摘要

网络管理已经成为计算机网络研究与建设中一个非常重要且必不可少的部分，它决定着网络资源的利用率和效益的发挥。基于 Web 的网络管理系统是有机的集成了 CORBA 技术与 Web 技术，具有三层 B/S 结构的网络管理系统。它是目前网络管理领域内最新的发展趋势和研究热点，对计算机网络管理具有重要的现实意义和良好的应用前景。

本文的研究是以 YSZ2002 电力监控网络管理系统为设计背景而展开的，提出了基于 Web 的具有 YSZ2002 电力监控系统特色的网络管理系统总体结构的设计思想，以及其安全管理模块的设计与部分实现。

本文首先介绍网络管理的一些基本概念，包括一般网络管理的体系结构、功能的划分及其国际标准和一些主要的网络管理协议，阐述在本系统中用到的中间件技术，并选择了 CORBA 技术作为本系统的中间件。随后，在分析基于 Web 的网络管理国际标准的基础之上，提出了一个基于 Web 的分布式网络管理模型，包括组织模型、通信模型、功能模型、信息模型，并把它应用于现有的 YSZ2002 分布式电力监控系统的网络管理功能部分的设计之上。在基于 Web 的网络管理系统中，为了使管理者能够更好的通过 Internet 管理被管设备，本文采用 Web Service 技术来完成管理者与被管设备之间的通信，详细的阐述了 Web Service 中的关键技术，并在 YSZ2002 分布式监控网络管理系统中设计和实现了这部分的功能。由于在网络管理功能中，安全管理功能是重中之重，因而本文设计和部分实现了基于 Web 的 YSZ 电力监控网络管理系统中的安全管理。

在本文的最后对基于 Web 的网络管理进行了展望，并提出了此网络管理模型今后需要改进和研究的地方。

关键字：基于 Web 的网络管理、中间件、网管协议、CORBA、Web Service、安全管理

Abstract

Network management becomes one of the very important and indispensable part of the research and construction of computer network, deciding to exerting using rate and benefit of the network resource . Web-based Management System is the Network Management System that possesses three-tied B/S level architecture integrated with CORBA and web technology . Being the newest tendency and present research focus, it is an object which has great realistic significance and extensive practical foreground in network management area .

This thesis is based on the research of YSZ2002 Network Management in Distributed Monitoring System, and comes up with the architecture of a web-based, YSZ2002 Monitoring System featured Network Management System; a Security Management Module is also designed and partly realized.

The article starts with the basal theories of network management, including its architectures, functions, international standards and protocols; it also expatiates the middleware technology used in this system and chooses CORBA as its middleware. After analyzing the international standards for Web-Based Network Management, this article raised a web-based distributed network management modal, including its organization modal, communication modal, function modal and information modal, and applies it into the YSZ2002 Network Management in Distributed Monitoring System. In order to facilitate the supervision of devices in web-based network management system, the essay accomplishes the web communication of the supervisor and the equipment. The key technology of Web Service is detailed explained, and the function is also designed and implemented in the YSZ2002 Network Management in Distributed Monitoring System. Since security is of vital importance in network management system, this paper designs and partly implements the security management in the web-based YSZ2002 Network Management in Distributed Monitoring System.

At the end of this thesis, the web-based network management is prospected, its future research and improvement aspect is also proposed.

Keywords:

Web-Based Network management, Middleware, NM protocol, CORBA, Web Service, Security management

第一章 绪论

1.1 选题背景及意义

随着网络技术的不断发展，网络的规模、速度和复杂度的增加，网络管理的重要性也不断增加。网络管理是控制一个复杂的计算机网络使得它具有最高的效率和生产力的过程 ，根据进行网络管理的系统的能力,这一过程通常包括数据收集，自动地或通过管理者的手动方式进行数据处理,然后提交给管理者。在网络操作中使用网络管理 ，还包括分析数据并提供解决方案，甚至可以不需要打扰管理者而自动处理一些情况，进一步还可以产生对管理者管理网络有用的事件报告。通常，网管系统主要是由管理者和管理代理两种实体组成 ，网络管理员通过管理者(Manager)与管理代理(Agent)之间的交互通信而达到对网络进行管理的目的。目前，网络管理协议主要有基于 TCP/IP 的简单网络管理协议(SNMP)和基于 OSI 的公共管理信息协议(CMIP)。网管功能主要包括配置管理、故障管理、性能管理、安全管理和计费管理五大部分。其实，这主要是针对电信运营网络而言的。对于企业计算机网络来说，一般不需要计费管理，所以网管功能基本包括配置、故障和性能、安全管理四个方面，而安全管理可以说在网管系统中是重中之重。

本文是要在基于中间件技术的具有分布式体系结构的 YSZ2002 电力监控管理系统的基础上，设计和实现一种基于 Web 技术的网络管理体系结构，它是对现有 YSZ2002 网络管理系统的改良和更新。

我们已有的 YSZ2002 电力监控系统是一个分布式的体系结构，以前单一主机的 Manager-Agent 管理模式已不能适应现在的监控系统的发展，因此需要一个分布式的网络管理模式，C/S 计算模式取代了传统的 Manager-Agent 管理模式，当今的网络管理理念是方便和快捷，要求管理员不必像以前一样登陆网络管理中心来进行网络管理，而是通过浏览器来完成管理任务，于是基于 Web 的网络管理模式就应运而生了。

本系统是基于 Web 技术的，客户是通过浏览器来对现场设备进行实时监控的，所以系统研究的是基于浏览器(browser)-服务器(server) B/S 结构的网管模型。基于 Web 技术的网络管理模式使网络管理者分布更加灵活、处理逻辑更加集中、管理能力更加强，在结构上为进一步提高系统的可重用性、安全性、可扩展性和维护性创造了良好的条件。有望最终实现以面向对象方式实现广泛的互联管理和控制。除了基于 Web 的网络管理部分，分布式网络管理还存在较多有待解决的问题。如解决异构环境互操作问题， 可以采用“中间件”的方法来解决分布异

第一章 绪论

1.1 选题背景及意义

随着网络技术的不断发展，网络的规模、速度和复杂度的增加，网络管理的重要性也不断增加。网络管理是控制一个复杂的计算机网络使得它具有最高的效率和生产力的过程 ，根据进行网络管理的系统的能力,这一过程通常包括数据收集，自动地或通过管理者的手动方式进行数据处理,然后提交给管理者。在网络操作中使用网络管理 ，还包括分析数据并提供解决方案，甚至可以不需要打扰管理者而自动处理一些情况，进一步还可以产生对管理者管理网络有用的事件报告。通常，网管系统主要是由管理者和管理代理两种实体组成 ，网络管理员通过管理者(Manager)与管理代理(Agent)之间的交互通信而达到对网络进行管理的目的。目前，网络管理协议主要有基于 TCP/IP 的简单网络管理协议(SNMP)和基于 OSI 的公共管理信息协议(CMIP)。网管功能主要包括配置管理、故障管理、性能管理、安全管理和计费管理五大部分。其实，这主要是针对电信运营网络而言的。对于企业计算机网络来说，一般不需要计费管理，所以网管功能基本包括配置、故障和性能、安全管理四个方面，而安全管理可以说在网管系统中是重中之重。

本文是要在基于中间件技术的具有分布式体系结构的 YSZ2002 电力监控管理系统的基础上，设计和实现一种基于 Web 技术的网络管理体系结构，它是对现有 YSZ2002 网络管理系统的改良和更新。

我们已有的 YSZ2002 电力监控系统是一个分布式的体系结构，以前单一主机的 Manager-Agent 管理模式已不能适应现在的监控系统的发展，因此需要一个分布式的网络管理模式，C/S 计算模式取代了传统的 Manager-Agent 管理模式，当今的网络管理理念是方便和快捷，要求管理员不必像以前一样登陆网络管理中心来进行网络管理，而是通过浏览器来完成管理任务，于是基于 Web 的网络管理模式就应运而生了。

本系统是基于 Web 技术的，客户是通过浏览器来对现场设备进行实时监控的，所以系统研究的是基于浏览器(browser)-服务器(server) B/S 结构的网管模型。基于 Web 技术的网络管理模式使网络管理者分布更加灵活、处理逻辑更加集中、管理能力更加强，在结构上为进一步提高系统的可重用性、安全性、可扩展性和维护性创造了良好的条件。有望最终实现以面向对象方式实现广泛的互联管理和控制。除了基于 Web 的网络管理部分，分布式网络管理还存在较多有待解决的问题。如解决异构环境互操作问题， 可以采用“中间件”的方法来解决分布异

构问题。本文采用 CORBA 作为中间件，所以本系统是一个以中间件为框架基础来构建的三层 B/S 结构的网络管理模型。对于这个网络管理系统功能的研究主要集中在对系统安全的管理上。

1.2 国内外发展状况

随着 Internet 的流行和发展，网络结构变得越来越复杂，这大大增加了网络管理的工作量，也给网络管理员真正管理好 Internet 带来了很大的困难。传统的网络管理方式已经不能适应当前网络发展的趋势。在网络管理技术的不断发展中，基于策略的网络管理、基于移动代理的网络管理、基于 CORBA 的网络管理、基于 XML 的网络管理、基于 Web 的网络管理等都是新一代网络管理技术研究的热点。

随着计算机网络和通信规模的不断扩大，网络结构日益复杂和异构化，网络管理也迅速发展，将 Web 技术应用于网络设备、系统和应用程序而形成的基于Web 的网络管理系统已经是时代不可逆转的潮流。作为一种全新的网络管理模式，WBNM (Web-Based Network Mannagement)从出现伊始就表现出强大的生命力，它以其特有的灵活性、易操作性等特点赢得了许多专家和用户的青睐，被誉为是“将改变用户网络管理方式的革命性网络管理解决方案”。在 WBNM 中的局域网内部建立有 Web 服务器，他们通过与超文本标记语言（HTML）有关的协议与其它用户通信。Internet 用户可以在任何一个网络节点或是网络平台上使用友好的、易操作的 Web 浏览器与服务器通信。基于 Web 的网络管理 WBNM（Web-Based Network Mannagement）的出现是网络管理领域内 Web 技术发展带来的必然趋势。WBNM 使得管理人员能够在任何站点、通过任何 Web 浏览器均可以监测和控制企业网络，并且能够解决很多由于多平台结构所产生的互操作性问题。

由于实际的需要，对网络管理的研究已成为涉及通信和计算机领域的全球性的热门课题，很多大计算机与网络通信厂商陆续推出了各自的网络管理系统，如 HP 的 OpenView、IBM 的 NetView、3Com 的 Transcend Networking、CA 的 Unicenter TNG 及 SunSoft 的 SunnetManager 等等，这些产品侧重点各不相同，在各自的领域内都获得了一定的成功，但是，他们的共同缺点是在综合性、分布性等方面有待完善。但国内的网络管理水平与国际先进水平尚有一定的差距，当前市场中的网络产品几乎为外国产品所垄断。本文要研究一种适于 YSZ2002 分布式电力监控系统的网络管理系统，在本网络管理系统中使用了 CORBA 技术和 Web 技术，并使用了 Web Service 技术来支持浏览器客户端与本地的应用系统之间信息的交互，是具有三层 B/S 结构的网络管理系统。它是目前网络管理领域内最新的发展趋势和研究热点，对计算机网络的管理具有重要的现实意义和良好的应用前景。

构问题。本文采用 CORBA 作为中间件，所以本系统是一个以中间件为框架基础来构建的三层 B/S 结构的网络管理模型。对于这个网络管理系统功能的研究主要集中在对系统安全的管理上。

1.2 国内外发展状况

随着 Internet 的流行和发展，网络结构变得越来越复杂，这大大增加了网络管理的工作量，也给网络管理员真正管理好 Internet 带来了很大的困难。传统的网络管理方式已经不能适应当前网络发展的趋势。在网络管理技术的不断发展中，基于策略的网络管理、基于移动代理的网络管理、基于 CORBA 的网络管理、基于 XML 的网络管理、基于 Web 的网络管理等都是新一代网络管理技术研究的热点。

随着计算机网络和通信规模的不断扩大，网络结构日益复杂和异构化，网络管理也迅速发展，将 Web 技术应用于网络设备、系统和应用程序而形成的基于Web 的网络管理系统已经是时代不可逆转的潮流。作为一种全新的网络管理模式，WBNM (Web-Based Network Mannagement)从出现伊始就表现出强大的生命力，它以其特有的灵活性、易操作性等特点赢得了许多专家和用户的青睐，被誉为是“将改变用户网络管理方式的革命性网络管理解决方案”。在 WBNM 中的局域网内部建立有 Web 服务器，他们通过与超文本标记语言（HTML）有关的协议与其它用户通信。Internet 用户可以在任何一个网络节点或是网络平台上使用友好的、易操作的 Web 浏览器与服务器通信。基于 Web 的网络管理 WBNM（Web-Based Network Mannagement）的出现是网络管理领域内 Web 技术发展带来的必然趋势。WBNM 使得管理人员能够在任何站点、通过任何 Web 浏览器均可以监测和控制企业网络，并且能够解决很多由于多平台结构所产生的互操作性问题。

由于实际的需要，对网络管理的研究已成为涉及通信和计算机领域的全球性的热门课题，很多大计算机与网络通信厂商陆续推出了各自的网络管理系统，如 HP 的 OpenView、IBM 的 NetView、3Com 的 Transcend Networking、CA 的 Unicenter TNG 及 SunSoft 的 SunnetManager 等等，这些产品侧重点各不相同，在各自的领域内都获得了一定的成功，但是，他们的共同缺点是在综合性、分布性等方面有待完善。但国内的网络管理水平与国际先进水平尚有一定的差距，当前市场中的网络产品几乎为外国产品所垄断。本文要研究一种适于 YSZ2002 分布式电力监控系统的网络管理系统，在本网络管理系统中使用了 CORBA 技术和 Web 技术，并使用了 Web Service 技术来支持浏览器客户端与本地的应用系统之间信息的交互，是具有三层 B/S 结构的网络管理系统。它是目前网络管理领域内最新的发展趋势和研究热点，对计算机网络的管理具有重要的现实意义和良好的应用前景。

1.3 研究内容及论文组织

1.3.1 研究内容

本文的研究内容主要包括以下几个方面：

- 现有网络管理技术的分析和研究
- 中间件技术的研究和中间件的选用
- 基于 Web 的网络管理体系结构的设计
- 将基于 Web 的网络管理应用于 YSZ2002 电力监控系统
- YSZ2002 网络管理服务的研究与设计
- 应用 Web Service 技术于网络管理中心与客户端的信息交互
- 使用 SOAP 协议在网络管理中心与客户端的信息交互时传输 XML 格式封装的实时数据的研究和设计
- YSZ2002 网络功能管理中安全管理的研究和设计

1.3.2 论文的组织结构

本文在抽象和总结工程项目的基础上，结合 CORBA 技术、Web 技术和现有的网络管理系统设计思想，从分布式监控系统的实现出发，提出了基于 Web 技术的网络管理模型。结合了现有的 YSZ2002 电力监控系统，阐明了此网络管理模型的体系结构及部分实现，给出了该系统中关于网络管理功能的安全管理部分的设计思想及其部分实现。

第一章 绪论。论文的绪论部分简要的介绍了研究网络管理的背景、目前国内外网络管理技术发展的状况，以及论文中的研究工作和论文内容的安排。

第二章 网络管理及其关键技术。本章从网络管理体系的定义、功能的划分及其它的体系结构，三个方面分析研究了现有的网络管理模型。给出了现有的网络模型 CMIP 模型和 SNMP 模型，并分别从组织模型、通信模型、信息模型和功能模型四个方面来对它们来进行描述。同时，在关键技术部分详细地分析了中间件的定义、作用、分类等内容；比较了现有的几个典型的面向对象的中间件模型确定了本系统所使用的中间件模型，并阐述了它在网络管理中的应用。

第三章 基于 Web 技术的分布式监控网络管理模型。本章首先介绍了基于 Web 的网络管理技术，包括它的实现策略、优越性及其两个国际标准。然后在此基础上设计了基于 Web 的分布式网络管理的体系结构，为了适应 YSZ2002 电力监控管理系统设计的需要，最后将基于 Web 的分布式网络管理应用于 YSZ2002 电力监控系统。

第四章 利用 Web Service 技术来实现基于 Web 的 B/S 网管体系结构。本章

1.3 研究内容及论文组织

1.3.1 研究内容

本文的研究内容主要包括以下几个方面：

- 现有网络管理技术的分析和研究
- 中间件技术的研究和中间件的选用
- 基于 Web 的网络管理体系结构的设计
- 将基于 Web 的网络管理应用于 YSZ2002 电力监控系统
- YSZ2002 网络管理服务的研究与设计
- 应用 Web Service 技术于网络管理中心与客户端的信息交互
- 使用 SOAP 协议在网络管理中心与客户端的信息交互时传输 XML 格式封装的实时数据的研究和设计
- YSZ2002 网络功能管理中安全管理的研究和设计

1.3.2 论文的组织结构

本文在抽象和总结工程项目的基础上，结合 CORBA 技术、Web 技术和现有的网络管理系统设计思想，从分布式监控系统的实现出发，提出了基于 Web 技术的网络管理模型。结合了现有的 YSZ2002 电力监控系统，阐明了此网络管理模型的体系结构及部分实现，给出了该系统中关于网络管理功能的安全管理部分的设计思想及其部分实现。

第一章 绪论。论文的绪论部分简要的介绍了研究网络管理的背景、目前国内外网络管理技术发展的状况，以及论文中的研究工作和论文内容的安排。

第二章 网络管理及其关键技术。本章从网络管理体系的定义、功能的划分及其它的体系结构，三个方面分析研究了现有的网络管理模型。给出了现有的网络模型 CMIP 模型和 SNMP 模型，并分别从组织模型、通信模型、信息模型和功能模型四个方面来对它们来进行描述。同时，在关键技术部分详细地分析了中间件的定义、作用、分类等内容；比较了现有的几个典型的面向对象的中间件模型确定了本系统所使用的中间件模型，并阐述了它在网络管理中的应用。

第三章 基于 Web 技术的分布式监控网络管理模型。本章首先介绍了基于 Web 的网络管理技术，包括它的实现策略、优越性及其两个国际标准。然后在此基础上设计了基于 Web 的分布式网络管理的体系结构，为了适应 YSZ2002 电力监控管理系统设计的需要，最后将基于 Web 的分布式网络管理应用于 YSZ2002 电力监控系统。

第四章 利用 Web Service 技术来实现基于 Web 的 B/S 网管体系结构。本章

首先研究了 Web Service 的体系结构、组成及其关键技术，随后设计和实现了YSZ2002 电力监控系统中的 Web Service 接口。完成了浏览器客户端和本地应用系统之间信息的交互。

第五章 基于 Web 的网络管理中的安全管理。基于 Web 的安全管理，可以从两个方面来考虑：一个是对本地应用系统中重要资源的访问权限，包括授权和鉴权；另一个就是数据在 Internet 上传输的安全性，本章就这两个方面给出了安全管理的具体解决方案。

第六章 结束语。展望网络管理技术的发展，总结本文的研究内容和成果，给出了基于 Web 技术的网络管理今后的研究方向。

第二章 网络管理及其关键技术

2.1 网络管理的概述及典型的网络管理模型

2.1.1 网络管理的定义

网络管理就是监视和控制一个复杂的计算机网络，以确保其尽可能长时间地正常运行，或当网络出现故障时尽可能快地发现故障和修复故障，使之最大限度地发挥其应用效益的过程。也就是说，网络管理包括网络监视和控制两个方面。

因此网络管理系统的重要任务就是：收集网络中各种设备和系统的工作参数，运行状态信息；将收集到的各种信息，以各种各样的、可视化的方式呈现给网络管理人员；接收网络管理人员的指令或根据对上述信息的处理结果向网络设备发出控制指令，即实施网络控制功能，同时监视指令执行的结果；保证网络设备按照网络管理系统的要求工作。

2.1.2 网络管理功能的划分

在OSI网络管理标准中，ISO将网络系统管理功能划分为五个功能领域，它们分别完成五个不同的网络管理功能。这种功能划分是目前建立网络管理体系结构的基础。

1、配置管理

一个计算机网络是由各种设备连接而成的。这些设备组成网络的各种物理结构和逻辑结构。在这些结构中，设备有许多参数、状态和名字等信息需要相互了解和相互适应。这对于一个大型计算机网络系统的运行是至关重要的。另外，网络运行的环境是经常变化的，网络系统本身也要随着用户的增加、减少或设备的维修而经常调整网络的配置。网络管理系统必须要有足够的手段支持这些调整或改变，使网络更有效地工作。这些手段构成了网络管理的配置管理功能。配置管理功能包括：识别被管网络的拓扑结构；标识网络中的各个对象；系统自动修改指定设备的配置；动态维护网络配置数据库等。配置管理是网络管理基本的功能。

2、故障管理

故障管理是网络管理功能中与故障检测、故障隔离、故障诊断和恢复等工作有关的部分，其目的是保证网络能够提供连续可靠的服务。网络服务的意外中断往往对社会和生产造成很大的影响，在大型计算机网络中，发现网络故障时，往往不能确定故障所在的具体位置，这就需要故障管理提供逐步隔离和最后定位故障的一整套方法和工具。有时，所发现的故障是随机性的，

第二章 网络管理及其关键技术

2.1 网络管理的概述及典型的网络管理模型

2.1.1 网络管理的定义

网络管理就是监视和控制一个复杂的计算机网络，以确保其尽可能长时间地正常运行，或当网络出现故障时尽可能快地发现故障和修复故障，使之最大限度地发挥其应用效益的过程。也就是说，网络管理包括网络监视和控制两个方面。

因此网络管理系统的重要任务就是：收集网络中各种设备和系统的工作参数，运行状态信息；将收集到的各种信息，以各种各样的、可视化的方式呈现给网络管理人员；接收网络管理人员的指令或根据对上述信息的处理结果向网络设备发出控制指令，即实施网络控制功能，同时监视指令执行的结果；保证网络设备按照网络管理系统的要求工作。

2.1.2 网络管理功能的划分

在OSI网络管理标准中，ISO将网络系统管理功能划分为五个功能领域，它们分别完成五个不同的网络管理功能。这种功能划分是目前建立网络管理体系结构的基础。

1、配置管理

一个计算机网络是由各种设备连接而成的。这些设备组成网络的各种物理结构和逻辑结构。在这些结构中，设备有许多参数、状态和名字等信息需要相互了解和相互适应。这对于一个大型计算机网络系统的运行是至关重要的。另外，网络运行的环境是经常变化的，网络系统本身也要随着用户的增加、减少或设备的维修而经常调整网络的配置。网络管理系统必须要有足够的手段支持这些调整或改变，使网络更有效地工作。这些手段构成了网络管理的配置管理功能。配置管理功能包括：识别被管网络的拓扑结构；标识网络中的各个对象；系统自动修改指定设备的配置；动态维护网络配置数据库等。配置管理是网络管理基本的功能。

2、故障管理

故障管理是网络管理功能中与故障检测、故障隔离、故障诊断和恢复等工作有关的部分，其目的是保证网络能够提供连续可靠的服务。网络服务的意外中断往往对社会和生产造成很大的影响，在大型计算机网络中，发现网络故障时，往往不能确定故障所在的具体位置，这就需要故障管理提供逐步隔离和最后定位故障的一整套方法和工具。有时，所发现的故障是随机性的，

需要经过很长时间的跟踪和分析，才能找到其产生的原因，这就需要有一个故障管理系统，科学地管理网络所发生的所有故障，具体记录每一个故障的产生、对故障跟踪分析以至最后确定并改正故障的全过程。

3、性能管理

性能管理涉及到网络通信信息的收集、加工和处理等一系列活动。其目的是保证在使用最少的网络资源和具有最小的延迟的前提下，网络提供可靠、连续的通信能力，并使网络资源的使用达到最优化的程度。性能管理的具体内容包括：从被管对象中收集与网络性能有关的数据；分析和统计历史数据；建立性能分析模型；预测网络性能的长期趋势，并根据分析和预测的结果，对网络拓扑结构、某些对象的配置和参数进行调整，逐步达到最佳。

4、安全管理

安全管理有两层含义，一方面，网络安全管理要保证网络用户和网络资源不被非法使用，另一方面，网络安全管理也要确保网络管理系统本身不被未经授权的访问。网络安全管理的主要内容包括：与安全措施有关的信息发布（如密钥的分发和访问权限设置等），与安全有关的事件通知（如网络有非法的侵入、无权用户对特定信息的访问请求等），安全服务设施的创建、控制和删除，与安全有关的网络操作事件的记录、维护和查阅等日志管理工作等。一个计算机网络的管理系统必须制定网络管理的安全政策，并根据这一政策设计和实现网络安全管理系统。在开放系统中，网络安全问题变得越来越重要了。早期 Internet 并未过多的考虑它的安全问题，使得今天的 Internet 网络的安全问题愈益突出。Internet 正在着手解决这个问题，但效果并不明显，这从另一个角度说明了网络安全管理问题的难度。

5、计费管理

计费管理功能有两个方面的用处：在网络通信资源有偿使用的情况下，计费管理功能能够统计哪些用户利用哪条通信线路传输了多少信息，访问的是什么资源等。因此，计费管理是商业化计算机网络的重要网络管理功能；另一方面，在非商业化的网络上，计费管理可以统计不同线路的利用情况，不同资源的利用情况，如果某条线路长期拥挤，那么是否考虑扩容，如果某些资源被频繁访问，那么是否考虑在近处设置一个镜像（mirror）服务器等。因此，从本质上讲，无论哪种情况，计费管理的根本依据都是网络用户使用网络资源的情况，例如信息传输量、占用线路的时间等统计量。一个网络管理系统必须记录这些信息，并选择一种用户可接受的计费方法。

2.1.3 网络管理的体系结构

网络管理的体系结构是用模型给出的，它描述了网络管理系统的整体结构

需要经过很长时间的跟踪和分析，才能找到其产生的原因，这就需要有一个故障管理系统，科学地管理网络所发生的所有故障，具体记录每一个故障的产生、对故障跟踪分析以至最后确定并改正故障的全过程。

3、性能管理

性能管理涉及到网络通信信息的收集、加工和处理等一系列活动。其目的是保证在使用最少的网络资源和具有最小的延迟的前提下，网络提供可靠、连续的通信能力，并使网络资源的使用达到最优化的程度。性能管理的具体内容包括：从被管对象中收集与网络性能有关的数据；分析和统计历史数据；建立性能分析模型；预测网络性能的长期趋势，并根据分析和预测的结果，对网络拓扑结构、某些对象的配置和参数进行调整，逐步达到最佳。

4、安全管理

安全管理有两层含义，一方面，网络安全管理要保证网络用户和网络资源不被非法使用，另一方面，网络安全管理也要确保网络管理系统本身不被未经授权的访问。网络安全管理的主要内容包括：与安全措施有关的信息发布（如密钥的分发和访问权限设置等），与安全有关的事件通知（如网络有非法的侵入、无权用户对特定信息的访问请求等），安全服务设施的创建、控制和删除，与安全有关的网络操作事件的记录、维护和查阅等日志管理工作等。一个计算机网络的管理系统必须制定网络管理的安全政策，并根据这一政策设计和实现网络安全管理系统。在开放系统中，网络安全问题变得越来越重要了。早期 Internet 并未过多的考虑它的安全问题，使得今天的 Internet 网络的安全问题愈益突出。Internet 正在着手解决这个问题，但效果并不明显，这从另一个角度说明了网络安全管理问题的难度。

5、计费管理

计费管理功能有两个方面的用处：在网络通信资源有偿使用的情况下，计费管理功能能够统计哪些用户利用哪条通信线路传输了多少信息，访问的是什么资源等。因此，计费管理是商业化计算机网络的重要网络管理功能；另一方面，在非商业化的网络上，计费管理可以统计不同线路的利用情况，不同资源的利用情况，如果某条线路长期拥挤，那么是否考虑扩容，如果某些资源被频繁访问，那么是否考虑在近处设置一个镜像（mirror）服务器等。因此，从本质上讲，无论哪种情况，计费管理的根本依据都是网络用户使用网络资源的情况，例如信息传输量、占用线路的时间等统计量。一个网络管理系统必须记录这些信息，并选择一种用户可接受的计费方法。

2.1.3 网络管理的体系结构

网络管理的体系结构是用模型给出的，它描述了网络管理系统的整体结构

和系统中的每个组件以及这些组件之间的关系，图 2－1 给出了网络管理的体系结构模型。在网络管理中普遍采用的是管理者/代理模型，包括四个组成部分：

- 管理节点

管理节点（Managed Node）即被管设备可以是主机、路由器、交换机、集线器、打印机，以及其他任何可与外界交流状态信息的设备。通常节点上能够运行网管代理 Agent。代理把来自管理者的命令或信息请求转换为本设备特有的指令，完成管理者的指示，或返回它所在地设备的信息。另外，代理也可以把在自身系统中发生的事件主动通知给管理者。每个代理还都维护一个本地数据库，库中存放它的状态、历史并影响它的运行。

- 管理工作站

管理工作站（NMS，Net Management Station）实际上是一台运行特殊管理软件的计算机。这种特殊软件就是管理者（Manager），它在网络上与代理通信，发送命令和接受应答。在这种模型中，绝大部分的计算工作由 Manager 完成，这是为了使 Agent 尽可能地减小对设备的影响。

- 管理信息库

大多数的实际运行网络采用了多个制造商的设备，为了使管理工作站能够与所有这些设备通信，由这些设备保持的信息必须有一个严格定义的标准。这个标准是网络管理模型中最大的组成部分，它详细规定了代理应该维护的确切信息以及信息的定义格式。简而言之，每个设备都具有一个或多个变量来描述其状态，这些变量被称作被管对象，不过此对象与面向对象系统中的对象含义不同，它只有状态，没有方法。网络中所有的对象都存放在一个叫管理信息库（MIB，Management Information Base）的数据结构中。

- 管理协议

管理工作站与代理的通信使用了管理协议。该协议允许管理工作站查询代理的本地对象的状态，必要时可以做出修改。网络管理协议的操作命令通常包括查询、设置和通知三大类型。

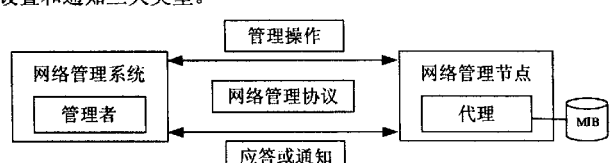


图 2－1　网络管理的体系结构模型

2.1.4 网络管理的国际标准

网络管理的国际标准主要有 ISO 的基于 OSI 参考模型的 CMIP 和 IETF 的基于 TCP/IP 参考模型的 SNMP。

2.1.4.1 CMIP 模型

公共管理信息协议 CMIP（Common Management Information Protocol）是由国际标准化组织 ISO（International Organization for Standardization）为要运行在 OSI 协议集上的开放系统提供的网络管理框架。它的特点是精心设计，复杂全面，功能强大，然而很难实现。

CMIP 是一个完全面向对象的设计，应用了面向对象的所有概念，包括继承、封装、管理对象间的关联等。其体系结构主要由四个部分组成，它们是信息模型、组织模型、通信模型和功能模型。

- 组织模型

CMIP（公共管理信息协议）的组织模型是建立在 OSI 系统管理体系结构的基础上的，如图 2-2 所示。图 2-2 左边开放系统的应用层上配置了一个管理者（Manager）实体，在右边开放系统的应用层上配置了一个代理（Agent）实体，这样就形成了两个实体之间的管理与被管理的关系。管理系统通过管理者发出访问被管系统中的管理信息的操作（operation）请求，来实现访问，同时将访问的结果发回给管理者。反过来，被管理者上的代理（Agent）也可以通过给管理者发出通知（notification），管理系统通过管理者接收请求来获得被管理者的信息，并做出必要的反应。

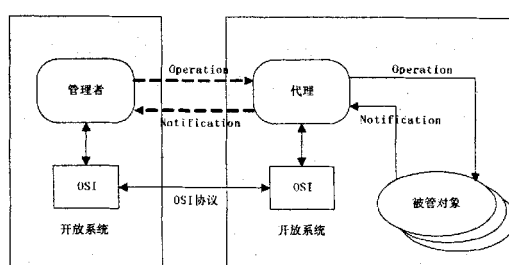


图 2-2 OSI 系统管理体系结构

- 信息模型

如果采用图 2-3 所示的远程监控的管理框架，则必须对不同厂商的网络设备以及异构网络的信息进行统一、一致和规范的描述。否则管理者就无法读取、设

置和理解远程的管理信息。为此 OSI 提出了基于 CMIP 的管理信息模型作为标准管理信息模型。管理信息模型采用面向对象技术，提出了被管对象的概念对被管资源进行描述，定义的各种标准被管对象类被赋予全局唯一的对象标识符，对被管对象的命名采用包含树的方法进行。

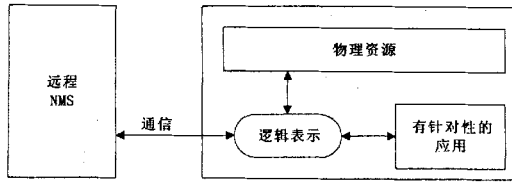


图 2-3 基于远程监控的管理框架

OSI 的管理信息模型具有以下基本特征：

1、 管理信息的定义与 CMIS 兼容，能够通过 CMIP 进行访问

2、 有一个公共的全局命名结构，对管理信息进行标示

3、 用面向对象的方法建立信息模型，管理信息被定义在被管对象中

图 2-4 是一个一般的 CMIP 被管对象的抽象。为了明确地描述管理信息模型中的被管对象，管理模型采用了著名的 ASN.1(Abstract Syntax Notation One)来有效地描述对象。

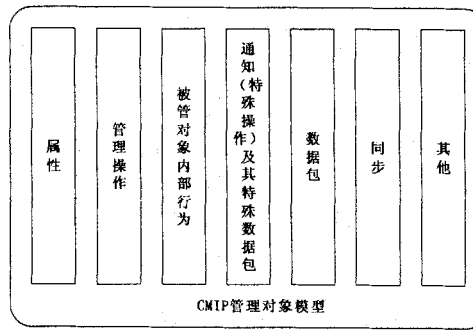


图 2-4 一般的 CMIP 被管对象的抽象

- 通信模型

OSI 提出的 CMIP 协议就是为了更好的支持管理者和代理之间的通信。CMIP 位于应用层，直接为管理者和代理提供服务。提供 CMIP 服务的实体被称为 CMIP 协议机，实体中包含三个服务元素：公共管理信息服务元素（CMISE）、联系控制元素（ACSE）和远程操作服务元素（ROSE）。

如下图所示，管理者和代理利用 CMISE 提供的服务建立联系，实现管理信息的交换。而 CMISE 利用 ACSE 控制联系的建立、释放和撤销，利用 ROSE 实

现远程操作和事件报告。ROSE 向远程系统以异步的方式发送请求和接收应答，即发出一个请求后，收到它的应答之前可以继续发送其他请求或接收其他请求的应答。

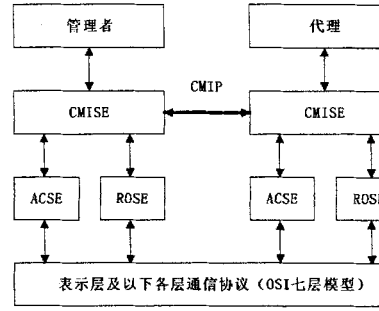


图 2-5 CMIP 通信模型

OSI 通信协议分两部分定义。一部分是对上层用户提供的服务，另一部分是对等实体之间的信息传输协议。在管理通信协议中，公共管理信息服务（CMIS）是向上提供的服务，CMIP 是 CMIS 实体之间的信息传输协议。

- 功能模型

CMIP 的功能模型包括 2.1.2 节所提到的 ISO 规定网络管理的五大管理功能：故障、配置、计费、性能和安全管理。

2.1.4.2 SNMP 模型

简单网络管理协议（SNMP，Simple Network Management Protocol）是由因特网工程任务组（IETF，Internet Engineering Task Force）制定的基于 TCP/IP 参考模型的用于计算机网络管理的标准。与 CMIP 相比，SNMP 的突出特点在于其简单性，易于实现。现在此协议发展了 SNMPv1、SNMPv2、SNMPv3 版本。

- 组织模型

SNMP 的基本体系结构是非对称的二级结构。在 OSI 系统管理体系结构中，开放系统之间存在对等的管理关系。例如：有两个系统 A、B，那么 A 可以是 B 的管理者，也可以作为 B 的被管理对象存在，反之亦然。SNMP 为了便于实现，采用了与 OSI 不同的管理体系，SNMP 的管理者和被管对象是不对称的，管理者和被管对象一般分别配置如图 2-6 所示。

