系统状态，有助于远程用户作出决策。当系统出现故障时，复示终端有利于远程用户作出系统故障诊断。

综上所述，复示系统通过将 Internet 技术带入到电力系统监控领领域中，为提高牵引供电系统调度管理综合自动化水平，为迅捷、快速分析、查找、切除故障点、提供系统智能化水平，确保系统安全、可靠、高效运行提供了有力的技术支持。

第2章 复示系统的需求分析

2.1 复示系统功能方面的需求

复示系统和调度中心的操作员工作站的功能有些相似，但又不尽相同。它主要的功能是使远程用户能够及时了解当前系统的运行状态和系统的历史记录，这里的远程用户包括调度中心内部的局域网和拨号网络上的远程用户。它不要求像操作员工作站那样对监控系统进行远动操作和本地操作。复示系统功能方面的具体需求有：系统状态显示、网络报文显示、事件显示、遥信和遥测全召、报表功能以及远程用户管理功能。

2.1.1 系统状态显示

系统状态的显示是复示系统的一个主要功能之一。远程用户通过访问调度中心的复示服务器，得到系统的当前状态，然后以图形界面的形式显示给用户。系统的状态显示要正确，应该和所监视的电力系统的当前状态相一致。系统状态的显示是一个实时的显示过程，当用户得到系统的当前状态后，它还必须实时刷新本地的状态，使得呈现给用户的系统状态和系统的实际状态相一致，也就是说当下面的设备发生一次遥信变位后，复示系统的远程用户也必须得到这次遥信变位的信息，来刷新画面，以新的状态呈现给用户，所以复示系统还必须接收调度中心网络上的网络报文，来实时刷新本地的状态。同时，复示系统软件还必须具有解释网络报文的功能，能够对所接收的网络报文按照规约作出正确的解释。

2.1.2 报文显示

由于复示系统的远程客户还必须接收调度中心网络上的网络报文，所以为了调试系统方便和理解网络规约的需要，将接收的网络报文以灵活的方式呈现给用户，用户可以选择显示网络报文，也可以选择不显示。另外，不同的网络报文所显示的字体和颜色应有所

不同以示区别。

2.1.3 事件显示

事件显示功能就是指系统能够将系统当前所发生的事件显示出来。系统根据接收的网络报文，除了将相应的状态刷新以外，还要将按照网络规约，将这次系统状态的改变翻译成易理解的事件像：开关闭合，故障等。

2.1.4 遥信和遥测全召

遥信全召指的是复示系统为了防止因通讯中断等因素可能引起的状态不一致而下发给 RTU 的命令，RTU 收到命令后，将所辖的遥信（位置遥信和非位置遥信）状态上报，复示系统收到上送的信息以后根据信息来刷新系统的状态。

遥测全召和遥信全召命令相似，只是 RTU 要上送的是被控站的遥测量，如进行电压、主变电流、馈线电流等供电运行参数。

2.1.5 报表功能

复示系统本身不进行任何的报表统计工作，报表统计是调度中心中的后台服务器完成的。复示系统的报表功能指的是远程用户通过请求服务器得到所要显示的数据，然后以图形或表格的形式呈现给用户。报表功能包括（日报、月报、年报、跳闸报、越限报、故测报等）。

2.1.6 远程用户管理

复示系统所反应的是系统的一种实时运行状态，和系统的历史运行情况，要想获得这些信息，必须有相应的操作权限和系统登录。所以系统须对远程用户进行管理，包括添加用户、删除用户、更改用户密码、分配权限等。

2.2 复示系统非功能方面的需求

复示系统除了满足功能方面的需求外，还必须满足一些非功能方面的需求。复示系统非功能方面的需求是对所考虑的可能解决方案的一种约束和限制。它主要有以下几点：可用性的要求、实时性的要求、可靠性的要求、可移植性的要求、系统安全方面的要求等。下面分别加以叙述。

2.2.1 可用性方面的要求

系统可用性方面的需求是一个很实际的需求，系统必须首先满足可用性方面的要求。复示系统可用性方面的要求指的是系统的人机界面友好、使用简单、无需复杂配置、可理解性好、可修改性好等等，同时便于系统升级和扩张。

2.2.2 实时性的要求

由于复示系统所反映的系统状态是系统的一种实时状态，所以复示系统应具有实时性方面的要求。复示终端应该实时得到调度中心局域网内的信息，然后将此信息实时的显示给用户，使得复示系统所呈现给用户的系统状态和系统的当前实际运行状态相同步。

复示终端所接收的网络报文实际上是调度中心内的网络报文，复示终端和复示服务器之间采用网络通讯。这就要求网络通讯模块具有实时主动上传网络信息的功能，另外，复示终端为了防止因通讯中断等因素可能引起的状态不一致而下发给RTU的遥信全召和遥测全召命，所以网络通讯模块除了实时接收网络信息外，还必须具有实时发送网络报文的功能。

复示终端接收到网络报文后，按照网络通讯规约解释报文，同时还必须实时的将状态的改变反映给用户。这时的画面实时显示就要求数据的实时处理。如果将数据的处理也提交到服务器，一方面加中服务器的负荷，使系统的负荷不均，导致系统的实时响应能力下降，所以复示终端应具有数据的处理能力。在系统初始化时，复始终端应该从复示服务器得到系统数据（包括系统的画面库、节点表、初始状态等）构建本地的实时数据库。这样复示终端在收到网络报文后，根据报文内容按照网络规约对系统中的节点状态加以改变，

然后再按照新的节点状态，实时地将新地系统状态呈现给用户。

2.2.3 可靠性的要求

为了满足系统实时状态的显示的要求，复示服务器要把调度中心的网络报文转发到复示终端，同时要将复示服务器上的实时数据库拷贝到本地，将远程数据处理转化为本地处理。所以为了系统能够可靠的工作，至少要满足两点可靠性的要求。

一是网络通讯的可靠性，复示服务器应可靠的接收调度中心局域网内的数据，同时复示终端和复示服务器之间的网络通讯模块应可靠的将复示服务器所接收到的网路报文转发到复示终端。

二是本地数据的获取必须正确可靠。复示终端的画面显示是根据本地的数据来显示。系统状态的更新也是通过本地的数据来进行的，所以复示终端启动开始获得的数据必须可靠。画面与遥信对象，遥测对象一定要关联正确。

2.2.4 数据一致性的要求

由于复示系统中的服务器和终端都都一个本地的实时数据库，所有用户根据本地的数据得到的系统状态应是相同的，所以客户端和服务器之间的数据要一致。后台服务器是数据的源头，复示服务器系统初始化的时候，从后台服务器的上的数据库服务器上取得数据构建本地的实时数据库。复示终端初始化的时候，再从复示服务器上取得数据构建本地得实时数据库。当复示服务器出现故障时，这时复示服务器不能够实时将网络数据远传复示终端，当复示服务器重新启动候，要有能够保证复示终端的本地数据库和当前的系统相一致，保证复示终端所反映的系统状态和当前系统状态相一致的措施。

2.2.5 系统可移植性的要求

复示系统的远程用户可能采用不同的操作平台，所以复示系统的复示终端应具有可移植性、跨平台的要求，使得系统在不同的操作环境下都可以稳定运行。

在这点上，java具有很大的优势，它的优点在于：编码一次，可以运行于任何地方。因为java语言是一种解释性的语言，它由java虚拟机来解释执行，由于不同平台上的java虚拟机采用统一的规范，java的数据类型在各种操作平台下是一致，所以用java语言编写的程序可以运行于各种平台，且其执行效果是一致的。另外，CORBA（公共对象请求代理体系结构）也是实现异构环境的不同软件相互协作的规范。可以使得处于操作平台、采用不同具体编程语言实现的对象之间能够透明的相互通讯，相互调用。这也是系统采用这两种技术来实现的一个主要原因。

2.2.6 系统安全方面的要求

复示系统也应该具有系统安全方面的要求。复示系统所提供的电力系统运行信息和系统状态只有在用户具有相应的权限才可以查看。系统应具有用户访问权限、操作权限方面的设置和远程用户管理等方面的功能。

第3章 系统的设计方案

3.1 系统设计概述

复示系统的设计就是采用合适的方案、合适的技术实现复示系统功能和非功能方面的两个需求，同时使系统具有良好的可扩展性。复示系统是对现有的微机监控系统的一个补充，它的主要功能是为远程用户提供当前运行的电力系统的系统状态。在功能上，它应有系统状态显示、网络报文显示、事件显示、遥信和遥测全召、报表功能以及远程用户管理功能。在非功能上，它应满足可用性的要求、实时性的要求、可靠性的要求、可移植性的要求、系统安全方面的要求等。在考虑系统设计方案时，应兼顾上述两方面的要求。

复示系统在组成上包括两部分：复示服务器和复示终端。复示服务器位于调度中心内，它为复示终端准备系统初始化的数据和转发调度中心局域网络上网络报文（主要是上行信息）。复示终端主要是显示系统状态和根据接收到的网络报文来实时刷新系统的状态，使得呈现给用户的系统状态和当前系统的运行状态相一致。

根据两者的组成关系和功能关系学，复示服务器和复示终端之间的系统结构我们采用基于 Web 的 B/S（浏览器/服务器）模式。复示服务器作为应用程序服务器和 Web 服务器，Web 服务器通过 HTTP 协议为复示终端提供页面和运行的代码，应用程序服务器转发网络报文到复示终端。复示终端只是一个浏览器，没有任何应用程序。B/S（浏览器/服务器）模式简化了客户机的工作，客户机无需配置任何软件，对数据库的访问和应用系统的执行在服务器上完成。B/S 模式是一种典型的三层分布式结构。如下图所示。

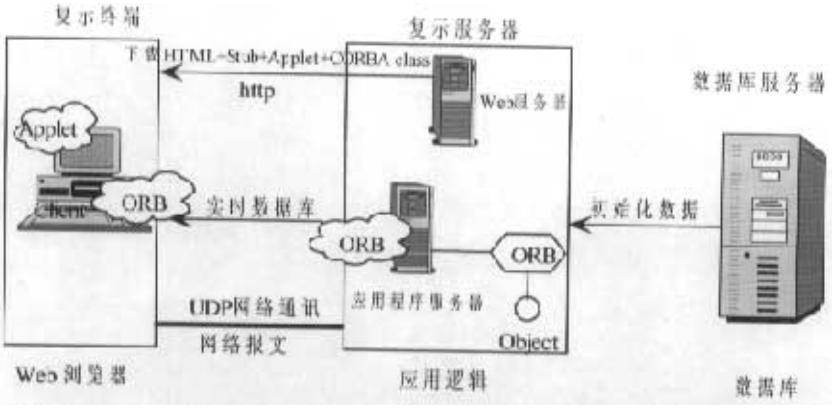

图3－1复示系统的三层结构

Web浏览器是人机界面部分，它不必关心业务逻辑是如何访问数据库的，只需把精力集中在人机界面上即可。

中间业务逻辑层包含了大量的供客户端程序调用的业务逻辑规则，以帮助其完成业务操作。它的优点就在于它具有的可伸缩性，可使其随具体业务的变化而改变，但在客户层和数据服务层所做的改动较小。

数据服务层主要提供对数据库进行各种操作的方法。它主要由中间业务层来调用来完成业务逻辑，当数据库的结构确定后，对于改模块的改动就比较小啦。

复示系统采用B/S（浏览器/服务器）模式，具有以下优势：

1. 开放的而非专用的标准。Browser/Server技术所基于的标准是开放的、非专用的，是经标准化组织指定而非单一厂商指定的。

2. 较低的应用开发及管理成本。在客户/服务器模式中，无论是安装、配置还是升级都需要在所有的客户机上实施，而Browser/Server技术较为低廉，一般只需安装、配置在服务器上，而客户机上的工作较少，从而降低了开发成本及管理成本。

3. 对信息及应用系统的自由访问。现在许多计算机用户已经建立起网络，由于信息和应用系统可通过Web浏览器进行访问，因此几乎所有的客户均可自由地、主动地访问信息系统。

4. 较低的培训成本。浏览器地技术简明易用，一旦用户掌握了浏览器地用法，也就掌握了运行系统上各种资源的钥匙。

3.2 实时性的解决方案

复示系统的实时性归根结底是数据处理的实时性。如何使复始终端能够实时刷新系统状态并快速显示给用户的关键是复示系统能够实时接收报文并根据报文的内容做出实时处理。为了实现这个数据的实时处理，可以将静态数据转移到本地，使数据的处理变为本地化处理，这样做同时也将任务转移到客户机上，从而降低了服务器的负荷，增强了系统的响应时间；对于动态数据（RTU 上传的网络报文），采用基于 UDP 的网络通讯方式，采用 UDP 的网络传输方式，无需建立连接，从而降低系统开销，提高系统的实时性。下图是复示系统实时性解决方案的示意图。