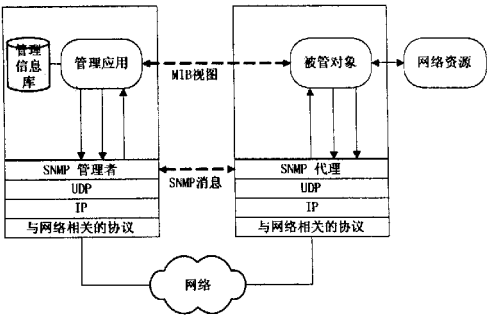


图 2-6 SNMP 的管理者和被管对象的配置

SNMP 体系结构由管理站、代理、管理信息库和通信协议构成，是一个管理站、多个代理的集中式体系结构。管理站采用轮询的方式访问各个代理中的管理信息。然而如果下面的代理太多的话，轮询将会消耗大量的资源；为了避免资源的消耗，SNMP 采用了一种陷阱引导轮询的模式。所谓陷阱引导轮询是指，管理站以不太频繁的周期进行一次初始化。在初始化期间，管理站轮询所有代理，将需要检测的关键信息独处。此后，便由各个代理通过发送 trap 消息向管理站报告重要事件。管理站一旦发现异常情况，可以直接查询报告事件的代理或者他的相邻代理，对事件进行诊断或获取关于异常情况的更多的信息。

基本的 SNMP 体系结构是 Manager/Agent 两级结构，但为了获得更高的性能和灵活性，有时 SNMP 也被配制成三级结构，也就是提出了一个代管（Proxy）的概念，并将其置于管理站和没有配置 SNMP 协议的被管设备之间，将这些设备接入 SNMP 管理系统。这样就使得一些小系统无需专门配置 SNMP 协议就可以用 SNMP 进行管理。

- 信息模型

SNMP 模型中的管理信息其实也是一种对象，只不过这种对象的抽象不同于 OSI 定义的被管理对象的模型。OSI 被管理对象完全是利用基于面向对象的分析方法将被管理资源进行的抽象，其中包括了对象的属性、操作、行为等要素；SNMP 管理信息定义的对象只是对被管理资源的一种简单抽象，SNMP 将这些被管理资源抽象为一些特定的数据结构，这些数据结构简单的反映了需要用来描述管理对象的参数的类型特征和一些数据变量。因为 SNMP 管理信息模型的简单性，SNMP 管理对象是不能被定义为严格意义上的对象的，但是它的简单性使其对象很易于管理和组织。SNMP 对象是通过树状结构将这些数据结构组织起来的，并且通过 MIB 将其管理起来。SNMP MIB 的定义主要包括了一个对象的下列信息：对象名称、对象 ID、数据类型、访问控制方式和简要描述信息。

为了规范管理信息模型，SNMP 发布了 SMI(Structure of Management

Information)。这个标准为定义 SNMP 管理信息结构和构造 MIB 提供了一个通用的框架。SMI 的基本思想是追求 MIB 的简单性和可扩充性，同时也达到了易于实现和高可互操作性的目的。因此，MIB 只存储简单的数据类型：标量和标量的二维矩阵。

SNMP 的被管对象也被组织在对象标示符注册树之中，下图为一个对象标示符注册树的 internet 节点之下构成 SNMP 被管对象标示符子树，这个子树被称为 internet MIB。MIB 树上的每个节点对应一个被管对象，节点的所属关系也就是被管对象的包含关系。因此，一个高级被管对象会包含多个层次的众多的被管对象。这一点与 OSI 系统管理模型是一致的，但是与 OSI 系统管理模型不同的是，在 SNMP 模型中，只有处于叶子节点的对象是可以直接访问的，这些对象被称为基本被管对象。

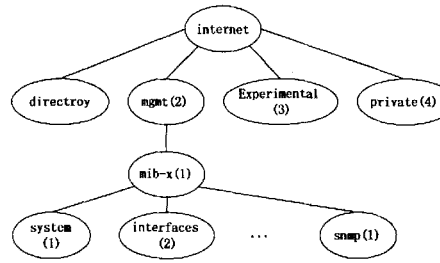


图 2-7　SNMP 被管对象标示符注册树

- 通信模型

SNMP 通信模型对 SNMP 的服务功能、SNMP 的对象访问策略、SNMP 协议以及 SNMP MIB 进行了定义。

SNMP 管理系统由配置了 Manager 的管理站和驻留了 Agent 的代理组成。SNMP 是管理站和代理之间交换管理信息的协议。为了简化代理所要实现的管理功能，SNMP 规定只能交换基本被管对象，SNMP 通过 5 种消息对网络进行管理，他们包含 3 种基本消息：get、set 和 trap。管理站通过轮询的方式访问代理，获取管理信息；代理站也可以发出 trap 信息向管理站报告特殊事件。

在 SNMP 体系结构中，管理站和代理是 SNMP 协议的应用实体。在管理的过程中，管理站和代理可能是多对多的关系，也就是说，一个代理可以接受来自多个管理站的操作，因此要进行多个被管理对象的访问控制。为了实现访问控制，需要考虑 3 个问题：首先，要限制对 MIB 的访问权限在授权管理站的权限范围内，这需要认证服务的支持。其次，针对不同的访问站要有效的制定不同的访问策略。最后，在代理系统中也要实现托管站的认证服务和访问权限的问题。SNMP 是通过在代理端定义一个访问公用体来实现的，该公用体记录了所有的认证、访

问控制和代理特性的信息，并且该公用体拥有代理内部唯一的公用体名称。管理站将公用体和代理联系起来对代理进行管理控制。

SNMP 协议除了定义了上述的认证服务、访问控制策略和代理服务之外，还定义了 MIB 中的每一个被管对象的唯一的对象标示符；同时，SNMP 还定义了管理站和代理之间交换的消息和用户数据包的格式；另外，SNMP 还定义了一些操作，这些操作包括管理站向代理发出的 get-request、get-next-request 和 set-request 三种操作，代理使用的 get-response 对上述三种操作的应答操作，以及代理使用 trap 向管理站报告异常事件的操作等。

SNMP 需要利用传输层的服务来传递 SNMP 消息，但是 SNMP 对传输层服务的可靠性，以及传输层服务是面向连接的还是无连接的都没有提出要求。在实际中 SNMP 的实现几乎都是建立在无连接的 UDP 协议的基础上的。因为 UDP 协议不能保证传输报文的可靠性，那么 SNMP 消息就有可能丢失，另外 SNMP 本身并不能保证传输报文的可靠性，因此，管理信息可能会丢失。

- 功能模型

SNMP 的功能模型包括 2.1.2 节所提到的 ISO 规定网络管理的五大管理功能：故障、配置、计费、性能和安全管理。

2.2 关键技术

2.2.1 中间件的概述

2.2.1.1 中间件的概念

中间件的定义比较多，大多是从一个特殊的角度反映了中间件的一个或几个特性。比较多的人认为中间件，就是位于操作系统和应用软件之间的一个软件层，它向各种应用软件提供服务，使不同的应用进程能在屏蔽掉平台差异的情况下，通过网络互相通信。另外，存在一个更加宽泛的定义，认为所谓中间件从本质上讲，就是一个连接应用程序并允许它们之间互相交换数据的软件层。通常，在实际使用中，把一组中间件集成在一起构成一个平台（包括开发平台和运行平台），其中必须要有一个通信中间件完成中间件之间的通信。从这个意义上讲，中间件应该包括平台和通信两个部分。中间件的示意图如图 2-8 所示：

问控制和代理特性的信息，并且该公用体拥有代理内部唯一的公用体名称。管理站将公用体和代理联系起来对代理进行管理控制。

SNMP 协议除了定义了上述的认证服务、访问控制策略和代理服务之外，还定义了 MIB 中的每一个被管对象的唯一的对象标示符；同时，SNMP 还定义了管理站和代理之间交换的消息和用户数据包的格式；另外，SNMP 还定义了一些操作，这些操作包括管理站向代理发出的 get-request、get-next-request 和 set-request 三种操作，代理使用的 get-response 对上述三种操作的应答操作，以及代理使用 trap 向管理站报告异常事件的操作等。

SNMP 需要利用传输层的服务来传递 SNMP 消息，但是 SNMP 对传输层服务的可靠性，以及传输层服务是面向连接的还是无连接的都没有提出要求。在实际中 SNMP 的实现几乎都是建立在无连接的 UDP 协议的基础上的。因为 UDP 协议不能保证传输报文的可靠性，那么 SNMP 消息就有可能丢失，另外 SNMP 本身并不能保证传输报文的可靠性，因此，管理信息可能会丢失。

- 功能模型

SNMP 的功能模型包括 2.1.2 节所提到的 ISO 规定网络管理的五大管理功能：故障、配置、计费、性能和安全管理。

2.2 关键技术

2.2.1 中间件的概述

2.2.1.1 中间件的概念

中间件的定义比较多，大多是从一个特殊的角度反映了中间件的一个或几个特性。比较多的人认为中间件，就是位于操作系统和应用软件之间的一个软件层，它向各种应用软件提供服务，使不同的应用进程能在屏蔽掉平台差异的情况下，通过网络互相通信。另外，存在一个更加宽泛的定义，认为所谓中间件从本质上讲，就是一个连接应用程序并允许它们之间互相交换数据的软件层。通常，在实际使用中，把一组中间件集成在一起构成一个平台（包括开发平台和运行平台），其中必须要有一个通信中间件完成中间件之间的通信。从这个意义上讲，中间件应该包括平台和通信两个部分。中间件的示意图如图 2-8 所示：

问控制和代理特性的信息，并且该公用体拥有代理内部唯一的公用体名称。管理站将公用体和代理联系起来对代理进行管理控制。

SNMP 协议除了定义了上述的认证服务、访问控制策略和代理服务之外，还定义了 MIB 中的每一个被管对象的唯一的对象标示符；同时，SNMP 还定义了管理站和代理之间交换的消息和用户数据包的格式；另外，SNMP 还定义了一些操作，这些操作包括管理站向代理发出的 get-request、get-next-request 和 set-request 三种操作，代理使用的 get-response 对上述三种操作的应答操作，以及代理使用 trap 向管理站报告异常事件的操作等。

SNMP 需要利用传输层的服务来传递 SNMP 消息，但是 SNMP 对传输层服务的可靠性，以及传输层服务是面向连接的还是无连接的都没有提出要求。在实际中 SNMP 的实现几乎都是建立在无连接的 UDP 协议的基础上的。因为 UDP 协议不能保证传输报文的可靠性，那么 SNMP 消息就有可能丢失，另外 SNMP 本身并不能保证传输报文的可靠性，因此，管理信息可能会丢失。

- 功能模型

SNMP 的功能模型包括 2.1.2 节所提到的 ISO 规定网络管理的五大管理功能：故障、配置、计费、性能和安全管理。

2.2 关键技术

2.2.1 中间件的概述

2.2.1.1 中间件的概念

中间件的定义比较多，大多是从一个特殊的角度反映了中间件的一个或几个特性。比较多的人认为中间件，就是位于操作系统和应用软件之间的一个软件层，它向各种应用软件提供服务，使不同的应用进程能在屏蔽掉平台差异的情况下，通过网络互相通信。另外，存在一个更加宽泛的定义，认为所谓中间件从本质上讲，就是一个连接应用程序并允许它们之间互相交换数据的软件层。通常，在实际使用中，把一组中间件集成在一起构成一个平台（包括开发平台和运行平台），其中必须要有一个通信中间件完成中间件之间的通信。从这个意义上讲，中间件应该包括平台和通信两个部分。中间件的示意图如图 2-8 所示：

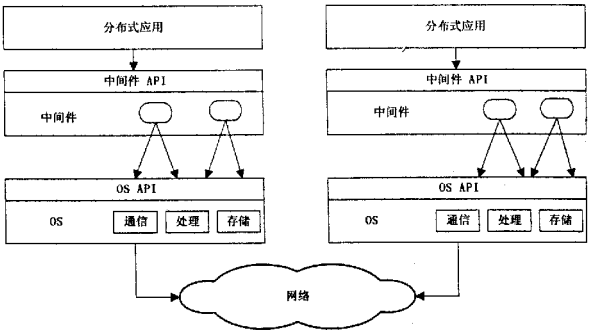


图 2-8 中间件示意图

另外还有人指出所谓中间件，或“粘结”，是网络层和应用层之间的一层软件。该软件提供了诸如：身份识别服务、鉴权授权服务、目录服务和安全服务等一系列服务。由于当今因特网应用通常都提供自己特定的上述服务，这往往容易导致标准的不统一和竞争。中间件通过推进了标准性和可互操作性，就使网络应用更加简单和有效。

2.2.1.2 中间件的作用

中间件的定义反映了中间件的桥梁特性，同时中间件还有下面一些重要的作用：

首先，中间件极大的简化了设计和开发的过程。协同计算要求各个应用之间要进行数据通信。采用中间件可以使它们之间的接口得到很大程度上的减少，中间件将原来的为 m 个应用开发 n 种功能时需要做 m×n 个接口简化为：为 m 个应用做 m 个同中间件的接口，另外将 n 个功能做 n 个同中间件的接口，也就是说总共只需要做 m + n 个接口。上面的化减在 m 和 n 都比较小的时候看不出很大的优势，但是当 m、n 很大时，采用中间件将极大地减化设计，节约软件开发的时间和成本。

其次，中间件可以从一个应用中获得数据并保存直到该数据没有使用价值为止。这就使数据的生存时期得到了保证，并且由于中间件对有用数据的专门处理可以使该数据发挥最大的功效。

最重要的一点是，中间件帮助不同的软件开发人员集中精力做好有针对性的软件开发，而不需要让他们更多的考虑到操作系统平台和底层通信设施对应用的影响。

总之，中间件可以使用户透明地应用分布式的计算机、网络、数据等资源；发展了更有效的协同和交互性的技术，例如：网格（Grid）计算等推动了教育和

科研的发展；同时，中间件还发展了一套可以扩大用户使用网络范围的体系结构。

2.2.1.3 中间件的分类

中间件的产品种类很多，分类的方式也各不相同，这里根据中间件采用的技术不同，大致划分为以下几种：

面向消息中间件（MOM，Message Oriented Middleware）：面向消息的中间件能够在客户和服务器之间提供同步和异步的连接，并且在任何时刻都可以将消息进行传送或者存储转发。MOM 的这两点都是建立在消息队列（message queue）这一关键技术的基础上的。另外消息中间件不会占用大量的网络带宽，可以跟踪事务，并且通过将事务存储到磁盘上实现网络故障时系统的恢复。消息中间件适用于需要在多个进程之间进行可靠的数据传送的分布式环境。

事务处理中间件（TPM，Transaction Processing Monitor）：与其他的中间件不同，事务处理中间件更加注重分布式事务的处理。这种中间件广泛地应用于数据管理、银行转账、定点分发等大型、分布式应用中，这些应用对物理上分散的节点之间的数据一致性有着很高的要求。TPM 是针对复杂环境下分布式应用的速度和可靠性要求而实现的。它对外提供了事务处理的 API，程序开发人员可以使用这个程序接口编写高速、可靠、基于事务处理的分布式应用程序。

面向对象的中间件（OOM，Object Oriented Middleware）：随着面向对象技术的发展，出现了面向对象的中间件。对象请求代理（ORB，Object Request Broker）就是其中的一个典型例子，它可以看作具有与位置、协议和平台无关的中间件的特性。从管理和封装的模式上看，ORB 和 RPC 有类似之处，不过 ORB 可以包含比 RPC 和消息中间件更复杂的信息，并且可以适用于非结构化的或者非关系型的数据。目前有两种对象请求代理的标准，分别是 CORBA 和 DCOM。

2.2.2 典型的面向对象的中间件模型

2.2.2.1 CORBA 模型

通用对象请求代理体系结构（CORBA）是对象管理组 OMG（Object Management Group）提出的一个面向对象中间件规范。CORBA 具体化了中间件模型并将其体现为：使用 IDL（接口定义语言）进行对象建模；采用 ORB 进行客户和服务器对象间的通信；采用 IIOP/GIOP（Internet Inter-ORB Protocol/ Generic Inter-ORB Protocol）进行不同 ORB 之间的通信。

ORB 是 CORBA 的核心，即对象请求代理，它能实现 Client 请求与目标对象实现之间的透明通信。通过对接请求代理，Client 方可以透明地调用服务方对象的方法，而无论服务方是在同一机器上还是跨网络，ORB 接收调用，并查找可

科研的发展；同时，中间件还发展了一套可以扩大用户使用网络范围的体系结构。

2.2.1.3 中间件的分类

中间件的产品种类很多，分类的方式也各不相同，这里根据中间件采用的技术不同，大致划分为以下几种：

面向消息中间件（MOM，Message Oriented Middleware）：面向消息的中间件能够在客户和服务器之间提供同步和异步的连接，并且在任何时刻都可以将消息进行传送或者存储转发。MOM 的这两点都是建立在消息队列（message queue）这一关键技术的基础上的。另外消息中间件不会占用大量的网络带宽，可以跟踪事务，并且通过将事务存储到磁盘上实现网络故障时系统的恢复。消息中间件适用于需要在多个进程之间进行可靠的数据传送的分布式环境。

事务处理中间件（TPM，Transaction Processing Monitor）：与其他的中间件不同，事务处理中间件更加注重分布式事务的处理。这种中间件广泛地应用于数据管理、银行转账、定点分发等大型、分布式应用中，这些应用对物理上分散的节点之间的数据一致性有着很高的要求。TPM 是针对复杂环境下分布式应用的速度和可靠性要求而实现的。它对外提供了事务处理的 API，程序开发人员可以使用这个程序接口编写高速、可靠、基于事务处理的分布式应用程序。

面向对象的中间件（OOM，Object Oriented Middleware）：随着面向对象技术的发展，出现了面向对象的中间件。对象请求代理（ORB，Object Request Broker）就是其中的一个典型例子，它可以看作具有与位置、协议和平台无关的中间件的特性。从管理和封装的模式上看，ORB 和 RPC 有类似之处，不过 ORB 可以包含比 RPC 和消息中间件更复杂的信息，并且可以适用于非结构化的或者非关系型的数据。目前有两种对象请求代理的标准，分别是 CORBA 和 DCOM。

2.2.2 典型的面向对象的中间件模型

2.2.2.1 CORBA 模型

通用对象请求代理体系结构（CORBA）是对象管理组 OMG（Object Management Group）提出的一个面向对象中间件规范。CORBA 具体化了中间件模型并将其体现为：使用 IDL（接口定义语言）进行对象建模；采用 ORB 进行客户和服务器对象间的通信；采用 IIOP/GIOP（Internet Inter-ORB Protocol/ Generic Inter-ORB Protocol）进行不同 ORB 之间的通信。

ORB 是 CORBA 的核心，即对象请求代理，它能实现 Client 请求与目标对象实现之间的透明通信。通过对接请求代理，Client 方可以透明地调用服务方对象的方法，而无论服务方是在同一机器上还是跨网络，ORB 接收调用，并查找可

以实现请求的对象，再传递参数并调用相应方法，最后将结果返回给 Client 方。Client 方可以完全不用考虑对象的位置，实现的编程语言，操作系统等与对象接口无关的信息。ORB 能实现分布环境中位于不同机器上的应用之间的互操作以及多对象系统之间的无缝连接。

COBRA 定义了通用 ORB 通信协议 GIOP，以及 GIOP 到 TCP / IP 协议的映射 IIOP（Internet Inter－ORB Protocol）。GIOP 是信息表示协议，描述了所传输信息的格式，而 IIOP 则描述了 CORBA 所支持的传输协议，即 GIOP 信息如何进行交换的。

下图 2－9 反映了 CORBA 的基本组成：

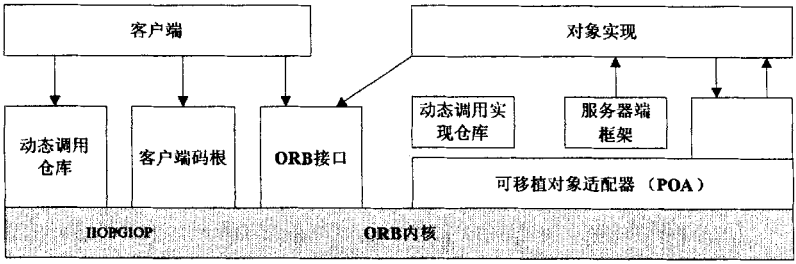


图 2-9　CORBA 体系结构及基本组成

在 CORBA 体系规范中定义了下面 16 种类型的服务（Service）：命名服务(Naming)、事件服务(Event)、通知服务(Notification)、生命周期服务(Life Cycle)、持久对象服务(Persistent Object)、事务服务(Transaction)、并发服务(Concurrency Control)、关系服务(Relationship)、具体化服务(Externalization)、查询服务(Query)、许可服务(Licensing)、属性服务(Property)、时间服务(Time)、安全服务(Security)、交易对象服务(Trading Object)、对象集合服务(Object Collection)。CORBA 服务与对象请求代理之间的关系如图 2-10 所示

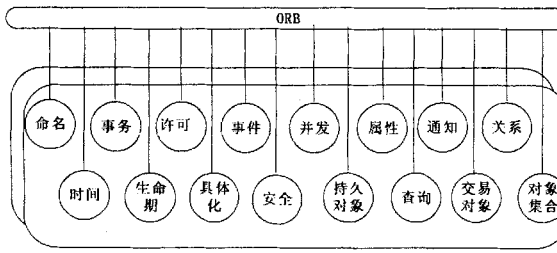


图 2-10　CORBA 服务及其与 ORB 的关系

2.2.2.2 COM/DCOM 模型

COM 是 Microsoft 提出的组件对象模型技术，而 DCOM 是 COM 的扩展，即分布式 COM。下图 2－11 反映了 COM/DCOM 的基本组成：

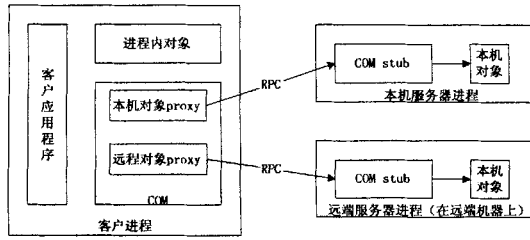


图 2-11　COM/DCOM 的基本组成

COM 可以以动态连接库和可执行程序的形式存在，DCOM 以可执行程序的形式存在。另外，组件之间的联系可以是在客户程序不知道 COM 的存在形式的情况下使用统一的接口调用方式来进行的，也就是说，COM 对象的位置对客户程序是透明的。

2.2.2.3 Java/RMI 模型

按照 Sun 和 JavaSoft 对 Java 的界定，Java 是一个应用程序开发平台，提供可移植性、可解释性、高性能和面向对象的编程语言及运行环境，而且 Java 也是一种分布式计算平台。Java 计算的本质就是利用分布在网络中的各类对象共同完成相应的任务。RMI 是分布在网络中的各类 Java 对象之间进行方法调用的 ORB 机制。但是 RMI 没有解决如何管理和访问异地其他大量非 Java 对象的问题，并且 RMI 没有提供分布对象事务管理等服务。而是由相关的应用服务器，如 WebLogic 提供这些服务。

2.2.3 本系统所使用的中间件

勿庸置疑，为了满足分布式电力监控管理系统的复杂功能和性能的要求，我们在实现网络管理时肯定使用分布式对象模型，即采用面向对象中间件。而在这三种主要的面向对象中间件模型中，通过考虑电力监控管理系统的应用环境、开发环境、集成需求和企业应用要求等因素，我们选取了 CORBA。

- 电力监控管理系统的实时和历史数据要能够进入 ERP、CRM 等企业信息管理系统，就要求所用中间件可以跨平台，而 DCOM/COM+只适用于 Windows 平台，所以不符合要求。
- 考虑到系统的开放性和扩展性，所用中间件必须可以使用各种编程语言实现

2.2.2.2 COM/DCOM 模型

COM 是 Microsoft 提出的组件对象模型技术，而 DCOM 是 COM 的扩展，即分布式 COM。下图 2－11 反映了 COM/DCOM 的基本组成：

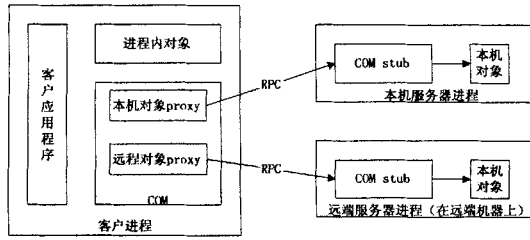


图 2-11　COM/DCOM 的基本组成

COM 可以以动态连接库和可执行程序的形式存在，DCOM 以可执行程序的形式存在。另外，组件之间的联系可以是在客户程序不知道 COM 的存在形式的情况下使用统一的接口调用方式来进行的，也就是说，COM 对象的位置对客户程序是透明的。

2.2.2.3 Java/RMI 模型

按照 Sun 和 JavaSoft 对 Java 的界定，Java 是一个应用程序开发平台，提供可移植性、可解释性、高性能和面向对象的编程语言及运行环境，而且 Java 也是一种分布式计算平台。Java 计算的本质就是利用分布在网络中的各类对象共同完成相应的任务。RMI 是分布在网络中的各类 Java 对象之间进行方法调用的 ORB 机制。但是 RMI 没有解决如何管理和访问异地其他大量非 Java 对象的问题，并且 RMI 没有提供分布对象事务管理等服务。而是由相关的应用服务器，如 WebLogic 提供这些服务。

2.2.3 本系统所使用的中间件

勿庸置疑，为了满足分布式电力监控管理系统的复杂功能和性能的要求，我们在实现网络管理时肯定使用分布式对象模型，即采用面向对象中间件。而在这三种主要的面向对象中间件模型中，通过考虑电力监控管理系统的应用环境、开发环境、集成需求和企业应用要求等因素，我们选取了 CORBA。

- 电力监控管理系统的实时和历史数据要能够进入 ERP、CRM 等企业信息管理系统，就要求所用中间件可以跨平台，而 DCOM/COM+只适用于 Windows 平台，所以不符合要求。
- 考虑到系统的开放性和扩展性，所用中间件必须可以使用各种编程语言实现

的分布式对象，也就是说不但要可以使用 C++编写的分布式对象，而且要可以使用 Java 编写的分布式对象。Java Bean 只能支持 Java 编写的分布式对象，而 CORBA 对象却通过 IDL 文件可以映射到 C++、Java 或其他编程语言，从而支持多种编程语言编写的分布式对象。

- CORBA 是由鼎鼎大名的 OMG 提出的一个中间件标准，其功能是非常全面的；特别是比起其他中间件更有优势的是 CORBA 制定了许多种服务，而这些服务为分布式系统的开发提供非常大的便利和帮助。
- 另外，我们的整个电力监控管理系统是用 Borland 公司的 C++ Builder 开发。在这个开发平台下，Borland 提供的 Visibroker 作为一种易用的、可伸缩的和可迁移的对象请求代理系统，符合 CORBA 规范，使用较为简单，并且完全满足当今复杂的、异构的应用环境的诸多挑战，也成为我们首选的分布式开发平台。

综合以上几点，我们选用了 CORBA。

2.3 CORBA 模型在网络管理上的应用

CORBA 的提出并不是针对网管领域的，但 CORBA 体系结构却非常合适应用于网络管理。采用基于 CORBA 技术的方法来实现网络管理，管理方与被管理方都可作为 CORBA 分布对象来实现，被管理方所维持的管理信息模型可以用代理的 IDL 接口描述来体现，由 ORB 提供的底层模块支撑管理方与被管理方之间的通信，这对于管理方与被管理方来说是透明的。网管系统运行时，管理方通过调用被管理方提供的服务来完成管理数据的采集，并在数据分析的基础上完成管理功能。

运用 CORBA 技术能够实现标准的网络管理系统。由于 CORBA 是一种分布对象技术，基于 CORBA 的网络管理系统能够克服传统网管技术的不足，使网管的分布性、可靠性和易开发性方面达到一个新的高度。

- CORBA 规范下管理进程和代理进程交互的方式是事件驱动，能克服 SNMP 下采用的轮询或陷阱的方式的缺点。CORBA 事件服务提供了一个异步事件通知的框架。事件服务由生产者、消费者和事件通道组成。其中生产者主要是用来产生数据，而消费者负责处理数据。事件通信有两种模型：PUSH 方式和 PULL 方式。在 PUSH 方式中，生产者主动提供事件给消费者，调用消费者的 PUSH 操作：在 PULL 方式中，由消费者向生产者获取事件，调用生产者的 PULL 操作。CORBA 事件服务的这种机制，可以使网络管理更加方便。
- CORBA 的对象可以保证应用程序跨多个网络管理平台，同时提供一系列可

的分布式对象，也就是说不但要可以使用 C++编写的分布式对象，而且要可以使用 Java 编写的分布式对象。Java Bean 只能支持 Java 编写的分布式对象，而 CORBA 对象却通过 IDL 文件可以映射到 C++、Java 或其他编程语言，从而支持多种编程语言编写的分布式对象。

- CORBA 是由鼎鼎大名的 OMG 提出的一个中间件标准，其功能是非常全面的；特别是比起其他中间件更有优势的是 CORBA 制定了许多种服务，而这些服务为分布式系统的开发提供非常大的便利和帮助。
- 另外，我们的整个电力监控管理系统是用 Borland 公司的 C++ Builder 开发。在这个开发平台下，Borland 提供的 Visibroker 作为一种易用的、可伸缩的和可迁移的对象请求代理系统，符合 CORBA 规范，使用较为简单，并且完全满足当今复杂的、异构的应用环境的诸多挑战，也成为我们首选的分布式开发平台。

综合以上几点，我们选用了 CORBA。

2.3 CORBA 模型在网络管理上的应用

CORBA 的提出并不是针对网管领域的，但 CORBA 体系结构却非常合适应用于网络管理。采用基于 CORBA 技术的方法来实现网络管理，管理方与被管理方都可作为 CORBA 分布对象来实现，被管理方所维持的管理信息模型可以用代理的 IDL 接口描述来体现，由 ORB 提供的底层模块支撑管理方与被管理方之间的通信，这对于管理方与被管理方来说是透明的。网管系统运行时，管理方通过调用被管理方提供的服务来完成管理数据的采集，并在数据分析的基础上完成管理功能。

运用 CORBA 技术能够实现标准的网络管理系统。由于 CORBA 是一种分布对象技术，基于 CORBA 的网络管理系统能够克服传统网管技术的不足，使网管的分布性、可靠性和易开发性方面达到一个新的高度。

- CORBA 规范下管理进程和代理进程交互的方式是事件驱动，能克服 SNMP 下采用的轮询或陷阱的方式的缺点。CORBA 事件服务提供了一个异步事件通知的框架。事件服务由生产者、消费者和事件通道组成。其中生产者主要是用来产生数据，而消费者负责处理数据。事件通信有两种模型：PUSH 方式和 PULL 方式。在 PUSH 方式中，生产者主动提供事件给消费者，调用消费者的 PUSH 操作：在 PULL 方式中，由消费者向生产者获取事件，调用生产者的 PULL 操作。CORBA 事件服务的这种机制，可以使网络管理更加方便。
- CORBA 的对象可以保证应用程序跨多个网络管理平台，同时提供一系列可

以重复使用的有关被管理对象的类，而且可以减少实现 SNMP、CMIP 程序所需的专门知识和技巧。

- 基于 CORBA 的网管系统是独立于协议和编程语言的。开发基于 CORBA 网管系统时，开发者需要关注的是计算对象的接口定义及其实现，而不需关心管理者和代理之间如何通信。即从开发者看来：这样的网管系统开发基本上不涉及通信，而传统网管开发中通信往往是最复杂、最不可靠和最费事的环节。有了 ORB 提供的可靠的标准的通信支持，网管系统本身的可靠性也得到了很大的提高。
- CORBA 规范本身就是面向对象的，IDL 定义的接口、属性和多继承性的对象非常接近于 C++描述。OMG 组织已经采用统一建模语言 UML 作为建模语言，CORBA 的应用实际上是面向对象技术应用的最好体现。因此，基于 CORBA 的网管系统开发适合采用面向对象技术，而且加上 CORBA 本身提供的大量的开发工具支持，基于 CORBA 的网管系统的组织和开发将比传统网管系统更具有规范性和可重用性，使得开发过程也更容易一些。

CORBA 具有的适于作为网管系统体系的特点，人们已经对 CORBA 在网管中的应用做了比较深入的研究。目前，SNMP 在网络设备级管理中仍将具有很大的市场，而 CMIP 在电信网络管理中也日趋流行，因此，未来一段时期内，必然是 CORBA、SNMP 和 CMIP 共存的局面。因而研究 CORBA 与 SNMP 和 CMIP 的互操作成为热点。

CORBA 主域管理者判断请求是发给基于哪一种网管接口的被管设备(如 CORBA、SNMP、CMIP)，如果是发给基于 SNMP 或是 CMIP 网管接口的被管设备，则将请求通过 IIOP 协议转发给相应的 CORBA 子域管理者，由 CORBA 子域管理者再发送给 CORBA/ SNMP 或 CORBA/ CMIP 网关，网关激活相应的目标对象，把 CORBA 请求转换成 SNMP 或 CMIP 请求，最后通过 SNMP 或 CMIP 协议把请求发送给相应的被管设备 Agent。如果请求是发给基于 CORBA 的网管接口的被管设备，则请求直接通过 IIOP 协议经由相应的 CORBA 子域管理者，发送到相应的被管设备 Agent。被管设备 Agent 处理请求，将响应以上述相反的顺序返回给管理用户。CORBA 网关起着收集网络设备信息，转换协议操作和数据格式的作用。对整个网络管理的结构来说，CORBA 和底层网管之间的透明传输保证了应用开发者不需要知道被管远地设备的具体协议。

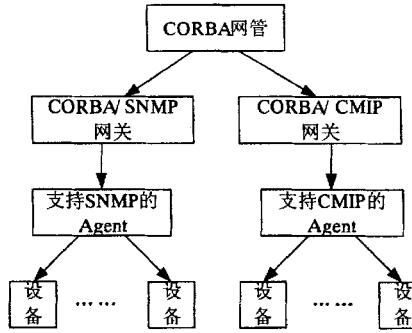


图 2－12　CORBA 与 SNMP 和 CMIP 的互操作

第三章 基于 Web 技术的分布式监控网络管理

3.1 基于 Web 的分布式网络管理技术

传统的网络管理界面是网络管理命令驱动的远程登陆屏幕，必须由专业网管工作人员操作。使用和维护网络管理系统需要专门培训的技术人员，随着网络规模增大，网管功能复杂化，使传统网管界面的友好程度愈来愈差了。为了减轻网管复杂性，降低网管费用，急需研究和开发一种跨平台的，方便使用的新的网管模式。基于 Web 的网络管理模式（WBM，web based management）可以实现这个目标。这种新的网络管理模式融合了 Web 功能和网络技术，他允许网络管理人员通过与 www 同样的形式去监测、管理网络系统，可以使用 Web 浏览器在网络任何节点上方便迅速的配置，控制及访问网络和它的各个部分，这种新的网络管理模式的魅力在于它是交叉平台，可以很好的解决很多由于多平台结构产生的互操作问题，这能提供比传统网管界面更直接，更易于使用的图形界面（浏览器操作和 Web 页面对 www 用户来讲是非常熟悉的），从而降低了对网络管理操作和维护人员的特殊要求。基于 Web 的网络管理模式是网络管理的一次革命，它将使用户管理网络的方法得以彻底改变，从而为实现“自己管理网络”和“网络管理自动化”迈出关键的一步。

3.1.1 基于 Web 的分布式网络管理的实现策略

WBM 有两种基本的实现策略：

（1）基于平台的 WBM 应用实现

网络平台是指一个包含网络管理进程软件和基于网络管理应用的主管系统，网络平台就是满足最低管理需要的主管系统，有了网络平台，管理人员就可以经由网管平台的管理者与被管系统中的代理通信。

这种方案将一个 Web 服务器与网管工作站相结合,如图 3－1 所示。

第三章 基于 Web 技术的分布式监控网络管理

3.1 基于 Web 的分布式网络管理技术

传统的网络管理界面是网络管理命令驱动的远程登陆屏幕，必须由专业网管工作人员操作。使用和维护网络管理系统需要专门培训的技术人员，随着网络规模增大，网管功能复杂化，使传统网管界面的友好程度愈来愈差了。为了减轻网管复杂性，降低网管费用，急需研究和开发一种跨平台的，方便使用的新的网管模式。基于 Web 的网络管理模式（WBM，web based management）可以实现这个目标。这种新的网络管理模式融合了 Web 功能和网络技术，他允许网络管理人员通过与 www 同样的形式去监测、管理网络系统，可以使用 Web 浏览器在网络任何节点上方便迅速的配置，控制及访问网络和它的各个部分，这种新的网络管理模式的魅力在于它是交叉平台，可以很好的解决很多由于多平台结构产生的互操作问题，这能提供比传统网管界面更直接，更易于使用的图形界面（浏览器操作和 Web 页面对 www 用户来讲是非常熟悉的），从而降低了对网络管理操作和维护人员的特殊要求。基于 Web 的网络管理模式是网络管理的一次革命，它将使用户管理网络的方法得以彻底改变，从而为实现“自己管理网络”和“网络管理自动化”迈出关键的一步。

3.1.1 基于 Web 的分布式网络管理的实现策略

WBM 有两种基本的实现策略：

（1）基于平台的 WBM 应用实现

网络平台是指一个包含网络管理进程软件和基于网络管理应用的主管系统，网络平台就是满足最低管理需要的主管系统，有了网络平台，管理人员就可以经由网管平台的管理者与被管系统中的代理通信。

这种方案将一个 Web 服务器与网管工作站相结合,如图 3－1 所示。

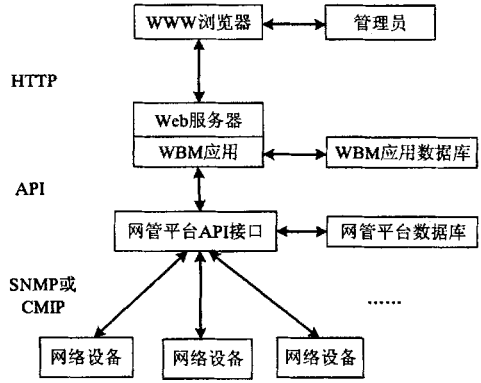


图 3－1　基于网管平台的 WBM 实现

其中，网管平台通过 SNMP（简单网络管理协议）或 CMIP（公用管理信息协议）与被管设备通信，收集、过滤、处理各种网管信息，维护网管平台数据库。WBM 应用通过网管平台提供的 API 接口获取网络信息，维护 WBM 专用的数据库。网管人员对网络的监视、调整和控制通过浏览器向 Web 服务器发送 HTTP（超文本传输协议）请求，Web 服务器通过 CGI（公用网关接口）调用相应的 WBM 应用，WBM 应用把网管信息转换为 HTML 形式返回给 Web 服务器，由 Web 服务器响应浏览器的 HTTP 请求。

这种方式融合了基于平台的网管系统和 Web 技术的优点,可以充分利用网管平台成熟的高效率的算法,保留对现有网管标准和传统设备的支持，保护用户对原有设备的投资。由于本系统是在现有的 YSZ2002 电力监控系统原有的网管系统上改造的，所以我们使用的 WBM 的实现是基于平台的。

（2）嵌入的 WBM 应用实现

这种方案将 Web 能力真正地嵌入到网络设备中，如图 3－2 所示。

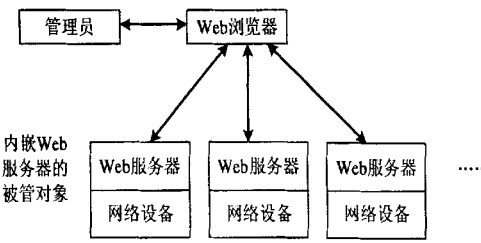


图 3－2　嵌入式 WBM 实现

其中，每个设备就是一个 Web 服务器系统，拥有全局唯一的 IP 地址，管理人员通过浏览器直接访问该设备并实施调整和控制。

这种方案给独立设备带来了完全的图形化管理，保障了简单易用的网管界面。网管系统完全采用 Web 技术，如通信协议采用基于 HTTP 的协议，网管信息库用基于 HTML 的语言描述，网络的拓扑算法采用高效的 Web 搜索、查询、索引技术，网络管理层次和域的组织采用灵活的虚拟形式，不再拘泥于地理位置等条件的约束。

嵌入式 WBM 是真正的 Web 网络管理，网络设备、系统、应用增加 Web 能力并不需要太多的资源开销。因此，越来越多的网络设备开始支持 Web 管理。

3.1.2 基于 Web 的网络管理的国际标准

3.1.2.1 WBEM

基于 Web 的企业管理标准 WBEM (Web-Based Enterprise Management),WBEM 建议采用面向对象的方式，抽象出各种管理数据对象，通过多种协议(SNMP、CMIP 等)从多种网络资源(如设备、系统、应用程序等)中收集数据并进行管理，WBEM 是当前网管标准协议的兼容和扩展，它提出了一个新的架构、协议、管理轮廓和一个对象管理器。WBEM 被认为是“兼容和扩展”了当前的标准，如 SNMP、CMIP，而并不是取而代之。

3.1.2.2 JMAPI

SUN 公司提出了 JMAPI(Java 管理应用程序接口)， JMAPI 为 Java 计算环境的扩展，为异构网络系统和服务管理的无缝管理提供了一组丰富的扩展对象及方法，从而使得使用一种语言来开发异构环境的网络管理软件成为可能,并可缩短开发系统的周期。

JMX（Java Management Extensions）通过以下各种形式提供给开发人员基于 JAVA 技术的管理应用程序的能力：建立只能的代理和管理者、跨平台运行的能力、实现分布式网管中间件和集成现有的网络管理系统。JMX 的体系结构分为三个层次：

- 设备级别
- 代理者级别
- 管理者级别

3.1.2.3 XOJIDM

XOJIDM 是由 X/Open 和 Network Management Forum 制定的用于在 CORBA、CMIP 和 SNMP 这三个网络框架之间互操作的标准。它的主要内容是怎样把

这种方案给独立设备带来了完全的图形化管理，保障了简单易用的网管界面。网管系统完全采用 Web 技术，如通信协议采用基于 HTTP 的协议，网管信息库用基于 HTML 的语言描述，网络的拓扑算法采用高效的 Web 搜索、查询、索引技术，网络管理层次和域的组织采用灵活的虚拟形式，不再拘泥于地理位置等条件的约束。

嵌入式 WBM 是真正的 Web 网络管理，网络设备、系统、应用增加 Web 能力并不需要太多的资源开销。因此，越来越多的网络设备开始支持 Web 管理。

3.1.2 基于 Web 的网络管理的国际标准

3.1.2.1 WBEM

基于 Web 的企业管理标准 WBEM (Web-Based Enterprise Management),WBEM 建议采用面向对象的方式，抽象出各种管理数据对象，通过多种协议(SNMP、CMIP 等)从多种网络资源(如设备、系统、应用程序等)中收集数据并进行管理，WBEM 是当前网管标准协议的兼容和扩展，它提出了一个新的架构、协议、管理轮廓和一个对象管理器。WBEM 被认为是“兼容和扩展”了当前的标准，如 SNMP、CMIP，而并不是取而代之。

3.1.2.2 JMAPI

SUN 公司提出了 JMAPI(Java 管理应用程序接口)， JMAPI 为 Java 计算环境的扩展，为异构网络系统和服务管理的无缝管理提供了一组丰富的扩展对象及方法，从而使得使用一种语言来开发异构环境的网络管理软件成为可能,并可缩短开发系统的周期。

JMX（Java Management Extensions）通过以下各种形式提供给开发人员基于 JAVA 技术的管理应用程序的能力：建立只能的代理和管理者、跨平台运行的能力、实现分布式网管中间件和集成现有的网络管理系统。JMX 的体系结构分为三个层次：

- 设备级别
- 代理者级别
- 管理者级别

3.1.2.3 XOJIDM

XOJIDM 是由 X/Open 和 Network Management Forum 制定的用于在 CORBA、CMIP 和 SNMP 这三个网络框架之间互操作的标准。它的主要内容是怎样把

SNMP 或 CMIP 的管理信息库 MIB 的规范映射成 OMG 的 IDL 规范，以达成 CORBA 与 SNMP 互操作的目的。

WBEM 和 JMAPI 分别制定了各自的管理体系结构，这两者从总体上都分为三部分：浏览器用户界面，管理运行模块或管理服务器，被管元素。概括起来讲，WBM 系统的一般结构是基于浏览器(Browse)—服务器(Server)结构，如图 3—3 所示。图 3—3 每个被管理的网络设备都含有一个代理(Agent)，它维持一个局部的信息管理库（MIB），并且和驻留在服务器的管理应用通过网管协议 SNMP 和 CMIP 进行通信，管理应用和管理代理之间的交互允许对管理信息库的检索和修改，从而实现网管功能。

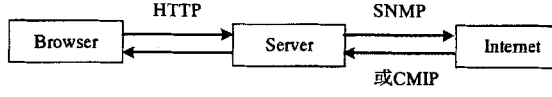


图 3—3 WBEM 和 JMAPI 标准的 WBM 实现

在本系统中我们没有单独使用哪个具体的网络管理标准，而是综合了各个网络管理标准，形成了具有本系统特色的网络管理体系结构，在本章的 3.4 节有详细的论述。

3.1.3 基于 Web 的网络管理的特点和优越性

随着 Web 的流行，带来的是网络管理和 Web 技术的结合。基于 Web 的网络管理其主要思想就是：使网络管理者可以通过浏览器来进行网络管理。基于 Web 的网络管理就是利用 World Wide Web 工具与技术在互联网上对网络设备实现管理。使用 TCP/IP 协议及 HTTP 协议，在标准的浏览器上监控由相关 HTTP 服务器程序所支持的设备。利用支持工具、编程语言和协议，如 HTML（超文本标记语言）、CGI（公共网管接口）、Java，C++和 SNMP 等，可以产生和传送管理信息，并把结果以静态、动态或表格的形式在网页上表示出来，其特点如下：

- 简单的用户界面：对于使用者来说，浏览器的界面是非常直观和简单的。网络管理者通过使用基于 Web 的网络管理系统，即使在没有培训的情况下，也可以很快掌握。
- 可移植性：浏览器几乎可以在各种硬件设备上运行。
- 费用：浏览器可以在低价位的 PC 机上或者在掌上设备上运行。
- 远程管理：由于使用浏览器使得远程管理变的简单了，可以在网络的每个接入点上进行管理。无论是在何处，网络管理者看到的界面都是一致的。
- 与其它应用的集成：许多应用系统开始与 Web 结合。例如可以通过计算机网络资源的使用量来管理一个公司的工作流程，以及记录设备的故障，这些都可以通过 Web 来完成。如果把网络管理系统也集成到 Web 上，那么就

SNMP 或 CMIP 的管理信息库 MIB 的规范映射成 OMG 的 IDL 规范，以达成 CORBA 与 SNMP 互操作的目的。

WBEM 和 JMAPI 分别制定了各自的管理体系结构，这两者从总体上都分为三部分：浏览器用户界面，管理运行模块或管理服务器，被管元素。概括起来讲，WBM 系统的一般结构是基于浏览器(Browse)—服务器(Server)结构，如图 3—3 所示。图 3—3 每个被管理的网络设备都含有一个代理(Agent)，它维持一个局部的信息管理库（MIB），并且和驻留在服务器的管理应用通过网管协议 SNMP 和 CMIP 进行通信，管理应用和管理代理之间的交互允许对管理信息库的检索和修改，从而实现网管功能。

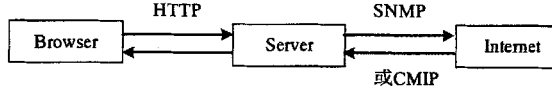


图 3—3 WBEM 和 JMAPI 标准的 WBM 实现

在本系统中我们没有单独使用哪个具体的网络管理标准，而是综合了各个网络管理标准，形成了具有本系统特色的网络管理体系结构，在本章的 3.4 节有详细的论述。

3.1.3 基于 Web 的网络管理的特点和优越性

随着 Web 的流行，带来的是网络管理和 Web 技术的结合。基于 Web 的网络管理其主要思想就是：使网络管理者可以通过浏览器来进行网络管理。基于 Web 的网络管理就是利用 World Wide Web 工具与技术在互联网上对网络设备实现管理。使用 TCP/IP 协议及 HTTP 协议，在标准的浏览器上监控由相关 HTTP 服务器程序所支持的设备。利用支持工具、编程语言和协议，如 HTML（超文本标记语言）、CGI（公共网管接口）、Java，C++和 SNMP 等，可以产生和传送管理信息，并把结果以静态、动态或表格的形式在网页上表示出来，其特点如下：

- 简单的用户界面：对于使用者来说，浏览器的界面是非常直观和简单的。网络管理者通过使用基于 Web 的网络管理系统，即使在没有培训的情况下，也可以很快掌握。
- 可移植性：浏览器几乎可以在各种硬件设备上运行。
- 费用：浏览器可以在低价位的 PC 机上或者在掌上设备上运行。
- 远程管理：由于使用浏览器使得远程管理变的简单了，可以在网络的每个接入点上进行管理。无论是在何处，网络管理者看到的界面都是一致的。
- 与其它应用的集成：许多应用系统开始与 Web 结合。例如可以通过计算机网络资源的使用量来管理一个公司的工作流程，以及记录设备的故障，这些都可以通过 Web 来完成。如果把网络管理系统也集成到 Web 上，那么就

可以方便地与其他应用集成。

- 易于推广：Web 的流行使得基于 Web 的网络管理系统也便于推广。

3.2 基于 Web 的分布式网络管理的体系结构

在网络和系统管理的实现中较有影响的模型是 SNMP 和 CMIP，这两个模型各有优点，同时又存在着许多不足之处。SNMP 在 Internet 上被广泛接受，它最主要的一个特点就是简单，但是在需要完成十分复杂的任务时，它就不能充分满足要求。许多通信厂商的网络结构是基于 CMIP 的，但是 CMIP 受到自身过于复杂以及标准化过程太慢的限制，至今仍未获得像 SNMP 那样广泛的支持。可以预见，这两种管理体系框架在很长时间内将会同时存在。

随着面向对象的分布式处理模型的出现，CORBA 作为第三种解决方案被提出。CORBA 提供了统一的资源命名、事件处理以及服务交换等机制。虽然它最初的提出是针对分布式对象计算，而并非针对网络管理任务的，但是在很多方面它都适合于管理本地以及很大范围的网络。与现有模型相比，CORBA 提供的功能比 SNMP 更强大，而且不像 CMIP 那么复杂。此外，CORBA 支持 C++，Java 等多种被广泛使用的编程语言，因此它已经迅速被大量的编程人员所接受。通过 CORBA，他们可以使自己的程序具有分布式的特点，而且不必在逻辑上有很大的变动。正因如此，现在普遍认为，CORBA 会在网络管理和系统管理中占有越来越重要的位置。将 WEB 与 CORBA 技术相融合将是网络管理发展的必然趋势。

3.2.1 基于 Web 的分布式网络管理的组织模型

图 3－4 所示的组织模型表示为四个层次：

（1）管理者在浏览器层通过 HTTP 协议下载 Applet。通过 JAVA 虚拟机可以运行下载的 JAVA Applet 程序，这样可以解决使用应用系统时跨平台的问题。

（2）网络管理中心除充当 Web 服务器外，还接收浏览器层发出的 CORBA 请求，并将该请求传递给能回答它的管理代理（子管理者）处。网络管理中心还管理一主数据库，该数据库用于存储管理代理的相关信息。

（3）每一个管理代理管理一个子网，它可以是子网的一部分，运行在子网的某一台计算机上，也可以独立于被管子网。管理代理配有一个本地的 MIB，该数据库用来存储被管子网的网管数据。

（4）设备层包含被管设备,每个子网中都包含有多个被管设备。

可以方便地与其他应用集成。

- 易于推广：Web 的流行使得基于 Web 的网络管理系统也便于推广。

3.2 基于 Web 的分布式网络管理的体系结构

在网络和系统管理的实现中较有影响的模型是 SNMP 和 CMIP，这两个模型各有优点，同时又存在着许多不足之处。SNMP 在 Internet 上被广泛接受，它最主要的一个特点就是简单，但是在需要完成十分复杂的任务时，它就不能充分满足要求。许多通信厂商的网络结构是基于 CMIP 的，但是 CMIP 受到自身过于复杂以及标准化过程太慢的限制，至今仍未获得像 SNMP 那样广泛的支持。可以预见，这两种管理体系框架在很长时间内将会同时存在。

随着面向对象的分布式处理模型的出现，CORBA 作为第三种解决方案被提出。CORBA 提供了统一的资源命名、事件处理以及服务交换等机制。虽然它最初的提出是针对分布式对象计算，而并非针对网络管理任务的，但是在很多方面它都适合于管理本地以及很大范围的网络。与现有模型相比，CORBA 提供的功能比 SNMP 更强大，而且不像 CMIP 那么复杂。此外，CORBA 支持 C++，Java 等多种被广泛使用的编程语言，因此它已经迅速被大量的编程人员所接受。通过 CORBA，他们可以使自己的程序具有分布式的特点，而且不必在逻辑上有很大的变动。正因如此，现在普遍认为，CORBA 会在网络管理和系统管理中占有越来越重要的位置。将 WEB 与 CORBA 技术相融合将是网络管理发展的必然趋势。

3.2.1 基于 Web 的分布式网络管理的组织模型

图 3－4 所示的组织模型表示为四个层次：

（1）管理者在浏览器层通过 HTTP 协议下载 Applet。通过 JAVA 虚拟机可以运行下载的 JAVA Applet 程序，这样可以解决使用应用系统时跨平台的问题。

（2）网络管理中心除充当 Web 服务器外，还接收浏览器层发出的 CORBA 请求，并将该请求传递给能回答它的管理代理（子管理者）处。网络管理中心还管理一主数据库，该数据库用于存储管理代理的相关信息。

（3）每一个管理代理管理一个子网，它可以是子网的一部分，运行在子网的某一台计算机上，也可以独立于被管子网。管理代理配有一个本地的 MIB，该数据库用来存储被管子网的网管数据。

（4）设备层包含被管设备,每个子网中都包含有多个被管设备。

可以方便地与其他应用集成。

- 易于推广：Web 的流行使得基于 Web 的网络管理系统也便于推广。

3.2 基于 Web 的分布式网络管理的体系结构

在网络和系统管理的实现中较有影响的模型是 SNMP 和 CMIP，这两个模型各有优点，同时又存在着许多不足之处。SNMP 在 Internet 上被广泛接受，它最主要的一个特点就是简单，但是在需要完成十分复杂的任务时，它就不能充分满足要求。许多通信厂商的网络结构是基于 CMIP 的，但是 CMIP 受到自身过于复杂以及标准化过程太慢的限制，至今仍未获得像 SNMP 那样广泛的支持。可以预见，这两种管理体系框架在很长时间内将会同时存在。

随着面向对象的分布式处理模型的出现，CORBA 作为第三种解决方案被提出。CORBA 提供了统一的资源命名、事件处理以及服务交换等机制。虽然它最初的提出是针对分布式对象计算，而并非针对网络管理任务的，但是在很多方面它都适合于管理本地以及很大范围的网络。与现有模型相比，CORBA 提供的功能比 SNMP 更强大，而且不像 CMIP 那么复杂。此外，CORBA 支持 C++，Java 等多种被广泛使用的编程语言，因此它已经迅速被大量的编程人员所接受。通过 CORBA，他们可以使自己的程序具有分布式的特点，而且不必在逻辑上有很大的变动。正因如此，现在普遍认为，CORBA 会在网络管理和系统管理中占有越来越重要的位置。将 WEB 与 CORBA 技术相融合将是网络管理发展的必然趋势。

3.2.1 基于 Web 的分布式网络管理的组织模型

图 3－4 所示的组织模型表示为四个层次：

（1）管理者在浏览器层通过 HTTP 协议下载 Applet。通过 JAVA 虚拟机可以运行下载的 JAVA Applet 程序，这样可以解决使用应用系统时跨平台的问题。

（2）网络管理中心除充当 Web 服务器外，还接收浏览器层发出的 CORBA 请求，并将该请求传递给能回答它的管理代理（子管理者）处。网络管理中心还管理一主数据库，该数据库用于存储管理代理的相关信息。

（3）每一个管理代理管理一个子网，它可以是子网的一部分，运行在子网的某一台计算机上，也可以独立于被管子网。管理代理配有一个本地的 MIB，该数据库用来存储被管子网的网管数据。

（4）设备层包含被管设备,每个子网中都包含有多个被管设备。

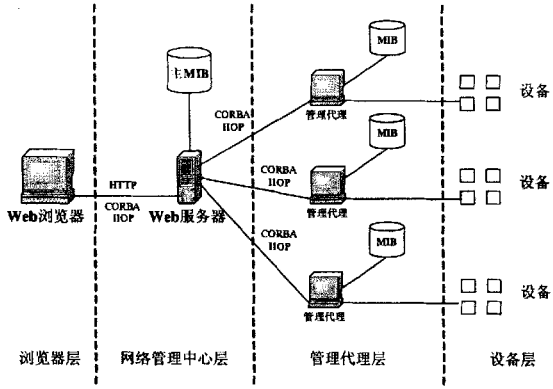


图 3－4 基于 CORBA 的网管模型组织模型

3.2.2 基于 Web 的分布式网络管理的通信机制

网络管理是通过客户与网络管理中心以及网络管理中心与管理代理（子管理者）的通信来实现的。其通信机制如图 3－5 所示：

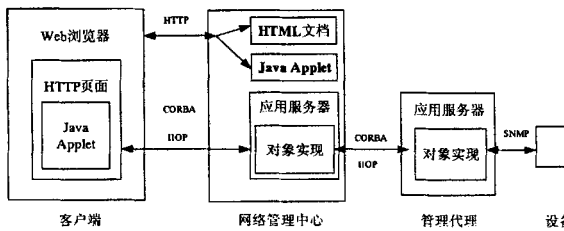


图 3－5 通信机制

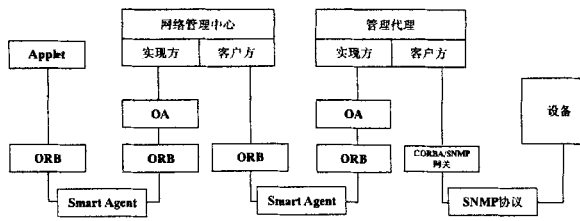


图 3－6 基于 Visibroker 平台的通信机制

（1） 浏览器层和网络管理层之间的 Web 页面交互通过 HTTP 协议实现，CORBA 请求和应答则通过 IIOP 协议实现。

（2） 网络管理中心和管理代理之间的交互是纯 CORBA 的，因而通过 IIOP

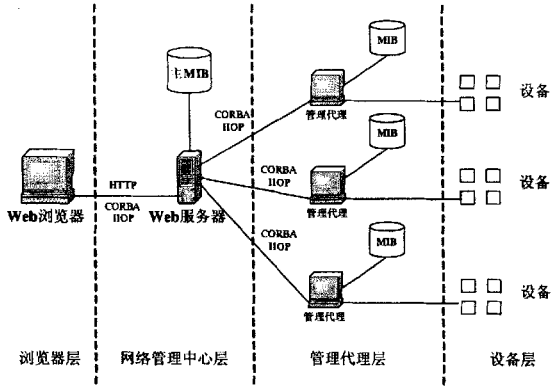


图 3－4 基于 CORBA 的网管模型组织模型

3.2.2 基于 Web 的分布式网络管理的通信机制

网络管理是通过客户与网络管理中心以及网络管理中心与管理代理（子管理者）的通信来实现的。其通信机制如图 3－5 所示：

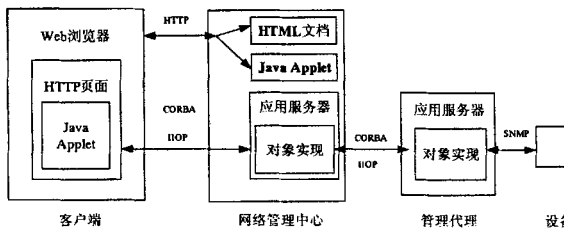


图 3－5 通信机制

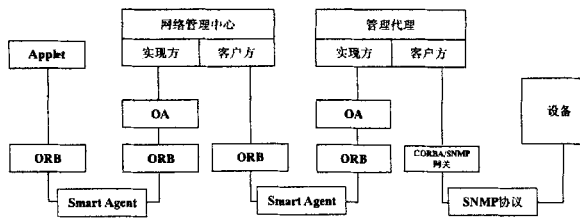


图 3－6 基于 Visibroker 平台的通信机制

（1） 浏览器层和网络管理层之间的 Web 页面交互通过 HTTP 协议实现，CORBA 请求和应答则通过 IIOP 协议实现。

（2） 网络管理中心和管理代理之间的交互是纯 CORBA 的，因而通过 IIOP

协议通信。

（3） 管理代理（子管理者）利用 SNMP 协议（或是 CMIP 协议）与设备进行通信。

本模型实现了两层 CORBA 通信。Java Applet 和网络管理中心、网络管理中心和管理代理以及管理代理之间都是客户方和实现方的关系。Applet 和网络管理中心之间的 IDL 接口定义了需实现的各项管理功能。网络管理中心和管理代理之间的 IDL 接口则将每个管理功能细化为若干操作(Health Function)。这些操作在实现时往往需要分解为若干个 SNMP 操作，并将其结果进行综合、分析，才得以实现。CORBA/SNMP 网关实现 CORBA 到 SNMP 协议的转换。子管理者通过调用 CORBA/SNMP 网关实现了 IDL 定义的所有接口，并发出 SNMP 请求，并轮询检测陷阱的发生。CORBA 管理者指网络管理中心，SNMP 管理者指管理代理，对设备的操作通过 CORBA/SNMP 网关转换为 SNMP 操作。

在 CORBA/SNMP 网关中，管理者就是 CORBA 的客户方，对被管理对象的描述以 IDL 的形式给出，按 SNMP 语法给出的传输结果被转换为 CORBA IDL 的形式。其中，管理代理与设备进行交互时使用的是 SNMP。因此，在 CORBA 管理者和 SNMP 代理之间的交互必须通过一个 CORBA/SNMP 网关，该网关负责对这些交互进行翻译。CORBA 管理者接收并处理 SNMP 的管理信息、Trap 和通知，它对 MIB 的访问是通过 IDL 完成的，完全忽略被管理系统中 SNMP 的存在。

上面提出的基于 Web 的网络管理体系结构是综合了 WBEM 和 JMAPI 两个国际标准后形成的。在这两个系统中，WBEM 使用的管理协议和管理对象模型都是新提出的，而且它主要是基于 OLE/DCOM 的，但是接受 DCOM 标准的平台并不是很多；JMAPI 是 SUN 公司提供的，受到了 Javasoft 等众多厂商的支持，它使用的管理协议和管理对象模型都是基于 Java 技术的，由于它也是基于 CORBA 的，因此可以跨越所有平台。

3.2.3 基于 Web 的分布式网络管理的功能模型

基于 Web 的网络管理模型的功能模型如图 3－7。

协议通信。

（3） 管理代理（子管理者）利用 SNMP 协议（或是 CMIP 协议）与设备进行通信。

本模型实现了两层 CORBA 通信。Java Applet 和网络管理中心、网络管理中心和管理代理以及管理代理之间都是客户方和实现方的关系。Applet 和网络管理中心之间的 IDL 接口定义了需实现的各项管理功能。网络管理中心和管理代理之间的 IDL 接口则将每个管理功能细化为若干操作(Health Function)。这些操作在实现时往往需要分解为若干个 SNMP 操作，并将其结果进行综合、分析，才得以实现。CORBA/SNMP 网关实现 CORBA 到 SNMP 协议的转换。子管理者通过调用 CORBA/SNMP 网关实现了 IDL 定义的所有接口，并发出 SNMP 请求，并轮询检测陷阱的发生。CORBA 管理者指网络管理中心，SNMP 管理者指管理代理，对设备的操作通过 CORBA/SNMP 网关转换为 SNMP 操作。

在 CORBA/SNMP 网关中，管理者就是 CORBA 的客户方，对被管理对象的描述以 IDL 的形式给出，按 SNMP 语法给出的传输结果被转换为 CORBA IDL 的形式。其中，管理代理与设备进行交互时使用的是 SNMP。因此，在 CORBA 管理者和 SNMP 代理之间的交互必须通过一个 CORBA/SNMP 网关，该网关负责对这些交互进行翻译。CORBA 管理者接收并处理 SNMP 的管理信息、Trap 和通知，它对 MIB 的访问是通过 IDL 完成的，完全忽略被管理系统中 SNMP 的存在。

上面提出的基于 Web 的网络管理体系结构是综合了 WBEM 和 JMAPI 两个国际标准后形成的。在这两个系统中，WBEM 使用的管理协议和管理对象模型都是新提出的，而且它主要是基于 OLE/DCOM 的，但是接受 DCOM 标准的平台并不是很多；JMAPI 是 SUN 公司提供的，受到了 Javasoft 等众多厂商的支持，它使用的管理协议和管理对象模型都是基于 Java 技术的，由于它也是基于 CORBA 的，因此可以跨越所有平台。

3.2.3 基于 Web 的分布式网络管理的功能模型

基于 Web 的网络管理模型的功能模型如图 3－7。

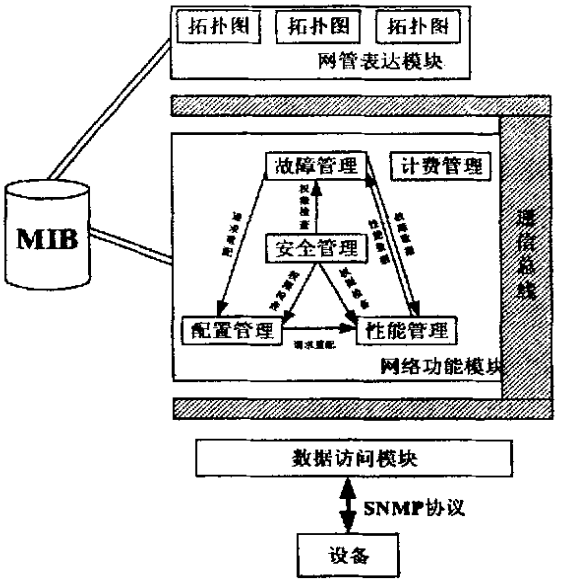


图 3—7 功能模型

功能模型主要由数据访问模块、网管功能模块和网管表达模块组成。通信总线主要用的是 CORBA 的 ORB 总线。

在本系统中，数据访问模块处于基础地位，它的任务就是通过 SNMP 协议同被管对象进行通信，采集网络管理所需的各种数据（主要为配置数据、报警数据和性能数据），并将这些数据经过一定的处理后写入数据库中。该模块并非随机调用，而是按照系统规定的采集周期定期采集除 Trap 信息之外的数据。Trap 信息的采集则通过陷阱接收事件实现，即一旦设备发出 Trap，系统便触发这个事件，从而保证报警的实时性。

网管功能模块的作用就是按照一定的规则对数据访问模块采集到的管理数据进行分析、整合，提供给网管表达模块使用。例如，数据访问模块每隔一定时间间隔从被管网络中采集性能数据，这些数据经过网管功能模块的分析和加工（如统计、计算等），形成各种报表，管理人员就可以通过浏览器对性能数据进行浏览。因此，网管功能模块在本系统中处于中间层位置，它包含了大量的应用逻辑，这些逻辑分别用于实现网络管理的各项管理功能，包括配置管理、性能管理、故障管理、安全管理和计费管理。配置管理的主要功能包括自动网络拓扑搜索；实时显示通断状态；对网络状态进行修改；重新配置网络参数；用户权限的配置等功能。性能管理主要提供性能检测功能、性能分析功能以及性能管理控制功能，另外还有在发现性能严重下降时启动故障管理系统的功能；设置设备使用门限。故障管理的主要功能包括轮询检测网络设备的故障；接受网络设备发出的 Trap 消息；维护故障日志。安全管理的主要功能包括检查用户的合法性；限制用户只能进行权限规定的操作。

网管表达模块的功能就是对数据访问模块采集到的网管数据进行有效的、直

观的表达，同时支持网络管理人员进行必要的管理操作，它是网络管理人员同本系统打交道的人机接口。在设计网管表达模块时，我们遵循了 Windows 的界面设计风格，网络管理系统的每一个操作都充分考虑到用户操作的便利性。

3.3 YSZ2002 电力监控管理系统的设计

集中式电力监控系统存在着：系统规模受限制、远距离监控困难和系统性能和应用瓶颈明显等问题。针对集中式监控系统的存在的问题，我们提出了在分布式电力监控系统的网络设计和实现的目标：子站分段管理，提高管理能力和可靠性；长距离管制，即指两个子站网段距离较远；提高工程效率和实时性；分布式的数据管理；易于发布信息；具有高可扩展性和高可用性。

考虑到分布式电力监控系统需要一套完整的专用监控网络管理的体系，要实现对监控网络的管理我们要先来熟悉一下整个监控网络的体系结构。

3.3.1 分布式电力监控管理系统整体体系结构

首先看一下整个系统的宏观上的网络拓扑图，如图 3—8。

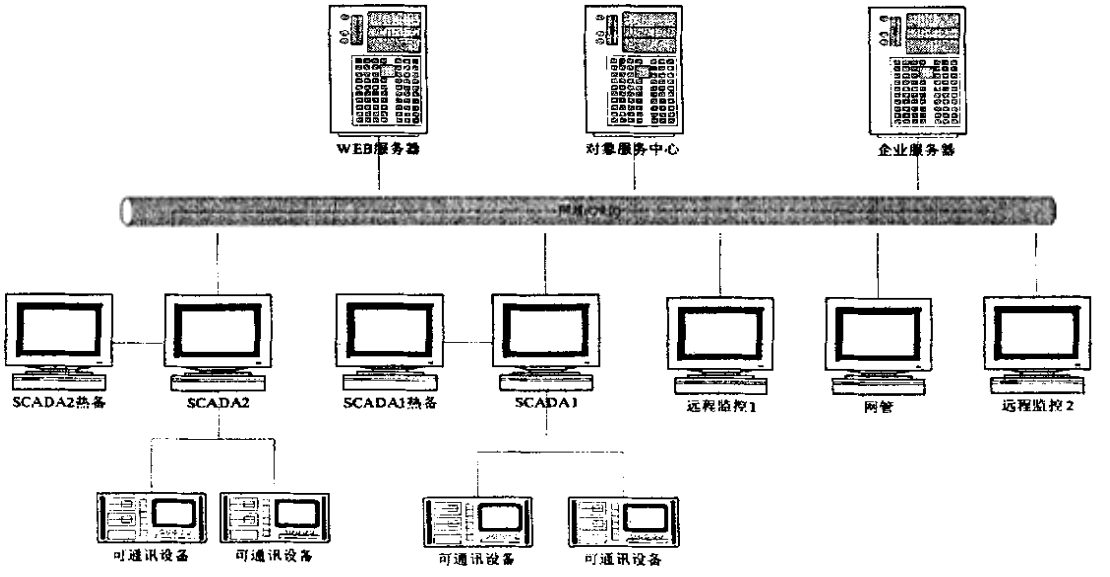
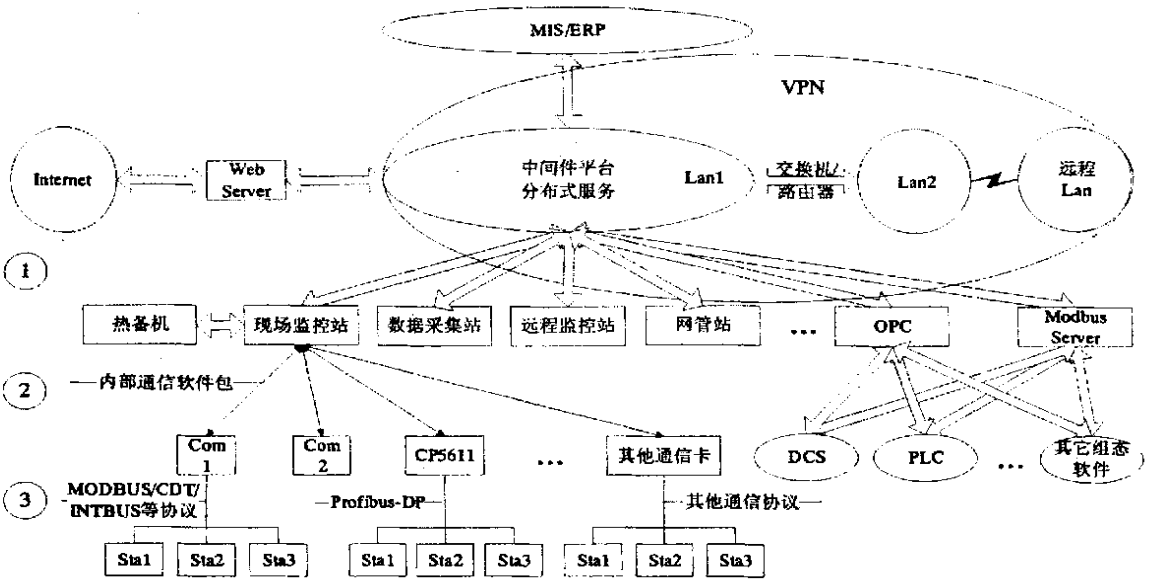