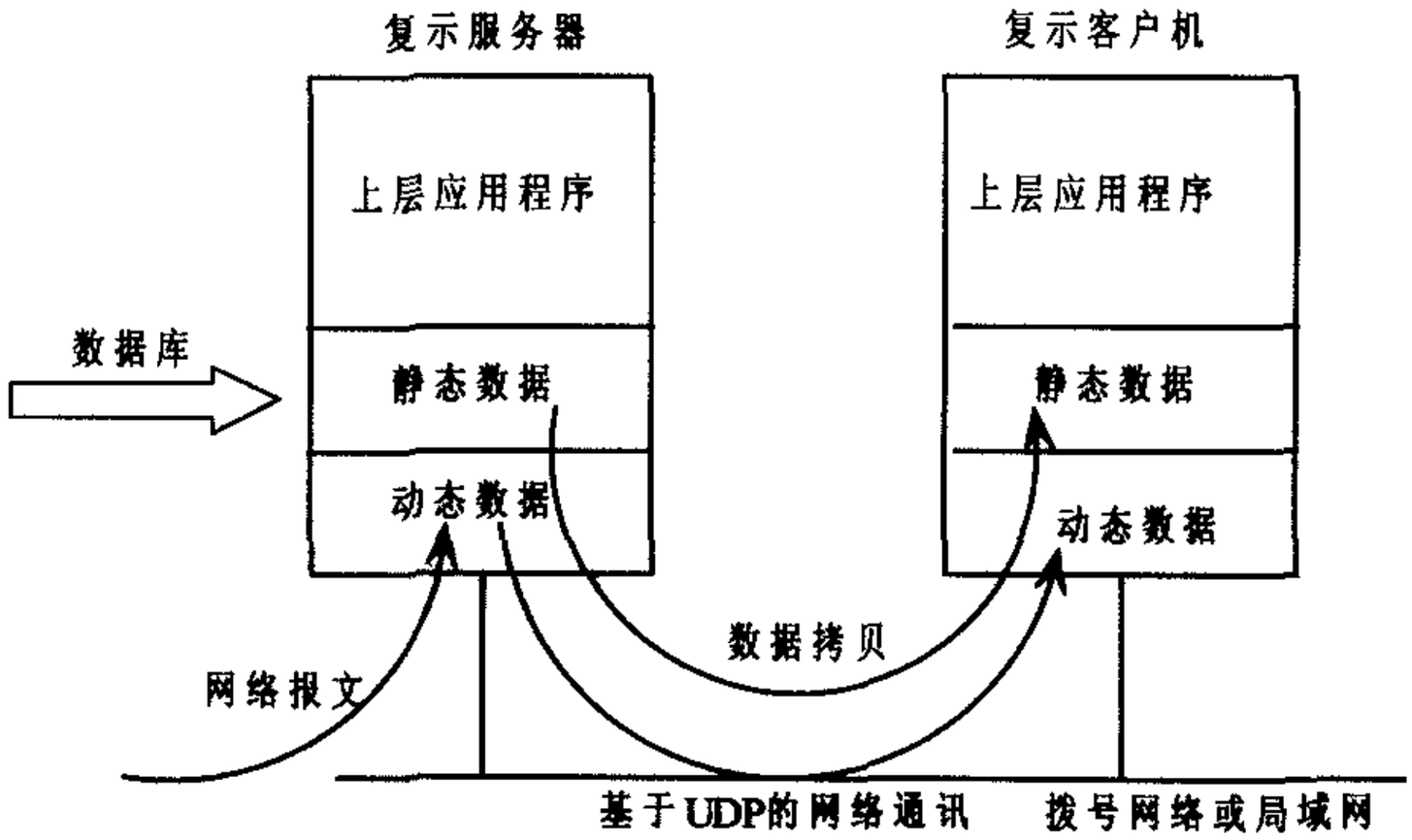

图 3－2 复示系统实时性的解决方案

3.2.1 静态数据的本地化处理

复示系统的数据有两个方面：静态数据和动态数据。静态数据包括被控站的结构、画面数据、节点表等；动态数据主要是节点状态信息，具体的说动态数据就是 RTU 上传到通讯前置机，通讯前置机又发布到调度中心网路内的数据，复示服务器又将该信息转发到复示终端。静态数据的主要用来进行系统初始化，画面显示、画面与系统遥信对象、遥测对象的关联。动态数据主要用来更新系统状态，使复示终端到达能够实时显示系统状态的目的，从而实现系统的远

程监视，移动监视。

数据的本地化就是为了实现静态数据的本地化，这样系统的显示就可以根据本地数据做出显示，同时将遥信对象、遥测对象通过节点表和画面关联，而不需要向服务器提出请求实现这些功能，增加复示服务器的负荷。同时，当复示终端接收到动态数据和实时数据后，根据本地的静态数据，实时刷新系统的状态。

1. 实现数据本地化的方法

复示服务器的数据的本地化是从后台数据库服务器读出被控站库、画面库以及节点关联表构件自己的内存数据。同时发送遥测，遥信全召取得当前电力系统的状态。

复示终端的数据的静态数据的本地化可以通过两种方法：一种服务终端直接访问后台的数据库服务器，通过 JDBC 访问被控站库、画面库以及节点库的方式，另一种方式是复示终端采用通过和复示服务之间的通讯，将复示服务器的数据转移到本地。前一种方式一个方面访问数据库需要花费大量的时间，另一个方面，所取得数据还不是系统的当前状态，当复示终端获得这些数据后，还必须进行遥信全召、遥测全召来更新自己本地的实时数据库，这也需要一定的时间，另外也增加网络的流量和服务服务器的负荷，影响整个系统的性能。而采用后一种方式，由于复示服务器的数据已经是内存数据，所以处理比较快，更重要的复示服务器的内存的数据已经反映了系统当前的状态，所以当复示终端获得这些数据后，不需要进行遥信全召和遥测全召来更新本地的实时数据，复示终端这时只需实时接收网络报文来现在的系统状态相同步就可以了。下表是复示终端数据本地化两种方式的详细比较。

表 3—1 复示终端数据本地化的两种方式的比较

	采用 JDBC 访问后台数据库服务器取得本地数据	采用拷贝复示服务器的实时数据的方法
花费时间	多	比较少
所取得数据的状态	初始态	系统的当前状态
是否遥信、遥测全召	不要	要

网络负荷	重	不重
复示服务器的负荷	重	比较轻

2. 实现数据本地化所需要的技术。

复示服务器的数据的本地化是从后台数据库服务器读出被控站库、画面库以及节点关联表构件自己的内存数据。同时发送遥测，遥信全召取得当前电力系统的状态。所以，服务服务器的数据本地化所采用的技术是数据库访问技术，具体的说就是采用 JDBC 技术。JDBC 是一种可用于执行 SQL 语句的 Java API（Application ProgrammingInterface)。使用它的好处是它给数据应用开发人员，数据库前台工具开发人员提供了一种标准的应用程序设计接口，使开发人员可以用纯 Java 语言编写完整的数据库应用程序。由于使用 Java 编写的程序可以在任何支持 Java 的平台上运行，不必在不同的平台上编写不同的程序。Java 与 JDBC 的结合可以让开发人员在开发数据时正真实现“Write Once，Run Anywhere”。

复示终端的数据的获取方法时通过上面的分析采用拷贝复示服务器的内存数据的方法。实现复示服务器和复示终端之间的数据拷贝大致有两种方法，一种方式是将复示服务器本地的实时数据库中的对象的属性封装成网络报文发送到复示终端，复示终端在获得这些网络报文后，根据网络报文传输规则解析报文得到构成内存对象的数据，调用对象的构造函数构件本地的内存对象。另外一种方式是复示服务器和复示终端之间直接传递内存对象。目前能够远程传递对象的方法又有两种：Java RMI（Remote Method Invocation）和 CORBA（common object request broker architecture)。采用前一种方式首先是编程比较麻烦，必须自己按照复示服务器和复示终端之间的数据拷贝网络规约将内存中的对象的属性组装成网络报文，自己编写复示服务器和复示终端之间的网络通讯，并且要确保通讯的可靠性。另外一个确定是不同用，复示服务器和复始终端之间的网络通讯规约是专门针对内存的对象类型，如果系统要扩展，需要传递别的数据类，这是就要修改数据传输规约，同时也要修改复示服务器上的应用程序和客户机上的 Applet 程序。它的优点是自己编写的

通讯模块无需建立太多的连接传送报文的效率要高一些。复示服务器和复示终端之间如果传递远程对象的话，首先是编程简单，要传递的对象只要实现某些接口（在 Java 中是 Remote 接口，在 CORBA 中要继承 IDL 编译器生成和类相应的 POA 接口）就可以。而且它们都支持直接传递数据和序列。另外系统的可靠性要高一些。Java RMI 和 CORBA 自身都提供可靠的机制来保证通讯的可靠性。在一点是它们具有很好的通用性。只要将所要传递的对象实现某些接口就可传递，而不要更改复示服务器和复始终端直接的通讯规约。和前一种方法相比，其缺点是效率要低一些。应为 Java RMI 和 CORBA 要实现通用性和可靠性必然要付出相应的代价。

CORBA 和 Java RMI 相比的优点是完全语言中立，系统中立。把它比作为一个软件总线是在恰当不过了，通过 ORB，处于异构环境下的并且连接在 ORB 上的对象可以相互通讯，相互操作。你可以采用不同的语言来实现客户机和服务器。Java RMI 只能用于 Java 环境下，它只能使 Java 对象和 Java 对象直接互操作。不过，相比 CORBA 来说，Java RMI 更容易理解一些。下表是实现复示终端和复示服务器之间数据传输的技术的比较。

表 3－2 实现复示终端和复示服务器之间数据传输的技术的比较

	网路报文传输方式	对象传递方式
程序设计的难易度	难实现	较易实现
通用性	不通用	通用
可靠性	不可靠（需要自己提供机制保证）	可靠
效率	较高	较低

表 3－3 CORBA 和 Java RMI 之间的技术比较

	Java RMI	CORBA
对象之间的互操作性	Java 对象和 Java 对象之间进行互操作	不同的对象都可以首先互操作
对象的实现方式	只能采用 Java	Java，C++等
效率	高	低
实现的难易程度	容易	较难

根据以上的比较结果，我们采用对象拷贝技术来实现复示终端数据的本地化，同时为了处于异构平台下的复示终端能够和复示服务器之间或互操作，我们采用 CORBA 技术来实现对象之间的数据拷贝。在这里我们重点考虑了系统以后的可扩展性，通用性以及跨平台性。而且在系统初始化的时候，可以适当放宽对时间的要求。

3.2.2 基于 UDP 的网络通讯

为了满足复示系统对实时性的要求，我们采用了静态数据的本地化处理方式，复示终端的画面显示以及系统状态都是在本地，这样大大提高了本地数据的处理的性能，从而达到系统状态的实时显示，为了实现系统状态的实时更新，复示终端还需要接收复示服务器转发过来的网络报文。复示服务器和复示终端之间的报文无外乎基于 UDP 的传输方式和基于 TCP 的传输方式。

UDP 和 TCP 都是 TCP/IP 中传输层的传输控制协议。TCP 和 UDP 最根本的区别是：TCP 是面向连接的，两个端点在传输数据之间，必须先经过三次握手建立连接，在数据传输完以后，还必须经过三次握手释放连接。UDP 在传输数据之前，无需建立连接。

TCP 被用于在网络上提供有序可靠数据传输能力的需电路服务。TCP 在不可靠的分组传输网络上（这种网上随时都有可能出现数据的丢失、损坏、重发重复重发、延迟和错序）提供可靠的进程间的通讯。为了取得可靠传送，TCP 必须执行检测分组丢失，收不到确认时自动重传，以及诸如处理延迟重复数据报得问题等许多操作。

UDP 采用的是无连接得方式提供高层协议间得事务处理服务，允许它们互相发送数据报。它就不保证可靠投递。它与远方的 UDP 实体不建立端到端的连接。而只是将数据发送到网络上，或者从网络上接收数据。它不具备诸如接收保证和避免重复等有序投递功能。在这里，我们重要关心复示服务器和复示终端之间网路通讯的实时性，由于 TCP 为了提供可靠的传输服务器，必须付出额外的系统开销。所以系统的响应较慢。而 UDP 无需建立连接就可以发送你跟报文，所以系统的响应较快。另外，复示终端和复示服务器之间所传输的报文长度较短，不需要进行分割，所以不用考虑报文不按序到达的问题，所以复示服务器和复示终端之间的网络通讯模块采用基

于 UDP 的网络通讯。同时网络模块必须处理以下几种情况：报文丢失、报文重传以及报文延时、可靠接收。

表 3－4 TCP 和 UDP 两种传输方式的比较

	UDP	TCP
是否建立连接	否	是
是否提供接收保证	否	是
是否能否避免接收重复	否	是
报文是否有序到达	否	是
系统的响应	快	慢

由于用 Java 实现的 Applet 为了避免网络风暴，不支持基于 UDP 的组播 （Multicast）方式。所以复示终端和复示服务器之间只能采用基于 UDP 的点对点的通讯方式。当复示服务器接收到一条新的网络报文时，它将按照目前的客户机的地址依次发送。