图 3—8 宏观上的网络拓扑图

整个系统中包括完成现场数据采集监视任务的 SCADAs 及其热备机，还包括对 SCADA 进行远程监控的工作站，以及可以管理整个网络的网管系统。这些子系统又通过 ORB 软总线与各种分布式对象服务、WEB 服务器以及企业服务器互连。其中分布式对象服务中心包括各种服务对象，但因为 CORBA 对象在一个网域中是可以任意部署的，因此它实际上是一个抽象的概念。各种分布式对象可以部署在网域中的任何一台电脑上，它们的位置是对使用者透明的。

更详细的系统结构图如图 3—9 所示。



从图上可以看到，本系统将整个监控环境从上往下划分为三个层次：

1. 第一层是 ORB 软总线层，主要用于资源共享以及系统互连。同时也可以通过 OPC 和 MODBUS Server 组件和其他系统互连，这些系统包括 DCS、PLC 以及其他的电力监控系统或自动化软件。而且这种互连和共享不仅可以在单一的 LAN 中实现，还可以通过交换机和路由器与其他 LAN 互连，甚至组成 VPN，形成一个统一的分布式服务平台；
2. 第二层是各子系统内部实现层，例如 SCADA 的实现；
3. 第三层是现场总线层，在这一层上 SCADA 通过各种现场总线与下位机通讯，从而进行数据采集和指令发送。

其中第二层和第三层传统的 C/S 模型系统已经做到，这里就不再赘述，但第一层是它所缺少的。第一层其实就是对分布式对象中间件的一种应用，是分布式系统层。这也正是分布式对象模型不同与 C/S 模型的地方，所以本文只对第一层的设计做详细介绍。

3.3.2 分布式对象中间件系统结构

在此之前，先了解两个下面将要用到的基本术语。

1. 分布式对象服务（简称服务）：服务其实就是指通过单个分布式对象独立工作或多个分布式对象的协同工作向外提供各种应用的一种手段。CORBA 规范已经定义了许多服务，但在本系统中我们要在一些 CORBA 服务的基础上实现电力监控系统所需要的服务。
2. 模块：在本系统中，模块是指使用各种服务来完成具体工作的子系统，是服务的消费者。同时模块也可以是一个 CORBA 对象，向外提供各种

应用接口。

整个分布式系统由各种服务和模块来构筑。这些服务和模块之间相互连接，共享资源，甚至可以相互操作，完成分布式电力监控任务。同时通过服务向系统外部的 ERP、WEB Server 和其他应用系统提供接入解决方案。整个分布式系统的分层式的体系结构如下：

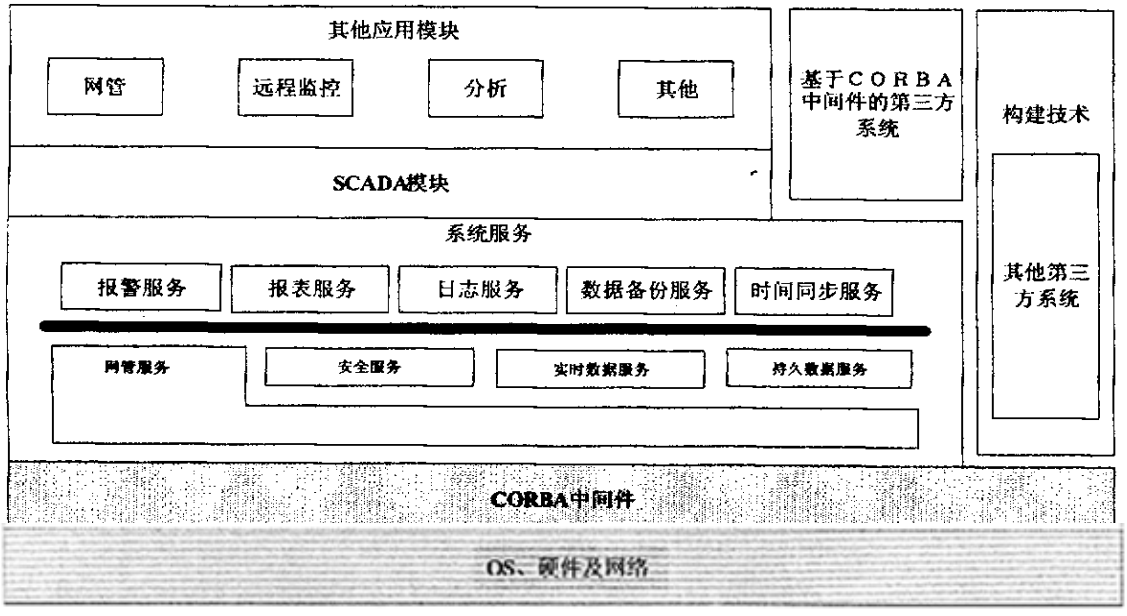
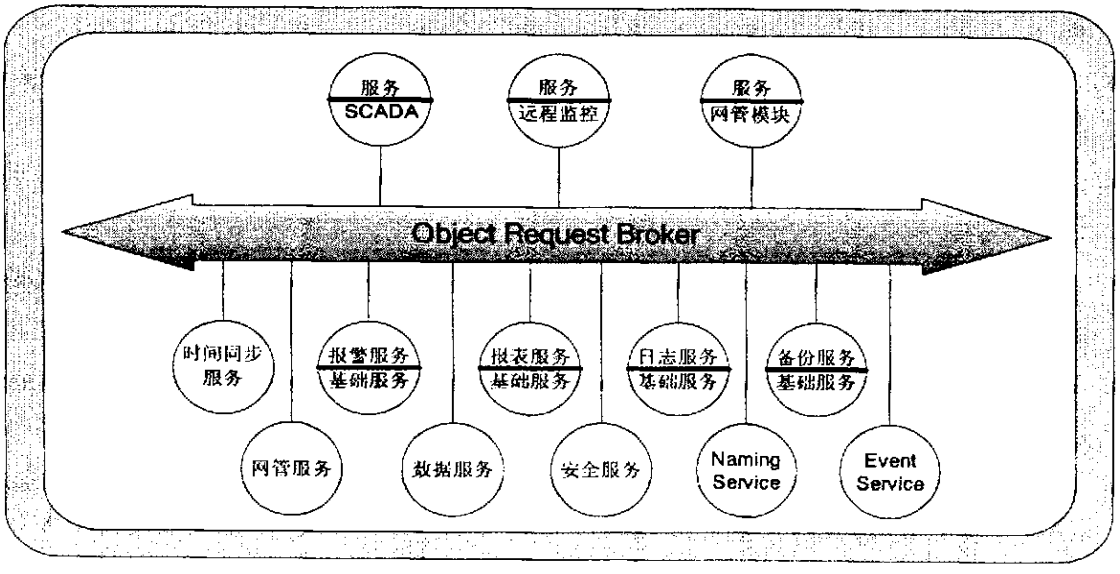


图 3—10 分布式系统层体系结构

如图所示，根据电力监控系统的需求，并达到分布式监控的目的，整个分布式系统可以被划分为几个层次。其中我们需要实现包括 8 种服务和 3 种模块。网管服务、安全服务、实时数据服务和持久数据服务是基础服务，其他服务在这些服务上构筑，而所有的模块又在系统服务上实现。

下图 3—11 则给出了系统的分布式对象框架：



3.3.2.1 服务

1．数据服务：分为实时数据服务和持久数据服务两部分。这两种服务提供分布式实时数据库和分布式历史数据库，使得监控系统的信息可以轻松接入 ERP 或其他企业信息系统，实现网内分布式对象之间的资源共享，而这时企业信息系统只需加入数据服务代理即可。

- 实时数据服务：实时数据服务用来从系统实时数据库或监控变量索引表中，检索并提供用户指定的实时数据。
- 持久数据服务：持久数据是指保存在系统数据库中的数据，和保存在工程文件上的数据，同实时数据库中的实时数据概念相对。服务支持用户通过服务规定的特定接口操纵数据库，包括对数据库的检索以及记录或字段值的增、删、改等。

2．安全服务：提供对系统用户的安全认证、授权和鉴权，系统安全记录，安全策略实施、管理等功能。分布式电力监控系统中所有的子系统的用户登录、操作等涉及安全性的问题的解决都由安全服务统一完成。

3．网管服务：提供网络管理功能，并封装 CORBA 的名字服务，实现了系统的命名服务，用于监视系统中各子系统的工作状况。

4．报警服务：提供报警素材管理，警情通知和报警信息管理功能。

5．时间同步服务：因为监控系统数据对时间敏感，所以我们将用时间同步服务实现整个系统中的时间同步。时间同步服务采用 GPS 时钟或时间服务器作为时间标准的提供者，并在每个子系统上配置时间同步代理。

6．报表服务：按报表配置生成系统报表，例如历史数据的年报表、月报表等，同时对报表模板进行管理。报表服务同样使得报表数据的获取、报表的整理和生成对用户是透明的，用户只需向报表服务发送请求即可获取已处理好的报表。

7．日志服务：收集整个系统的日志信息，并提供日志查询和删除操作。

8．数据备份服务：提供对指定数据对象的转储和导入功能。

3.3.2.2 模块

1．SCADA 模块：除了完成基本的与现场设备（本文中有时也称为下位机）的通讯和数据交互功能外，还具有分布式对象接口，以便与其他子系统资源共享和互操作。SCADA 模块主要对现场设备进行数据采集、测量以及各类信号报警等功能，对现场设备进行数据访问和数据交互是其本质目的。在整个监控系统中，SCADA 模块向下与下位机交互，向上却为各种服务和其他子系统提供所需数据，是整个系统的数据源泉。其人机界面可有可无。同时为了对整个系统进行有机的

管理，我们把 SCADA 模块所投运或者所监控的实体称为一个工程或子工程，而其所提供的 CORBA 对象则称为工程对象。

2．远程监控模块：远程监控模块可以配置在系统的任何一个地方，却可以通过使用网管服务、安全服务、数据服务同时对多个 SCADA（即多个工程或子工程）进行监控，满足了可以从一个地点看到多个信息的信息集成的需求。

3．网管模块：网管模块使用网管服务对整个系统的工作状态进行监视和管理，以及使用安全服务进行用户管理。但与一般网管不同的是它不考虑网络的可用性、可靠性等问题，也不提供进行错误定位的工具。它仅仅是提供了一种对整个网域中所有分布式对象状态信息的汇总视图。

3.3.2.3 系统部署

本系统采用嵌套工程组织的方式来部署整个监控系统。从图 3－10 的系统结构图可以看到，我们可以将整个系统看成一个大的工程组，，这个大工程组又可以分为几个子工程组。而在每个监控前端部署的一个 SCADA 模块被作为工程组或子工程组中的一个子工程，这些 SCADA 模块（子工程）通过 ORB，利用各种服务就可以将数据上传到网上，共享数据给远程监控模块和网管模块；同时我们已经可以清楚的知道，上述所谓的模块和服务其实都是一个个 CORBA 对象，所以远程监控模块和网管模块又可以利用各种服务或者直接调用 SCADA 的功能接口来操作 SCADA，例如进行四遥操作、查询数据库等等。

又由于分布式对象可以在网络系统中任意部署，因此在整个监控系统中，除了 SCADA 必须部署在监控前端以外，其他的模块和服务却是可以部署在网域中的任意上位机上。这样所有的上位机可以被概念的分为两种类型，服务器和工作站。

- 服务器：在其上运行某一服务的节点就被称为服务器。一个网域中可以有一个或几个服务器，但至少要有一个。服务器功能是为本身和整个网络上的其他工作站提供各种服务。
- 工作站：在其上运行某一模块的节点就称为工作站。因此一个节点可以即是服务器又是工作站。

同时根据工作站上运行的模块的配置不同，工作站可以有如下几种典型类型：

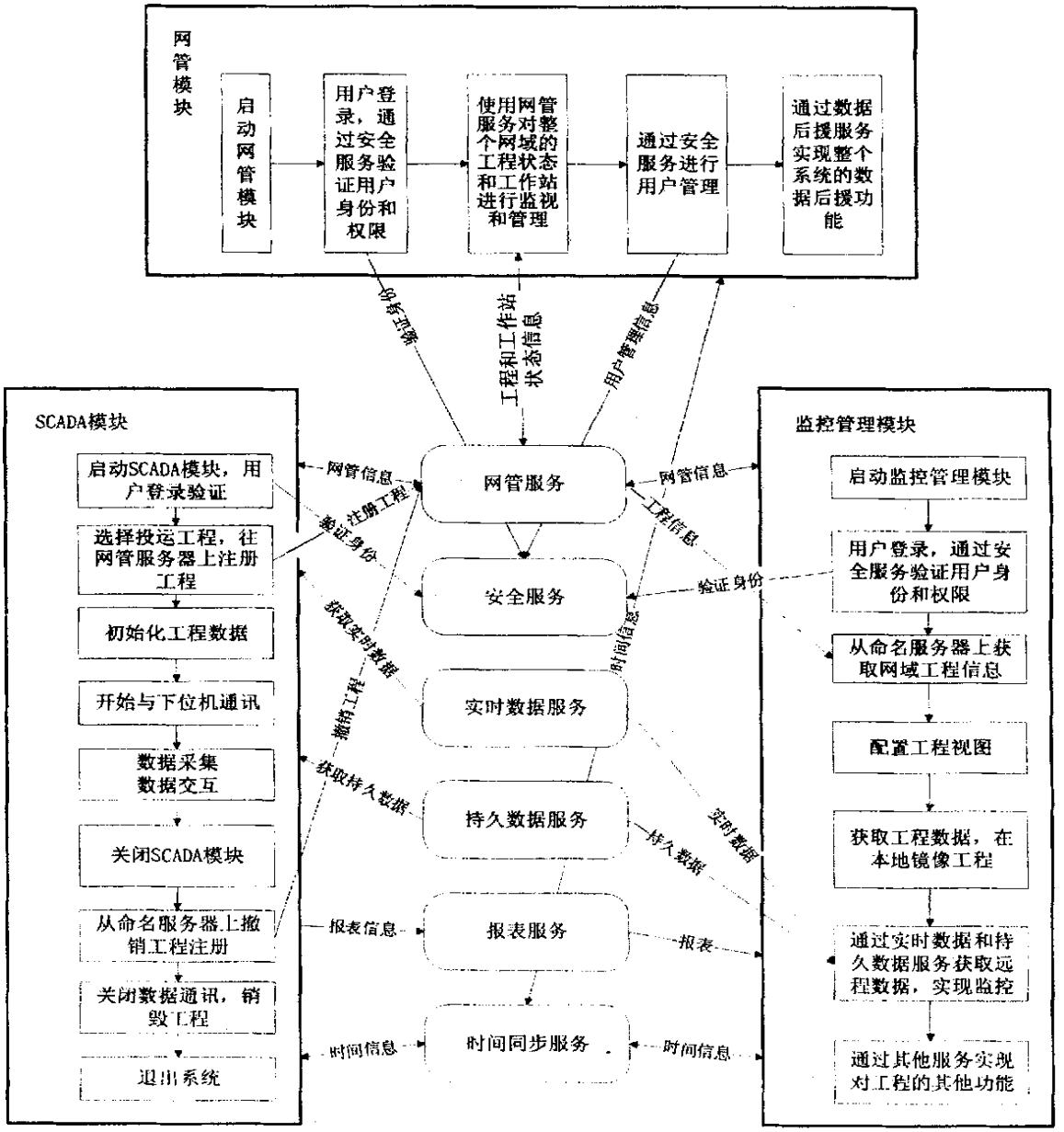
- 数据采集站：具备 SCADA 模块，可进行数据采集和数据交互；
- 现场监控站：具备数据访问和远程监控模块，既可以进行现场数据采集，又可以监控其他 SCADA 模块；
- 远程监控站：具备远程监控模块，可对指定 SCADAs 进行监控；

- 网管站：具备网管模块；
- 热备站：用于双机热备，实现对数据访问的备份，以保证下位机通讯网的稳定。

通过合理有效的组合和配置上述服务和模块，就可以构筑一个开放的、可扩展的分布式电力监控系统。

3.3.2.4 工作流程及协同工作机制

分布式系统的总体工作流程以及服务和模块之间的协同工作机制如下图 3—12 所示，图中每一个大方框和园角矩形都是一个 CORBA 对象。



3.4 将基于 Web 的分布式网络管理应用于 YSZ2002 电力监控系统

3.4.1 基于 Web 的 YSZ2002 分布式电力监控系统的网络管理体系结构的需求

本系统对网络管理体系结构有如下需求：

- 用户能够通过 Internet 在浏览器中监视和控制 YSZ2002 分布式电力监控系统的各现场设备；

（1）由 SCADA 模块实现对数据采集、数据存储，然后提供给其他服务和模块，要处理的数据包括实时和历史数据、工程组织数据、报警数据、故障录波数据和用电数据。这些数据通过数据服务向外提供。用户可以在由工程子站组织的工程窗口中看到这些实时数据并能够对他们进行相应的处理。用户还可以从浏览器端调用远程监控模块提供对所选工程的实时数据和历史数据的查询。远程监控模块提供一个实时数据字典，在这个字典中，用户可以方便的查询到每个子站的每一个变量的值；同时还提供一个历史数据字典，用户可以设置查询时间段，从远端的 SCADA 上将该时间段内的工程历史数据库获取并显示。

（2）用户在浏览器端可以查看到本地 SCADA 模块本地发生的报警信息，并对报警的内容做出相应的处理。SCADA 模块本地发生的报警信息必须传递到其他的远程监控模块上去。这些报警信息包括报警、故障等等。一旦发生报警，则 SCADA 要马上使用网管服务的通知服务来通知所有在线的远程监控模块；远程监控模块的得到通知后，则马上通过持久数据服务从 SCADA 中获取报警的详细情况，包括时间、报警原因、报警环境等，然后通过 Web 服务器的 Webservice 接口将报警信息的具体情况提供给远程的浏览器用户。对于报警的管理包括报警响应、报警查询、报警设置。

（3）SCADA 要可以接受客户浏览器端的远程遥测、遥信、遥调和遥控操作。这就要求 SCADA 不仅要向外提供数据，还要可以接受远程接口的调用，完成遥调和遥控。

（4）SCADA 模块中子站的登陆、卸载、挂牌、摘牌可以由客户在浏览器端来完成，它可以向 Web 服务器发出对子站的上述操作的请求，由网络管理中心调用管理代理来完成。这些操作的步骤应为远程监控模块将登陆、卸载、挂牌、摘牌请的求发向 SCADA 模块，SCADA 模块执行相应的动作，并发送相应的子站信息通知网管模块，网络模块将记录和通知其他的网管模块和远程监控模块这一子站变更的消息。

（5）报表功能是自动化系统一个非常重要的部分。远程监控模块通过报表

3.4 将基于 Web 的分布式网络管理应用于 YSZ2002 电力监控系统

3.4.1 基于 Web 的 YSZ2002 分布式电力监控系统的网络管理体系结构的需求

本系统对网络管理体系结构有如下需求：

- 用户能够通过 Internet 在浏览器中监视和控制 YSZ2002 分布式电力监控系统的各现场设备；

（1）由 SCADA 模块实现对数据采集、数据存储，然后提供给其他服务和模块，要处理的数据包括实时和历史数据、工程组织数据、报警数据、故障录波数据和用电数据。这些数据通过数据服务向外提供。用户可以在由工程子站组织的工程窗口中看到这些实时数据并能够对他们进行相应的处理。用户还可以从浏览器端调用远程监控模块提供对所选工程的实时数据和历史数据的查询。远程监控模块提供一个实时数据字典，在这个字典中，用户可以方便的查询到每个子站的每一个变量的值；同时还提供一个历史数据字典，用户可以设置查询时间段，从远端的 SCADA 上将该时间段内的工程历史数据库获取并显示。

（2）用户在浏览器端可以查看到本地 SCADA 模块本地发生的报警信息，并对报警的内容做出相应的处理。SCADA 模块本地发生的报警信息必须传递到其他的远程监控模块上去。这些报警信息包括报警、故障等等。一旦发生报警，则 SCADA 要马上使用网管服务的通知服务来通知所有在线的远程监控模块；远程监控模块的得到通知后，则马上通过持久数据服务从 SCADA 中获取报警的详细情况，包括时间、报警原因、报警环境等，然后通过 Web 服务器的 Webservice 接口将报警信息的具体情况提供给远程的浏览器用户。对于报警的管理包括报警响应、报警查询、报警设置。

（3）SCADA 要可以接受客户浏览器端的远程遥测、遥信、遥调和遥控操作。这就要求 SCADA 不仅要向外提供数据，还要可以接受远程接口的调用，完成遥调和遥控。

（4）SCADA 模块中子站的登陆、卸载、挂牌、摘牌可以由客户在浏览器端来完成，它可以向 Web 服务器发出对子站的上述操作的请求，由网络管理中心调用管理代理来完成。这些操作的步骤应为远程监控模块将登陆、卸载、挂牌、摘牌请的求发向 SCADA 模块，SCADA 模块执行相应的动作，并发送相应的子站信息通知网管模块，网络模块将记录和通知其他的网管模块和远程监控模块这一子站变更的消息。

（5）报表功能是自动化系统一个非常重要的部分。远程监控模块通过报表

服务提供对各个分布式工程的持久数据以表格或图表的形式进行集成显示、保存以及打印的功能，并可以在客户浏览器端现实。

（6）各个 SCADA 的运行日志存储在各自的数据库中，在本地只能看到自己的运行记录。而客户在浏览器端可以通过 Web 服务器调用远程监控模块通过数据服务提供了对所有日志信息的集成，方便用户统一查询、打印。

- 对于整个系统的网管服务有如下需求：

（1）整个分布式系统的名字存储、管理和解析功能：也就是说名字服务需要提供名字绑定、解除绑定，命名环境对象的创建、删除以及列表、排序等操作，命名对象采用分层组织图（命名树）的方式组织。支持对对象提供的服务的查询和索引等功能。同时，当命名对象发生改变时，命名服务要通知用户，使其能及时更新工程逻辑视图。

（2）整个分布式系统中消息的通知传递功能：在一个模块（或服务）需要唤醒其他模块（或服务）的数据更新或需要该模块（或服务）主动获取其他模块（或服务）的数据时，需要借助通知服务将数据打包发送给被通知端，如果这个数据包还不足以反应数据的变化，那么就应该让目的端主动调用源端的接口来更新数据。

（3）整个 YSZ2002 分布式电力监控系统中运行服务和模块状况的管理功能：维护服务信息注册表，登记系统已经运行的服务类型及其对象；当服务变更时刷新服务注册表。维护模块信息注册表，登记模块当前功能配置信息。获取方式有两种：模块启动时主动注册以及由服务在当前网段内搜索并登记。提供上述信息表的访问接口。

- 对于整个系统的网管模块有如下需求：

YSZ2002 网管模块的配置管理主要提供了网络拓扑管理、网络资源管理和用户权限的配置等功能；YSZ2002 网络管理模块的故障管理的含义和 OSI 有所不同，它是负责管理物理故障、逻辑错误和系统异常的一个管理单元；系统的性能管理应该是依靠 YSZ2002 分布式监控系统的网络管理协议中 QoS 适配器提供的一系列的策略实现的；YSZ2002 网络管理模块的计费管理，需要在 YSZ2002 网络管理模块上运行一个下面多级计费管理系统的汇总、察看和整理的系统；安全管理服务包括：用户鉴权、顶级用户对其他级别用户的授权和管理、安全管理信息的建立和维护三个方面。

由于在网络管理发展的历程中，对配置管理、安故障管理、性能管理、计费管理的研究已经非常的成熟，所以本文不再对这几方面的内容进行阐述，本文重点对现在各系统中最为关心的安全管理提出解决方案，这部分的内容将在稍后的第五章做出详细的阐述。

3.4.2 基于 Web 的 YSZ2002 分布式电力监控系统的网络管理体系结构

按照上述 3.2 节和 3.3 节的内容，本系统的体系结构可分为：网络管理的组织模型、网络管理的通信模型、网络管理的功能模型、网络管理的信息模型。

3.4.2.1 组织模型

网络管理组织布局的灵活性取决于管理部件的分散程度，取决于数据采集、处理、网络控制和监视功能的分布程度。所有这些方面构成了网络管理体系结构的组织模型。

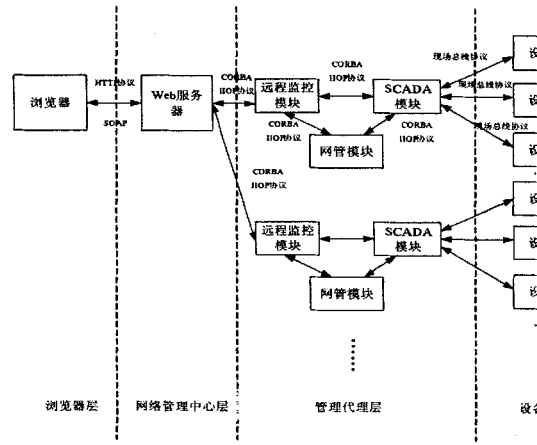


图 3－13 组织模型

- 管理者在浏览器层安插插件来扩展浏览器的功能，它在浏览器环境中工作，可以支持各种标准及格式，如网络通信、用户特定的应用等。插件是与浏览器紧密集成的，当浏览器被加载时，它自动搜索可用的插件，注册其支持的信息格式，当收到一个插件支持的信息时，就自动加载该插件进行处理。插件不同于应用程序，插件是与浏览器紧密集成在一起的，必要时由浏览器自动加载。利用插件技术可以使浏览器端具有自动处理网络信息的能力，自动处理特定类型的 HTML 信息。本系统中所安插的插件是 TXMLHTTP 和 SVGVIEW，TXMLHTTP 控件可以完成客户端到服务器的各种动作，如向服务发出请求的动作；SVGView 主要是为了改变在 HTML 页面中图形的显示。因为本系统是在原有的应用系统的基础上改建的基于 Web 的网络管理系统，所以使用如 JAVA 的方法来构建客户端平台是不划算的，所以我们放弃了 JAVA 的实现方案，采用一般的基于浏览器的客户端来实现。

- 网络管理中心可以有 Web 服务器来充当，管理代理则由远程监控模块、SCADA 模块和网管模块共同担当，其中管理代理的数据库放在实时监控现场设备的 SCADA 模块上。在 Web 服务器上的网络管理中心就不需要专用的 MIB 了。
- 服务器端用客户端的 SOAP 网管请求调用远程监控模块的 CORBA 对象，远程监控模块通过调用网管模块的 CORBA 对象，使远程监控模块得到 SCADA 模块的 CORBA 对象，SCADA 模块控制管理着多个子站，每个子站就是一个现场设备，SCADA 模块收到 SOAP 网管请求后，根据请求对设备进行管理，最后将结果返回给 WebService 接口，再由它将结果通过 HTTP 协议传送给客户端。
- 设备层包含被管的现场设备，每个 SCADA 模块控制管理多个被管设备，每个被管设备还可以有一个被管设备代理，用于与管理代理进行通信。

3.4.2.2 通信模型

网络管理体系结构的通信模型定义了在多个网络管理实施者之间交换管理信息的方案和机制。

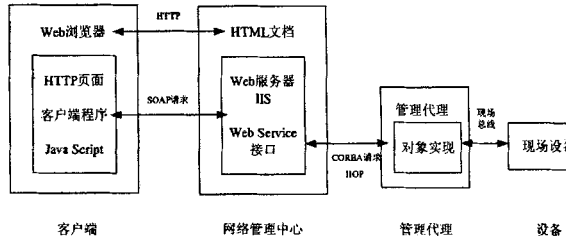


图 3－14 通信模型

- 浏览器层和网络管理层之间的 Web 页面交互通过 HTTP 协议实现，CORBA 请求和应答则通过 IIOP 协议实现。
- 网络管理中心的 Web 服务器上我们使用了 WebService 规范，在客户端的浏览器发出的网络管理请求通过 XML 格式书写，用 SOAP 封装，通过 HTTP 协议传输到服务器的 WebService 接口上，通过 WebService 接口来调用管理代理的 CORBA 对象来管理现场的设备。
- 网络管理中心和管理代理之间，以及管理代理内部的交互是纯 CORBA 的，因而是通过 IIOP 协议通信的。
- 由于传统的网络管理是管理的设备都是网络设备，如路由器，交换机等等，所以要使用 SNMP 协议，SNMP 与 CORBA 的转换还要使用 SNMP/CORBA

网关，而本系统使用的设备是现场的实时数据因此使用的现场总线协议，如MODELBUS 等等。

本模型实现了多层 CORBA 通信。客户端和网络管理中心、网络管理中心和管理代理以及管理代理之间都是客户方和实现方的关系。客户端和网络管理中心之间的 IDL 接口以及网络管理中心和管理代理之间的 IDL 接口定义了需实现的各项管理功能。SCADA 模块则将每个管理功能细化为若干对现场设备的具体操作，也就是 CORBA 请求的具体实现。远程监控模块得到 SCADA 模块的 CORBA 对象实现了 IDL 定义的所有接口，再由 SCADA 模块对现场设备进行具体的操作。

3.4.2.3 功能模型

网络管理活动被组织成一系列不同的功能组。网络管理体系结构的功能模型将所有的网络管理任务分成几个不同的功能域，如配置管理、计费管理、故障管理等，并试图为每一个功能域定义通用的管理功能。在一个功能域的范围内生成所要求的功能、服务和对象的一个技术涉及为该功能域创建功能性或描述性的模型。

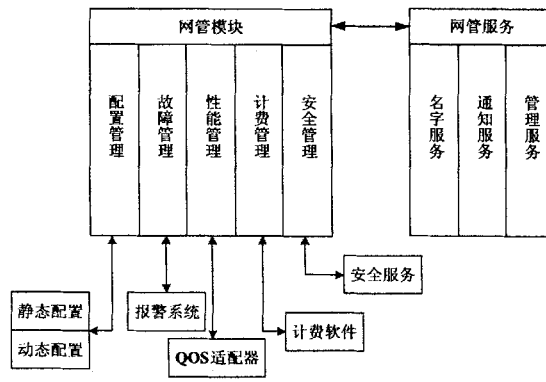


图 3—15 功能模型

YSZ2002 监控网络管理系统分为网络管理服务和网络管理模块两个主要部分。其中网络管理服务是通过实现网络管理协议来为整个分布式系统提供网络管理的支持的。而网络管理模块是在利用网络管理服务提供的服务的基础上对整个网域的问题、配置、计费、安全和性能进行管理的。

网络管理服务主要是完成在需求中提到的命名、通知、模块和服务信息管理三大服务功能。这三大服务将在 3.4.3.2 节里阐述。

网络管理模块则是对 OSI 管理框架中对网络管理定义的故障管理、配置管

理、计费管理、安全管理、性能管理 5 种功能的一种体现。当然，这种体现不是全面的实现上述五种功能，而是根据 YSZ2002 电力监控系统实际需要，及在网络管理服务提供的信息的基础上实现的动态网络拓扑配置管理、多级故障管理、性能管理、计费管理和安全管理。

本文将主要对本系统的五大管理功能中的安全管理功能进行研究和设计，这部分的内容将在第五章进行详细的阐述。

3.4.2.4 信息模型

网络管理信息库是一个网络管理系统的基础，管理信息库及描述管理信息的信息模型是一个网络管理体系结构的重要和核心的组成部件。

分布式对象模型中需要实现分布式数据库(即 MIB)，包括分布式历史数据库和分布式实时数据库。这两种数据库都由数据服务来提供。分布式数据库系统是地理上（或物理上）分散而逻辑上集中的数据库系统。所谓地理上分散是指各个节点分布在不同的地方，而逻辑上统一则指网络联结的各节点共同组成单一的数据库。对用户来说，一个分布式数据库从逻辑上看，如同集中式数据库一样，用户可以在任何一个场地执行全局应用。分布式数据库主要有数据共享、可靠性高、可用性好等特点。在本系统中，系统数据库，包括实时数据库和历史数据库都是分散存储的，即每一个子系统（SCADA 模块）的实时和历史数据都存储在本地，但通过数据服务可以相互共享，并向外提供一致接口，而数据存取是完全透明的。

系统在使用实时数据库和历史数据库时，使用了两种服务：实时数据服务和历史数据服务。这两种服务使用 SCADA 模块提供的分布式实时数据库和分布式历史数据库，统称 MIB，使得监控系统的信息可以轻松接入 ERP 或其他企业信息系统，实现网内分布式对象之间的资源共享，而这时企业信息系统只需加入数据服务代理即可。

- 实时数据服务：实时数据服务用来从系统实时数据库或监控变量索引表中，检索并提供用户指定的实时数据。
- 持久数据服务：持久数据是指保存在系统数据库中的数据，和保存在工程文件上的数据，同实时数据库中的实时数据概念相对。服务支持用户通过服务规定的特定接口操纵数据库，包括对数据库的检索以及记录或字段值的增、删、改等。

3.4.3 基于 Web 的 YSZ2002 分布式电力监控系统的网络管理体系的设计

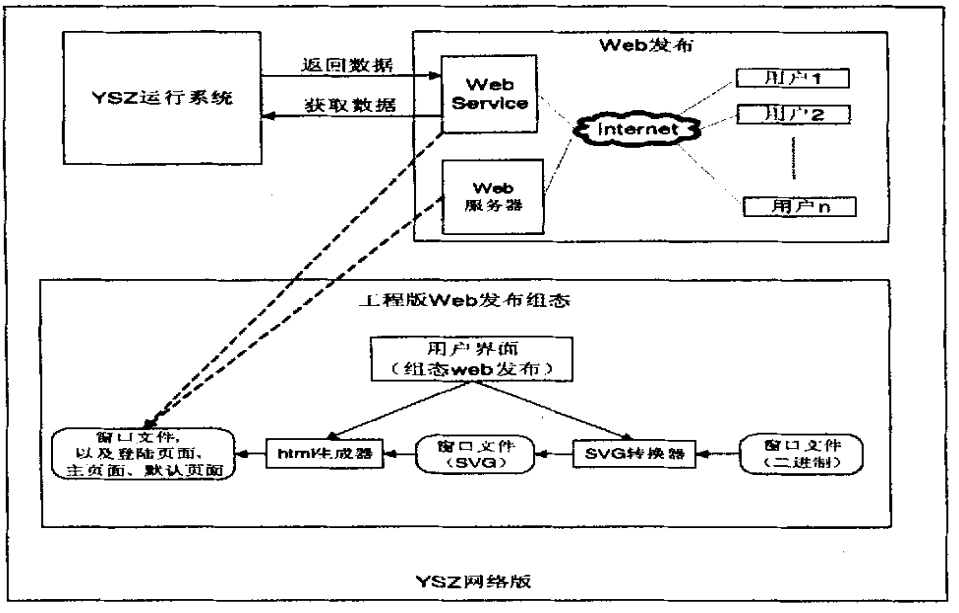
3.4.3.1 对客户端所能实现的功能的设计

由于本系统是根据实际的需要而设计的，所以目前只实现了基于 Web 的网络管理的一部分功能，争取在以后的开发过程中将其不断的完善。

本系统所实现的客户端的基于 Web 的网络管理的功能如下：

- 用户能够通过 Internet 在浏览器中查看电力系统的工程窗口的内容。工程窗口的内容可以根据现场设备的情况进行实时的窗口元素的刷新。对于一个工程的哪些窗口可以查看已经在工程版的组态过程中定义好了，用户只要拥有对这个工程的操作权限就可以看到整个工程里定义好的所有窗口。
- 可以通过 Web 服务器调用管理代理的 CORBA 对象，来实现查看某个工程窗口的权限，这部分将会在第五章的安全管理中详细阐述。

3.4.3.1.1 总体设计框图



从图中可以看到此 Web 功能的实现包括两个方面：工程版 Web 的发布组态（它和客户在浏览器中所看到的网页内容有直接的关系）及窗口刷新；以 WebService 为接口技术的窗口内容的实时数据的获得和传输。

本系统采用的技术为：SVG＋JavaScript＋XMLHTTP＋Web Service

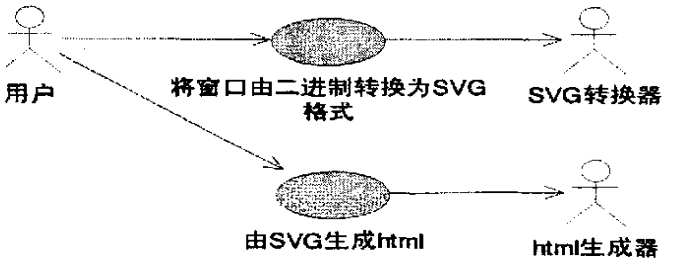
其中，SVG 用于显示各窗口。JavaScript 用于在客户端处理实时数据并对网页进行刷新。XMLHTTP 一直被用于实现“无刷新”的 Web 页面，它和 JavaScript 配合，可以完成数据从服务器到客户端的传输。XMLHTTP 技术实现了将表示逻辑与业务逻辑分开，将浏览器作为客户端，将复杂的界面表示逻辑交给 JavaScript

的处理，只要在服务器端留下 XML 格式的远程接口就行了。Web Service 主要用于在服务器端向客户端提供身份的授权和鉴权以及实时数据传输的功能。

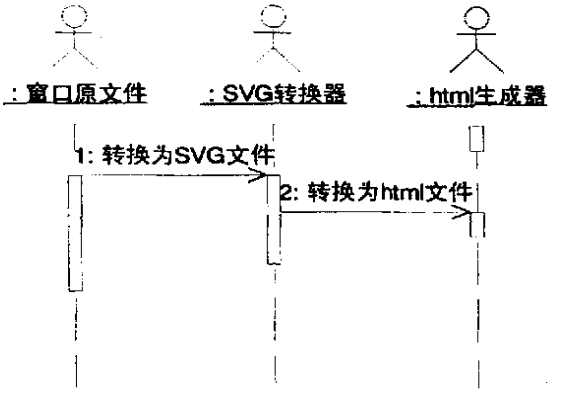
3.4.3.1.2 工程版发布窗口的设计

工程版基于 Web 发布组态主要实现将窗口源文件转换为网页形式。其中经历两个过程，一是先将源文件转换为 SVG 文件，另一方面是再由 SVG 生成 html。

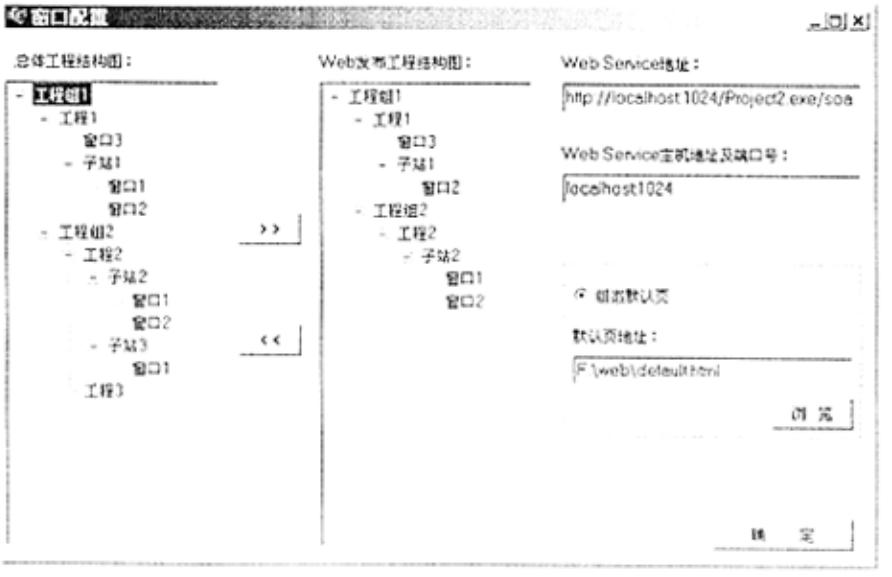
UserCase 图：



流程图：



组态界面原型：



3.4.3.1.3 利用 WebService 提供基于 Web 的网络管理

Web Service 主要是向客户端提供各种服务，其中包括身份验证，以及刷新数据的传输。Web Service 用 Soap 协议传递用 XML 标准封装的数据给客户端的浏览器，客户端的 Javascript 解析 XML 标准封装的刷新数据，并用这些数据刷新反映工程窗口的 HTML 页面。

1. 身份验证

用户访问 Web 发布的登陆页面，输入工号、用户名、密码，调用 Web Service 提供的身份验证服务，Web Service 通过安全服务来验证用户输入的工号、用户名、密码是否正确。用返回值（用户的权限值）来控制（客户端要做一些判断）用户是否可以进入主页面（如果验证通过，允许用户进入主页面，否则提示用户没有权限）。

返回值包括：用户身份验证结果，以及用户对各工程或工程组的查看权限。

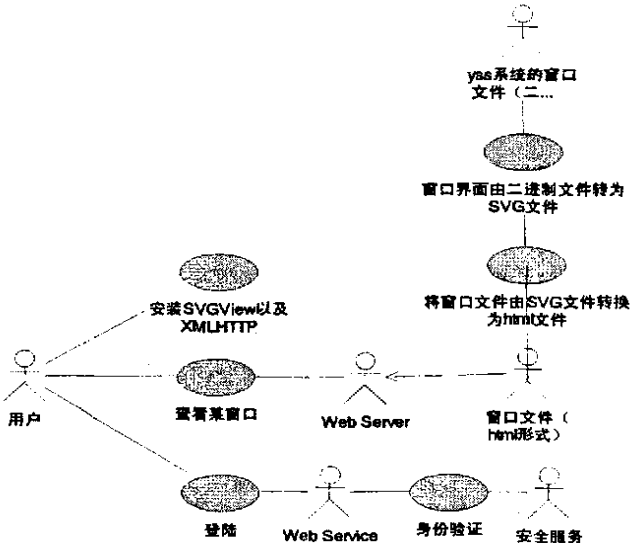
2. 刷新数据的传输。

a. 用户把想要浏览的正在投运的工程的 Web 文件夹中的子文件夹 WebServer 放置到 Web 服务器上，用户根据不同的网络地址进入不同的登陆页面，再进入相应的工程（工程组）的主页面，对于主页面左边的树状图是在工程版组态的时候进行初始化的。

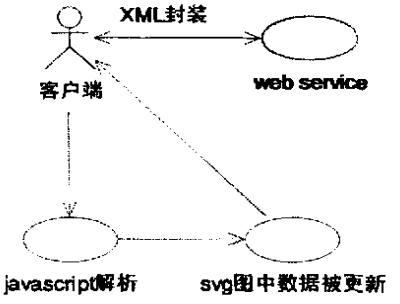
b. 树状图可以全部被展开，用户点击想要查看的窗口，如果用户有权限查看该窗口，则显示该窗口，否则，显示默认页。实时刷新时，调用 Web Service 的刷新接口，根据参数（工程路径和窗口名）到 WebService 文件夹去查原工程文件，看看窗口元素的哪些属性需要刷新，调用实时数据服务到具体投运工程的 SCADA 里获取实时数据，然后再转换为 XML 格式的属性数据，最后打包封装到客户端，再由客户端解析数据内容进行刷新。

UserCase 图：

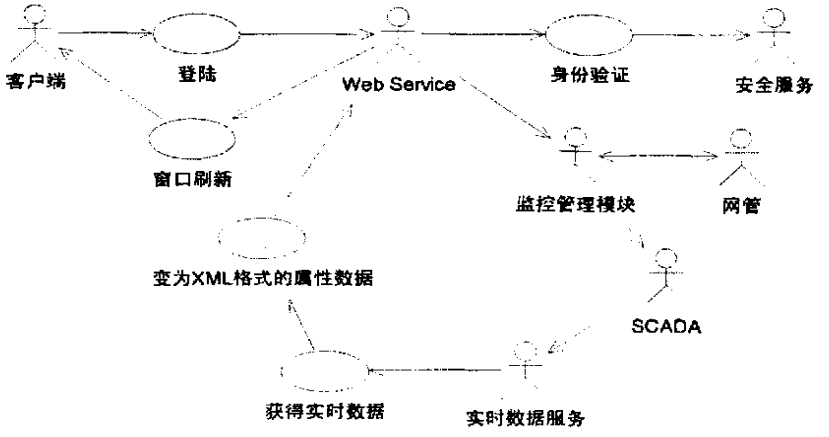
A. 用户通过 Web Server 访问工程版组态的 Web 页面



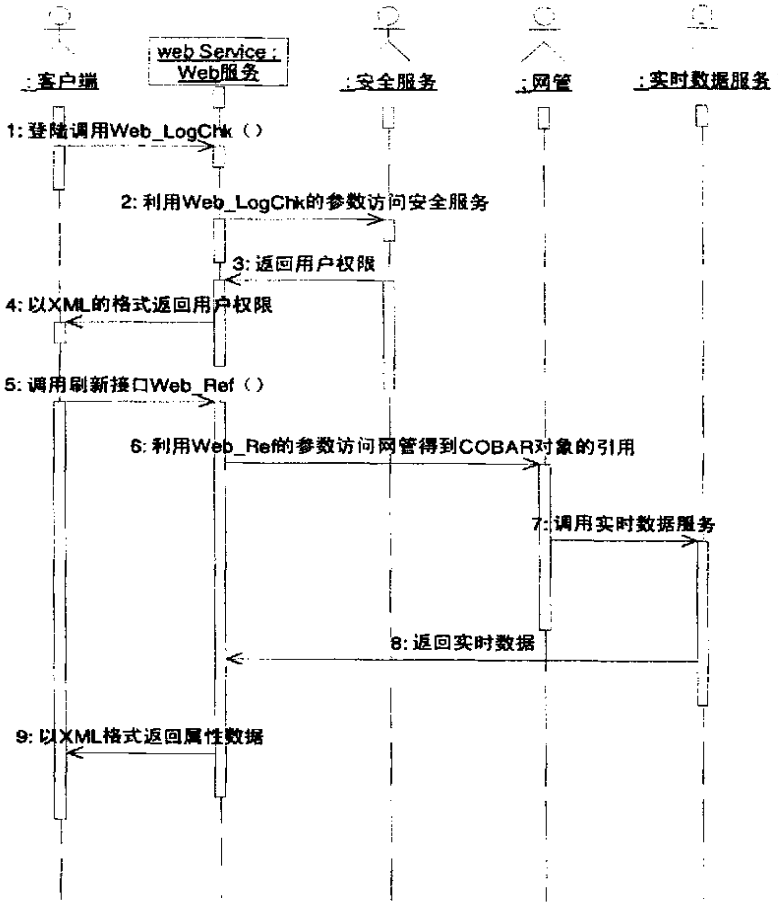
b、客户端访问 Web Service 的数据流



c、Web Service 实现的功能



流程图：



3.4.3.2 网管服务所能实现的功能的设计

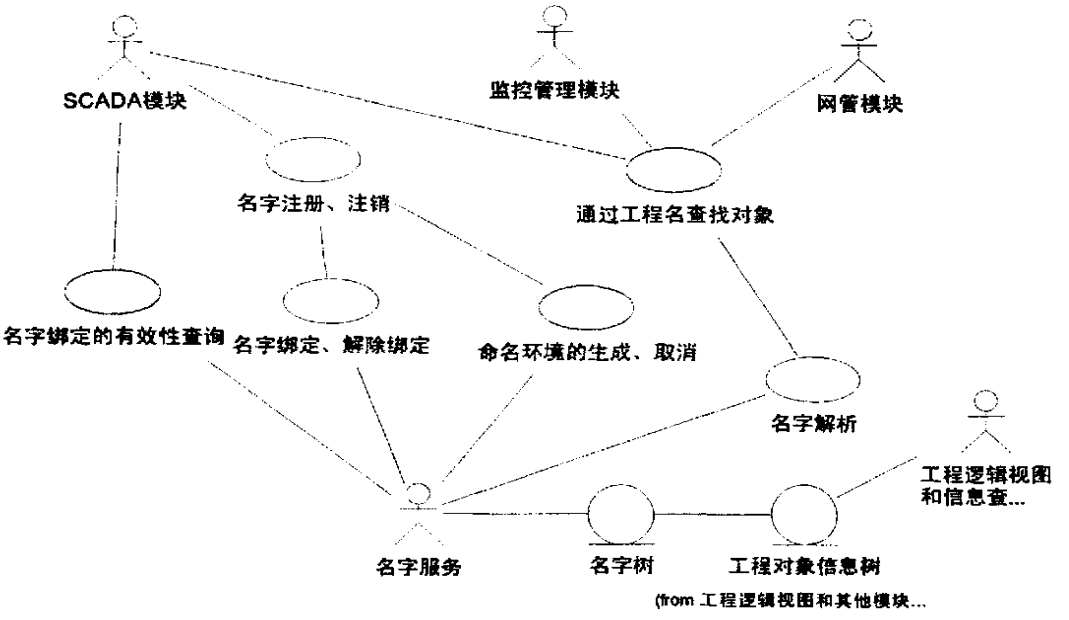
网管服务主要用来提供网络管理功能，并通过封装 CORBA 的名字服务，实现了系统的命名服务，同时还用于监视系统中各子系统的工作状况。可以用 UML 的 UserCase 图一一描述。

- 命名服务

要对整个分布式系统进行管理，就必须提供一种灵活而方便的手段来管理各个分布式对象，而本系统就是通过命名服务来提供对所有 SCADA 中工程对象的管理。命名服务其实是提供了名字到对象引用（模块和服务）的映射。其他客户端只需知道对象名称就可以通过命名服务来得到其对象引用，从而完成所需功能。例如，SCADA 的工程对象首先得向命名服务注册，然后如果远程监控模块想要得到某个工程对象引用，他只需简单的将此工程的名称作为参数来调用命名服务的接口就可以获取此工程对象的引用，而不必在远程监控模块中对系统中所有的工程对象进行管理。最后在关闭 SCADA 时，还得从命名服务注销工程对象。

下图是命名服务的 UserCase 图，从图中可以清楚的知道命名服务需要提供如下接口：

- 工程对象注册和注销；
- 工程对象引用绑定；
- 工程对象信息树。



- 通知服务

在一个模块（或服务）需要唤醒其他模块（或服务）的数据更新或需要该模块（或服务）主动获取其他模块（或服务）的数据时，需要借助通知服务将数据打包发送给被通知端，如果这个数据包还不足以反应数据的变化，那么就应该让目的端主动调用源端的接口来更新数据。可能要用到通知服务的包括以下几个方面：

- SCADA 模块的子站状态改变
- SCADA 模块的磁盘空间不足信息
- SCADA 模块的自动控制、电器连锁和报警调节修改信息
- SCADA 模块的报警和故障信息
- 安全服务的用户权限变更

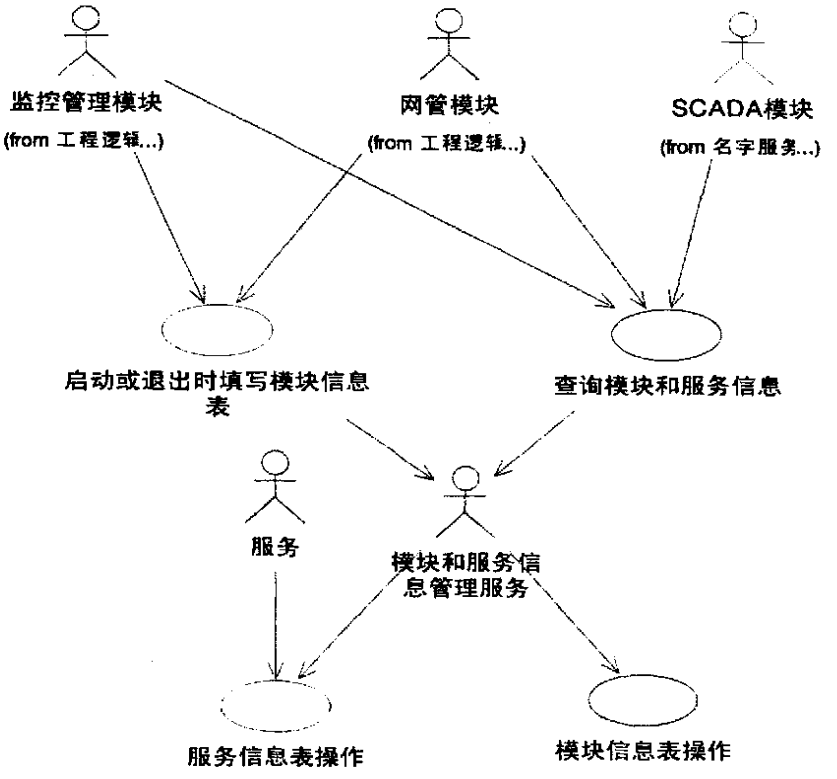
下图是通知服务的 UserCase 图。

- 模块和服务信息管理服务

除了要对工程对象进行管理外，还要对服务和其他模块进行管理，至少要知道在整个网域中存在哪些服务或模块，以及这些服务和模块所在的位置。因此系统还提供了模块和服务信息管理服务，其功能包括

- 维护服务信息注册表，登记系统已经运行的服务类型及其对象；当服务变更时刷新服务注册表；
- 维护模块信息注册表，登记模块当前功能配置信息。获取方式有两种：模块启动时主动注册以及由服务在当前网段内搜索并登记；
- 提供服务和模块信息查询。

下图是模块和服务信息管理服务的 UserCase 图。



第四章 利用 Web Service 技术来实现基于 Web 的 B/S 网管体系结构

4.1 Web Service 概述

随着 Internet 在各个领域应用的普及和深化，人们迫切需要能够方便地实现 Internet 上跨平台、语言独立、松散耦合的异构应用之间的交互和集成，这对分布式计算提出了新的要求。Web Service 作为一种新的技术应运而生，提出了面向服务的分布式计算模式。Web Service 的主要目标就是在现有的各种异构平台的基础上构筑一个通用的与平台无关、语言无关的技术层，各种不同平台之上的应用依靠这个技术层来实施彼此的连接和集成。在我们的 YSZ2002 电力监控系统中我们使用 Web Service 技术来具体实现基于 WEB 的网络管理。

Web Service 平台需要一套协议来实现分布式应用程序的创建。任何平台都有它的数据表示方法和类型系统。要实现互操作性，Web Service 平台必须提供一套标准的类型系统，用于沟通不同平台、编程语言和组件模型中的不同类型系统。组成 Web Service 平台有四个关键技术：XML、SOAP、WSDL、UDDI。

4.2 Web Service 技术的特点

采用 Web Service 构成的应用系统主要有以下特点：

- Web Service 允许在不同平台上、以不同语言编写的各种程序以基于标准的方式相互通信，这充分体现了 Web Service 的异构性。
- Web Service 使用标准的 Web 协议 HTTP 和 TCP/IP 等。

拥有以上两大特点的 Web Service 统一了 DCOM(Distributed Component Object Model)、CORBA(CommonObject Request Broker Architecture)、RMI(Remote Method Invocation)等技术，通过基于 HTTP 的 SOAP 协议在 Web 上提供软件服务，使用 WSDL 文件对服务进行说明，并通过 UDDI 进行注册。这样以 Web Service 方式提供的现有应用程序服务，可以构建新的、更强大的应用程序。

4.3 Web Service 的体系结构

Web Service 是用面向对象技术封装起来的对象，它对外以对象方法的形式提供服务，在 Internet 环境中，提供远程发现和调用的机制。首先，对象在 UDDI 中注册，则应用开发就可以利用其发现机制，查找所需的服务，UDDI 就会返回

第四章 利用 Web Service 技术来实现基于 Web 的 B/S 网管体系结构

4.1 Web Service 概述

随着 Internet 在各个领域应用的普及和深化，人们迫切需要能够方便地实现 Internet 上跨平台、语言独立、松散耦合的异构应用之间的交互和集成，这对分布式计算提出了新的要求。Web Service 作为一种新的技术应运而生，提出了面向服务的分布式计算模式。Web Service 的主要目标就是在现有的各种异构平台的基础上构筑一个通用的与平台无关、语言无关的技术层，各种不同平台之上的应用依靠这个技术层来实施彼此的连接和集成。在我们的 YSZ2002 电力监控系统中我们使用 Web Service 技术来具体实现基于 WEB 的网络管理。

Web Service 平台需要一套协议来实现分布式应用程序的创建。任何平台都有它的数据表示方法和类型系统。要实现互操作性，Web Service 平台必须提供一套标准的类型系统，用于沟通不同平台、编程语言和组件模型中的不同类型系统。组成 Web Service 平台有四个关键技术：XML、SOAP、WSDL、UDDI。

4.2 Web Service 技术的特点

采用 Web Service 构成的应用系统主要有以下特点：

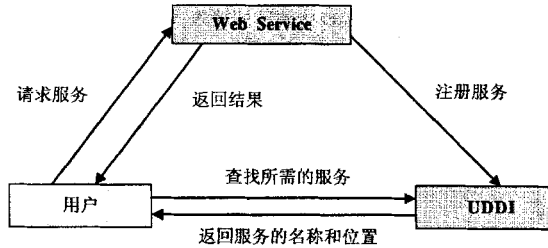
- Web Service 允许在不同平台上、以不同语言编写的各种程序以基于标准的方式相互通信，这充分体现了 Web Service 的异构性。
- Web Service 使用标准的 Web 协议 HTTP 和 TCP/IP 等。

拥有以上两大特点的 Web Service 统一了 DCOM(Distributed Component Object Model)、CORBA(CommonObject Request Broker Architecture)、RMI(Remote Method Invocation)等技术，通过基于 HTTP 的 SOAP 协议在 Web 上提供软件服务，使用 WSDL 文件对服务进行说明，并通过 UDDI 进行注册。这样以 Web Service 方式提供的现有应用程序服务，可以构建新的、更强大的应用程序。

4.3 Web Service 的体系结构

Web Service 是用面向对象技术封装起来的对象，它对外以对象方法的形式提供服务，在 Internet 环境中，提供远程发现和调用的机制。首先，对象在 UDDI 中注册，则应用开发就可以利用其发现机制，查找所需的服务，UDDI 就会返回

查找结果(服务名称和位置)。应用开发就可以通过 SOAP 协议访问远程的 Web Service，远程的 Web Service 通过身份认证后，就会调用其方法，并将结果返回应用程序。Web Service 这些服务是用 WSDL 来描述的。Web Service 的体系结构图 4—1 如下：



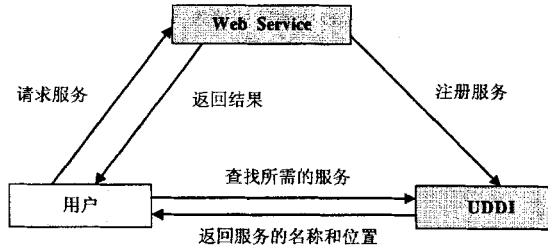
4.4 Web Service 的组成及其关键技术

- XML 和 HTTP 这是 Web Service 最基本的平台。当前最流行的系统集成方案就是 B/S 结构,而这正是建立在 HTTP 协议之上的应用, Web Service 的 SOAP 协议支持 HTTP 消息绑定,这就为 Web Service 部件通过 Interlnet 交互奠定了协议基础,并具有穿透防火墙的良好特性。XML 是一种元语言,可以用来定义和描述结构化数据,它是 Web Service 得以实现的语言基础。Web Service 的其它协议规范都是以 XML 形式来描述和表达的。
- SOAP 协议定义了服务请求者和服务提供者之间的消息传输规范。SOAP 用 XML 来格式化消息,用 HTTP 来承载消息。SOAP 包括三部分:定义了描述消息和如何处理消息的框架的封包、表达应用程序定义的数据类型实例的编码规则、以及描述远程过程调用和应答的协定。
- WSDL为服务提供者提供以XML格式描述Web Service请求的标准格式,将网络服务描述为能够进行消息交换的通信端点的集合,以表达一个 Web Service 能做什么,它的位置在哪里,如何调用它等。
- UDDI 是 Web Service 的信息注册规范,以便被需要该服务的用户发现和使用它。UDDI 规范描述了 Web Service 的概念,同时也定义了一种编程接口。通过 UDDI 提供的标准接口,企业可以发布自己的 Web Service 供他人查询、调用;也可以查询特定服务的描述信息,并动态绑定到该服务上。

4.4.1 XML

可扩展标记语言 XML (Extensible Markup Language)是 Web Service 平台中表

查找结果(服务名称和位置)。应用开发就可以通过 SOAP 协议访问远程的 Web Service，远程的 Web Service 通过身份认证后，就会调用其方法，并将结果返回应用程序。Web Service 这些服务是用 WSDL 来描述的。Web Service 的体系结构图 4—1 如下：



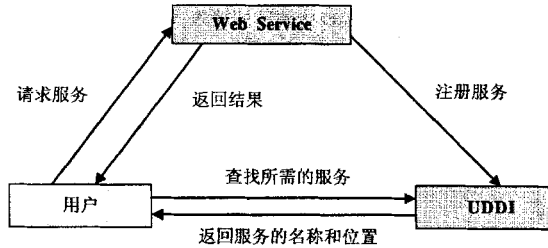
4.4 Web Service 的组成及其关键技术

- XML 和 HTTP 这是 Web Service 最基本的平台。当前最流行的系统集成方案就是 B/S 结构,而这正是建立在 HTTP 协议之上的应用, Web Service 的 SOAP 协议支持 HTTP 消息绑定,这就为 Web Service 部件通过 Interlnet 交互奠定了协议基础,并具有穿透防火墙的良好特性。XML 是一种元语言,可以用来定义和描述结构化数据,它是 Web Service 得以实现的语言基础。Web Service 的其它协议规范都是以 XML 形式来描述和表达的。
- SOAP 协议定义了服务请求者和服务提供者之间的消息传输规范。SOAP 用 XML 来格式化消息,用 HTTP 来承载消息。SOAP 包括三部分:定义了描述消息和如何处理消息的框架的封包、表达应用程序定义的数据类型实例的编码规则、以及描述远程过程调用和应答的协定。
- WSDL为服务提供者提供以XML格式描述Web Service请求的标准格式,将网络服务描述为能够进行消息交换的通信端点的集合,以表达一个 Web Service 能做什么,它的位置在哪里,如何调用它等。
- UDDI 是 Web Service 的信息注册规范,以便被需要该服务的用户发现和使用它。UDDI 规范描述了 Web Service 的概念,同时也定义了一种编程接口。通过 UDDI 提供的标准接口,企业可以发布自己的 Web Service 供他人查询、调用;也可以查询特定服务的描述信息,并动态绑定到该服务上。

4.4.1 XML

可扩展标记语言 XML (Extensible Markup Language)是 Web Service 平台中表

查找结果(服务名称和位置)。应用开发就可以通过 SOAP 协议访问远程的 Web Service，远程的 Web Service 通过身份认证后，就会调用其方法，并将结果返回应用程序。Web Service 这些服务是用 WSDL 来描述的。Web Service 的体系结构图 4—1 如下：



4.4 Web Service 的组成及其关键技术

- XML 和 HTTP 这是 Web Service 最基本的平台。当前最流行的系统集成方案就是 B/S 结构,而这正是建立在 HTTP 协议之上的应用, Web Service 的 SOAP 协议支持 HTTP 消息绑定,这就为 Web Service 部件通过 Interlnet 交互奠定了协议基础,并具有穿透防火墙的良好特性。XML 是一种元语言,可以用来定义和描述结构化数据,它是 Web Service 得以实现的语言基础。Web Service 的其它协议规范都是以 XML 形式来描述和表达的。
- SOAP 协议定义了服务请求者和服务提供者之间的消息传输规范。SOAP 用 XML 来格式化消息,用 HTTP 来承载消息。SOAP 包括三部分:定义了描述消息和如何处理消息的框架的封包、表达应用程序定义的数据类型实例的编码规则、以及描述远程过程调用和应答的协定。
- WSDL为服务提供者提供以XML格式描述Web Service请求的标准格式,将网络服务描述为能够进行消息交换的通信端点的集合,以表达一个 Web Service 能做什么,它的位置在哪里,如何调用它等。
- UDDI 是 Web Service 的信息注册规范,以便被需要该服务的用户发现和使用它。UDDI 规范描述了 Web Service 的概念,同时也定义了一种编程接口。通过 UDDI 提供的标准接口,企业可以发布自己的 Web Service 供他人查询、调用;也可以查询特定服务的描述信息,并动态绑定到该服务上。

4.4.1 XML

可扩展标记语言 XML (Extensible Markup Language)是 Web Service 平台中表

示数据的基本格式。除了易于建立和易于分析外，XML 主要优势在于，它提供了标准的数据封装技术，让数据交换跨越了各种平台、操作系统和开发工具。通过 XML，各种应用程序之间交换数据将不再困难。XML 提供了数据交换的标准，当不同的 Internet/Intranet 应用系统需要调用彼此提供的服务时，不需要知道对方的实现技术，只要以 XML 封装彼此的服务进入点，就可以顺利地使用对方的服务。这样不但达成了标准交换数据的机制，也顺利的使用 XML 集成各种不同的 Internet/Intranet 应用系统。

XML 是一种元语言，使用它可以生成并格式化文档标记。在 HTML 下，目前使用的标记是静态的，例如：<HEAD>及<BODY>是被紧紧地集成在 HTML 标准中，且不能改变或扩展。XML 则允许创建自己的标记标签并可以对其进行配置。每一个元素均可通过自定义的文档类型定义（Document Type Definition，DTD）和样式表（Style Sheet）来定义，并可在一个或多个 XML 文档中使用，这是他与 HTML 语言相比的优点。

XML 应用程序目前支持级联样式表（Cascading Style Sheet，CSS），但是还存在一种可扩展性更强的样式表规范，叫做可扩展样式表语言（Extensible Stylesheet Language，XSL）。使用 XSL，可以确保 XML 文档以相同的方式被格式化，不管它们是用于何种应用程序或出现在何种平台上。

XML 使用标记来标记内容以传送信息，标记用于界定内容而语法允许用户自行定义任何复杂性的结构，对于跨平台的大量数据交换和存储而言，XML 是一个非常理想的解决方案，它强大的生命力在于可以根据需要任意扩展，可以自动地将这类扩展信息传递给任何读取信息的实体。XML 具有以下主要优点：

（1） 适于异构应用系统间的数据共享；

（2） 强大的数据检索能力；

（3） 方便的数据存储机制；

4.4.2 SOAP

SOAP 和 Web Service 将会是现在以及未来 Internet 应用的主流技术，SOAP 是目前业界中的标准技术，而 Web Service 是数据技术混合运用的构想。用各种分布式技术以及组件模型来开发应用系统时，例如使用 COM、CORBA、或是 EJB 等技术来开发分布式应用系统，每一种组件模型都使用自己的数据封装方式以及通信协议，因此当不同的组件模型想要集成在一起的时候，了解对方的数据封装方式和通信协议成为一个技术难题。如何让用户自由的使用 WEB 服务而不受底层实现的影响，针对于此提出了一个标准规格—简单对象访问协议 SOAP（Simple Object Access Protocol）。SOAP 使用 XML 标准规范来定义不同系统之间的沟通，封装通信协议，并通过 HTTP 传送到远程的系统之中，因此解决了许

示数据的基本格式。除了易于建立和易于分析外，XML 主要优势在于，它提供了标准的数据封装技术，让数据交换跨越了各种平台、操作系统和开发工具。通过 XML，各种应用程序之间交换数据将不再困难。XML 提供了数据交换的标准，当不同的 Internet/Intranet 应用系统需要调用彼此提供的服务时，不需要知道对方的实现技术，只要以 XML 封装彼此的服务进入点，就可以顺利地使用对方的服务。这样不但达成了标准交换数据的机制，也顺利的使用 XML 集成各种不同的 Internet/Intranet 应用系统。

XML 是一种元语言，使用它可以生成并格式化文档标记。在 HTML 下，目前使用的标记是静态的，例如：<HEAD>及<BODY>是被紧紧地集成在 HTML 标准中，且不能改变或扩展。XML 则允许创建自己的标记标签并可以对其进行配置。每一个元素均可通过自定义的文档类型定义（Document Type Definition，DTD）和样式表（Style Sheet）来定义，并可在一个或多个 XML 文档中使用，这是他与 HTML 语言相比的优点。

XML 应用程序目前支持级联样式表（Cascading Style Sheet，CSS），但是还存在一种可扩展性更强的样式表规范，叫做可扩展样式表语言（Extensible Stylesheet Language，XSL）。使用 XSL，可以确保 XML 文档以相同的方式被格式化，不管它们是用于何种应用程序或出现在何种平台上。

XML 使用标记来标记内容以传送信息，标记用于界定内容而语法允许用户自行定义任何复杂性的结构，对于跨平台的大量数据交换和存储而言，XML 是一个非常理想的解决方案，它强大的生命力在于可以根据需要任意扩展，可以自动地将这类扩展信息传递给任何读取信息的实体。XML 具有以下主要优点：

（1） 适于异构应用系统间的数据共享；

（2） 强大的数据检索能力；

（3） 方便的数据存储机制；

4.4.2 SOAP

SOAP 和 Web Service 将会是现在以及未来 Internet 应用的主流技术，SOAP 是目前业界中的标准技术，而 Web Service 是数据技术混合运用的构想。用各种分布式技术以及组件模型来开发应用系统时，例如使用 COM、CORBA、或是 EJB 等技术来开发分布式应用系统，每一种组件模型都使用自己的数据封装方式以及通信协议，因此当不同的组件模型想要集成在一起的时候，了解对方的数据封装方式和通信协议成为一个技术难题。如何让用户自由的使用 WEB 服务而不受底层实现的影响，针对于此提出了一个标准规格—简单对象访问协议 SOAP（Simple Object Access Protocol）。SOAP 使用 XML 标准规范来定义不同系统之间的沟通，封装通信协议，并通过 HTTP 传送到远程的系统之中，因此解决了许

多以往难以解决的集成困难。SOAP 是一个标准的功能规范，而不是如何实现的标准，因此各种不同的开发工具、应用程序服务器或是软件场上都可能使用不同的方法来实现，但是这些不同的工具在最后产生的 SOAP 封装却是一致的，并符合 SOAP 功能规范。

4.4.2.1 概念及定义

SOAP 是一种把各种程序语言对于远程的调用和参数转化为以 XML 封装的机制，再通过底层的传送通信协议传递到远程的应用系统。SOAP 是一个以 XML 技术为基础的通信协议，它在分布式环境中，以交换信息为主要工作。本身包括四个部分：

（1）一个数据包封套（Envelope），其中包含了描述此 SOAP 封装的信息，以及如何处理这些数据报的信息；

客户端在进行远程调用时，可以指定要在远程执行的行动，因此在 SOAP 封装的数据之中可以选择性的把这些指定的动作封装起来，并加入一些说明信息。

（2）一群编码规则以代表应用程序定义的数据类型；

由于各种不同的程序员都拥有不同的数据类型，因此 SOAP 定义了一些基本的数据类型，各种程序员必须把各自使用的数据类型转化为 SOAP 定义的数据类型，以便能够彼此使用标准的格式交换数据。

（3）一个代表远程调用以及调用结果的约定；

除了数据之外，客户端调用远程的方法，以及远程方法执行的结果也需要在 SOAP 中表达。

(4) 一个绑定约定，以便使用底层的传送通信协议来交换数据。

SOAP 封装数据和传送通信协议绑定的信息，用来代表远程服务者在什么地方，由于 SOAP 功能规格中第一个实现的传输通信协议是 HTTP，因此这个绑定信息经常是一个指定到远程被调用目标的 URL。

4.4.2.2 SOAP 的功能规范

SOAP 就是使用标准的格式把程序语言对于远程的调用转化为 XML 代表的内容。XML 不但能够封装有意义的信息，客户端和服务器端也能轻易的使用 XML 解析器，从中取出封装的信息，而且现在几乎所有的平台都支持 XML 和提供 XML 解析器，因此最适合用来封装调用信息的内容。

4.4.2.2.1 SOAP 的标准

SOAP 是定义如何封装交换信息的标准，而使用 SOAP 功能规格封装的信息称为 SOAP 包（SOAP Packet）。SOAP 包由固定的元素组成，然后使用标准的