3.3 可靠性的解决方案

根据前面的需求分析，复示系统的可靠性就是指复示终端所显示的数据必须真实可靠。这就要求系统的本地数据要可靠，复示服务器转发网络报文到复示终端要可靠。复示终端的本地数据是通过 CORBA 将复示服务器的实时数据拷贝到复示终端的。CORBA 本身提供了一种可靠的传输机制。为了保证复示服务器和复示终端之间的网络通讯的实时性，它们之间的网络通讯传输采用 UDP 传输方式。而 UDP 不保证报文的可靠传输，所以复示服务器和复示终端之间的网络通讯必须处理报文丢失、报文重传以及报文延时、可靠接收等问题。这里我们采用是带计数的报文重传机制。和 TCP 不同的是，它对报文不进行确认，也就是说复示服务器不需要知道报文是否正确到达。这一点的目的也是为了提高系统网络通讯的实时性。

报文计数的机制是这样的：发送端为其发送的每一条报文编上报文编号 N1，客户端也保留着来自发送端的下一条报文的编号 N2。当接收端收到的报文编号 N1 不等于 N2 时，说明该条报文不是自己希

望得到的来自该发送端的报文。如果N1大于N2，说明前一条报文丢失或者自己收到但没有做放到接收缓冲区，这时接收端可以保留编号为N1的这条报文也可以放弃该条报文。这里我们采用的方法时保留该条报文到临时缓冲区，然后在自己的临时缓冲区内查找编号为N2的报文，如果没有找到就发送请求重传报文编号为N2的报文到发送端。如果报文编号小于N2，说明该条报文已经收到过，这是报文的重传现象，所以放弃该条报文。

当报文丢失时，接收单就会出现接收到的报文编号和应接收报文的编号不一致的情况，接收端就会发送重传请求。

当报文重传时，接收端就会发现接收到的报文编号小于应接收报文的编号，接收端放弃该条报文。

对于报文延时的现象，网络通讯模块没有启用定时器机制，这是系统的实时性能将会下降。

下面的一个表格说明了网络通讯模块对报文丢失、报文重传以及报文延时的处理方式情况。其中，N1为发送端发送的报文标号，N2为接收端期望接收的报文编号。

表3—5 网络通讯模块对报文的处理方式

	N1和N2之间的关系	发送端	接收端
正确接收	N1＝N2		接收N1
报文丢失	N1>N2	重发编号为N2的报文	将N1临时存储放入
报文重传	N1<N2		丢弃N1
报文延时	N1＝N2		接收N1

综上所述，采用带计数的报文重传机制，网络通讯模块基本上实现了复示服务器和复示终端之间的可靠网络通讯。

3.4 数据的一致性的保证

由于复示系统中，有可能出现多个客户机的情况，复示服务器和复示终端都有可能出现故障，复示服务器和复示终端本地都有实时数据库。当系统出现故障时如何保证系统中各个客户机之间以及和服务器之间的数据一致性的问题？在这里，探讨一种情况：当复示

服务器出现故障在恢复时，如何实现复示终端正确显示当前系统运行状态，也就是如何实现复示终端的实时数据库和当前系统相一致？

由于复示系统所反映的状态来源自下面的 RTU 监控到的系统状态，所以当系统状态不一致时，可以采用下发命令到 RTU，让 RTU 将当前的系统状态上送到调度中心，复示服务器接收到这些新的系统状态信息后，依次转发到所有的复示终端，复示终端收到信息后，根据本地的数据库，实时处理，实时显示给远程用户。

复示系统提供了召系统状态的遥信全召和遥测全召，复示服务器在系统初始化完以后主动发送。复示系统还提供了定时发送、手动发送两种命令的发送方式。确保当复示系统所反映的状态不一致时，可以采取相应的措施。

3.5 系统安全性的解决方案

为了保障复示系统的安全运行，系统必须有相应的安全措施。在这里，我们可以分两种情况进行讨论，对与基于网络用户来说，通过系统登陆来保障系统安全，只有具有监视权限的用户才可以进入。对于拨号网络的用户来说，可以采用拨号网络帐号来限制用户进入调度中心的局域网，当用户拨号进来以后，通过系统登陆进一步验证将身份的合法性，从而实时系统的安全性。

3.6 系统的可用性及可移植性

由于客户机采用通用的浏览器方式，大大简化了客户机的系统开销（无需任何配置）；由于客户机的代码是从服务器下载过来的，只需要更新服务器上的代码就可方便实现客户机系统的升级；另外由于客户机是通用的浏览器而不是专用的客户端软件，它使用户的操作变得更简单。由于客户机是采用 Java 语言编写的 Applet 对象，所以具有 java 语言的优势：扩平台性、可移植性。正真实现：编写代码一次，可以到处运行。复示服务器和复示终端上的实时数据不仅是 Java 对象，也是 CORBA 对象，系统可以很好的在异构平台下的

运行，实现系统的移植。

3.7 系统总的解决方案

整个复示系统的总方案是：复示服务器既是 Web 服务器，又是应用程序服务器，复示终端采用浏览器方式，无需任何配置。复示服务器上的 Web 服务器部分为复示终端提供客户端程序，复示终端通过下载 Applet 到本机，利用本地的资源来运行客户程序。应用程序服务器为复示终端提供系统初始化的数据，同时将实时接收到的网络报文转发到复示终端使它实时更新系统状态。图 3－2 是其系统的结构图。

系统实时性的保证是通过数据本地化和基于 UDP 的网络通讯来保证的。系统的可靠性一方面是通过 CORBA 机制保证静态数据的移植的可靠性，另一方面采用带计数的报文重传方式来保证复示服务器和复示终端之间网络通讯的可靠性。

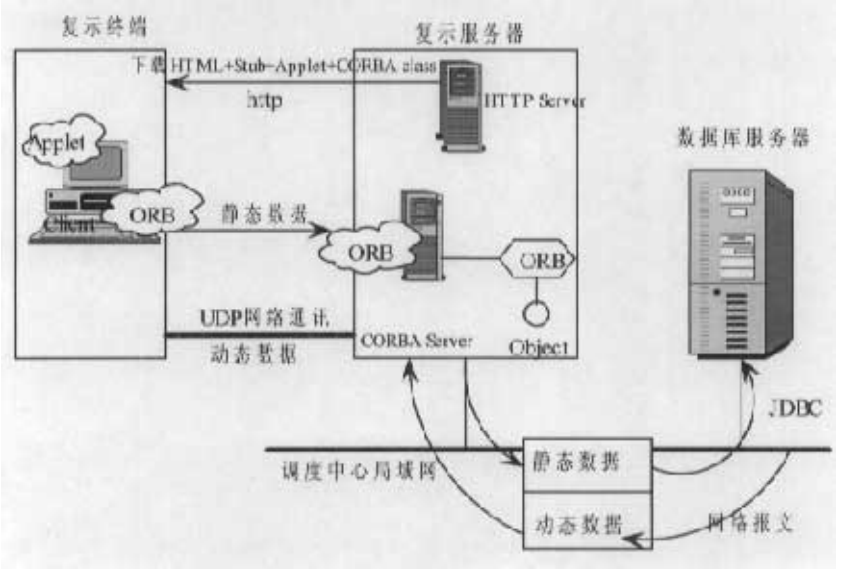

图 3－2 系统框架

下表说明了是系统设计中说遇到的问题和相应的解决方案。

表 3－6 系统设计中说遇到的问题和相应的解决方案

系统设计中的问题	解决方案
实时性的问题	数据的本地化＋实时的网络通讯（基于UDP的网络通讯方式）
网络通讯的可靠性，数据移植的可靠性	带计数的报文重传机制＋CORBA
系统的安全性	系统登录认证
系统的移植性	Java＋CORBA

第4章 系统的实现

4.1 系统实现概述

复示系统的具体实现包括两个部分的实现：复示服务器的实现和复示终端的实现。复示服务器既是 Web 服务器又是应用程序服务器。复示服务器为复示终端提供客户机代码和系统初始化数据，同时将接收到的网络报文转发到复示终端上；复示终端采用通用的浏览器方式，通过 URL 将客户机的 Applet 代码下载到本地，然后利用本机的资源运行，Applet 同时也是一个远程的 CORBA 对象，通过 ORB 提供的命名服务查找到复示服务器上的实时数据库对象，调用其接口，拷贝实时数据到本地，构建本地的实时数据库。复示系统采用 Java 技术和 CORBA 技术相结合的方式来实现。下面分别介绍复示服务器和复示终端的实现细节。重点阐明系统是若何在保证功能需求的同时来保证实时性和可靠性的要求。所谓实时性的要求就是指服务器要及时地将信息转发到客户机，客户机同时及时处理来自 Web 服务器中转上来的有效信息。也就是要保证客户机、Web 服务器以及操作员工作站所反应的系统的状态应相一致。所谓可靠性的要求就是指客户机的系统运行要可靠、客户机和服务器之间的数据通讯要可靠，确保实时信息及时可靠地传递到客户机。

4.2 复示服务器的设计与实现

复示服务器是放在调度中心，之所以称之为服务器是从它所提供的功能来说的。如前所述，它最基本的功能就是：为客户机提供所需运行的代码（采用 Java 语言编写的 Applet 形式）和系统初始化时所需的数据、构建客户机本地的实时数据库；及时可靠地将调度中心中的网络数据转发到客户机，确保客户机所反映的系统状态和当前地系统状态相同步。

复示服务器又分为三部分：SCADA 应用程序部分、实时数据库部分和网络通讯部分。SCADA 应用程序所完成的主要任务有站的画

面显示（和操作员工作站相同）、远程用户的管理等；实时数据库是系统初始化时从数据库得到的初始数据，它包括系统配置数据、被控站数据以及画面显示数据等，实时数据库一方面为复示服务器提供数据服务，另一方面也为客户机提供数据服务，客户机在系统初始化时，将服务器端的实时数据库拷贝到本地，以后客户机上有关数据的查询、修改都在本地运行，这样可以提供系统的响应速度，满足系统对实时性的要求；网络通讯模块也有两方面的作用：一方面为本地 SCADA 程序提供网络数据，也就是将接收的网络数据交给 SCADA 程序以便它来更新数据和画面显示，另一方面网络通讯模块还需将接收到的网路数据准确可靠地发送到客户机，以便客户机利用这些数据来更新实时数据库和画面显示。整个复示服务器的工作过程如图 4-1 所示。

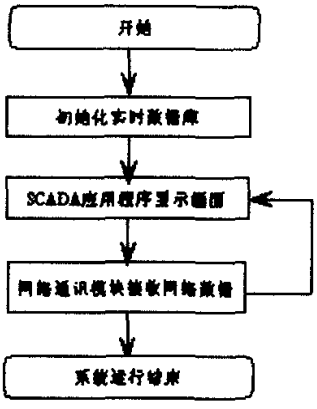

图 4－1 复示服务器工作流程图

下面就分别介绍这三大模块的设计与实现。

4.2.1 实时数据库的设计与实现

实时数据库是在系统初始化的时候根据从数据库中取出的数据构造的，对内将数据加以封装，对外提供访问接口供 SCADA 应用程序查询、修改。它以链表的形式存在于服务器的系统中，并被封装在一个类（class）中。类的构造及类与类之间的关系和系统的逻辑构成相一致。实时数据库是可重用的，客户机和服务器上的实时数据库是一样的，因为它们的本质上是一样的，只是运行的位置不一

样，客户机在系统初始化的时候，将服务器上的数据库拷贝到本地，以后所有有关数据的操作都在本地运行，而不需要提交到服务器上运行，减少服务器的负荷，同时提高系统的响应。服务器和客户机之间的数据库关系如下图所示

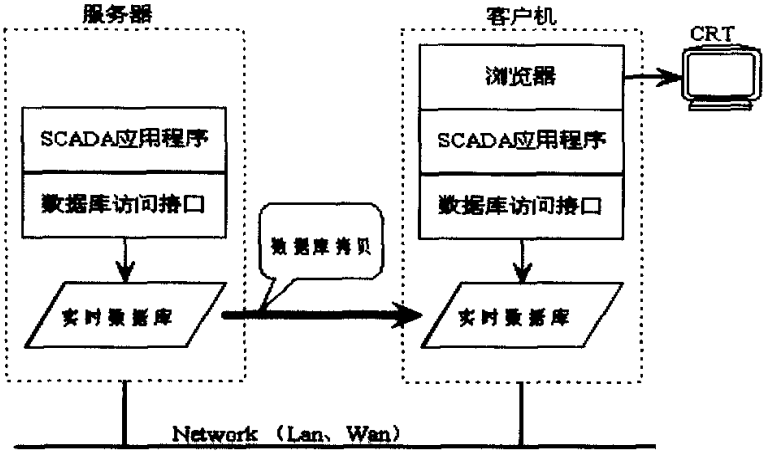

图 4－2 调度端的系统组成图

实时数据库包括两个部分：被控站对象库和画面显示库。被控站库中有被控站表、信号对象表、遥测对象表、脉冲对象表、遥调对象表、事故关联、以及开关联动表。画面显示库有画面显示描述表、位图对象表、线条对象表、文本对象表标准块对象表以及节点表。画面显示库和被控站中的遥信、遥测对象是通过节点表来关连。一个被控站中有多个对个遥信对象、多个遥测对象、多个脉冲对象、多个遥调对象。它们之间是一对多的关系。被控站中的对象关系如下图所示，其中 CBase 是一个基类，CSubStation 代表被控站，CSingalObject 代表遥信对象，CPulseObject 代表脉冲对象，CValueObject 代表遥测对象。CSubStation、CSingalObject、CPulseObject 和 CValueObject 分别继承自 CBase 类。它们重用了名称、站码和站标识符三个变量以及对这三个变量的操作。

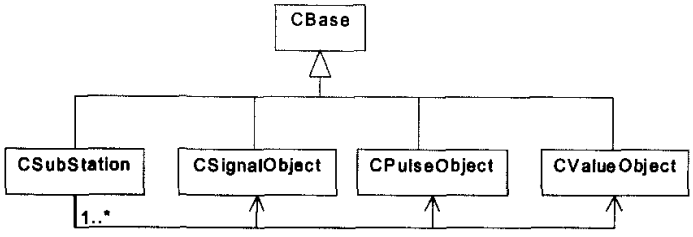

图4－3数据类之间的继承关系

被控站在构建的时候，根据自身的站码，查找遥信表、遥测表、脉冲表来构建属于本站的遥信链表、遥测链表以及脉冲链表。在CSubStation类中，这些链表都是以私有属性加以封装。CSubStation类负责提供查询、添加、删除等操作的接口。下面是节选自CSubStation.java中的部分代码。

```
/*遥信对象链表*/
protected LinkedList   SignalObjList= new LinkedList();
/*遥测对象链表*/
protected LinkedList   ValueObjList= new LinkedList();
/*脉冲对象链表*/
protected LinkedList   PulseObjList= new LinkedList();
构建接口
public   boolean BuildSignalObjList(Connection con)
/*根据 CSubStation 中的站码创建遥信对象链表*/
public   boolean BuildPulseObjList(Connection con)
/*根据 CSubStation 中的站码创建脉冲对象链表*/
public   boolean BuildValueObjList(Connection con)
/*根据 CSubStation 中的站码创建遥测对象链表*/
访问 CSignalObject 对象的接口
public int GetSignalObjectNum()
public CSignalObject GetSignalObject(int index)
public CSignalObject SearchSignalObject(short code, boolean yk)
public CSignalObject SearchSignalObject(String id)
访问 CValueObject 对象的接口
```

public CValueObject SearchValueObject(short code)
public CValueObject SearchValueObject(String id)
public CValueObject GetValueObject(int index)
public int GetValueObjectNum()
访问 CPulseObject 对象的接口
public CPulseObject SearchPulseObject(short code)
public CPulseObject SearchPulseObject(String id)
public CPulseObject GetPulseObject(int index)
public int GetPulseObjectNum()

再将整个系统中的站信息封装起来，形成链表，放在 CRealDB 类中。CRealDB 类中除了站链表外，还有节点表（开关节点和其它节点）、开关状态、画面关连表、画面被控站关联表以及测量-节点关联表。下面是节选自 CRealDB.java 文件中的部分代码。

```
public Vector KeyNode=new Vector();//开关节点
public Vector UseNode=new Vector();//其它节点
private LinkedList stateArray=new LinkedList(); //开关状态
private LinkedList PictureList=new LinkedList();//画面连表
private LinkedList PicRelateList=new LinkedList();//画面被控站关联表
private LinkedList YcRelateNodeList=new LinkedList();//测量-节点关联表
```

由于这些链表都是以私有数据的形式存在于类中，CRealDB 类也提供了相应的公用操作接口供外部对象调用，这里就不多述啦。

画面库主要有这样几个类：Cpainted（基类）、CpaintLine（线条）、CpaintStdblock（标准块）、CpaintText（文本）、CpaintValue（数值）、CpaintBitmap（图片）以及 CpaintCombo（组合图元），它们之间的关系如下图。

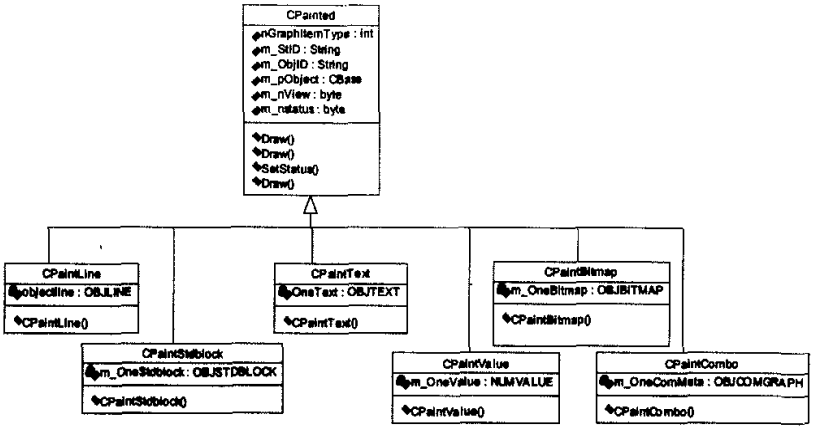

图 4－4 画面库中各类之间的继承关系图

所有这些画面库封装在 CMyPicture 类中，下面是 CMyPicture.java 中的部分代码。

类属性部分

```
private Vector m_Objects=new Vector();//object lists
```

//画面元素链表，将一个 picture 中所有线条对象、文本对象等放入该向量中。

```
privateGRAPH_PROPERTYm_PicProperty=new RAPH_PROPERTY();
```

//画面的属性（包括画面名称、标识、类型、画面高度和宽度等）。

Get 操作部分

```
public String getPicID(){return m_PicProperty.picID;}
public short getType(){return m_PicProperty.PicType;}
public short getPicNo(){return m_PicProperty.picNo;}
public String getGraphName(){return m_PicProperty.GraphName;}
public short getHight(){return m_PicProperty.PicHeight;}
public short getWidth(){return m_PicProperty.PicWidth;}
```

构件画面元素向量

```
private boolean BuildLineList(Connection con)//线条对象表
private boolean BuildBmpList(Connection con)//位图对象表
private boolean BuildConPicPatternList(Connection con)//组合图元表
private boolean BuildNumValueList(Connection con)//数值对象表
```

private boolean BuildStandardBlockList(Connection con)//标准块表

private boolean BuildTextList(Connection con)//文本对象表

最后将所有的 CmyPicture 对象封装在 CRealDB 中。整个数据库类关系如下图所示。

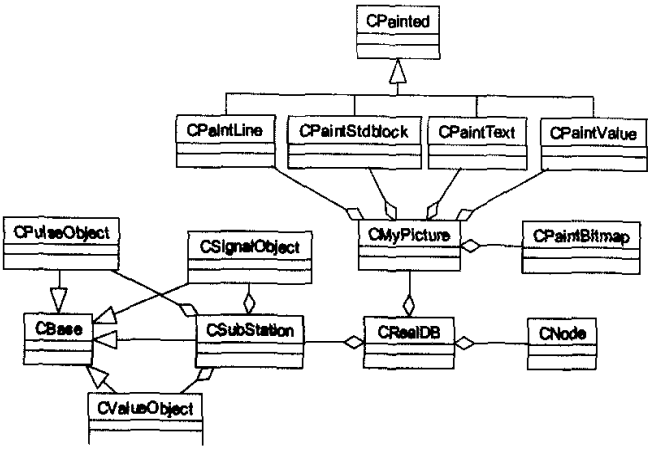

图 4－5 系统中各类之间的关系

CRealDB 的一个作用是为服务器提供系统数据，另外客户机在初始化的时候还需要将客户机所需要的数据复制到客户机本地。在这里，将 CRealDB 做成一个远程 CORBA 对象，通过提供一些访问接口供客户机来调用来获得所需的数据。客户机在系统初始化的时候通过 ORB 提供的命名服务来获得 CRealDB 的引用。通过调用接口来获得系统初始化所需的数据。具体的开发流程如下图。

步骤 1：首先用 IDL 定义访问远程对象接口。下面的代码节选自 getRealDBFromServer.idl 文件。

```
interface getRealDBFromServer
{
//数据结构定义
    typedef sequence<CNode> KeyNodeSeq;
    typedef sequence<CNode> UseNodeSeq;
    typedef sequence< CStatebuff> stateSeq;
    typedef sequence< CMyPicture> PictureSeq;
    typedef sequence< picRelate> picRelate Seq;
```

typedef sequence< CYcRelateNode> YcRelateNodeSeq;

typedef sequence< CSubStation> SubStationSeq;

//异常定义

exception BadRealDB{String Reanson};

//接口方法定义

KeyNodeSeq getKeyNode() raise BadRealDB:// 开关节点

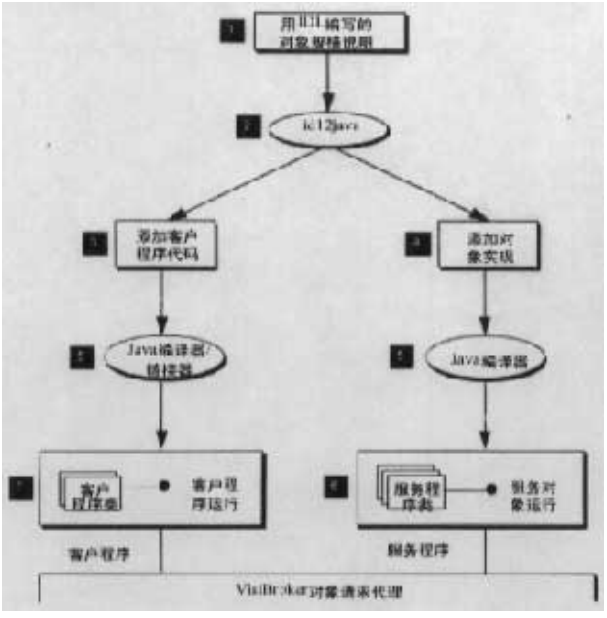

图 4－6 开发流程图

UseNodeSeq getUseNode()raise BadRealDB;// 其它节点

StateSeq getstateArray() raise BadRealDB; //开关状态

PictrueSeq getPictureList() raise BadRealDB;//画面连表

picRelate getPicRelateList()raiseBadRealDB;//画面被控站关联表

YcRelateNodeSeq getYcRelateNodeList()raise BadRealDB;//测量-节点关联表

SubStationSeq getSubStationList()raise BadRealDB;//被控站对象

}

步骤 2：生成客户程序桩和服务程序的 serverant

用 idl2java 编译器编译 getRealDBFromServer.idl 文件生成如下

文件：

_getRealDBFromServerStub.java：客户端 getRealDBFromServer 对象的桩代码。

getRealDBFromServer.java：getRealDBFromServer 的接口声明。

getRealDBFromServerHelper.java：声明 getRealDBFromServerHelper 类，该类定义帮组工具的方法。

getRealDBFromServerHolder.java：声明 getRealDBFromServerHolder 类，该类为传递 getRealDBFromServer 对象提供支持。

getRealDBFromServerOperation.java 该类定义了 getRealDBFromServer 接口定义的方法的声明。

GetRealDBFromServerPOA.java：服务器端 getRealDBFromServer 对象实现的 POA servant 代码（实现的基类代码）。

GetRealDBFromServerPOATie.java：该类用于以组带机制实现服务端的 getRealDBFromServer 对象。

步骤 3：实现客户程序。客户机执行一下几步：

（1）初始化 ORB

（2）帮定到一个 CRealDB 对象。

（3）通过调用 getRealDBFromServer 对象上的方法获得客户机所需要的数据。

```
public class ClientApplet extends JApplet implements MouseListener
{
   public void getDatafromRealDB()
   {
    // 初始化 ORB
    org.omg.CORBA.ORB orb=org.omg.CORBA.ORB.init(args,null);
    //取得 CRealDB 的 ID
    byte[] CRealDBId= "CRealDB".getBytes();
    // 获得 CRealDB 对象引用
    getRealDBFromServer realdb=
    RealDB.getRealDBFromServerHelper.bind(orb,"/realdb_agent_po
    a", CRealDBId);
    If(realdb!=null)
```

```
    {
      try
      {
       KeyNodeArray=realdb.getKeyNode();// 开关节点
       UseNodeArray=realdb getUseNode();// 其它节点
       StateArray=realdb.getstateArray() ; //开关状态
       PictrueArray=realdb.getPictureList() ;//画面连表
       PicRelateArray=realdb.getPicRelateList();//画面被控站关联
       表
       YcRelateNodeArray=realdb.getYcRelateNodeList();//测量-节
       点关联表
       SubStationArray=realdb.getSubStationList();//被控站对象
      }
      catch(Exception e)
      {
        //异常处理
      }
    }
    else
    {
      // 提示错误
    }
    //数据定义
    CNode[] KeyNodeArray=null;
    CNode[] UseNodeArray=null;
    CStatebuff[] stateArray=null;
    CMyPicture[] PictureArray=null;
    PicRelate[] picRelateArray=null;
    CYcRelateNode[] YcRelateNodeArray=null;
    CSubStation[] SubStationArray=null;
}
```