规则封装交换信息，因此使用 SOAP 标准沟通的双方就可以遵照 SOAP 标准而解析并且了解 SOAP 包中封装的信息意义，进而达成交换信息的目的。

一个 SOAP 包主要由三个元素组成，它们分别是 Envelope，Header 和 Body 元素。其中 Envelope 元素是根元素，它包含了子元素 Header 和 Body 元素。其中 Header 元素的目的是封装额外的信息，而 Body 元素则包含了封装的远程调用的信息。这些远程调用信息包含了调用的方法以及这个调用方法的所有参数信息，在 Header 元素和 Body 元素中可以包含其他符合 XML 规则的子元素，整个 SOAP 封装可以由图 4－2、图 4－3 表示。

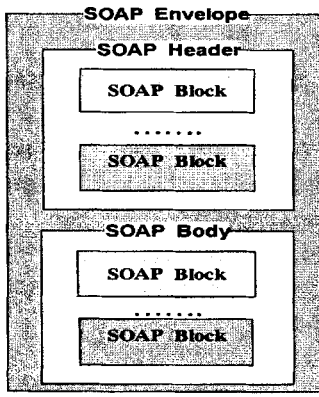


图 4－2 SOAP 包的组成元素

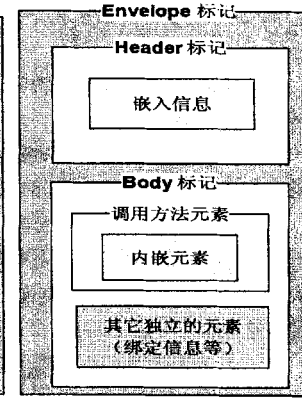


图 4－3 SOAP 包的详细组成元素以及元素的功能

由于 Body 元素是封装远程调用的内容，因此在 Body 元素中第一个子元素便是远程调用的方法，至于远程调用方法的参数信息，可以是远程调用方法的元素的子元素，或是跟随在远程调用方法的元素之后独立的元素。

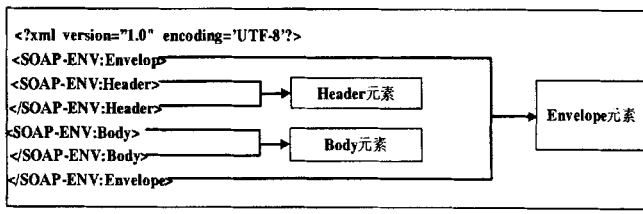


图 4－3 SOAP 包的代码格式

当任何程序语言使用 SOAP 标准进行远程调用或是交换数据时，程序语言把调用的信息转换为 SOAP 包的过程称为 Serialozation。在 SOAP 功能规格中把处

理 SOAP 包的操作点称为 SOAP 节点（SOAP Node），而第一个把远程调用转换为 SOAP 包的节点称为起始节点。当 SOAP 通过传递通信协议传送 SOAP 包时，应用系统可以拦截传送的 SOAP 包，加以处理之后，再继续传送 SOAP 包，一直到最后一个 SOAP 节点，而这整个 SOAP 封装传送的流程称为 SOAP 传送路径。SOAP 功能规格定义了如何封装各种数据类型、使用的命名空间和 SOAP 封装的内容以及结构。如图 4－4 所示 SOAP 的传送路径。

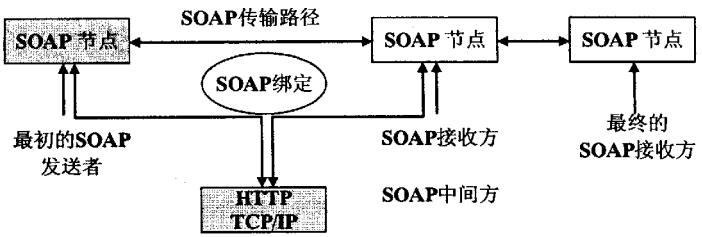


图 4－4 SOAP 节点和 SOAP 传送路径

4.4.2.2.2 SOAP　Envelope

SOAP 的 Envelope 元素是 SOAP 封装的 XML 的根元素，它的标准格式如下：

<SOAP-ENV：Envelope>

</SOAP-ENV：Envelope>

在这个根元素之中将内嵌 Header 和 Body 元素。通常在 Envelope 元素中会包含若干个重要的属性（Attribute），以便让实现 SOAP 的机制能够正确地处理 SOAP 封装之中的信息。

SOAP 的功能规格定义了 Envelope 元素的使用规则如下：

（1） 在 Envelope 元素中可以包含 SOAP－ENV：encodingStyle 属性，以定义使用的封装机制，SOAP－ENV：encodingStyle 属性的作用域是所有包含在 Envelope 元素中的内容；

（2） Envelope 元素可以包含额外的命名空间；

（3） Envelope 元素可以包含其它的子元素，但是必须使用命名空间来指定。

Envelope 元素的 SOAP－ENV：encodingStyle 属性的意义在于叙述如何封装数据。因此如果在 SOAP 数据包中加入了 SOAP－ENV：encodingStyle 属性指定的纲要，那么实际的程序代码在进行把 SOAP 数据包转换成调用堆栈时必须遵

照这些纲要定义的封装规则。一般来说通用的SOAP数据包在缺省时是使用http://schemas.xmlsoap.org/soap/encoding/定义的规则来封装数据的。

在SOAP功能规格中使用了许多的命名空间。在SOAP数据包中使用命名空间的目的是为了在SOAP数据包中加入定制的纲要以规定如何Serialization数据以及转换SOAP数据包的内容为调用堆栈。不同的SOAP实现技术可能在SOAP封装中使用不同的命名空间。在一般的SOAP封装中经常会使用下面的命名空间：

Xmlns:SOAP-ENC= "http：//schemas.xmlsoap.org/soap/encoding/"

4.4.2.2.3 SOAP Header

SOAP的Header元素允许传递额外或是定制的信息给Web Service服务器。我们可以在Header元素加入许多应用系统需要的特性。

（1） Header元素是可选的，这个意思是说在SOAP数据包中可以包含Header，也可以没有Header元素。如果SOAP数据包中包含Header元素，那么它必须是第一个在Envelope元素之后的子元素。

（2） Header元素必须遵守SOAP的功能规格，除非在Envelope元素的SOAP－ENV：encodingStyle属性中有另外的定义。

（3） Header元素可以包含SOAP－ENV：mustUnderstand属性

（4） Header元素的子元素必须使用命名空间来指定。

mustUnderstand属性的意义是说，接受这个SOAP数据包的SOAP服务器必须了解Header元素中这个子元素的意义，而且期望能够收到这个子元素的数值。如果在SOAP数据包中拥有指定mustUnderstand属性的子元素，而接受这个SOAP数据包的服务器无法处理此子元素，或是没有期望收到此子元素，那么SOAP服务器必须返回一个SOAP错误包给客户端。因此mustUnderstand属性可以让客户端确定SOAP服务器是否成功地处理客户端地要求。

4.4.2.2.4 SOAP Body

SOAP的Body元素是真正封装远程调用的地方，在这个元素中我们将会发现它包含了调用方法的名称、参数以及返回的结果。在SOAP功能规格中Body元素可以包含三种不同类型的元素，它们是：

（1） 远程调用；

（2） SOAP服务器返回结果；

（3） SOAP发生的错误。

当Body元素是封装远程调用时，那么在<SOAP－ENV：Body>之后的第一个子元素必须是以调用的方法名称作为标记名称的一个独立元素，而在此子元素之中则是此方法的参数信息。此外方法名称元素必须使用命名空间来指定。方法名称之内的参数子元素也是使用参数名称作为标记名称。

当 SOAP 服务器接受了客户端的 SOAP 请求并且运算完成之后，就同样地会把运算的结果以 SOAP 数据包返回给客户端。SOAP 服务器返回结果封装的规则和一般的 SOAP 封装一样，只是返回的结果是位于<SOAP－ENV：Body>之后的第一个子元素，而且这个独立元素的标记名称是使用如下的格式：

调用方法名称+Response

至于远程调用方法的返回值则是使用下面的一对起始/结尾标记包含的：

<return></return>

当 SOAP 服务器在处理 SOAP 请求发生错误时，SOAP 会根据发生的错误种类来决定返回什么错误包。SOAP 错误包的封装规则也和一般的 SOAP 规则一样，只是在 SOAP 错误封装中，在<SOAP－ENV：Body>元素之后的第一个子元素是使用<SOAP－ENV：Fault>作为标记名称的子元素。一般来说在<SOAP－ENV：Fault>独立元素之内包含两个内嵌元素：

（1）<SOAP－ENV：Faultcode>，代表发生错误的代码。

（2）<SOAP－ENV：Faultstring>，代表发生错误的原因。

4.4.2.2.5 SOAP　Action 字段

在 SOAP 功能规格中 SOAPAction 是可选字段，这个意思是说 SOAPAction 可以出现在 SOAP 包之中，也可以不出现。不过在目前大部分的 SOAP 实现中几乎都会使用 SOAPAtion 字段。SOAPAtion 字段的目的是让客户端指明想让 SOAP 服务器处理的动作。因此当包含 SOAPAtion 字段的封装传递时，SOAP 节点可以只分析 SOAP 包表头中的 SOAPAtion 字段内容就可以知道是不是需要处理此 SOAP 包，而无需处理整个 SOAP 包之后才知道是否需要处理此 SOAP 包，因此可以加快 SOAP 包被处理的效率。由于 SOAPAtion 是给 SOAP 服务器处理的，因此它并没有固定的格式，只要客户端和服务器端遵照相同的规则即可。在 C++Builder6 中 SOAPAtion 字段是使用如下格式定义的：

Urn：Web Service 实现类文件名称－Web Service 接口名称#调用的方法名称

4.4.2.3 SOAP 的特点

SOAP 的主要特点是简明性和可扩展性。对于各 Web Service 之间的弱耦合关系，采用一种简单的调用描述方法，符合其需求和特点。SOAP 是完全基于 XML 之上的。他也继承了 XML 的可扩展性和可描述性。同时，SOAP 利用了 XML Schema 所定义的丰富的数据结构，使得其对数据的描述功能变得更为丰富。

SOAP 以 XML 形式提供了一个简单、轻量的用于无中心分布式环境下交换结构化和类型信息的框架。因为 SOAP 本身并没有规定任何编程模型和应用语义，所以 SOAP 实际上只是提供了消息框架，可以用多种下层协议(如

TCP,HTTP,SMTP 甚至 POP3)来支持此框架，规范提供一个灵活的框架来定义任意的协议绑定。在此框架上，也可以用 SOAP 实现各种消息交换模式(Message Exchange Patterns,MEP)，比如请求/响应模式、通知模式、发布/订阅模式以及对等会话模式等，从而提供对上层应用的有力支持。

4.4.3 WSDL

4.4.3.1 概述

随着通信协议和消息格式在 Web 中的标准化，以某种格式化的方法来描述通信变得越来越重要，其实现的可能性也越来越大。Web Service 采用 Web Service 的描述语言 WSDL (Web Service Description Lanugage)来描述其服务接口。WSDL 采用 XMLSchema 定义，能够对各种语言实现的服务接口进行描述，具有语言无关性。WSDL 将 Web Service 定义为网络端点的集合，使用类型、消息、端口等元素来描述服务接口。请求者据此可以知道服务要求的数据类型、消息结构、传输协议等，从而实现对 Web Service 的调用。

使用 WSDL，Web 服务被描述为一组服务访问点(端口 Port)，端口会处理包含面向文档或面向过程的消息，其中操作和消息都是被抽象描述的，然后它们被绑定到一个具体的网络协议和消息格式上。本质上，WSDL 至少需要说明 Web 服务三个方面的信息：

1)服务做什么，即服务所提供的操作；

2)如何访问服务—数据格式以及访问服务操作的必要协议；

3)服务位于何处—由特定协议决定的网络地址。用 WSDL 来表达的服务描述文档其组成如图 4－5 所示。

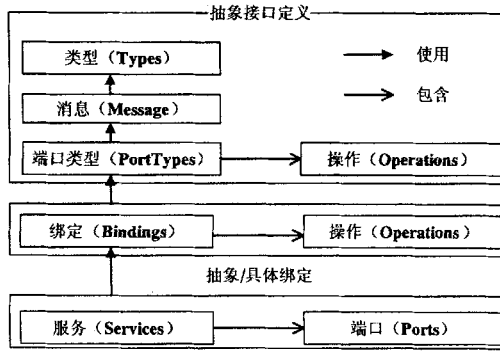


图 4－5 WSDL 的信息模型

TCP,HTTP,SMTP 甚至 POP3)来支持此框架，规范提供一个灵活的框架来定义任意的协议绑定。在此框架上，也可以用 SOAP 实现各种消息交换模式(Message Exchange Patterns,MEP)，比如请求/响应模式、通知模式、发布/订阅模式以及对等会话模式等，从而提供对上层应用的有力支持。

4.4.3 WSDL

4.4.3.1 概述

随着通信协议和消息格式在 Web 中的标准化，以某种格式化的方法来描述通信变得越来越重要，其实现的可能性也越来越大。Web Service 采用 Web Service 的描述语言 WSDL (Web Service Description Lanugage)来描述其服务接口。WSDL 采用 XMLSchema 定义，能够对各种语言实现的服务接口进行描述，具有语言无关性。WSDL 将 Web Service 定义为网络端点的集合，使用类型、消息、端口等元素来描述服务接口。请求者据此可以知道服务要求的数据类型、消息结构、传输协议等，从而实现对 Web Service 的调用。

使用 WSDL，Web 服务被描述为一组服务访问点(端口 Port)，端口会处理包含面向文档或面向过程的消息，其中操作和消息都是被抽象描述的，然后它们被绑定到一个具体的网络协议和消息格式上。本质上，WSDL 至少需要说明 Web 服务三个方面的信息：

1)服务做什么，即服务所提供的操作；

2)如何访问服务—数据格式以及访问服务操作的必要协议；

3)服务位于何处—由特定协议决定的网络地址。用 WSDL 来表达的服务描述文档其组成如图 4－5 所示。

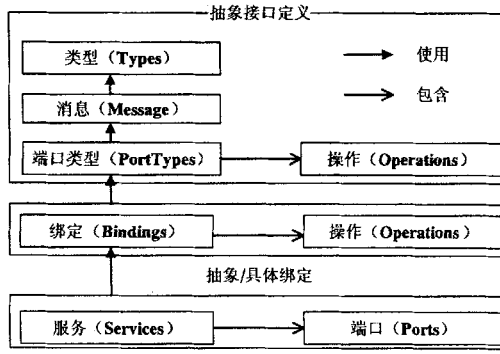


图 4－5 WSDL 的信息模型

WSDL从两个层面来描述Web服务，一个是抽象层面一个是具体层面，即图中的服务接口定义(抽象层面)和服务实现定义(具体层面)。服务实现定义描述了服务提供者如何实现特定的服务接口，服务接口定义和实现定义结合在一起，组成一个完整的Web服务描述。抽象层包括类型、消息和端口类型，因为binding元素又是在接口定义层中出现，为了不引起混淆，将其单独列为绑定层。

4.4.3.2 组成

(1)definition元素:它是整个WSDL文档的根元素,包含所有其它WSDL元素。WSDL允许根据定义的抽象级别,将同一definition中的不同元素分别放入不同文档中,需要时仅用import元素将其导入即可。这样,既有助于编写更为清晰的服务定义,又能最大限度地对各种元素的定义再利用;

(2)types元素:它包含与交换的消息相关的数据类型定义。为了获得最大程度的互操作性与平台的中立性,WSDL选用XMLSchema作为标准类型系统,也允许通过扩展性元素来添加类型,以增强数据表达能力。types定义可以进行单独存放,使其利于复用;

(3)message元素:它代表所传输内容的抽象定义。message(消息)由一些逻辑片段(part)构成,片段(part)是一种用于描述消息的逻辑抽象内容的灵活机制,每个片段分别与某个类型系统中的定义相关联,绑定(binding)就是使用片段名称来指定有关片段的绑定专用信息的;

(4)portType元素:它代表抽象操作(operation)的集合,以及所涉及的抽象消息。

(5)binding元素　它为特定portType所定义的操作和消息指定消息格式和协议细节。WSDL为了能够描述在多种环境下的传输要求,针对各种协议进行单独的绑定细节描述,并将它们分别定义在不同的命名空间中,提供在不同的环境下使用。绑定时必须先指定一个协议,然后按该协议的绑定细节,指定绑定风格、传输方式、操作(operation)地址以及所传消息内各片段(part)的编码方式等内容;

(6) port元素　它为绑定(binding)指定一个地址,但不能指定除地址信息以外的任何绑定信息;

(7) service元素　它用于将一组相关端口(port)聚合在一起,一个服务(service)中的所有端口(port)间都不能相互通信。由于一个portType可以有多个binding,因此,为使某种服务(service)能适应多种环境的要求,可以使该服务(service)下几个端口(port)都属于同一端口类(portType),但使用不同的绑定(binding)或地址。这样,调用者就可根据自己的情况,调用相应的端口(port)进行通信。

4.4.4 UDDI

Web Service 就是由 WSDL、SOAP 标准和实现层共同形成的概念。Web Service 以 WSDL 对外提供逻辑描述接口，让外界了解 Web Service 提供的服务以及如何使用它，并且要求使用 SOAP 标准封装来封装交换的数据和进行远程调用，并且最后提供 WSDL 和实现层之间的绑定以便能够让任何的程序语言或是开发工具提供真正的实现代码。通过逻辑描述接口和实现层隔离的机制，Web Service 可以让不同的应用系统只需要关心对方的 WSDL 即可，而不需要知道对方是不是和自己使用相同的程序语言或是开发工具进行开发，也不需要估计最方是否和自己兼容，因此它解放了平台和开发工具的束缚。

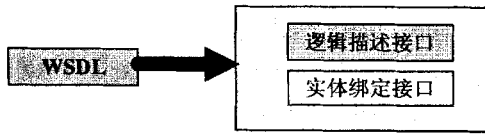


图 4－6 WSDL 的组成

Web Service 所产生的 WSDL 内容，前半部分描述此 Web Service 提供的服务信息，而后半部则是实现此 Web Service 的实体绑定信息。由于 WSDL 中包含了绑定 Web Service 服务器的绑定接口，因此如果客户端取得了 Web Service 的 WSDL 内容，并且据此绑定了 Web Service 服务器，那么一旦 Web Service 服务器的实现位置改变，那么客户端就无法再绑定到 Web Service 服务器了。为了克服 WSDL 的缺点，UDDI 应运而生了。UDDI 能够提供 Web Service 服务集群的功能，并且具备将 Web Service 的服务接口和绑定接口分开的机制，这样可以让 Web Service 应用系统更容易管理，并且克服了 WSDL 和 Peer-To-Peer 的缺点。

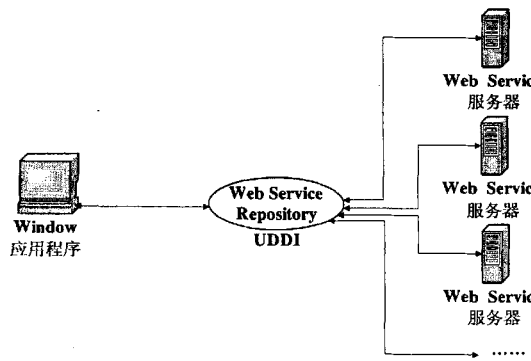


图 4－7 集群的 Web service 应用系统结构

4.4.4.1 UDDI 的定义

UDDI(Universal Description Discovery and Integration)是一个公开的标准，它以结构化的方式来注册、管理 Web Service 服务器的信息以及相关的服务信息。

通过 UDDI，可以发布或是搜寻 Web Service 信息，从而达到资源共享的目的，在 UDDI 中的信息是以分门别类的方式管理和存储的，因此可以很方便的找到所需要的信息。它提供了 Web Service 应用系统的基础虚拟平台，让所有的 Web Service 应用系统都能够通过一个共同的标准相互的沟通。

4.4.4.2 UDDI 与 Web Service 中的其他标准之间的关系

UDDI 标准是建立在 SOAP 和 WSDL 标准之上提供公布、搜寻和使用服务的机制，而 Web Service 的功能则是由这几个业界标准共同合作而形成的。这些技术是以阶梯方式提供的，每一个上层的机制都是由下面的标准机制支持的，如图 4－8：

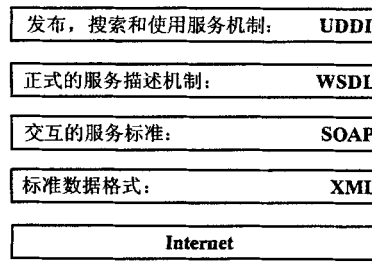


图 4－8 XML、SOAP、WSDL、UDDI 之间的关系

4.4.4.3 UDDL 的组成

UDDL 用三个不同的元素来提供对外服务的详细信息：

- 服务提供者的描述信息
  1. 提供者名称，说明信息以及联络信息等
  2. 服务分类和服务识别信息
- 服务本身的描述信息
  1. 服务名称，服务描述信息
  2. 服务细节分类信息
- 连接服务和提供实现服务程序代码的信息
  1. 实现的描述
  2. 服务存取 URL

3．服务需要的参数等

这三个元素分别分成逻辑层和实现层。在逻辑层中用户可以得知服务的意义，决定是否使用该服务，实现层则是提供服务真正实现的程序代码。当用户决定使用逻辑层中特定的服务之后，就可以依据逻辑层和实现层之间的连接找到实现服务的程序代码，并且调用和使用这些 Web Service。

图 4－9 为三个元素和 tmodel 之间的关系结构图

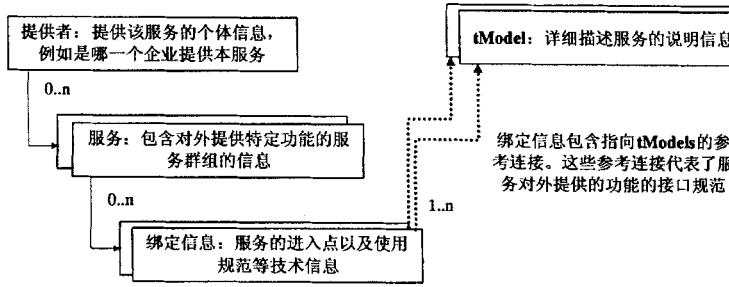


图 4－9 UDDI 中提供服务的三个元素和 tModel 之间的关系

其中 tmodel 代表了 UDDL 所提供的服务。

在 UDDI 中，tModel 是描述服务的接口，美个 tModel 会绑定一个提供实现信息的 XML 元素 businessEntity。在 businessEntity 元素中每一个 tModel 描述的服务由一个子元素 businessService 代表，而在 businessService 元素中则包含由这个 tModel 实现的 Web Service 的绑定信息 bindingTemplate，其中 businessEntity 元素相当与描述了服务的提供者，而 businessService 元素则相当与提供的各种服务，每个 businessService 元素下有针对服务的详细描述，以及实现此服务的 bindingTemplate 元素，每个 bindingTemplate 元素都指明了实现此服务的 Web Service 的 URL。此外每个 bindingTemplate 元素也会连接到它相关的 tModel 信息，而 tModel 信息则是系统对外提供描述服务的接口信息。

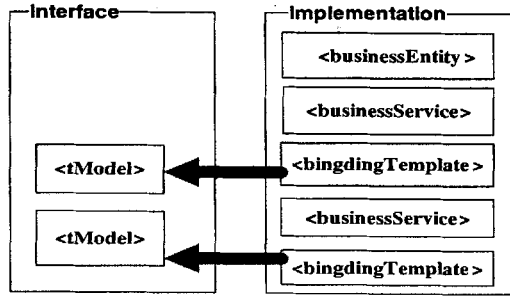


图 4－10 为 tModel 和 businessEntity 之间的交互关系。

由于 tModel 是 UDDI 中对外描述服务的接口，因此 tModel 就像 Web Service 的 WSDL 内容的前半断一样是用来描述 Web Service 提供的功能的。而 WSDL 的后半断有关 Web Service 绑定的 URL 地址和 Port 信息则可以转换为对应 tModel 的 bindingTemplate 信息。图 4－11 说明三者之间的转换流程

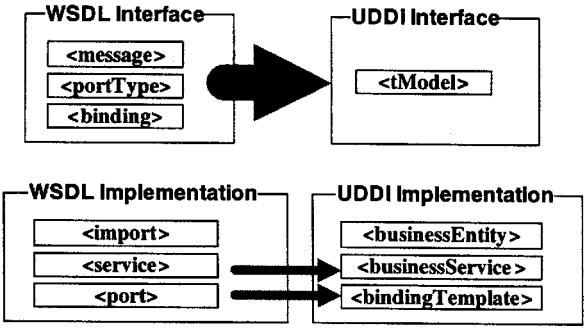


图 4－11 WDSL 和 UDDI 之间的关系

4.4.4.4 UDDI 的注册和开发

首先，可以使用各种程序语言或是开发工具来实现企业对外定义的 tModel 服务。

其次，使用 UDDI API 来注册对外支持和提供的服务信息，这是可以确定哪些服务是开放的。UDDI API 可以使用 SOAP 的 Request/Response 包来和 UDDI 注册节点交互，UDDI API 会自动地根据程序员调用的函数组成正确的 SOAP 包传送给 UDDI 注册节点。至于 UDDI 注册节点如何注册和管理、存储 Web Service 的信息则可以使用不同的技术。

最后，当各个系统注册了支持的服务以后，UDDI 注册节点的服务器便会为每一个系统的服务产生一个唯一的 GUID 数值，并且使用这个数值来绑定 tModel、服务信息以及 Web Service 的程序代码，这一步是 UDDI 注册节点服务器内部形成连接的流程。此时在 UDDI 注册节点服务器中的 tModel 信息算是完整了。此时用户便可以使用 UDDI API 来向 UDDI 注册节点服务器查询或是搜寻可以使用的适当的服务信息，并且绑定到实际的 Web Service，最后再真正的调用此 Web Service 应用程序。

目前 Web Service 技术大致上在三个领域快速的发展，SOAP 标准是属于封装的领域，WSDL<XML<tModel 是描述信息的领域，而 UDDI 则是属于搜索和

发现服务的领域。

下图 4－12 是客户通过 UDDI 搜寻各个系统的 Web Service 服务的结构示意图：

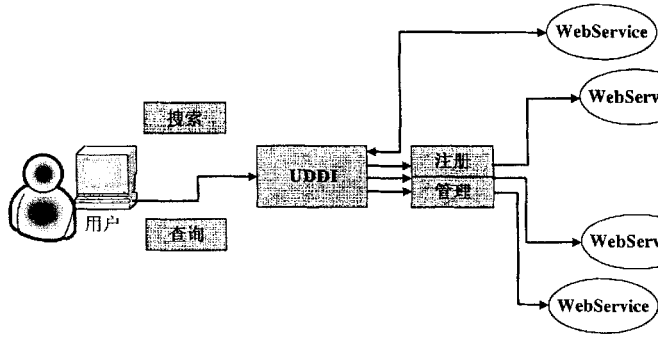


图 4－12 通过 UDDI 搜寻 Web Service 服务

4.5 YSZ2002 电力监控网管系统中使用 Web Service 技术的设计

4.5.1 YSZ2002 电力监控网管系统中 Web Service 的功能设计

YSZ2002 电力监控网管系统中的 WebService 是由远程监控模块改造产生的，通过在 WebService 中封装远程监控模块的获取数据模块，从而达到与 SCADA 通信并且向外提供数据的目的，也达到了网络管理中的监视和控制目的。

Web Service 主要是向客户端提供各种服务，其中包括身份验证，以及刷新数据的传输。Web Service 用 SOAP 协议传递用 XML 标准封装的数据给客户端的浏览器，客户端用 Javascript 解析 XML 标准封装的刷新数据，并用这些数据刷新反映工程窗口的 HTML 页面。

1. 身份验证

用户访问 Web 发布的登陆页面，输入工号、用户名、密码，调用 Web Service 提供的身份验证服务，Web Service 通过安全服务来验证用户输入的工号、用户名、密码是否正确。用返回值（用户的权限值）来控制（客户端要做一些判断）用户是否可以进入主页面（如果验证通过，允许用户进入主页面，否则提示用户没有权限）。

返回值包括：用户身份验证结果，以及不同用户对各工程或工程组的查看权限。

客户浏览器端的登陆鉴权界面：

发现服务的领域。

下图 4－12 是客户通过 UDDI 搜寻各个系统的 Web Service 服务的结构示意图：

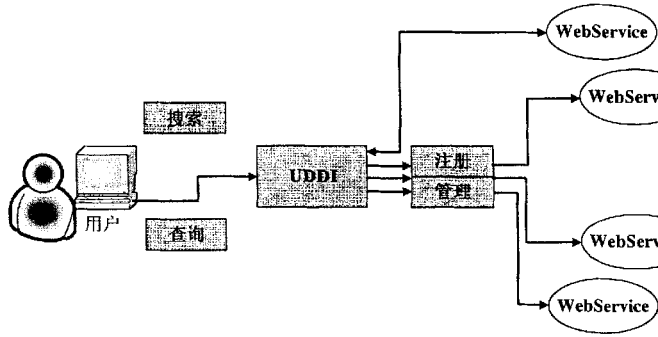


图 4－12 通过 UDDI 搜寻 Web Service 服务

4.5 YSZ2002 电力监控网管系统中使用 Web Service 技术的设计

4.5.1 YSZ2002 电力监控网管系统中 Web Service 的功能设计

YSZ2002 电力监控网管系统中的 WebService 是由远程监控模块改造产生的，通过在 WebService 中封装远程监控模块的获取数据模块，从而达到与 SCADA 通信并且向外提供数据的目的，也达到了网络管理中的监视和控制目的。

Web Service 主要是向客户端提供各种服务，其中包括身份验证，以及刷新数据的传输。Web Service 用 SOAP 协议传递用 XML 标准封装的数据给客户端的浏览器，客户端用 Javascript 解析 XML 标准封装的刷新数据，并用这些数据刷新反映工程窗口的 HTML 页面。

1. 身份验证

用户访问 Web 发布的登陆页面，输入工号、用户名、密码，调用 Web Service 提供的身份验证服务，Web Service 通过安全服务来验证用户输入的工号、用户名、密码是否正确。用返回值（用户的权限值）来控制（客户端要做一些判断）用户是否可以进入主页面（如果验证通过，允许用户进入主页面，否则提示用户没有权限）。

返回值包括：用户身份验证结果，以及不同用户对各工程或工程组的查看权限。

客户浏览器端的登陆鉴权界面：

发现服务的领域。

下图 4－12 是客户通过 UDDI 搜寻各个系统的 Web Service 服务的结构示意图：

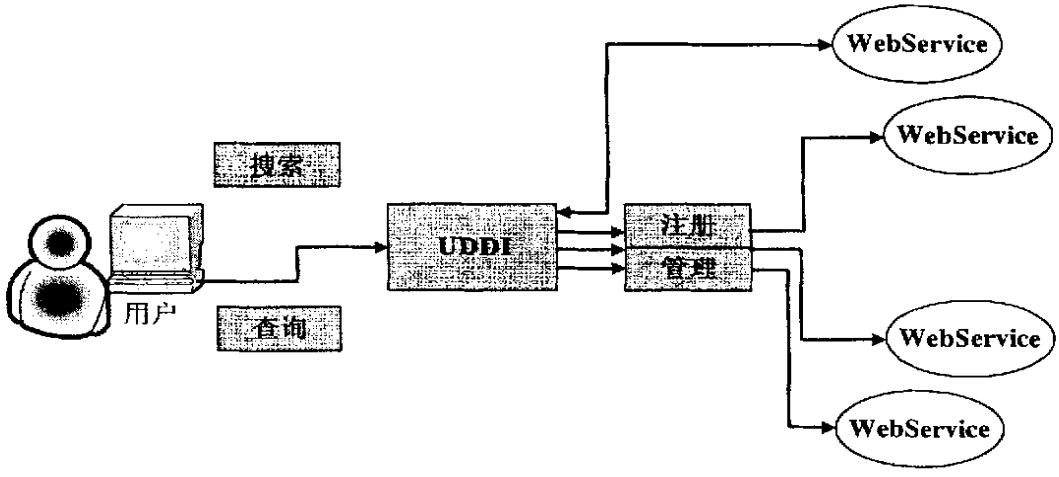


图 4－12 通过 UDDI 搜寻 Web Service 服务

4.5 YSZ2002 电力监控网管系统中使用 Web Service 技术的设计

4.5.1 YSZ2002 电力监控网管系统中 Web Service 的功能设计

YSZ2002 电力监控网管系统中的 WebService 是由远程监控模块改造产生的，通过在 WebService 中封装远程监控模块的获取数据模块，从而达到与 SCADA 通信并且向外提供数据的目的，也达到了网络管理中的监视和控制目的。

Web Service 主要是向客户端提供各种服务，其中包括身份验证，以及刷新数据的传输。Web Service 用 SOAP 协议传递用 XML 标准封装的数据给客户端的浏览器，客户端用 Javascript 解析 XML 标准封装的刷新数据，并用这些数据刷新反映工程窗口的 HTML 页面。

1. 身份验证

用户访问 Web 发布的登陆页面，输入工号、用户名、密码，调用 Web Service 提供的身份验证服务，Web Service 通过安全服务来验证用户输入的工号、用户名、密码是否正确。用返回值（用户的权限值）来控制（客户端要做一些判断）用户是否可以进入主页面（如果验证通过，允许用户进入主页面，否则提示用户没有权限）。

返回值包括：用户身份验证结果，以及不同用户对各工程或工程组的查看权限。

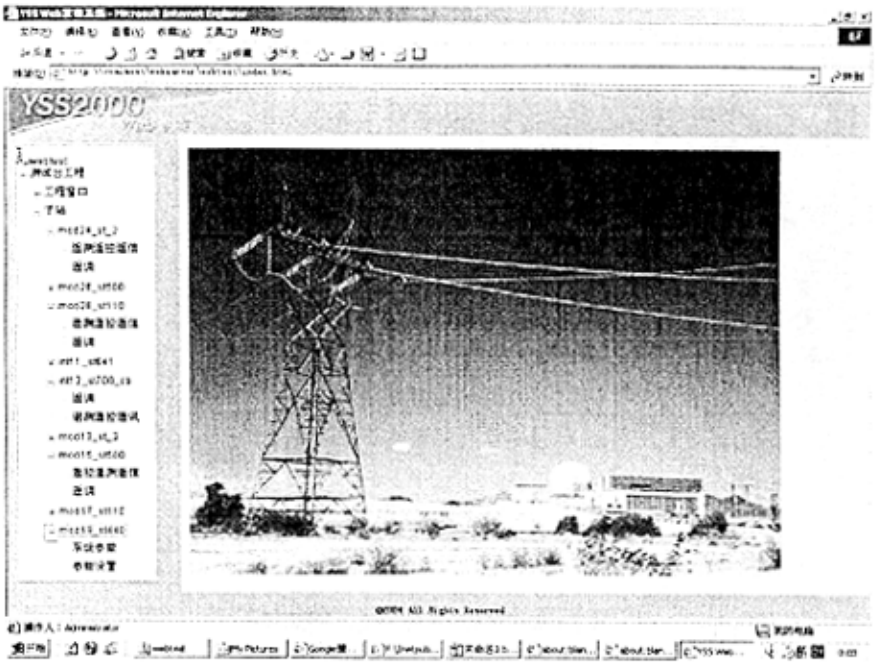
客户浏览器端的登陆鉴权界面：

2．刷新数据的传输。

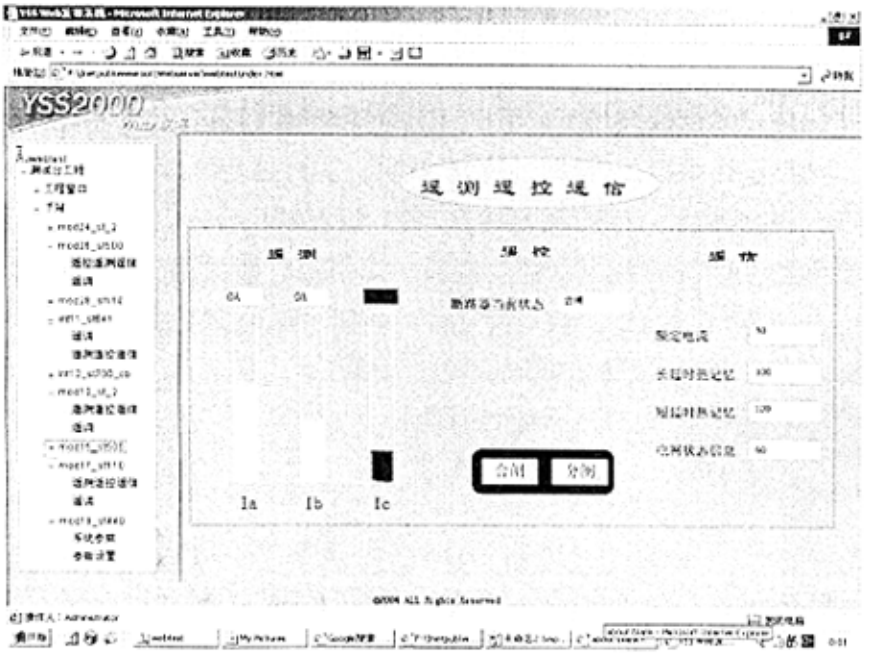
a. 用户把想要浏览的正在投运的工程的 Web 文件夹中的子文件夹 WebServer 放置到 Web 服务器上，用户根据不同的网络地址进入不同的登陆页面，再进入相应的工程（工程组）的主页面，对于主页面左边的树状图是在工程版组态的时候进行初始化的。

b. 树状图可以全部被展开，用户点击想要查看的窗口，如果用户有权限查看该窗口，则显示该窗口，否则，显示默认页。实时刷新时，调用 Web Service 的刷新接口，根据参数（工程路径和窗口名）到 WebService 文件夹去查原工程文件，看看窗口元素的哪些属性需要刷新，调用实时数据服务到具体投运工程的 SCADA 里获取实时数据，然后再转换为 XML 格式的属性数据，最后打包封装到客户端，再由客户端解析数据内容进行刷新。