步骤4：实现服务器程序。服务器要执行以下几步：

（1）初始化对象请求代理ORB

（2）用所需策创建一个可移植对象适配器POA

（3）创建servant

（4）激活servant

（5）激活POA管理器

（6）等待接入的请求。

下面是选自服务器的代码：

```
// WebServer.java
import org.omg.PortableServer.*;
public class WebServer
{
  public static void main(String[] args)
  {
    try
    {
      // 初始化ORB.
org.omg.CORBA.ORBorb=org.omg.CORBA.ORB.init(args,null);
      // 获得根POA引用
    POA rootPOA=
    POAHelper.narrow(orb.resolve_initial_references("RootP
    OA"));
    // 创建持久POA策略
    org.omg.CORBA.Policy[]policies=
    {
        rootPOA.create_lifespan_policy(LifespanPolicyValue
        .PERSISTENT)
     };
     // 用正确的策略创建 myPOA
     POA  myPOA  =  rootPOA.create_POA(  "bank_agent_poa",
        rootPOA.the_POAManager(),policies);
      // 创建服务器对象
```

```
CRealDB RealDB = new CRealDB ();
// 获得服务对象的 ID
byte[] realdbId = "CRealDB".getBytes();
//采用 ID 激活服务对象的方法
myPOA.activate_object_with_id(realdbId,RealDB);
// Activate the POA manager
rootPOA.the_POAManager().activate();
  // 等待客户机的调用
  orb.run();
 }
 catch (Exception e)
 {
  e.printStackTrace();
 }
 }
}
```

4.2.2 SCADA 应用程序的设计与实现

Web 服务器上的 SCADA 应用程序应具有这几方面的功能：远程用户管理功能、画面的显示功能（和操作员工作站相似）以及报文处理功能。远程用户管理功能就是对远程的客户对服务器的访问权限加以管理（包括添加用户、删除用户以及根改用户信息等）；画面显示的功能和操作员工作站相似（显示各个变电站站的主接线图，分区所图等）；报文处理功能主要是根据有效上行信息来实时更新画面（使得画面所反映的状态和当前系统状态相吻合）。除此之外，SCADA 应用程序还负责创建实时数据库和网络通讯模块。SCADA 应用程序是整个 Web 服务器调度中心。它的主要工作流程如下图：

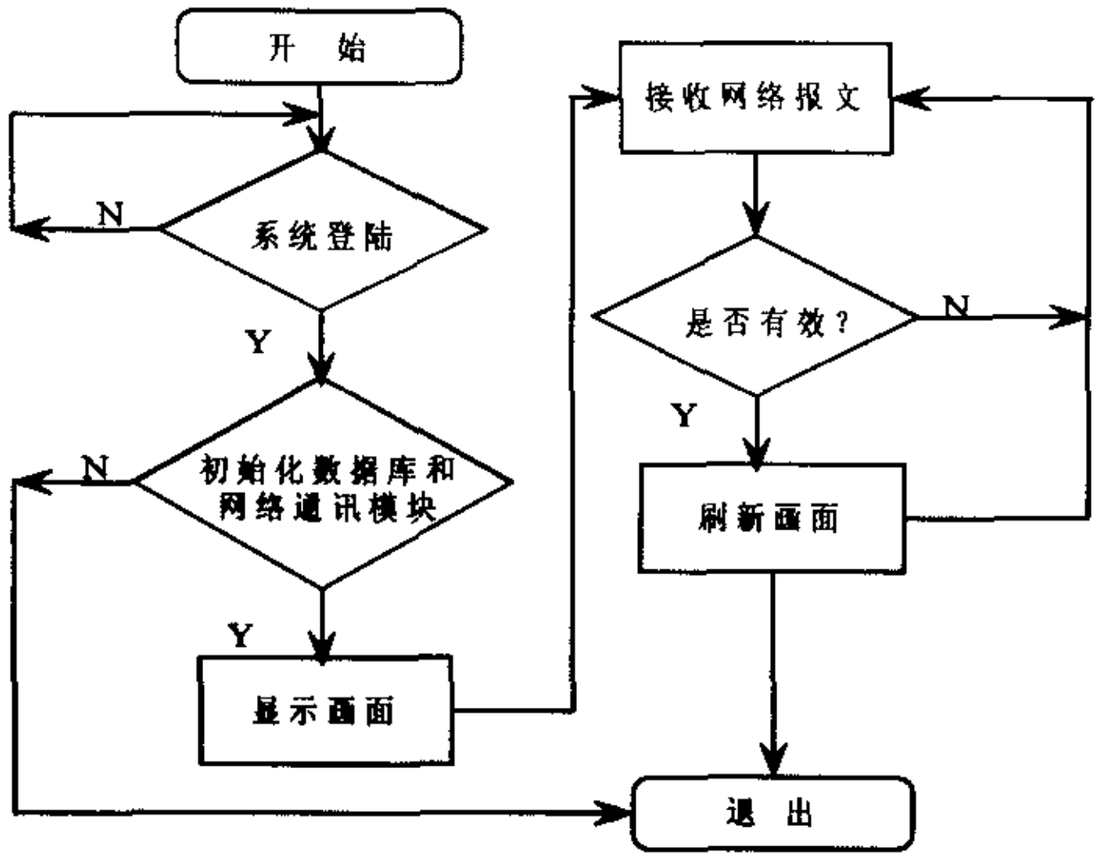

图 4－7 SCADA 应用程序工作流程图

在这里主要介绍一下报文接收处理模块。在介绍报文处理模块之前，先介绍一下网络通讯规约。

网络通讯规约定义了调度管理自动化系统调度端局域网络内各节点机间进行数据传输时的会话层/应用层数据帧格式。标准适用于 IEEE 802.3 以太网，数据传输方式可以是全双工或半双工。规约定义网络通讯帧由固定长度的会话层报头及应用层报文组成，帧格式如下定义：

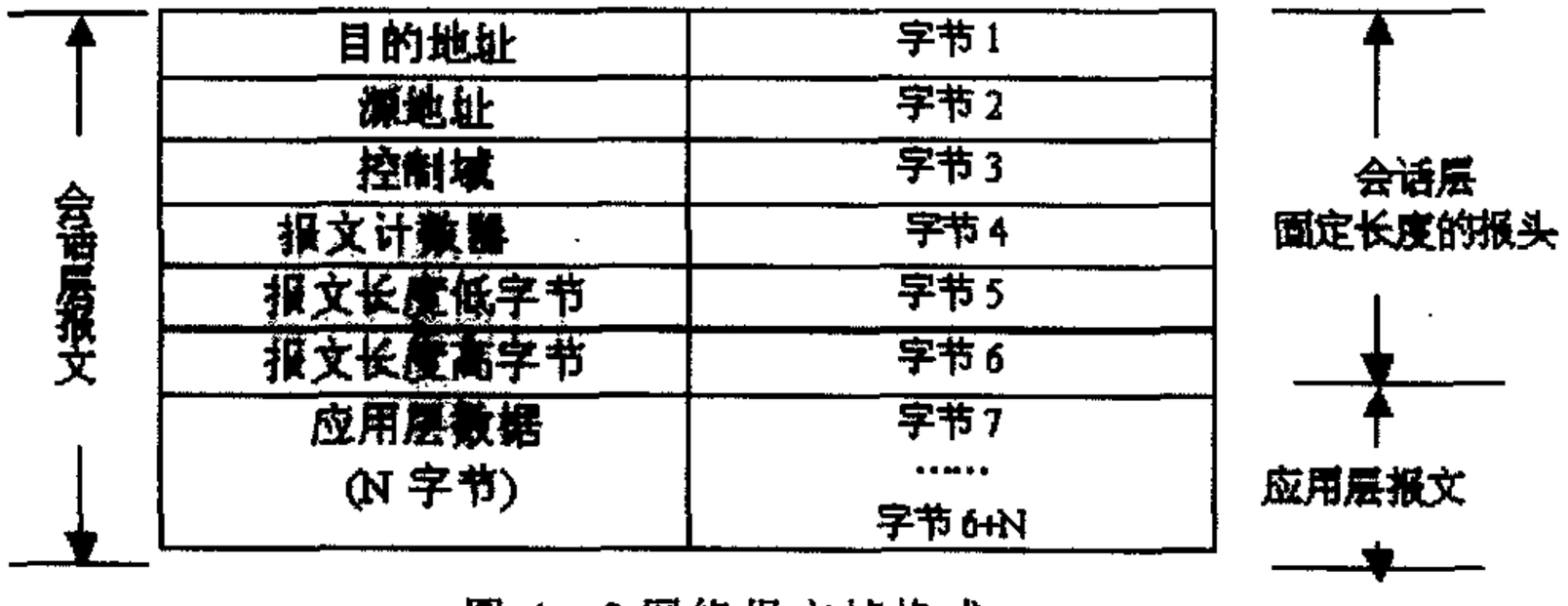

图 4－8 网络报文帧格式

应用层的报文有 RTU 通讯类、网络节点操作类、系统时钟类以及操作员工作站操作类。在 Web 服务器上报文处理模块主要用到了 RTU 通讯类的上行报文。报文产生方是前置通讯处理机。目的地址是后

台处理机、相应台的操作员工作站。应用层报文的格式如下：

字节 0	报文类型 =1		报文头部
字节 1	操作员代码		
字节 2	模拟操作标志	00H：正常操作；FFH，模拟操作	
字节 3	方向 = 1	0：下行信息；1：上行信息	
字节 4	被控站地址		
字节 5	命令码		
字节 6	长度= n+2		
报文内容			报文内容

图 4－9 应用层报文格式

其中报文类型为 1 表示是 RTU 通讯类报文。命令码表示是那一类具体的上行报文。不同类型的上行报文其数据格式是不一样的。详细的数据格式参阅《调度管理自动化系统网络通讯协议》。SCADA 应用程序每取出一条报文，首先判断报文头部的报文类型，再判断命令码，根据这两个字段找到相应的报文处理函数，调用该函数处理报文。图 3－13 是报文处理流程图。下面的部分代码说明这个工作过程。

```
public void   Dispatcher()
{
    Buffer = Netserver.CommGetNetData();
    if(Buffer!=null)
    {
        MainFrame.DisplayDatagram(Buffer,false);/*显示接收报文*/
        switch (Buffer[0])
        {
            case 1://   RTU 通讯类报文
            if (Buffer[3]==1)//判断上下行
            {
            switch (Buffer[5])
            {
            case 1:YXReply();break;//遥信应答
```

```
case 2:YCReply();break;//遥测应答
case 3:YXVaried();break;//遥信变位
case 4:PulseReply();break;//脉冲应答
case 5:YCVaried();break;//遥测变位
case 6:YKSelected();break;//遥控选择
case 7:FaultDetectReport();break;//故测
case 8:YXChanged();break;//
        case 9:YKZXResult();break;
        }
    ........
}
```

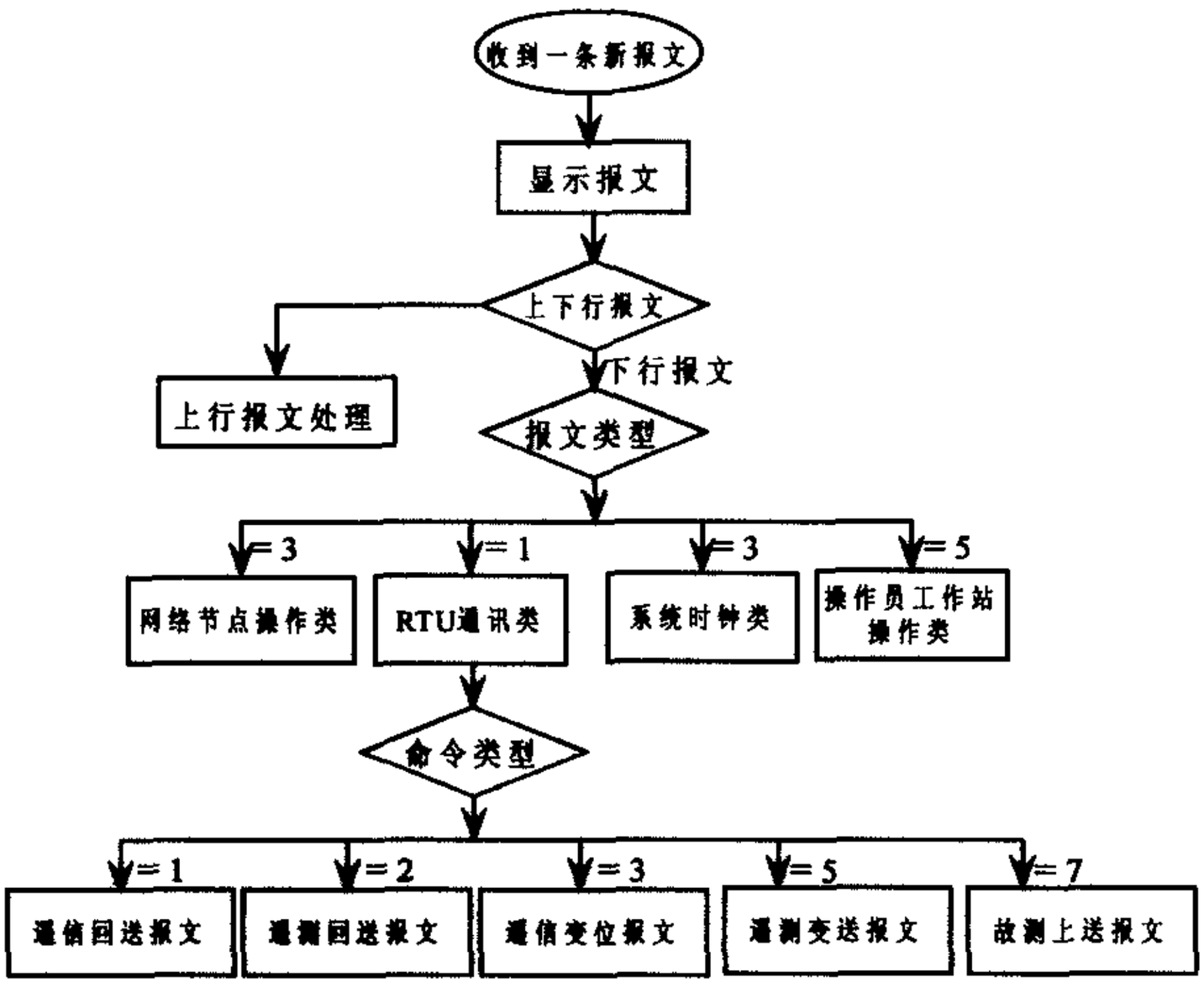

图 4－10 报文处理流程

这里的新报文来自 Web 服务器上的网络通讯模块，它负责接收网络报文，它提供一种可靠的无连接服务，保证报文的可靠性，准确性。SCADA 只负责处理报文，而不用管接收报文的。每当处理完一条报文，就从缓冲区中取下一条报文，一直循环下去。

下面以一条典型的遥信变位报文的处理过程来阐述一下 SCADA 应用程序具体处理报文的过程。遥信变位的报文定义格式如下表。

表 4－1 遥信变位报文的格式

字节 7	原因	0：突发 1：被请求 2：远动 3：当地 4：未知（光芒规约）
字节 8	对象地址低字节	
字节 9	对象地址高字节	
字节 10	对象变位性质	0: 分 1: 合 2:异常
字节 11	时	0-23
字节 12	分	0-59
字节 13	毫秒低字节	
字节 14	毫秒高字节	
....	....	

SCADA 接收报文处理模块首先获得遥信变位报文中的被控站地址、遥信对象的地址、变位的时间和原因以及是否是模拟标志等参数。再根据被控站地址、遥信对象的地址查找实时数据库找到相应的被控站和遥信对象。检查是否这两个对象引用是否不为空，否则返回。再判断该遥信对象是否遥信变位禁止，如禁止，则返回，否则显示遥信变位事件，改变遥信对象的状态，根据新的对象状态来刷新遥信对象所在的画面。下面是具体的程序流程图：

4.2.3 网络通讯模块的设计与实现

Web 服务器上的网路通讯模块的作用分为两个部分，一部分是负责接收网络通讯报文并把它放入缓冲区，另外还需管理和客户端的网络通讯，把有效的网络报文转发到客户机。

网络通讯模块工作在网络模型的会话层（参考 ISO-OSI 7 层参考模型）。它为应用层提供一种可靠的无连接服务。它是基于传输层的 UDP 协议，网络节点只要有报文发送，就发送，而不要考虑报文是否可靠的到达目的地。整个系统是一种尽最大努力投递服务系统。

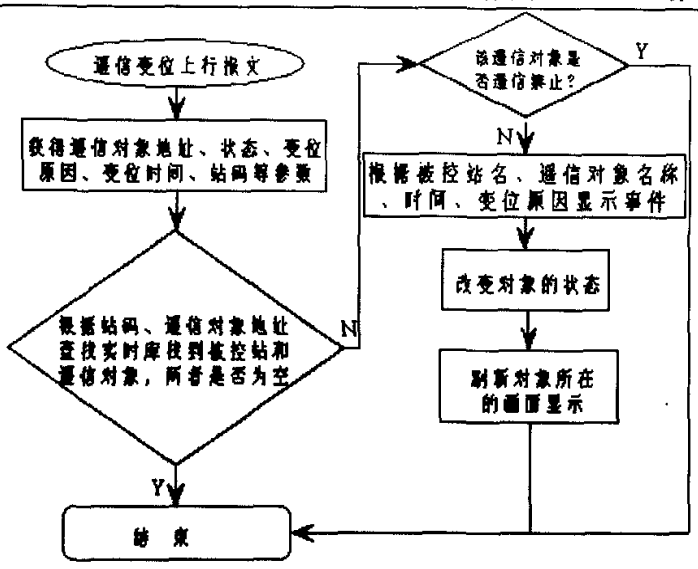

图 4－11 遥信变位处理流程

这主要是基于系统实时性要求的考虑而采用这种无连接的服务。由于整个系统对可靠性也有严格要求，所以工作在会话层的网络通讯模块必须提供一种可靠的服务。在这里，我们采用的是带重传的报文计数方式。发送端对发送的每一条报文进行编号，接收端也维持着来自不同发送端的下一条报文的编号的计数表。接收端收到一条网络报文，根据发送端的地址，查找计数表，得到应接收来自该发送端的报文的编号。如两者一致，就把该报文放入报文缓冲区，等待 SCADA 应用程序来取；如报文标号不一致，又分为两种情况：如接收报文编号大于应接收的报文编号，说明这条报文是一条新报文，将这条报文保存在临时缓冲区，供后来查询使用；如小于应接收的报文编号，说明这条报文已经接收过，是旧报文，丢弃这条报文。如果报文计数不一致，首先查找临时缓冲区，看是否以前收到过该条报文没有（一个报文计数循环后应清空该缓冲区，以免报文计数产生重复错误）。如找到就将该报文放入接收报文缓冲区，供 SCADA 应用程序来取；如没有找到，就发送一条要求重传应接收编号的报文。如过重传次数超限，应给用户提示。

由于要求报文重传，所以报文发送端在每发送一条报文，应该将报文保存一个副本，供下次重传使用。另外一点很重要的是，由于

报文计数只有一个字节，报文计数范围为（0～255），所以报文编号小的报文并不一定就是旧报文，有可能是下一轮的报文计数，所以报文计数的比较应该考虑这个问题，尤其是在一轮计数快要结束时。

网络通讯模块中共用到三个缓冲区：发送报文缓冲区、接收报文缓冲区和接收报文临时缓冲区。发送报文缓冲区用来缓存已发送的报文，如果接收端要求发送端重传某编号的报文，发送端就查找该缓冲区，找到相应报文重传此报文。接收缓冲区用来存放已经收到的正确报文，网络通讯模块提供外部接口供SCADA应用程序从该缓冲区中取报文。对该缓冲区的访问应该采用线程同步机制。报文临时缓冲区用来存放临时的正确报文，以免接收端请求发送端重发已经发送过的报文。三个缓冲区的大小都是256条报文，它们以链表的形式存在，是采用先进先出的原则的一个队列。

应用层、会话层以及传输层的报文的关系如下图。

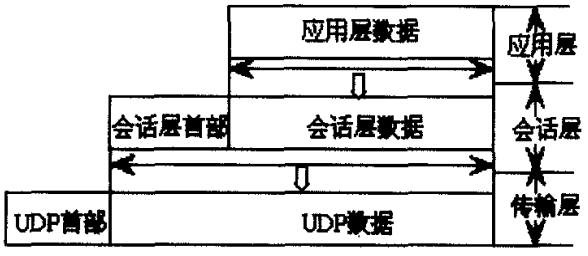

图4－12 应用层、会话层以及传输层的报文关系

应用层产生要在网络上发送的报文，它调用网络传输模块提供的发送接口，将报文提交到会话层作为会话层的数据，会话层加上会话层首部形成会话层报文。整个会话层的报文又提交到传输层传输到网络上。报文的接收过程是一个相反的过程，传输层将UDP首部去掉后得到会话层报文，会话层再将会话层的报头去掉得到应用层数据，将该数据放入接收缓冲区，等待SCADA程序处理。会话层的报文头如下图：

目的地址	字节1
源地址	字节2
控制域	字节3
报文计数器	字节4
报文长度低字节	字节5
报文长度高字节	字节6

图4－13 会话层的报文头

目的地址用来确定网络数据的接收处理方,为位使能定义。具体定义如下:

D7	D6	D5	D4	D3	D2	D1	D0
CC	SRV	MGW	MNP	OGW4	OGW3	OGW2	OGW1

图 4－14 目的地址

CC ：=1，通讯前置机将处理该报文
=0，通讯前置机不处理该报文
SRV：=1，后台处理机将处理该报文
=0，后台处理机不处理该报文
MGW：=1，维护员工作站将处理该报文
=0，维护员工作站不处理该报文
MNP：=1，模拟盘驱动器将处理该报文
=0，模拟盘驱动器不处理该报文
OGWi：=1，第 i 台操作员工作站将处理该报文
=0，第 i 台操作员工作站不处理该报文

目的地址中允许有多位同时置为 1，不允许有所有位同时置为 0。当全部为 1 时表示所有网络节点机均接收该报文。

源地址（即设备代码）用来标记网络数据的产生者，其取值范围为 0-254，255 为会话层专用。

控制域用于会话层传输控制及系统预留用，其定义如下：
会话层控制码： =1，会话层启动初始化
=2，会话层关闭
=3，重传请求
=其它，预留

D7	D6	D5	D4	D3	D2	D1	D0
会话层控制码				系统预留			

图 4－15 控制域

报文记数器用于当传输层采用 UDP 协议时会话层控制数据流的收发次序，或者当网络节点机为双网配置且无论传输层采用 UDP 或 TCP 协议时，对来自不同网络接口的冗余信息的判别，其值为 0-255 间的循环计数。

报文长度用于会话层确定后续应用层报文的总字节数，其计算公式为：应用层报文的总字节数 = 报文长度低字节 + 报文长度高字节×256。

下面仔细分析一下报文的发送过程和接收过程。

发送过程比较简单。SCADA 应用程序调用发送接口，发送接口在应用程序报文前面加上报文头信息，调用 Socket 发送出去。

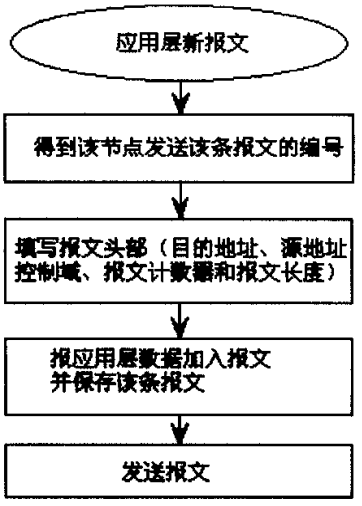

图 4－16 发送报文流程

网络通讯模块中专门有一个线程在接收网络报文。每收到一个网络报文，首先根据目的地址判断是否接收该条报文。再根据控制域字段来判断该条报文的类型：节点关机报文、节点关机报文、要求重传某编号的报文、对重传的应答报文、一般网络报文。如果是节点关机报文，在节点表中删除该节点，然后返回。如果是节点上电报文，在节点表中注册该节点，然后返回。如果是要求重传某编号的报文，根据编号和目的地址来查找发送报文缓冲区，如找到重发该报文，否则给出提示。如果是对重传的应答报文，去掉应答标志后，转到一般报文的处理过程。如果是一般报文，从节点表找到应接受的报文编号 datagramNo，和该条报文的标号相比较。这是考虑三种情况：相同、不同但 datagramNo 等于 1、不同且 datagramNo 不同等于 1。如两个编号相同说明新来的这条报文就是要接收的报文。如不同且 datagramNo 等于 1，说明 Web 服务器刚启动，也接收该报

文，将它如接收缓冲区。如不同且 datagramNo 不同等于 1，这时如 datagramNo 大，发起该条报文，否则，将该条报文，存入临时报文缓冲区中，再在临时报文缓冲区查找 datagramNo 编号的报文，看是否以前收到该报文，如找到就把该条报文放入接收缓冲区，否则发送请求重传编号为临时报文缓冲区的报文。图 3-20 时整个接收过程的流程图。