客户浏览器端登陆主页面的默认页面：



用户（有相应操作权限）查看工程窗口：



此时，窗口里的数据会进行实时刷新。

4.5.2 YSZ2002 电力监控网管系统中 Web Service 的流程设计：

用户第一次请求查看工程窗口时，先要输入这个窗口所在的 IIS 服务器的地址 URL，以获取这个窗口的页面（初始的页面，客户端刷新不影响），IIS 服务器在收到用户的这个请求以后，将预先存在的 dll 文件加载到服务器端，初始

化远端的实时数据服务的客户端代码。此时客户端可以动态刷新窗口，每次动态刷新，都会向服务器端发出 SOAP 请求，要求获取远程的实时数据，此时 IIS 服务器将解析 SOAP 包的内容，调用 Web service 的服务接口，接口利用 CORBA 对象再调用远程的程序代码，最后将返回的数据用 SOAP 包的格式来封装，用户在客户端收到 SOAP 包后，对它进行解析获取自己需要的部分。

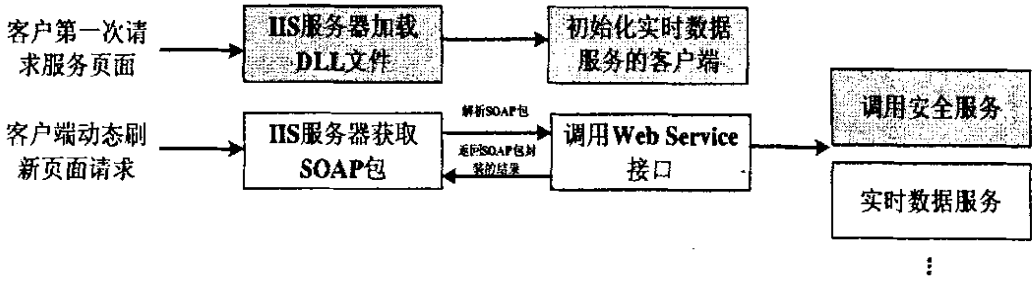


图 4－13 Web Service 的流程图

4.5.3 YSZ2002 电力监控网管系统中 Web Service 的接口设计：

1、安全服务接口 Web_LogChk（）

函数定义	AnsiString Web_LogChk(int iwUsrID, AnsiString swUsrName, AnsiString swPws)	
函数功能	根据浏览器的请求调用现有的安全服务接口，检查工号，用户名，密码是否匹配，并返回工程权限列表	
参数说明	int iwUsrID	用户工号
	AnsiString swUsrNam	用户名
	AnsiString swPws	用户密码
返回值	生成的 XML 文档。 1．对返回值格式的定义 如果是正确的用户名，口令和工号，则返回该用户对那些工程有权限及对哪些工程有什么权限。 {工程名 1#权限}{工程名 2#权限}{工程名 3#权限}。。。。。。 2．主要实现的方法是调用安全服务的接口 SFS_UsrChk。	

化远端的实时数据服务的客户端代码。此时客户端可以动态刷新窗口，每次动态刷新，都会向服务器端发出 SOAP 请求，要求获取远程的实时数据，此时 IIS 服务器将解析 SOAP 包的内容，调用 Web service 的服务接口，接口利用 CORBA 对象再调用远程的程序代码，最后将返回的数据用 SOAP 包的格式来封装，用户在客户端收到 SOAP 包后，对它进行解析获取自己需要的部分。

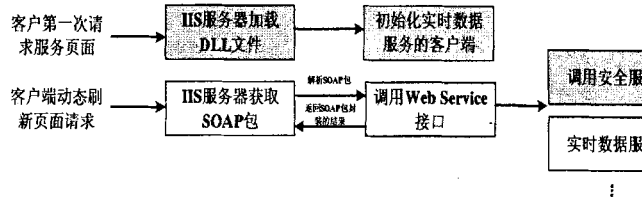
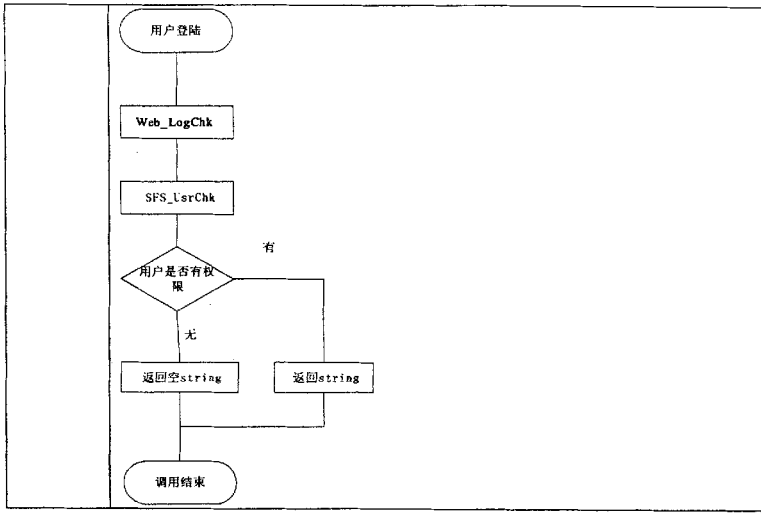


图 4－13 Web Service 的流程图

4.5.3 YSZ2002 电力监控网管系统中 Web Service 的接口设计：

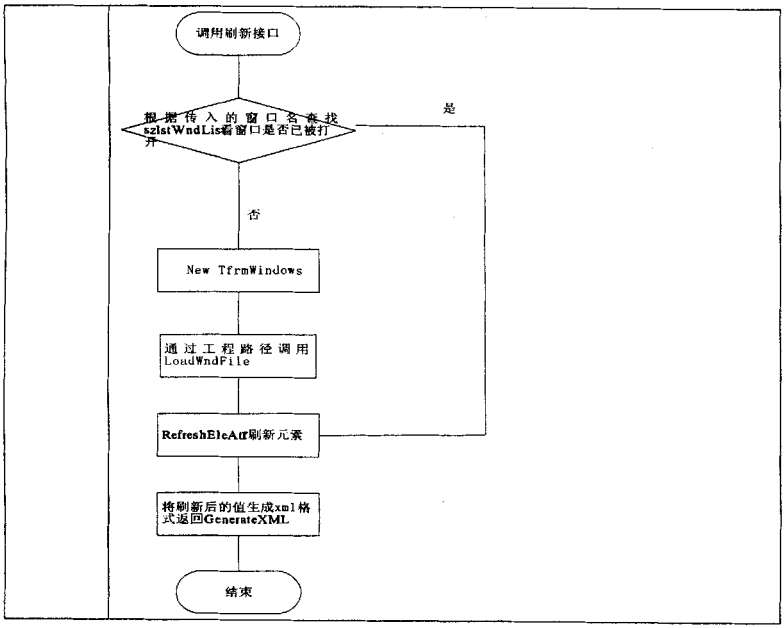
1、安全服务接口 Web_LogChk（）

函数定义	AnsiString Web_LogChk(int iwUsrID, AnsiString swUsrName, AnsiString swPws)	
函数功能	根据浏览器的请求调用现有的安全服务接口，检查工号，用户名，密码是否匹配，并返回工程权限列表	
参数说明	int iwUsrID	用户工号
	AnsiString swUsrNam	用户名
	AnsiString swPws	用户密码
返回值	生成的 XML 文档。 1．对返回值格式的定义 如果是正确的用户名，口令和工号，则返回该用户对那些工程有权限及对哪些工程有什么权限。 {工程名 1#权限}{工程名 2#权限}{工程名 3#权限}。。。。。。 2．主要实现的方法是调用安全服务的接口 SFS_UsrChk。	



2、窗口刷新接口 Web_Ref_Initial（）

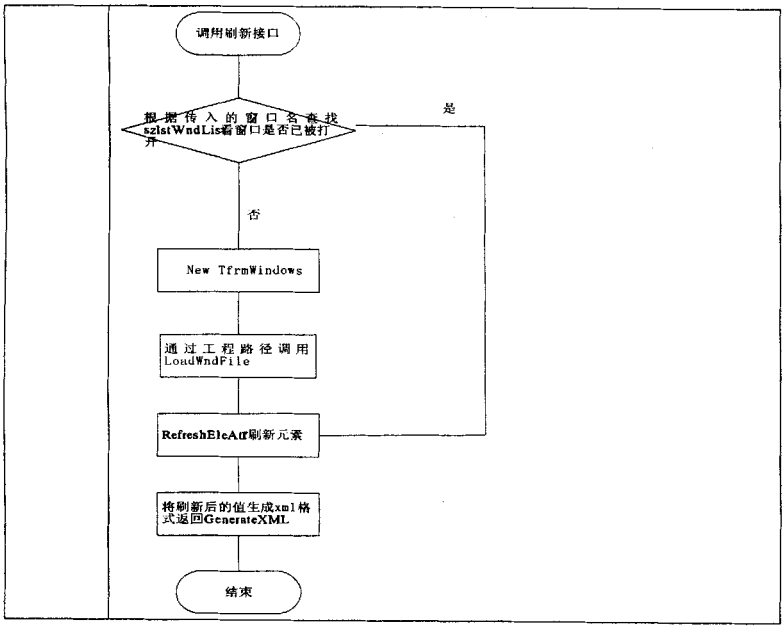
函数定义	AnsiString Web_Ref_Initial(AnsiString swProPath, AnsiString swWinName)	
函数功能	用户登录调用 Web Service 接口从 Web 服务获得实时数据。	
参数说明	名 称	含 义
	swProPath	工程路径（绝对路径）
	swWinName	窗口名（子站名^窗口名）
返回值	String	
实现思路	1. 对返回值格式的定义 XML 格式表示的属性数据以字符串的格式存放。 2. 实现过程	



注：其中对于返回的 XML 文档的定义格式，对于各观测窗口的图元、图标的属性标记（Tag）在事前已做了相应的规定。

4.6 YSZ2002 电力监控网管系统中的 Web Service 接口的实现

本文在实现 Web 服务时，用 C++Builder6 使用 Web Service 技术来实现具体的客户端与 Web 服务之间的通信。 C++Builder6 为了使用户更加方便地使用 Web Service，它提供了三个主要的控件：HTTPSoapDispatcher、HTTPSoapCppInvoker、WSDHLMLPublish。THTTPSoapDispatcher 控件主要用来分配客户端的 HTTP 请求到适当的服务器端方法来处理。THTTPSoapDispatcher 会识别 HTTP 的以 soap 为结尾的 Post 请求。当 THTTPSoapDispatcher 控件拦截到这种 HTTP 请求之后，就会调用 HTTPSoapCppInvoker 控件的 DispatchSoap 方法来处理这个 HTTP 请求。HTTPSoapCppInvoker 控件的 DispatchSoap 方法则会根据这个 HTTP 请求找到相对应的实现在 Web Service 服务器中的方法，然后调用这个方法来处理此 HTTP 请求。Web Service 服务器提出的以 wsdl 为结尾的 HTTP 请求都会由 TWSDLHTMLPublish 控件处理。而 TWSDLHTMLPublish 控件处理的方式就是返回此 Web Service 的 WSDL 文件内容，而 WSDL 内容就是



注：其中对于返回的 XML 文档的定义格式，对于各观测窗口的图元、图标的属性标记（Tag）在事前已做了相应的规定。

4.6 YSZ2002 电力监控网管系统中的 Web Service 接口的实现

本文在实现 Web 服务时，用 C++Builder6 使用 Web Service 技术来实现具体的客户端与 Web 服务之间的通信。 C++Builder6 为了使用户更加方便地使用 Web Service，它提供了三个主要的控件：HTTPSoapDispatcher、HTTPSoapCppInvoker、WSDHLMLPublish。THTTPSoapDispatcher 控件主要用来分配客户端的 HTTP 请求到适当的服务器端方法来处理。THTTPSoapDispatcher 会识别 HTTP 的以 soap 为结尾的 Post 请求。当 THTTPSoapDispatcher 控件拦截到这种 HTTP 请求之后，就会调用 HTTPSoapCppInvoker 控件的 DispatchSoap 方法来处理这个 HTTP 请求。HTTPSoapCppInvoker 控件的 DispatchSoap 方法则会根据这个 HTTP 请求找到相对应的实现在 Web Service 服务器中的方法，然后调用这个方法来处理此 HTTP 请求。Web Service 服务器提出的以 wsdl 为结尾的 HTTP 请求都会由 TWSDLHTMLPublish 控件处理。而 TWSDLHTMLPublish 控件处理的方式就是返回此 Web Service 的 WSDL 文件内容，而 WSDL 内容就是

描述此 Web Service 输出的服务，以及让客户端调用时使用的凭证。

Web Service 真正的意义是开放对外提供的服务，首先需要实现 Web Service，然后通过 WSDL 输出对外提供的服务功能。C++Builder6 在实现 Web Service 时是使用虚类机制，首先需要定义一个或是数个服务虚类，并且在虚类中定义 Web Service 提供的服务（方法），最后再使用实体类实现虚类提供的服务。使用 C++Builder6 的虚类定义 Web Service 的服务，再撰写 C++Builder6 类实现服务接口，接着必须注册服务接口信息以及实现的信息，这样就可以自动让客户端通过 Internet/Intranet 的 SOAP 数据包调用并且使用 Web Service 对外提供的服务。

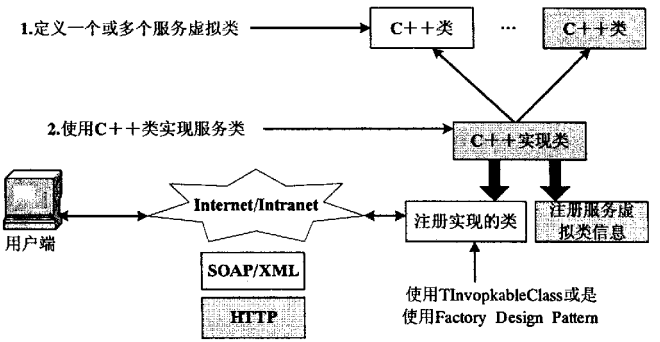


图 4－14 Web Service 应用程序的实现

A、实现 Web Service 服务接口：

本系统在生成的 TYssWebServiceImpl 类中实现安全服务接口 Web_LogChk（）和窗口刷新接 Web_Ref_Initial（）的虚函数，如下图：

```
__interface
  INTERFACE_UUID("{A1445547-1174-4609-815D-3BA2588A47D3}")
  IYssWebService : public IInvokable
{
public:
    //安全服务接口
    virtual AnsiString Web_LogChk(int iwUsrID, AnsiString swUsrName,
          AnsiString swPws)=0;
    //页面第一次刷新时的接口
    virtual AnsiString Web_Ref_Initial(AnsiString swProPath, AnsiString
          swWinName)=0;
```

```
};
typedef DelphiInterface<IYssWebService> _di_IYssWebService;
```

在 TYssWebServiceImpl 类的 Cpp 文件中添加两个接口的具体实现方法。

```
class TYssWebServiceImpl : public TInvokableClass, public IYssWebService
{
private:
    TfrmWindow *frmCurrentForm;
public:HRESULT STDMETHODCALLTYPE QueryInterface(const GUID&
IID, void **Obj){ return GetInterface(IID, Obj) ? S_OK : E_NOINTERFACE; }
ULONG STDMETHODCALLTYPE AddRef() {return TInterfacedObject::
_AddRef(); }
ULONG STDMETHODCALLTYPE Release() {return TInterfacedObject::
 _Release(); }
  virtual AnsiString Web_LogChk(int iwUsrID, AnsiString swUsrName,
  AnsiString swPws);
  virtual AnsiString Web_Ref_Initial(AnsiString swProPath, AnsiString
  swWinName);
  TfrmWindow *frmGetForm(AnsiString swProPath, AnsiString swWinName);
  void checkValid() { delete new TYssWebServiceImpl(); }
};
```

安全服务的接口实现：

```
AnsiString TYssWebServiceImpl::Web_LogChk(int iwUsrID, AnsiString
            swUsrName, AnsiString swPws)
{
   AnsiString ansiResult = SYSTEM->UserCheck
   (iwUsrID ,swUsrName ,swPws);
    return ansiResult;
}
```

刷新窗口的接口实现：

```
AnsiString TYssWebServiceImpl::Web_Ref_Initial(AnsiString swProPath,
            AnsiString swWinName)
{
        int iSharpPos = swWinName.AnsiPos("^");
        int iansiLength = swWinName.Length();
        AnsiString ansiStaName = swWinName.SubString(1,iSharpPos-1);
```

```
    SYSTEM->szCurrentStaName = ansiStaName;
    SYSTEM->szCurrentProjName = swProPath;
    frmCurrentForm = frmGetForm(swProPath, swWinName);
    frmCurrentForm->szPrjLogicName = swProPath;
    frmCurrentForm->RefreshEleAttr(1);
    AnsiString ansiXMLBack = frmCurrentForm->GenerateXML(1);
    return ansiXMLBack;
}
```

在 TYssWebServiceImpl 类的实现程序代码之后，产生了下面的注册程序代码：

```
static void RegTypes()
{
  InvRegistry()->RegisterInterface(__interfaceTypeinfo(IYssWebService));
  InvRegistry()->RegisterInvokableClass(__classid(TYssWebServiceImpl),
    YssWebServiceFactory);
}
```

这些程序注册代码的目的是，当 Web Service 服务器被激活以服务客户端的需求时，在内存中建立和产生 Web Service 的服务程序码表格以及 Web Service 服务接口表格，以便让客户端传来的 SOAP 数据包能够找到正确的 Web Service 服务，并且执行正确的 Web Service 实现程序代码。上面的程序代码中，RegisterInterface 方法会注册 Web Service 服务接口，而 RegisterInvokableClass 方法则会注册 Web Service 的实现程序代码。

从 WSDL 的内容来看，其中定义了 Port Name 以及存取的 URL 地址，客户端需要这些信息才可以绑定 Web Service。

B、实现客户端应用程序

客户端程序使用 THHTTPRIO 控件，他可以使用静态的或者动态的绑定方式实现调用 Web Service。具体的方法是：从 Web Service 中取得 WSDL 的内容，解析其中的信息，分析出 Web Service 提供的服务，并且让用户使用这些服务。其中，WSDL 的 URL 的具体格式为：

http://WebService 地址/Web Sevice 应用程序/wsdl/接口名称

客户端可以使用两种方法来取得远程 Web Service 的服务接口：

第一是调用 THHTTPRIO 的 QueryInterface 方法，另外一个就是使用 WSDL Import 向导产生的 cpp 文件中的程序代码。通常 WSDL Import 向导会在 cpp 文

件中产生一个“GetWebService 服务接口名称”的方法，可以调用远程的 Web Service 的服务接口。当客户端应用程序发出具有 SOAP 的 PathInfo 的 HTTP 请求后，便会由 Web Service 中的 THTTPSoapDispatcher 控件处理。而 THTTPSoapDispatcher 控件便会根据 HTTP 请求的内容调用指定的 Web Service 服务，最后再把结果使用 SOAP 的数据包返回给客户端。

在本系统中，客户端的程序是通过浏览器来实现的，在浏览器中使用 XMLHTTP 控件，通过调用 XMLHTTP 控件的一些方法来发出 SOAP 请求，根据发出的 SOAP 请求，可以知道客户想要查看的工程窗口是哪一个，通过服务器端的程序的调用，会返回给客户端一个 SOAP 包，它的格式也是 XML 文档格式，SOAP 包的 XML 文档的<return>标签中的是服务器端返回的数据的 XML 文档，里面有所有需要刷新的元素的 ID、属性、值，浏览器通过自带的 DOM 来解析 SOAP 返回的 XML 文档，实际上解析出来是个树状图，在从中获取数据的属性值，数据的 XML 格式是通过事先的客户端与服务器端的约定来定义的，这使客户端根据约定就可以找到需要的相应数据，来刷新 SVG 格式的图形，最后将它嵌入到 HTML 中形成刷新页面。以下是客户端的一段发送 SOAP 请求和接受 SOAP 包，以及解析出其中的数据的 XML 代码：

```
XmlSoapEnvelope.loadXML('<?xmlversion="1.0"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Bod
y
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><NS1:Web
_Refxmlns:NS1="urn:YssWebService-IYssWebService"><swProPathxsi:type="xsd:st
ring">treeview</swProPath><swWinName xsi:type="xsd:string">工程窗口
^Win_1</swWinName></NS1:Web_Ref></SOAP-ENV:Body></SOAP-ENV:Envelo
pe>');
    XmlHttp.open("POST","http://localhost/CGITest/YssWeb.exe/soap/IYssWebSer
vice", false);
XmlHttp.setRequestHeader("Cookie", "LastProgID=ServerInfo.ServerInfo");
XmlHttp.setRequestHeader("Cache-Control", "no-cache");
XmlHttp.setRequestHeader("Connection", "Keep-Alive");
XmlHttp.setRequestHeader("Host", "localhost/CGITest");
XmlHttp.setRequestHeader("Accept", "text/xml");
XmlHttp.setRequestHeader("SOAPAction",'"urn:YssWebService-IYssWebService#W
```

```
eb_Ref");
XmlHttp.setRequestHeader("User-Agent", "Borland SOAP 1.1");
XmlHttp.setRequestHeader("Content-Type", "text/xml");
XmlHttp.send(XmlSoapEnvelope);
var objDispatch = XmlHttp.responseXML;
var node = objDispatch.selectSingleNode("//return");
```

C、总体的流程

本系统在使用 C++Builder6 创建 Web Service 应用服务接口的相关步骤后，将产生 Web Module 和 YssWebService 文件，将这些文件和远程监控模块所联系的图元、图标文件放在一个 YssWeb.dll 文件中去，dll 文件一般存放在 IIS 服务器上(Microsoft IIS 是允许在公共 Intranet 或 Internet 上发布信息的 Web 服务器。Internet Information Server 通过使用超文本传输协议(HTTP)传输信息)，当客户发出管理请求后，客户端程序将管理请求的内容用 SOAP 的标准封装成 XML 的格式，使用 HTTP 通信传输协议，传输到 Iis 服务器上，IIS 服务器上集成的 dll 文件中的 Web Module 文件里的 THTTPSoapDispatcher 控件将自动的接受以 soap 为结尾的 Post 请求，并调用 HTTPSoapCppInvoker 控件的 DispatchSoap 方法来处理这个管理请求，根据这个管理请求找到相对应的实现在 Web Service 服务器中的方法，或许是安全服务接口或是刷新接口等等，然后调用这个方法来处理此网络管理请求。在这些方法中可能会查询到远程数据库的一些信息，可能会调用 CORBA 中的一些对象，最后将查询和控制的结果，用预先定义好的 XML 格式用 SOAP 包封装起来返回给客户端。客户端收到 SOAP 包后按事先定义好的 XML 格式对 SOAP 包的内容进行解析，以得到相应的服务器端的返回信息。

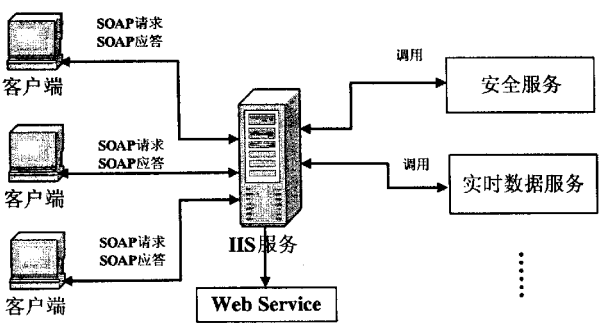


图 4—15 Web Service 的总体流程示意图

第五章 基于 Web 的网络管理中的安全管理

5.1 基于 Web 的网络管理中的安全管理的概述

由于 WBM 使用了 Internet 的 Web 技术，显然安全性成为必须考虑的问题。特别是管理网络中的重要设备时，安全性显得尤为突出。如果仅仅是系统内部使用，而不与外部网络构成连接,管理网络可能会安全一些，但这势必降低了 WBM 的管理功能。未来网管的目标是能够在任意地点、任何时间都能够有效管理网络资源和系统，所以一般应允许管理人员从外部也能接入网络。那么为了免受黑客的攻击或他人盗取管理信息，安全成为首要考虑的问题。为了保证网络管理系统的安全，可以采用以下措施：a 接入控制；b 鉴权；c 加密；d 连接控制；e 数据私有化；f 审核站点。这些是安全所考虑的一般方法。当然，针对不同的网络，还可以有不同的安全策略，还可以使用防火墙技术来增强网络的安全性。

WBM 的安全问题在众多网络安全问题中是首要的。通常一个安全网络需要使用防火墙与 Internet 隔离，以保护资源，防止外部 Internet 中非授权的访问。为了提高 Internet 的安全性，服务器访问必须通过口令和访问地址过滤等手段来加强控制。在一个网络中，WBM 控制着关键资源，它被严格要求只有授权的用户才能访问。Web 服务器可以通过管理人员设置访问授权，来实现对关键资源的访问。WBM 的安全技术与现存的操作系统实施的安全方法并不冲突，管理人员能够对 WBM 系统应用复杂的权限技术。WBM 也能充分利用网络安全成熟的技术如数据加密和认证来保护内部数据，保护浏览器到服务器的通信。

基于 Web 网络管理的安全性主要有两个方面来构成：

- 数据传输的安全,防止被第三方看到或恶意修改。
- 被管理资源信息的安全,即只有经过授权和通过认证的用户才能看到和修改信息。

5.2 基于 Web 的安全管理的设计

5.2.1YSZ2002 电力监控系统中的安全管理的设计

在一个网络中,WBM 控制着关键资源,它被严格要求只有授权的用户才能访问。在 YSZ2002 电力监控系统中，对于特定工程和工程组只有授权用户才可以访问。在网络管理中的安全管理在 YSZ2002 电力监控系统中是由安全服务来完成的。结合电力监控系统的特点，本系统的安全服务提供了：用户授权服务、用户鉴权服务、安全日志和用户权限表的维护服务、安全日志和用户权限表的查看

第五章 基于 Web 的网络管理中的安全管理

5.1 基于 Web 的网络管理中的安全管理的概述

由于 WBM 使用了 Internet 的 Web 技术，显然安全性成为必须考虑的问题。特别是管理网络中的重要设备时，安全性显得尤为突出。如果仅仅是系统内部使用，而不与外部网络构成连接,管理网络可能会安全一些，但这势必降低了 WBM 的管理功能。未来网管的目标是能够在任意地点、任何时间都能够有效管理网络资源和系统，所以一般应允许管理人员从外部也能接入网络。那么为了免受黑客的攻击或他人盗取管理信息，安全成为首要考虑的问题。为了保证网络管理系统的安全，可以采用以下措施：a 接入控制；b 鉴权；c 加密；d 连接控制；e 数据私有化；f 审核站点。这些是安全所考虑的一般方法。当然，针对不同的网络，还可以有不同的安全策略，还可以使用防火墙技术来增强网络的安全性。

WBM 的安全问题在众多网络安全问题中是首要的。通常一个安全网络需要使用防火墙与 Internet 隔离，以保护资源，防止外部 Internet 中非授权的访问。为了提高 Internet 的安全性，服务器访问必须通过口令和访问地址过滤等手段来加强控制。在一个网络中，WBM 控制着关键资源，它被严格要求只有授权的用户才能访问。Web 服务器可以通过管理人员设置访问授权，来实现对关键资源的访问。WBM 的安全技术与现存的操作系统实施的安全方法并不冲突，管理人员能够对 WBM 系统应用复杂的权限技术。WBM 也能充分利用网络安全成熟的技术如数据加密和认证来保护内部数据，保护浏览器到服务器的通信。

基于 Web 网络管理的安全性主要有两个方面来构成：

- 数据传输的安全,防止被第三方看到或恶意修改。
- 被管理资源信息的安全,即只有经过授权和通过认证的用户才能看到和修改信息。

5.2 基于 Web 的安全管理的设计

5.2.1YSZ2002 电力监控系统中的安全管理的设计

在一个网络中,WBM 控制着关键资源,它被严格要求只有授权的用户才能访问。在 YSZ2002 电力监控系统中，对于特定工程和工程组只有授权用户才可以访问。在网络管理中的安全管理在 YSZ2002 电力监控系统中是由安全服务来完成的。结合电力监控系统的特点，本系统的安全服务提供了：用户授权服务、用户鉴权服务、安全日志和用户权限表的维护服务、安全日志和用户权限表的查看

服务。

对其中的关键性名词做一下解释：

用户权限：按照系统提出要求，共有 18 个不同权限，限制用户针对指定工程可进行的操作。如，对数据库管理的操作、遥控操作、遥调操作、报警阀值修改、用电管理的操作等等。

安全日志：用来记录网络上用户登录退出工程的简单信息和用户对工程的操作记录。

用户类型：系统用户、工程组用户、工程用户。系统用户对所有工程组里工程都可以操作，工程组用户只能操作一个工程组里的工程，工程用户只能操作一个工程。

用户交班：一个管理者离开操作台时，并不关闭系统而是注销自己的登录。

用户接班：一个管理者来到已运行的操作台时，要重新登录。

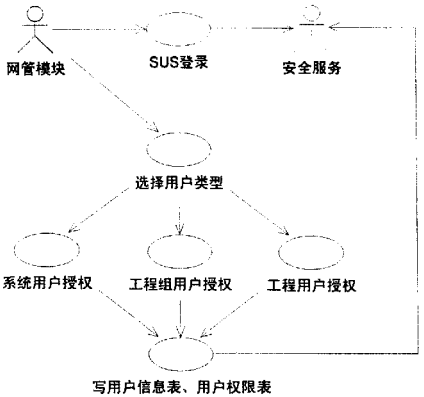
- 用户授权

SUS（系统超级用户）通过网管模块使用安全服务，给系统的新用户授权，授权的内容包括：用户名、工号、密码、用户类型以及对可以操作的工程的工程权限。网管模块将授权结果作为参数传递到安全服务，安全服务对授权结果进行分类保存，或放入用户权限表（存放用户及其相应的工程权限）里或放入用户信息表里（存放用户名、工号、密码、用户类型）。

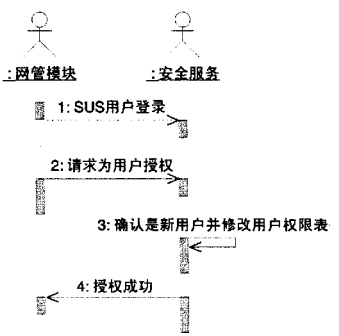
用户授权过程中工程的选择是 SUS 用户填写的工程逻辑名，对于其指定工程的存在与否暂时不做验证，只要保证将授权结果保存到用户信息表和用户权限表中就可以了。

用户授权过程中的各个用户角色与权限的对应关系由安全服务保存，角色的记录情况同用户的记录情况相同。角色就是对一种用户权限的集中定义，以后有相同权限定义的用户可以由角色来授权，用户的权限可以通过角色来授权给用户也可以直接授权给用户。

用户授权的 Usecase 图：



用户授权的流程图：



- 用户鉴权

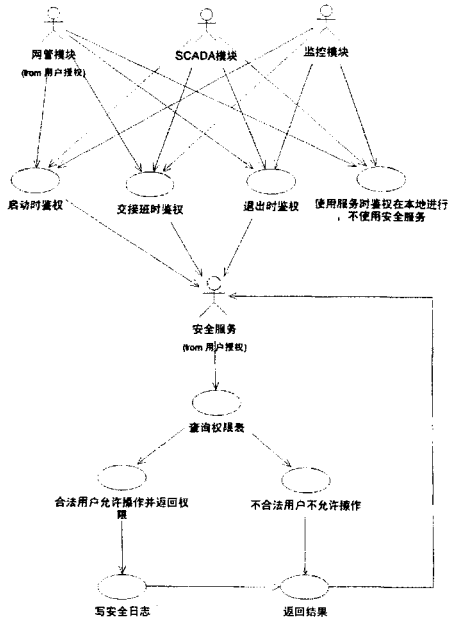
任何模块的投运和用户接班之前都必须使用安全服务进行用户鉴权，只有通过鉴权的合法用户才可以启动和登录模块。当鉴权结束后，安全服务主动将启动模块的动作写入安全日志，之后将该用户的所有权限发送到客户端，用作本地鉴权，以后用户对工程的某项操作是否合法就由下载到本地的权限来鉴别，不用在去安全服务鉴别了。

对于 SCADA 模块，用户投运工程和接班的时候鉴权针对指定工程，当用户拥有对工程投运的权限时，才将相应的工程权限作为返回值，并且填写安全日志

和投运工程用户表。用户退出和交班时鉴权，只是写安全日志和修改投运工程用户表，返回值为空。

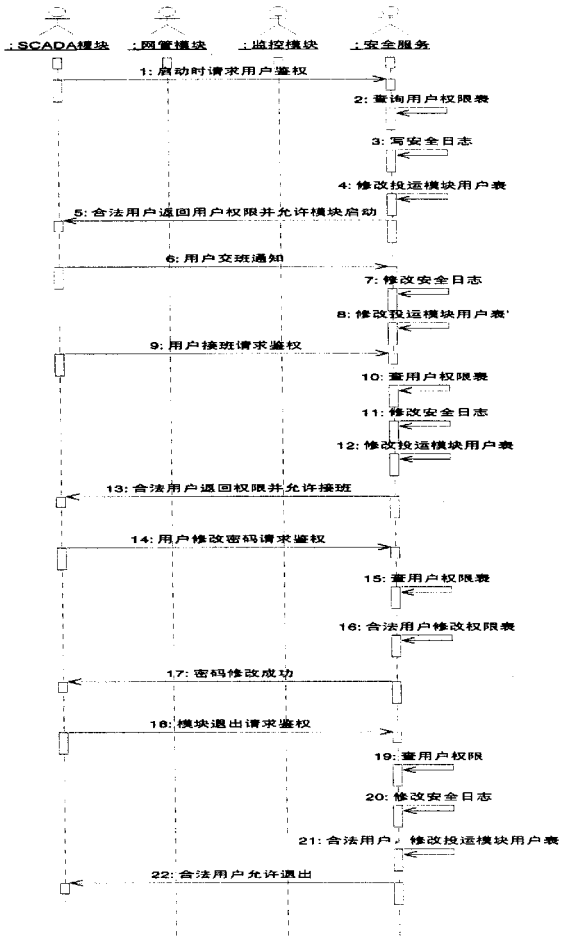
对于网管模块和监控管理模块的投运和接班鉴权，结果是合法用户可以登录，返回值为该用户所有的工程权限。退出和交班鉴权，只是写安全日志和修改投运工程用户表，返回值为空。