网络模块的另一个作用是管理和客户机之间的通讯，将来自调度中心网络上的报文转发给客户机。服务器上的网络通讯模块有一张客户机地址表，该表包含了客户机网络通讯模块接收来自服务器转发的网络报文的地址和端口号。客户机在系统初始化的时候给网络通讯模块发送一条上电报文，服务器上的网络通讯模块就可获得客户机用来接收报文的地址和端口号，将该地址加入客户机地址表中。

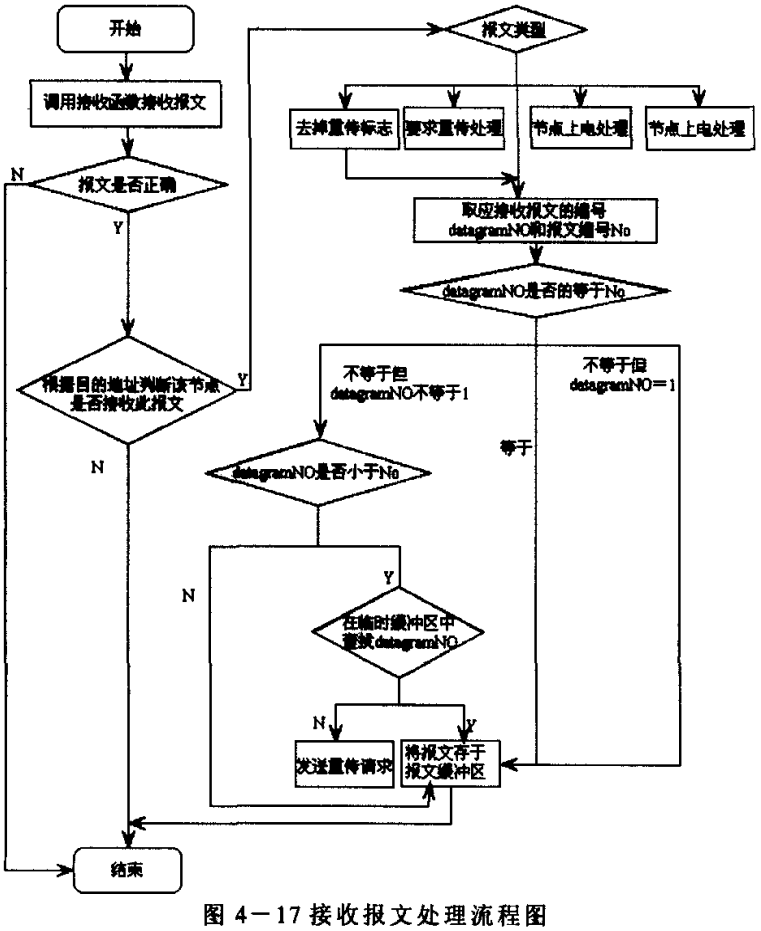

图 4－17 接收报文处理流程图

服务器在转发报文的时候根据客户机地址表来转发网络报文。当客户机系统退出时，客户机再给服务器发送一条退出系统报文，服务器删除此客户机在地址表中的地址。

4.3 客户机系统设计与实现

客户机系统相对服务器来说要简单一些，客户机采用通用的 Web 浏览器的方式，无需任何配置，简单易用。从系统结构上讲，客户机也有三大模块：SCADA 应用程序、实时数据库以及网络通讯模块。SCADA 应用程序部分和服务器的功能一样，只是执行简单的画面显示、事件显示以及网络报文显示；实时数据库部分为系统提供画面显示、被控站对象（遥信、遥测对象等）等数据；网络通讯模块用来和服务器进行通讯，获得服务器转发的网络报文，实时的刷新画面。系统初始化的时候，从服务器获得客户机运行代码（Applet）。Applet 再从服务器上拷贝系统初始化数据，初始化网络通讯模块，接收网络数据，刷新画面。下面是客户机的工作示意图。

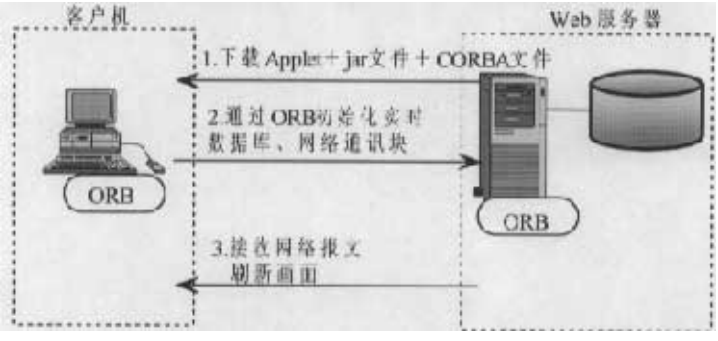

图 4－18 客户机工作示意图

4.3.1 实时数据库的设计与实现

客户机的数据是通过 ORB 将服务器上的数据拷贝到本机上的，拷贝的数据有：画面库、节点表、节点状态等。直接从服务器上拷贝数据，而不是从数据库服务器上取数据有两点好处：当前拷贝的库和系统的状态是一致而从数据库服务器上取出数据的处于初始

态。另一方面也提高了程序效率，不用频繁的访问数据库。关于这一部分的设计，请参见 Web 服务器端的数据库设计。

4.3.2 SCADA 应用程序的设计与实现

客户机上的 SCADA 应用程序和服务器上的 SCADA 应用程序几乎是一样的，都不需要进行远动操作（包括单控、程控等），它主要的功能是以图形的方式表达系统的当前状态。关于这部分的设计 Web 服务器端的 SCADA 应用程序设计。

4.3.3 网络通讯模块设计与实现

客户机端的网络通讯模块主要用来和服务器端的网络通讯模块进行通讯，接收服务器端转发的数据。由于 Applet 不支持组播的通讯方式，客户机端和服务器之间的通讯采用基于的 UDP 点对点通讯方式。由于 UDP 给应用层提供的是一种无连接的不可靠的网络通讯方式，所以网络通讯模块也要负责通讯的可靠性，这里采用的通讯方式也是带重传的报文计数机制。发送端为其转发的报文编上号，接收端收到报文后验证此编号的报文是否就是自己所要接收的报文，这一点和服务器端接收网络报文相类似。

第5章 系统的测试与结果

论文的最后阶段，搭建了一个测试系统，对整个复示系统的功能、性能（系统的响应时间、可靠性）进行了测试。

5.1 测试环境

测试系统的调度中心有四个控制台共监控坪石至广州 49 个站，RTU 端使用环路采集传输数据。后台服务器采用 HP9000UNIX 平台+Sybase 数据库。复示服务器采用 ICS 工控机（PIII500，128M 内存，Windows Professional 2000）Web 服务器采用操作系统自带的 IIS5.0。复示终端采用 ICS 工控机（PIII500，128M 内存，Windows Professional 2000）。

操作员工作站、后台服务器、通讯前置机以及复示服务器采用双以太网连接的方式。通讯前置机和被控站之间采用串口进行通讯，负责将调度中心内的网络报文按照规约转化为串口报文，它也负责将 RTU 上送的串口报文转化为网络报文。复示服务器和复示终端采用两种方式相连，以太网相连和拨号网路相连。以太网主要调度中心内部的局域网。拨号网络供远程用户使用，通过拨号到调度中心，远程访问服务器为它分配相应的 IP 地址，通过 IP 地址访问复示服务器，下载客户端代码，实时数据构建本地的实时数据库，同时根据此 IP 地址建立相应的网络通讯，接收来自调度中心的网络报文。

5.2 测试结果

测试结果分两种情况考虑（100M 以太网和 Modem 拨号）下面是一个测试表格（时间单位为秒）。

测试项目	100M 以太网下的测试结果	Modem 拨号的测试结果
系统初始化情况测试		
复示服务器的系统初始时间	166 秒	172 秒

复示终端下载数据库的时间	9 秒	70 秒
复示终端初始完了的时间（从启动 IE 到终了）	15 秒	128 秒
复示终端的初始状态和复示服务器是否一致	是	是
操作情况测试		
操作员工作站执行遥控后，复示服务器的画面是否作相应的改变	是	是
操作员工作站执行遥控后，复示终端的画面是否作相应的改变	是（部分状态不能刷新）	是（部分状态不能刷新）
操作员工作站执行遥控到复示服务器的画面作出相应的改变的时间	遥信变位:1 秒	遥信变位:3 秒
操作员工作站执行遥控到复示终端的画面作出相应的改变的时间	遥信变位:7 秒;	遥信变位:15 秒;
复示服务器和复示终端的画面是否显示正确	是	是
事件是否显示正确	是	是
网络报文是否显示正确	是	是
遥信全召、遥测全召能否下发，并能正确执行	能	能

结论

根据前一章的测试结果，复示系统在功能上基本满足了用户的要求，能够实时的、可靠的将电力系统的状态呈现给远程用户：复示服务器能够实时的、可靠的将调度中心网络内的 RTU 上行数据转发到复示终端；复示终端也能够及时可靠的呈现给用户。

本文在提出复示系统解决方案时，重点解决了复示系统中的以下几个关键问题。

1. 实时性问题。

复示系统实时性的解决是通过将复示系统中的静态数据复制到本地，和实时接收动态数据（系统新的状态） 来实现的。其中数据的复制转移是通过 CORBA 技术来实现的。数据数据的接收是通过采用基于 UDP 的网络通讯模块来实现的。

2. 可靠性问题。

复示系统的可靠性问题是通过可靠实现静态数据的本地化，可靠的接收网络报文来取得的。复示终端可靠实现静态数据本地化是通过 CORBA 本身提供的可靠传输机制来保证的。复示服务器和复示终端之间的网络通讯的可靠性是通过采用带计数的报文重传机制来处理报文丢失、报文重传以及报文延时、可靠接收等问题取得的

3. 数据的一致性。

复示服务器和复示终端之间的数据的一致性是通过系统在初始化、定时和手动发送给遥信全召、遥测全召命令到 RTU，RTU 将所辖的各设备的状态上传到调度中心局域网络内，复示服务器在更新自身系统状态的同时将网络报文转发到复示终端，使其本地的实时数据库得到更新来实现的。

由于客户机采用通用的浏览器方式，大大简化了客户机的系统开销（无需任何配置）；由于客户机的代码是从服务器下载过来的，只需要更新服务器上的代码就可方便实现客户机系统的升级；另外由于客户机是通用的浏览器而不是专用的客户端软件，它使用户的操作变得更简单。复示服务器和复示终端上的实时数据不仅是 Java 对象，也是 CORBA 对象，系统可以很好的在异构平台下的运行，实现系统的移植。

当然，系统也存在很多不足的地方，时间就是一个主要问题。造成这些的一个主要原因是：Java 本事一种解释性的语言，代码的执行效率低，所有客户机从服务器下载代码，数据到系统初始化完，需要一定的时间。在这里，可以考虑两种方式：一种是网络下载，素有的数据和代码都从网络下载，客户机无需任何配置，另外一种是客户机可以配置少量的数据文件，只需下载部分代码。另外，系统的程序代码需进一步调试和优化。

总的来说，随着 Internet 技术的发展，基于 Internet 技术的复示系统也将以其自身的灵活性被广泛应用于电力系统监控领域中。远程用户的移动监视、远程监视将提高牵引供电系统调度管理综合自动化水平，为迅捷、快速分析、查找、切除故障点、提供系统智能化水平，确保系统安全、可靠、高效运行提供了有力的技术支持。

致谢

首先感谢导师钱清泉教授在我研究生期间付出的辛勤培养及悉心指导。感谢电气化自动所的谭永东副教授对论文提出宝贵的修改建议和平时的悉心指导。特别要感谢电气化自动所的刘学军教授，是他把我领进计算机的大门，同时为我创造了好的学习环境，项目机会。还要感谢成都光芒公司的陈奇志总工，她为我的论文提出了好的修改意见，以及叶茂软件开发工程师，他和我一起开发了复示系统并对系统进行了严格的测试。最后，还要感谢在论文的完成过程种给予过帮助的何正友博士、甘亚东博士、刘贺锋硕士。

谭建龙

2002-04-18

参考文献

1. 钱清泉编著.电气化铁道微机监控技术.中国铁道出版社.2000.6：9—16
2. 钱清泉编著.监控系统理论研究. 西南交通大学出版社.1997.4
3. 盛寿麟著. 电力系统远动监控原理.中国电力出版社 1998.11
4. Yashio Ebata. Development of the Intranet-Based SCADA (Supervisory Control and Data Acquisition System).TOSHIBA Review. Vol.27,1999.
5. Yoshio Ebata. A SCADA System Based on Internet Technologies TOSHIBA 1999.
6. M.Benda. Internet Architecture:Its Evolution from an Industry Perspective. IEEE INTERNET COMPUTING,Vol.2NO2,1998
7. Stephen S.Yau, Shaowei Mao.A FrameWork for ADS Application Software Development Based On CORBA. Proceedings of the 3rd Int'l Symposium on Autonomous Decentralized System(ISADS'97). 1997, IEEE.
8. K.Kawano,M.Orimo and K.Mori,Autonomous Decentralized Systems:Concept, Data Field Architechure and Future Trend. Proceeding First Int'l Symp on Autonomous Decentralized Systems.1993,IEEE.
9. Douglas C.Schimidt and Steve Vinoski. Introduction to Distributed Object Computing.C++ Report ,January 1995.
10. Douglas C.Schimidt and Steve Vinoski. Modeling Distributed Object Computing.C++ Report ,February 1995.
11. OMG. The Common Object Request Broker. Architecuture and Specification 2.3 edition. October 1999.
12. S.S.Yau and H.Ying. A Clustering Algorithm for Object-Oriented Development of Distributed Computing System Software.Proc.5th IEEE Workshop on Future Trends of Distributed Computing System.1995
13. Groran Zecevic. Web based interface to SCADA System. 1998,IEEE