用户鉴权的 Usercase 图：



注：鉴权操作返回的用户权限结构对用户是透明的。用户将权限保留在本地，对工程进行操作前，所需要的鉴权在本地进行即可。

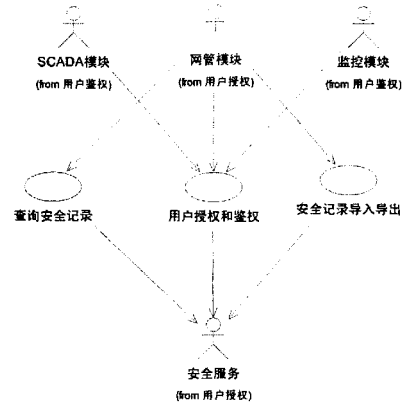
用户鉴权的流程图：



- 系统安全日志

模块投运、用户交接班、退出时都要写安全日志。安全日志在运行安全服务的主机保存，SUS 用户可以进行导入和导出和查看。

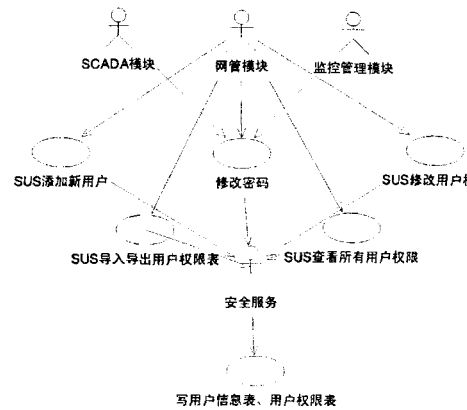
Usercase 图：



- 安全策略维护

SUS 用户可以通过网管模块对新用户进行授权，对已有用户权限进行更改，授权结果和更改结果将写入安全策略维护的用户信息表和用户权限表，该表保存在运行安全服务的主机上。同时，SUS 拥有通过网管模块对权限表进行导入导出和查看的权利。

Usercase 图：



5.2.2 基于 Web 的 YSZ2002 电力监控系统中的安全管理的设计

在基于 Web 的 YSZ2002 电力监控系统中的安全管理中，安全管理的设计主要分为两个部分：对浏览器用户的授权和鉴权，可以访问网管系统中被管设备的某些资源；对在 internet 上传输数据的加密。

5.2.2.1 对浏览器用户的授权和鉴权

基于 Web 的 YSZ2002 电力监控系统的安全管理所使用的安全服务是在原有的系统中增加了 Web 实现的部分。安全服务是直接为网管模块服务的，网管模块使用安全服务进行用户管理，下面就是它受到 Web 客户端调用的系统结构图。

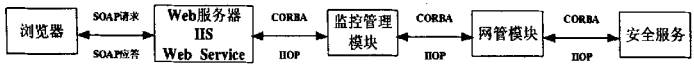


图 5－1　Web 服务器受 Web 客户端调用的系统结构图

- 浏览器用户授权

用户在浏览器端输入网管模块的地址，此时 web 服务器会给客户端传递一个网管模块的网页，系统超级用户（SUS）可以在网管模块的页面配置新的授权用户，并将授权的请求用 SOAP 包封装通过 Web service 接口调用监控管理模块的接口，再通过本地 CORBA 的 ORB 软总线机制通过本地网管模块使用安全服务，来确定系统的新用户授权。网管模块将授权结果作为参数传递到安全服务，安全服务对授权结果进行分类保存，或放入用户权限表里或放入用户信息表里。

这一部分的设计流程与原有的 YSZ2002 电力监控系统用户授权保持一致，仅在与监控管理模块交互时加入了 Web 服务器和浏览器客户端。具体的通信过程在第四章有详细的说明。

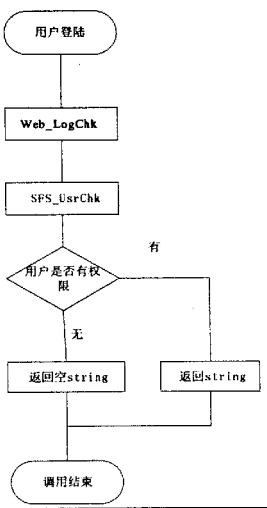
- 浏览器用户鉴权

浏览器用户想要对某个工程进行操作，首先要输入这个工程的网址，IIS 服务器将登陆页面传递给客户端，用户输入工号，密码和户用名，在客户端将这些信息用 SOAP 封装传递给 IIS 服务器的 Web service 接口，在 Web 服务器端调用监控管理模块的鉴权接口，再通过本地 CORBA 的 ORB 软总监机制通过本地网管模块使用安全服务进行用户鉴权，当鉴权结束后，安全服务主动将客户端用户的登陆写入安全日志，之后将该用户的所有权限发送到 IIS 服务器的 Web service 接口处，再由它通过 SOAP 协议发送给客户端，用作本地鉴权，以后用户对工程的某项操作是否合法就由下载到本地的权限来鉴别，不用再去安全服务鉴别了。只有通过鉴权的合法用户才可以向 IIS 服务器申请登陆到工程的操作页面。

这一部分的设计流程与原有的 YSZ2002 电力监控系统用户鉴权保持一致，

具体的通信过程在第四章有详细的说明。由于开发周期的原因，本系统只实现了鉴权部分。

Web Service 鉴权接口 Web_LogChk（）的定义：

<table>
<tr><td>函数定义</td><td colspan="2">AnsiString Web_LogChk(int iwUsrID, AnsiString swUsrName, AnsiString swPws)</td></tr>
<tr><td>函数功能</td><td colspan="2">根据浏览器的请求调用现有的安全服务接口，检查工号，用户名，密码是否匹配，并返回工程权限列表</td></tr>
<tr><td>参数说明</td><td>int iwUsrID</td><td>用户工号</td></tr>
<tr><td></td><td>AnsiString swUsrNam</td><td>用户名</td></tr>
<tr><td></td><td>AnsiString swPws</td><td>用户密码</td></tr>
<tr><td>返回值</td><td colspan="2">生成的 XML 文档。
3．对返回值格式的定义
如果是正确的用户名，口令和工号，则返回该用户对那些工程有权限及对哪些工程有什么权限。
{工程名 1#权限}{工程名 2#权限}{工程名 3#权限}。。。。。。。。。。
4．主要实现的方法是调用安全服务的接口 SFS_UsrChk。
 
</td></tr>
</table>

鉴权接口的实现：

AnsiString TYssWebServiceImpl::Web_LogChk(int iwUsrID, AnsiString swUsrName, AnsiString swPws)

{

```
    AnsiString
    ansiResult=SYSTEM->UserCheck(iwUsrID,swUsrName,swPws);
    return ansiResult;
}
```

注：UserCheck(iwUsrID,swUsrName,swPws)为监控管理模块调用安全服务的接口，通过这个函数调用安全服务的 CORBA 对象，才能调用安全服务的鉴权函数完成对客户端用户登陆的鉴权工作。

客户端用户登陆时的鉴权请求：

```
XmlHttp = new ActiveXObject("Msxml2.XMLHTTP.4.0");
XmlSoapEnvelope = new ActiveXObject("Microsoft.XMLDOM");
varenvlp='<?xmlversion="1.0"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Bod
ySOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><NS1:We
b_LogChkxmlns:NS1="urn:YssWebService-IYssWebService"><iwUsrIDxsi:type="x
sd:string">';
envlp=envlp+form1.txtUserNum.value+'</iwUsrID><swUsrName
xsi:type="xsd:string">';
envlp=envlp+form1.txtUserName.value+'</swUsrName><swPws
xsi:type="xsd:string">';
envlp=envlp+form1.txtPwd.value+'</swPws></NS1:Web_LogChk></SOAP-ENV:B
ody></SOAP-ENV:Envelope>';
XmlSoapEnvelope.loadXML(envlp);
XmlHttp.open("POST",      "http://localhost/isapi/YssWeb.dll/soap/IYssWebService",
false);
XmlHttp.setRequestHeader("Cookie", "LastProgID=ServerInfo.ServerInfo");
XmlHttp.setRequestHeader("Cache-Control", "no-cache");
XmlHttp.setRequestHeader("Connection", "Keep-Alive");
XmlHttp.setRequestHeader("Host", "localhost/isapi");
XmlHttp.setRequestHeader("Accept", "text/xml");
XmlHttp.setRequestHeader("SOAPAction",
'"urn:YssWebService-IYssWebService#Web_LogChk"');
XmlHttp.setRequestHeader("User-Agent", "Borland SOAP 1.1");
```

```
XmlHttp.setRequestHeader("Content-Type", "text/xml");
var objDispatch = XmlHttp.responseXML;
var node = objDispatch.selectSingleNode("//return");
```

5.2.2.2 数据在 Internet 上传输时的加密

管理者对被管设备进行管理时，形成管理域，我们将在局域网内形成的管理域定义为“安全管理域”，而跨越 Internet 进行管理的区域定义为“不安全管理域”，如图 5－2。当管理者在“不安全管理域”进行网络管理时，被管设备就可能面临着管理者(Manager)身份伪装、管理内容篡改、管理消息流顺序改变等安全威胁。为此，需要有一种机制来提供管理者身份验证、管理内容完整性校验以及该管理者是否具有管理设备的有效权限等方面的功能。对于管理者身份验证和管理者管理设备的有效权限方面的内容上一节 5.2.2.1 中的对浏览器用户的授权、鉴权中已完成，对于传输数据的机密性、正确性、完整性将是这一章研究的重点。SSL 协议保证数据在网络层及会话层的传输安全性；本网络管理系统在 internet 上传输的用 SOAP 协议封装的 XML 格式的数据，所以对 XML 格式的数据进行加密和签名可以保证传输数据的安全性；基于上述两种安全策略进而使用 SOAP 协议的安全封装规范 WS－Security，来到达更加安全的数据传输；在本小节的最后给出了基于 Web 通信的综合安全模型。

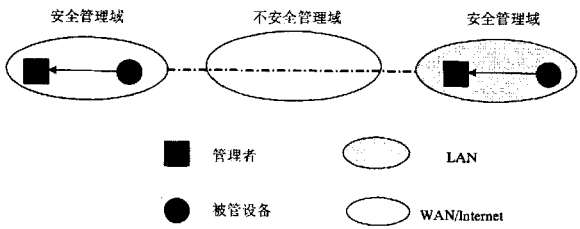


图 5－2 安全管理域与不安全管理域

- 使用 SSL 协议保证数据传输的安全性

由于 Internet 上有时要传输重要或敏感的数据，由 Netscape 公司开发的安全套接层（Secure Socket Layer，SSL）是一种保证网络安全的通信协议也可以说是一种在 Internet 基础上提供的一种保证私密性的安全协议，现在已经成为在 Internet 上广泛使用的客户机与服务器之间身份鉴别和加密通信的标准方式。SSL 采用公开密钥技术，其目标是保证两个应用间通信数据的机密性和完整性，可在服务器和客户机两端同时实现对 SSL 协议的支持。目前使用的 Web 浏览器普遍将 HTTP 和 SSL 协议相结合，从而实现安全的数据通信。SSL 安全协议主要提供三方面的服务：

1)认证用户和服务器,使得它们能够确信数据将被发送到正确的客户机和服务器上;

2)加密数据,隐藏被传送的数据;

3)维护数据的完整性,确保数据在传输过程中不被改变。

SSL 协议可以使客户服务器应用之间的通信不被攻击者窃听,并且始终对服务器进行认证,还可选择对客户进行认证。SSL 是工作在网络层与会话层之间的协议,它在 TCP/IP 和 HTTP 之间增加了一个加密层,即在 TCP 之上建立了一个加密通道,通过这一层的数据经过了加密,建立用户与服务器之间的加密通信,确保信息传递的安全性。但对于工作在 HTTP 协议以上的用户而言,加密是完全透明的,SSL 协议在应用层协议通信之前就已经完成加密算法、通信密钥的协商以及服务器的认证工作。在此之后应用层协议所传送的数据都会被加密,从而保证通信的私密性。

SSL 的通信过程分为两个阶段,即 SSL 握手阶段和 SSL 会话阶段。在 SSL 握手阶段,通过基于非对称加密算法和消息摘要算法的数字证书,鉴别服务器端进程和客户端进程的身份,并且交换通信密钥;在 SSL 会话阶段,客户机和服务器之间的通信数据全部通过对称加密算法(如 DES)加密,而密钥已经在 SSL 握手阶段协商好了,仅有通信双方知道。由此可见,SSL 可以很好的保证通信数据的机密性、正确性和完整性。

要实现 SSL 的应用,最复杂的就是对证书的颁发和管理。我们的网络管理系统需要维护一个 CA 服务器来颁发证书(先将它置于 Web 服务器上)。对于浏览器客户端,通过将系统 CA 加入到客户机浏览器信任的 CA 列表,并将由系统 CA 签发的数字证书安装到 Web 服务器上(可以鉴别服务器端进程和客户端进程的身份),就可以通过 HTTPS(即安全 HTTP 协议)进行安全的数据通信。

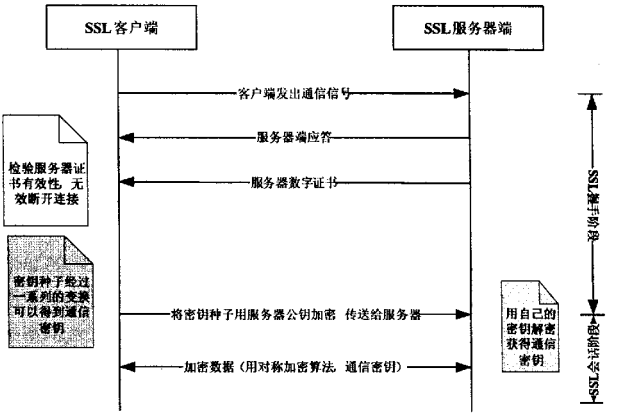


图 5－3 SSL 的通信流程

- 对 XML 格式的数据加密和签名来保证传输数据的安全性

通常为了建立基本的信任关系和安全会话，数据在交流过程中需要满足五个基本的安全性因素：①机密性(Confidentiality)，对私有和机密数据进行加密；②完整性(Integrity)，保证数据在传输过程中没有被偶然或恶意篡改；③身份验证(Authentication)，通过公共密钥体系和数字签名来确定通信实体身份；④不可否认性(Nonrepudiation),通过数字签名公钥密制保证信息发送者不能抵赖或否认对信息的发送；⑤授权(Authorizaion)，给实体分配资源请求的不同权限。关于权限的授权、鉴权已经在 5.2.2.1 节中作了详细的讲解，下面主要对在 Internet 中数据传输安全性的前 4 个因素作出相应的处理。

XML 数字签名和加密,将安全机制集成到 XML 文档中,从而能够保障传输数据的安全。首先，它通过对消息数据部分运用一个安全散列(securehash)的数学算法产生消息的“摘要”，然后将摘要和所有要被签名的附加信息再摘要一次，再加密，并把结果作为数字签名写入 XML 消息本身中的<SignatureValue>元素中。加密算法和解密密钥都置于<SignedInfo>中。XML 签名模板如下所示。这样,收件人可以验证收到的消息与发件人原本发送的消息是否相同,保证消息的完整和不可否认性。

```
<Signature Type = “URL”>
<SignedInfo>
< Signature   Method Algorihtm =…algorithm parameters…/>
<KeyValue> …public key value…</ KeyValue >
</SignedInfo>
< Signature Value > …signature generated…< /Signature Value >
```

<Object>...data used for signature...</Object>

</Signature>

将数字签名嵌入 XML 文档之后，下一步就是要加密文档，来保证数据传输的机密性。XML 加密通过进一步加密已被签名的消息来扩展 XML 数字签名系统的能力，它允许对全体 XML 消息、部分 XML 消息或者已加密部分的 XML 消息进行加密。XML 加密的基本实体是<EncryptedData>元素，它的 type 属性标志加密节点的类型。加密节点进行编码并把结果值置于<EncryptedData>中。加密运算法则在<DecryptionInfo>中标志出来。下面是 XML 加密模板。

< EncryptedData Type = "URL">

<DecryptionInfo>

< Method Algorihtm URL ="URL"/>...algorithm parameters

</ Method >

</DecryptionInfo>

<CipherData>

< CipherValue > ...signature generated...< /CipherValue >

</CipherData>

< /EncryptedData>

客户端利用 SSL 结合 HTTP 来完成对服务器的身份验证,并使用 SSL 协商的确定的通信密钥接收发送方加密的消息,从而在客户端和服务器端建立一条安全的通信连接，利用 XML 数字签名和 XML 加密确保消息的数据机密性和完整性，同时通过结合 SSL 对发送方身份验证来提供消息的不可否认性。用 CA 来作为网络管理系统数据传输安全体系结构的基础，同时,作为权威的认证机构，它产生、管理和分发证书，整个安全消息通信过程中所用到的通信密钥、公/私密钥和证书的产生、分发都由 CA 完成的。

- WS－Security 标准

上面的两个安全技术可保证在传输层、网络层和 XML 应用层数据传输的基本安全。而 SOAP 作为 Web 服务通信协议的基础，在实现基于 Web 的网络管理安全模型中应成为重要的角色。IBM、Microsoft 和 Verisign 联合发布了一个关于 Web 服务安全性(Web Services Security，WS Security)的规范，该规范提供了保护 SOAP 消息交换的机制，加强 SOAP 消息传递的安全性，保证数据的完整性和数据的机密性。这个规范还定义了如何在 SOAP 消息内附加并包含一种用于指定二进制编码的安全性令牌(例如 X.509 证书)的机制；提供了一种通用机制,让 SOAP 消息中能够结合多种安全凭证格式，如：Kerberos 票据、X.509 证书，以作为 Web Services 应用环境间相互通信的基础，这些机制可以独立使用也可以组合在一起使用来提供多种安全性模型和加密技术。

根据上述规范,我们可以在 XML Signature 和 XML Encryption 标准之上对 SOAP 消息进行扩展。使得应用程序可以在 SOAP 标头(Header)中加入安全性元素，进而确保消息的完整性、机密性和身份识别,并结合其他技术和标准来达到 WS Security 规范的要求。在对 SOAP 消息的扩展中,通过 XML Signature 来保证消息在传送过程中没有被更改，利用 XML Encryption 来确保 SOAP 消息安全部分处于加密状态。

由于 IT 环境由多种多样的系统构成,因此 Web 服务安全采用模块化方法：下一层为上一层提供基础，如图 5－4 。WS Security 为所有其它的安全规范提供基础；上层的 policy 层,包括了 Web 服务端点 policy (WS policy)，信任模型(WS Trust)，隐私模型(WS privacy)，这些规范旨在保证传输的信息已经得到认证；在此基础之上,提出旨在解决如何连接使用不同安全技术的客户机的规范，包括安全对话(WS Secure Conversation)，联合信任(WS Federation)和授权(WS Authorization)。

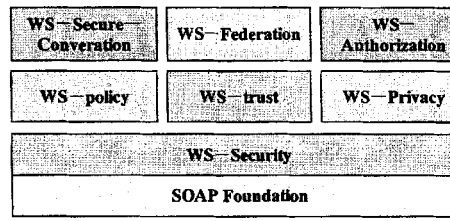


图 5－4 WS Security 的安全模型

规范说明：

WS Security：描述了如何为 SOAP 消息附加签名和加密头部,以及如何为 SOAP 消息附加二进制编码的安全令牌。XML 签名和安全令牌联合保证了数据完整性。SOAP 消息的保密性则由 XML 加密和安全令牌共同提供。

WS policy：定义了功能和安全政策的表述方法。具有可扩展性，可以描述任一方面的要求和性能，基本的服务性能包括:隐私性、编码格式、安全令牌要求和支持的算法。

WS Trust：描述了建立信任关系模型。

WS privacy：定义了 Web 服务中如何表述个人隐私政策。

WS Secure Conversation：描述相互交换的信息的管理及认证方法。

WS Federation：描述在不同环境下管理及从中协调信赖关系的方法。

WS Authorization：定义 Web 服务管理认证数据及策略的方法。根据此安全模型，我们可以给出一个安全方案,如图 5－5 示:

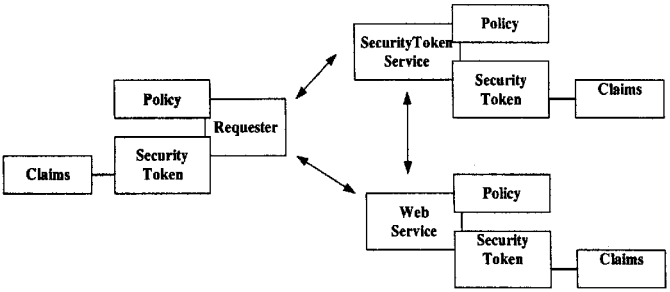


图 5－5 安全方案

解释一下相关术语：

Security Token：一组安全信息,例如,姓名、X.509 认证。

Claims：对一事物的描述,可以是任意形式任意内容。

Policy Web：服务在响应服务请求前需要此请求提供的 Claims 和相关信息。

Proof of possession：在授予安全令牌的所有权的过程中使用到的信息。例如,它可能是与安全令牌相关的一个私有密钥。

该模型通过以下方案保证了基于 Web 的网络管理传输数据端到端的安全：

A 安全令牌的发布

1、对于新收到的数据请求,Web 服务要求它提供安全令牌(如，姓名、密钥、授权等)。如果此数据请求没有所需的安全令牌,则 Web 服务可以忽略或者拒绝此请求。

2、Requester 与认证服务（Security Token Service）通信，以获得安全令牌,这里主要是对服务请求者的身份证明。

3、Requester 将消息和附加的安全令牌一并发送到 Web 服务，然后得到 Web 服务的响应。

B 安全令牌的获取

某些情况下,并不在传递的数据请求中包含安全令牌，只提供了对安全令牌的一个引用，通过此引用来获得安全令牌。

1、Requester 向 Web 服务发出请求,其中包含了对安全令牌的引用和 proof of possession。

2、Web 服务使用这些信息从 Security Token Service 出获得令牌,再对要求进行响应。

- 基于 Web 通信的综合安全模型

在分析了 Web 通信安全保障的若干技术之后，在实际的应用中,可以综合使用以上安全措施，为网络数据通信提供一个更安全的通信平台。以下是基于 Web 的客户端与服务器端通信的安全模型。

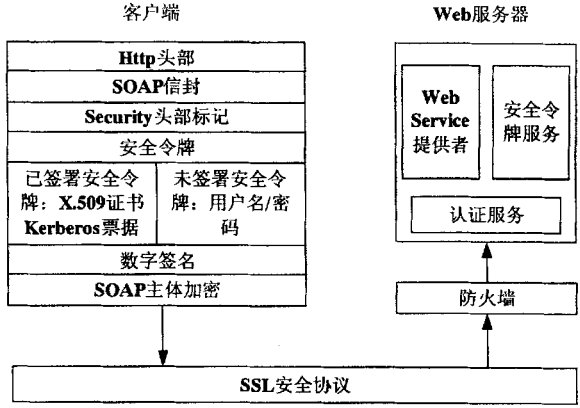


图 5－6 基于 Web 通信的综合安全模型

SSL 安全套接层协议及 XML 加密和数字签名的实现基础是建立在公钥密码技术之上的安全令牌和数字签名。该措施假设:通信双方已经使用某种信任机制(如 CA)建立了 Web 服务器对安全令牌的信任。实际运行中，服务器会核实安全令牌。数据发送方可能希望让接收方判别数据是否在传输过程中被更改，同时验证发送方的身份。由于对整个消息采用公钥算法加密会大大影响 Web 服务的性能,因此实际中常使用一个摘要来加快处理速度。当消息被接收时，Web 服务的验证程序会验证重新计算的摘要与对方发送的摘要是否匹配，以此来确认消息的正确性和完整性。

第六章 结束语

计算机网络和通信网络的应用日益广泛、规模日益扩展、结构日益复杂，如果没有有效的管理方法、管理体制和管理系统的支持和配合，就很难维持网络的正常运行。近年来，需求的驱动、技术的进步推动了网络管理的迅速发展。

传统的集中式的网络管理模式在网络规模急剧膨胀的现实面前已显得越来越力不从心，网络管理必须走向分布式管理的道路，CORBA 技术在分布式计算方面的成功，为分布式网络管理提供了有益的启示，基于 CORBA 的分布式网络管理是一条现实可行的、可实现多域交叉的管理方案；Web 技术的出现和流行为创建一个平台独立的通用网络管理系统提供了一条新的解决途径，基于 Web 的网络管理技术的一个先天优势是可以是很容易地实现分布式的网络监视和控制；网络管理的另一个发展趋势是向智能化、综合化的网络管理方向发展。

本文所研究的是基于 Web 的分布式监控网络管理系统，以下是本文所完成的工作：

- 本文以传统的网络管理体系结构为基础，结合了 Web 技术和 CORBA 技术构架了一个基于 Web 的分布式网络管理系统，其中包含它的组织模型、通信模型、信息模型以及功能模型。将这套理论模型结合现有的 YSZ2002 电力网络监控系统构架出一种符合 YSZ2002 系统的分布式监控网络管理系统的体系结构，并对系统所需实现的网络管理功能进行了详细的设计和开发。
- 针对用户可以在浏览器端获取的网管信息以及 Web 页面的发布，本文采用 IIs 服务器作为 Web 服务器，采用 Web Service 技术向浏览器客户端提供各种管理功能的接口，SOAP 协议封装 Xml 格式的网络管理信息来支持跨平台数据的传输，完成浏览器与本地子管理者之间的信息交换。
- OSI 规定的五大网络管理功能中，本文主要针对其中的安全管理进行详细的设计，对于安全问题我们分为两部分来考虑：对网络管理的内部信息访问权限的授权和鉴权；本地的子管理者与浏览器客户端交互时，通过 Internet 传输数据的安全性。
  a. 建立分层的网络管理结构，根据不同层次上的用户管理范围的不同，设立相应的权限，并为每一层次上的用户提供相应的身份验证，从而保证了数据的可靠性访问。
  b. 对于通过 Internet 传输的信息的安全性，由于我们使用的是 Web Service 技术，用 SOAP 协议封装 Xml 格式的数据进行传输，要保持数据传输的安全，本文采用了对 Xml 数据进行加密的方法来解决。

由于基于 Web 的网络管理系统是随着 CORBA 和 Web 技术的发展而刚刚兴起的新型网络管理技术，发展潜力还很大，还有很多地方可以作进一步的研究：

- 使用 CORBA 与 JAVA 技术的结合来实现基于 Web 的网络管理。在这种方法中，Web 服务器与浏览器相结合，成为 Java 小应用程序的分发管道。被下载的小应用程序在浏览器内局部运行后，成为 CORBA 服务器的 GUI 客户，承担部分应用逻辑和所有界面逻辑的功能。对于复杂应用逻辑，小应用程序通过特定的 ORB 域内协议直接与远程 CORBA 服务器通信，由服务器激活相应服务对象的方法而完成，从而实现面向对象的分布式计算。使用 Java 技术可以有效的解决跨平台的问题。
- 要加强对网络管理其他四个功能部分的研究，虽然故障管理、配置管理和计费管理在本系统中已经很完善了，但性能管理仍需要加强，要与 QOS 相结合，争取达到网络的各种性能参数的最佳值。

研究生阶段工作总结

感谢我的导师史浩山教授充分给予我锻炼的机会。在我硕士读书期间，让我参与了下列项目的研发工作：

- 苏州智能配电自动化有限公司：**YSS2000™**电力监控系统（V2.0）

我在其中主要和其他同学合作完成工程版组态界面由V1.0版本向V2.0版本升级的设计与实现、完成其中的编码、测试工作。

- 苏州智能配电自动化有限公司：**YSZ2002™** 分布式电力监控系统（V3.0）

我在其中主要完成报警信息发布服务的前期调研，需求分析、概要设计，基本实现了报警信息短信发布及报警信息电话发布的前端工作。

- 苏州智能配电自动化有限公司：**YSZ2002™** 分布式电力监控系统（V3.0）

我在其中主要和其他同学合作完成YSZ的Web信息发布服务的前期工作，需求分析、概要设计，完成部分编码和测试工作。

在参与上述科研项目的同时，我结合项目中技术发展方向和个人兴趣爱好进行了一些学习和研究，在此过程中我发表了一篇文章：

- 蒋毅 史浩山：一种基于移动Ad hoc网络的安全路由策略，计算机应用研究，2005年9月

致 谢

首先感谢我的导师史浩山教授。史老师以其为人师表的风范、诲人不倦的忘我精神以及严谨治学的态度深深地感染着我，从他身上，我学到了踏实的工作作风，学会了在研究和学习中分析问题、解决问题的能力。在论文编写期间，史老师给予了我无微不至的关怀与细心的指导，并为我提出了很多好的建议，使我得以顺利完成该论文。

感谢吴健教授。在项目研究期间，他主持的内容丰富、题材新颖的专题讲座给予了我开展研究的无数灵感，让我的研究工作免于陷入困境。在此向吴老师表示诚挚的感谢。

感谢我的师兄滑楠博士和陈丁剑博士，他们在 YSZ2002 电力监控系统的设计和实现方面作了大量的前期工作，并且无私地向我提供了他们的研究成果和许多宝贵思想；特别感谢我的合作伙伴侯蓉晖博士和董相均博士，在并肩战斗的日日夜夜里，我们携手并进，探讨各种难题，共同促进；感谢陈伟硕士、黄熙硕士、李恩菊硕士、王静硕士、王磊硕士，在论文的撰写过程中得到他们的大力帮助，并为我的论文提供了很多宝贵的资料和建议；感谢讯博 YSS 小组的所有成员，对于他们所给予的关心和帮助表示诚挚的感谢。

最后感谢我的父母，二十多年来，他们始终如一的支持我，给予我生活上无微不至的关心，使我能够安心学业，在人生道路上不断成长。希望他们和我一起分享收获的快乐。

感谢所有关心和帮助过我的人们。

参考文献

【1】朱其亮：CORBA 原理与应用，北京邮电大学出版，2001 年 10 月

【2】OMG：CORBA 服务，电子工业出版社，2002 年 7 月

【3】杨家海、任宪坤、王沛瑜：网络管理原理与实现技术，清华大学出版社，2000年9月

【4】Andrew S.Tanenbaum：计算机网络，清华大学出版社，1999年8月

【5】Stepben Spaibour E RobortEckstein: *WEBMASTER技术手册*, 清华大学出版社, 2004年4月

【6】Scott Short：构建***XML Web***服务，清华大学出版社，2002年10月

【7】李维 ***:C++ Builder 6 SOAP/Web Service*** 开发，华中科技大学出版社, 2002 年 4 月

【8】胡海路、彭接文、胡智宇***: XML Web Services*** 高级编程规范，2003 年 2 月

【9】孙淑铃， 应用密码学，清华大学出版社， 2004 年 3 月

【10】Mark O`Neill: ***Web*** 服务安全技术与原理，清华大学出版社, 2003 年 9 月

【11】 钟义杰::基于 ***CORBA/Web*** 的网络管理系统体系结构的研究与设计，华中师范大学硕士学位论文，2000 年 6 月

【12】李静：基于 ***Web*** 的分布式网络管理系统分析与设计，西安交通大学硕士学位论文，2000 年 1 月

【13】郭楠：基于 ***Web*** 的分层式网络管理系统的设计与实现，东北大学硕士学位论文，2001 年 8 月

【14】贺旻捷：基于 ***Web*** 方式网络管理的研究与实现，南京理工大学硕士学位论文，2001 年 1 月

【15】刘兰：统一网络安全管理的分析与设计，华中科技大学硕士学位论文，2002 年 5 月

【16】中国惠普有限公司：HP OpenView 网络管理手册，高等教育出版社，2001 年 1 月

【17】唐翰：基于中间件的分布式监控网络管理技术的研究与应用，西北工业大学博士论文，2004 年 3 月

【18】郭军：网络管理（第二版），北京邮电大学出版社，2003 年 9 月

【19】姜 飞, 夏靖波, 黄鑫洋, 常乔冈：一种基于 CORBA/Web 技术的综合网管系统的设计，计算机应用，2004 年 7 月 第 24 卷 第 7 期，pp. 87－89

【20】陈荣平,唐宝民：基于 Web 的网络管理，江苏通信技术，2000 年 8 月 第 16 卷 第 4 期，pp. 4－8

【21】Michah Lerner, George Vanecek: *Middleware Networks Concept, Design*

and Deployment of Internet Infrastructure, Kluwer Academic Publishers, May 15 2000

【22】HARLERC Web Based network management: Beyondthebrowser[M]United States of America: John Wiley&SonsInc, 1999

【23】苑迎春、张 伟、李宪松、王凤先：基于 CORBA 和 Web 的网络管理系统研究，计算机工程，2002 年 1 月 第 28 卷 第 1 期，pp. 267－277

【24】Object Management Group. The Common Objict Request Broker: Architecture and Secification. Revision2. 3, Object Management Group, Framingham, Mass, june 1999. 29～49

【25】李文超 杨妮妮：基于 CORBA 和Web 的网络管理模型，交通与计算机，2004 年 4 月 第 22 卷 第 119 期，pp. 63－66

【26】王海洲，李 晓：基于 Web 管理技术的安全网络管理系统，计算机工程，2002 年 1 月 第 28 卷 第 1 期，pp. 170-172

【27】RFC 2501 Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, Jan. 1999

【28】RFC 1157 A Simple Network Management Protocol (SNMP), May 1990

【29】RFC 2236 Internet Group Management Protocol, Version 2 Nov 1997

【30】RFC 2205 **Resource ReSerVation Protocol (RSVP) -Version 1 Functional Specification,** Sept. 1997

【31】RFC 1095 **The Common Management Information Services and Protocol over TCP/IP,** April 1989

【32】RFC 1156 Management Information Base for Network Management of TCP/IP-based internets, May 1990

【33】白显羽、李村合、张培颖：Web Service 的应用研究与实现，福建电脑，2004 年 8 月，pp. 20－21

【34】王世美、王元成、宿超：Web Service 原理与应用，山东电大学报，2004 年 3 月，pp. 15－16

【35】刘道军, 高会生, 刘童娜：SNMP、CMIP 和 CORBA 的安全性分析，电力系统通信，2002 年 3 月

【36】赵文玉 纪越峰 徐大雄：DCOM 与 CORBA 及其在网络管理应用中的分析与比较，计算机工程与设计，2001 年 12 月，第 22 卷 第 6 期，pp. 69-73

【37】邱雪松 李文景 孟洛明 陈俊亮：一个基于 CORBA 的网络管理体系结构，电路与系统学报，2000 年 12 月 第 5 卷 第 4 期，pp. 86-89

【38】林 勇、吕述望、陈辉焱：安全套接层（SSL）及其在网络安全管理中的应用，计算机应用研究，2003 年第 7 期，pp. 40－43

【39】唐瀚，史浩山：分布式电力监控系统中安全问题的研究与实现，电讯技术，2004 年 4 月，pp. 107－111

【40】徐　峰，宋如顺，赵青松：基于 Web Service 的混合型网络管理研究与实现，计算机应用研究，2004 年 3 月 第 22 卷 第 3 期，pp. 125－128

【41】杨　鹏：架构在 WebServices 上的电信级网络管理，数据通信，2003 年 4 月 第 25 卷 第 1 期，pp. 9－11

【42】滑楠：基于中间件的分布式数据服务设计与研究，计算机工程，2005 年

【43】李增智，刘康平，王志文：基于 OSI 网络的安全管理，微电子学与计算机，2000 年 2 月 第 2 期，pp. 18－21

【44】方伟，钟声扬，诸鸿文：基于 Web 的网络管理，通信技术，2002 年第 1 月 第 104 期，pp. 71－74

【45】朱菲，肖德宝：基于 Web/CORBA 的网管关键技术的研究，小型微型计算机系统，2000 年 8 月 第 21 卷 第 8 期，pp. 814－817

【46】龚新浩，熊齐邦：基于 SOAP 的分布式网络管理，计算机工程，2003 年 12 月 第 29 卷 第 22 期，pp. 131－134

【47】WonkiHJ, etal. Web-basedd Intranet services and network management. IEEE Communication Manazine , 1997(10)

【48】Kun Yang, Alex Galisl, Telma Mota, Xin Guo and Chris Todd: *Service and Network Management Middleware for Cooperative Information Systems through Policies and Mobile Agents*, Department of Electronic and Electrical Engineering, University College London

【49】Andrew Hughes: *Middleware for the management of a large heterogeneous programmable network: a regress report,* Department of Computing Science, University of College London

【50】Michael Champion, etal. Web Services Architecture, http://www.w3.org/TR/2002/WD-ws-arch-20021114/

【51】SOAP 协议规范. http://www.longen.org/S-Z/details～z/SOAP.htm

西北工业大学
学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西北工业大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文被查阅和借阅。学校可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律注明作者单位为西北工业大学。

保密论文待解密后适用本声明。

学位论文作者签名：蒋毅　　指导教师签名：

2005年3月20日　　2005年3月20日

西北工业大学
学位论文原创性声明

秉承学校严谨的学风和优良的科学道德，本人郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容和致谢的地方外，本论文不包含任何其他个人或集体已经公开发表或撰写过的研究成果，不包含本人或他人已申请学位或其它用途使用过的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

本人学位论文与资料若有不实，愿意承担一切相关的法律责任。

学位论文作者签名：蒋毅

2005年3月20日