14. Steve Vinoski. CORBA:Integrating Diverse Applications within Distributed Heterogeneous Environments.IEEE Communications Magazine.Vol.35,1997.2
15. 周之英.互操作和部件标准化接口 CORBA/DCOM.电子展望与决策.1998 年第 2 期.
16. E.Douglas Jensen. A Proposed Initial Approach to Distributed Real-Time Java. IEEE INTERNET COMPUTING. January.February 2001.
17. Java Remote Method Invocation(RMI) homepage.http:// java.sun.vom/j2se/1.3/docs/guide/rmi/
18. Java IDL. http://java.sun.com/j2se/1.3/docs/guide/idl/
19. CORBA Technology and the Java platform. Http://java.sun.com/j2ee/corba.
20. Marin Schaaf, Frank Maurer. Integrating Java an CORBA:A Programmer's Perspective. IEEE INTERNET COMPUTING. January.February 2001.
21. 王柏 王红熳 邹华编著. 分布式计算环境 北京邮电出版社 2000.8.
22. 周之英编著.现代软件工程(中) 科学出版者 2000.1.
23. 周之英编著.现代软件工程(下) 科学出版者 2000.1.
24. 王克宏等编著 Java 2 程序设计 清华大学出版社 2000.9.
25. 印旻编著.Java 语言与面向对象程序设计 .2000.9.
26. Cay S.Horstmann, Gary Cornell. Core Java 2,Volume I: Fundamentals.京京工作室译.机械工业出版社.2000.1.
27. Cay S.Horstmann, Gary Cornell. Core Java 2,Volume II: Adavanced Features.京京工作室译.机械工业出版社.2000.1.
28. Borland/Inprise 公司. Visibroker for Java Reference,Visibroker for Java Programmer's guide. 李文军 周小聪 李师贤等译. 机械工业出版社.2000.11.
29. Microsystem Sun Corperation.The Java Tutorial. Ocotorber,2000
30. R.Otte, P.Patrick. Understanding CORBA:the common object request broker architecture. 李师贤等译校.清华大学出版社

1999.10.

31. Andrew S.Tanenbaum. Computer Networks(third edition).熊桂喜等译.清华大学出版社 1998.7
32. 张龙祥编著.UML 与系统分析设计.人民邮电出版社.2001.8
33. 刘润东著.UML 对象设计与编程.北京希望电子出版社 2001.3
34. 汤子瀛,哲风屏,汤小丹著 计算机操作系统. 西安电子科技大学.1999.4.
35. 谢希仁编著. 计算机网络(第 2 版)电子工业出版社.1999.4.
36. Douglas E.Comer. Internetworking with TCP/IP Vol1:Principles,Protocol and Architecture.林瑶,蒋惠,杜蔚轩译.电子工业出版社.1998.4
37. 邵维忠,杨芙清著.面向对象的系统分析.清华出版社.1998.12
38. OMG 公司网页.http://www.omg.org.
39. IONA 公司网页.http://www.iona.com.
40. Sun 公司网页.http://java.sun.com
41. 冯玉琳等著.对象技术导论.科学出版社.1998
42. Peter Cord, Edward Yourdon 著.面向对象的分析.邵维忠等译.北京大学出版社.1991.6
43. *Dingbang Xu, Xin Han, Jianmin Wang and Yujian Chen.* A Strategy for Persistent Object Service under CORBA and WEB Environment. Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems.2000.IEEE.
44. Frank Sommers, Shahram Ghandeharizadeh and Shan Gao. Cluster-Based Computing with Active, Persistent Objects on the Web. Proceedings of the 2001 IEEE International Conference on Cluster Computing (CLUSTER'01).2001,IEEE
45. Brad Cox.Objects As Property On The Web IEEE INTERNET COMPUTING. January.February 2001.
46. Frank Manloa.Technologies for a Web Object Model. IEEE INTERNET COMPUTING. January.February 2001.
47. Philipper Merle,Christophe Gransart and Jean-Marc Geib.Intergrating CORBA Objects in the World Wide

Web.http:corbaweb.lifl.fr.

48. N. Narasimhan, L. E. Moser and P. M. Melliar-Smith.Transparent Consistent Replication of Java RMI Objects. The Proceedings of the International Symposium on Distributed Objects and Applications (DOA'00).2000, IEEE.
49. Matjaz B. Juric, Ivan Rozman, Alan P. Stevens, Marjan Hericko, Simon Nash Java 2 Distributed Object Models PerformanceAnalysis, Comparison and Optimization. 7th International Conference on Parallel and Distributed Systems (ICPADS'00) 2000 IEEE.
50. Daniel Hagimont and Fabienne Boyer.A Configurable RMI Mechanism for Sharing Distributed Java Objects. IEEE INTERNET COMPUTING. January.February 2001.
51. Eun Sook Cho, Soo Dong Kim, Sung Yul Rhew.Object-oriented Web application architectures and development strategies. Proceedings of the 4th Asia-Pacific Software Engineering and International Computer Science Conference (APSEC '97 / ICSC '97) 1997 IEEE.

附录

常用操作代码（PulicOperations.java）

```
package NetCommunication;
public class PublicOperations
{
    //将两个字节合成短整类型
    public static short ShortFromByte(byte highbyte,byte lowbyte)
    {
       short temp=(short)(((highbyte&0x00ff)<<8)|(lowbyte&0x00ff));
       return temp;
    }

    //将字节以十六进制输出
    public static String HexofByte(byte b)
    {
       byte low=(byte)(b&0x0f);
       byte high=(byte)((b&0xf0)>>4);
       return (invert(high)+invert(low));
    }

    //将短整型以十六进制输出
    public static String HexofShort(short t)
    {
       byte lowbyte=(byte)((t&0x00ff));
       byte highbyte=(byte)((t&0xff00)>>8);
       return (HexofByte(highbyte)+HexofByte(lowbyte));
    }

    //将有符号的字节转换为无符号的字节，但返回值为短整型
    public static short UnsignByte(byte b)
    {
          short t=(short)(b<0?(b+256):b);
          return t;
    }
    //将四位二进制转化为十六进制
    private   static String invert(byte b/*fout bits*/)
    {
        if(b==0x0a)
            return "A";
        else if(b==0x0b)
            return "B";
        else if(b==0x0c)
            return "C";
        else if(b==0x0d)
            return "D";
        else if(b==0x0e)
            return "E";
        else if(b==0x0f)
            return "F";
        else
            return Byte.toString(b);
    }

}
```

复示服务器端的网络通讯程序（NetInterfaceforServer.java）：

```
package NetCommunication;
```

```
import java.io.*;
import java.net.*;
import java.util.*;
public class NetInterfaceForServer extends Thread
{
   //the address of multicast group
    public static final String MUTIGROUP_ADDRESS="239.255.0.8";
   public static final int RECEIVE_PORT=4001;                    //the receive port
   public static final int SEND_PORT=3001;                       //the send port
   //the receive port and server port that is used to connect with applet
    public static final int SERVER_PORT=5000;

   //网络模块的初始化
public NetInterfaceForServer(byte b_OriginPcAddr,byte b_OriginRecvBit)
{

   try
   {
    receivesocket = new MulticastSocket(RECEIVE_PORT);
    sendsocket = new MulticastSocket(SEND_PORT);
   InetAddress address=InetAddress.getByName(MUTIGROUP_ADDRESS);
    sendsocket.joinGroup(address);
    receivesocket.joinGroup(address);
   }
   catch(UnknownHostException e)
   {
      System.out.println("Error:"+e);
   }
   catch(IOException e)
   {
      System.out.println("Error:"+e);
   }
   //发送上节点电报文
   byte[] b=new byte[6];
   b[0]=0;
   b[1]=(byte)OriginPcAddr;
   b[2]=PcPowerOn;
   b[3]=0;
   b[4]=b[5]=0;
   MutiSocketSend(b,6);
   m_ComServerandClient=new ComServerandClient(this);
   this.start();
}

//通过组播发送报文
public void MutiSocketSend(byte[] buf,int length)
{
    try
    {
    InetAddress group = InetAddress.getByName(MUTIGROUP_ADDRESS);
    DatagramPacket    packet    =    new    DatagramPacket(buf,    length,    group,
     RECEIVE_PORT);
      sendsocket.send(packet);
    }
    catch(UnknownHostException e)
    {
      System.out.println("Error:"+e);
    }
    catch(IOException e)
```

```
        {
          System.out.println("Error:"+e);
        }
    }
//接收网络报文
public void GroupMastRecv()
 {
    byte[] buf=new byte[300];
    DatagramPacket packet = new DatagramPacket(buf, buf.length);
    try
    {
      receivesocket.receive(packet);
   if(packet.getLength()<1) return;
buf=new byte[packet.getLength()];
System.arraycopy(packet.getData(),0,buf,0,packet.getLength());
 }
 catch(IOException e)
 {
    System.out.println("Receive Data Error!");
    return;
 }
 if(buf[1]==OriginPcAddr) //receive myself datagram; return;
 if(buf[2]==PcPowerOff)//PcPowerOff
 {
     //节点掉电报文处理
   return;
 }
 if(!FindOriginNameInStructNumReg(buf[1]))
 {
      //在本机注册新节点
 }
 if(buf[2]==PcPowerOn)
 {
    //节点掉电报文处理
}
//请求报文重传处理
if((buf[2] & ((byte)ResponeReDoContextType))!=0x00)
        buf[2] &=0x7f;
if((buf[2]==(byte)ReDoContext))
{
    if(buf[0]==OriginPcAddr)
  {
        byte[] b=null;
       //查找重传报文
        b=FindContextInSendBuf(OriginPcAddr,buf[6]);
    if(b!=null)
    {
         buf[2] |=ResponeReDoContextType;
        MutiSocketSend(b,b.length);
      }
    else
    {
   System.out.println("GroupMastRecv Redo Error! RegNo="+buf[6]);
    }
     return;
     }
}
byte temp=0;
Byte RegNo=GetRegNo(buf[1]);
```

```
    temp=RegNo.byteValue();
    if((buf[0]&OriginRecvBit)==0x00)
    {
       if(buf[3]==temp)
        IncRegNo(buf[1]);
else if(temp==1)
       {
        SetRegNo(buf[1],(byte)(buf[3]+1));
        CommPutNetData(buf);
       }
    return;
    }
    MyRecvPutNetData(buf);
if(buf[3]==temp)
{
    CommPutNetData(buf);
    IncRegNo(buf[1]);
 }
 else if(temp==1)
 {
      SetRegNo(buf[1],(byte)(buf[3]+1));
  CommPutNetData(buf);
 }
 else
 {
      byte[] b=null;
      //查找以前是否收到该条报文
       b=FindContextInRecvBuf(buf[1],temp);
       if(b!=null)
       {
            CommPutNetData(b);
               IncRegNo(buf[1]);
         }
         else
         {
              //发送重传请求
            GroupMastRedo(OriginPcAddr,buf[1],temp);
            }
     }
     return;
  }
  .....
  }
  //从网络上接收数据，转发给 NetInterfaceApplet
  public class NetInterfaceForClient extends Thread
  {

  .....
  //将网络报文转发给客户，采用点对点的网络通讯方式
public synchronized void CommGroupMastSend(byte DestationAddr,byte [] buf,short
length)
   {
   byte[] b=new byte[300];
   byte DatagramNo=0;
b[0]=DestationAddr;
   b[1]=OriginPcAddr;
   b[2]=0;
   IncRegNo(OriginPcAddr);
   Byte RegNo=GetRegNo(OriginPcAddr);
```

```
    if(RegNo==null)    return;
    b[3]=RegNo.byteValue();
    b[4]=(byte)(length & 0x00ff);
    b[5]=(byte)((length>>8)&0x00ff);

    System.arraycopy(buf,0,b,6,length);

    b[6+length]=0;      //crc
    length=(short)(length+7);
    MutiSocketSend(b,length);
    MySendPutNetData(b,length);
    return;
  }

  ..........................
}
```

攻读学位期间发表的论文

1. 谭建龙 钱清泉.基于 Intranet 的微机监控系统工程 电力自动化设备（已录用）