摘 要

通信网络的多技术集成和规模化扩展，使综合网络管理系统复杂性的关键在于网元管理技术。

论文以《海军综合网络管理系统》军事预研项目为背景，在深入分析面向对象范型的中间件技术 CORBA 和 Ice 的基础上，以 TMF 多技术网络管理理论和规范为参考，提出了综合网络管理系统的整体设计方案。该方案综合 TMF 多技术网管规范中的网元模型，吸取软件理论中的设计模式思想，提出了抽象网元模型。论文采用 Ice 技术描述和实现抽象网元，利用 Hibernate 技术解决抽象网元数据的持久化，并采用二类对象模型解决了管理信息的元数据管理和实例管理，实现了基于抽象网元模型的网管代理 TMFAgent。

论文实现的海军综合网络管理原型系统已通过鉴定。原型系统基于电子地图的智能化操作界面，多技术网络的综合管理能力，丰富的管理功能，均验证了抽象网元模型的合理性。

关键词：综合网管网元管理　网元模型　**CORBA　ICE　TMF　Hibernate**

Abstract

Multi-Technology integrated and scalable expansion of Communication Network makes the key of Integrated Network Management System's complexity focus on network element management techniques.

This paper considers the military pre-study project of "Navy Comprehensive Network Management System" as the background. On the foundation of deep analyze of Object-Orientated middleware CORBA and Ice, by reference the theories and specifications of TMF Multi-Technology Network Management this paper address the whole design scheme of an Integrated Network Management System. Following Design Pattern in software theory, the author put forward an abstract network element model in the scheme based on Managed Element Model in TMF MTNM specification. This paper use Ice to describe and instantiate the abstract network element model and use Hibernate to resolve the problem of abstract network element data's persistence. This paper use Second Object to implements the management of meta data and instantiate in management information. Finally, The paper implements the network management agent—TMFAgent which based on abstract network element model.

The Navy Comprehensive Network Management Prototype System realized based on this thesis has already past the authentication. The prototype system has intelligent graphic user interface which support GIS map. All the comprehensive management ability face to multi-technology networks and abundant management functions verified the rationality of the abstract network element model.

Keywords: INMS ME Model CORBA Ice TMF Hibernat

创新性声明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：胡明波　　日期：2006.1.

关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。本人保证毕业离校后，发表论文或使用论文工作成果时署名单位仍然为西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。(保密的论文在解密后遵守此规定)

本学位论文属于保密在2年解密后适用本授权书。

本人签名：胡明波　　日期：2006.1

导师签名：吴孝红　　日期：2006.1

第一章 绪 论

1.1 综合网络管理概述

当人类进入21世纪时，一个显著的特征就是社会的网络化。作为网络的重要组成部分，网络管理系统的建设和发展也面临着新的机遇和挑战。当今通信网络的发展特点是网络规模不断扩大、功能复杂性不断增加、异构类型的网络逐渐融合，这种趋势给网络管理带来了前所未有的挑战。在面向新一代网络(NGN)的特点和管理需求下，传统的SNMP和CMIP的优势不复存在，探索新的管理方法、构建新的管理体系结构、应用新的管理技术成为网络管理领域的研究热点和迫切需要解决的任务。

综合网络，又可以称作混合网络，它泛指含有多种技术的网络。混合网络有很多种类型，大致存在以下三种情况。

- 当前的电信网可以采用的特定技术或所能够提供的特定业务被划分为若干子网，这些子网通常被称为专业网。目前的电信业务提供商的网络一般采用如IP、SDH、WDM等多种技术，由多个专业技术通信网络构成，这些专业网相互协作为用户提供各种业务。可以将属于业务提供商的相互之间存在多种关系的多个专业网组成的网络称为混合网络。
- 随着技术的不断进步，不同网络技术不断融合，业务提供商将越来越多的采用多技术混合网元，即一台网络设备采用了多种技术。这样的由多技术混合网元构成的网络也可以称为混合网络。
- 由于市场竞争的结果和单个业务提供商能力的限制，客户往往需要跨多个业务提供商的端到端的连接性业务。为了能够迅速提供此类业务，并完成业务的故障管理、性能监视管理等功能，就必须对这种由多个业务提供商的网络构成的混合网络进行管理，从而完成跨多个业务提供的端到端连接性业务的管理。

可见，综合电信网络是由许多独立管理的业务网和支撑网互联而成，网络资源和网络业务紧密结合。每个网管系统建立在各自的专业网上，使用不同厂商的设备，从而使不同的业务和资源分别对应不同的操作、管理、维护的运营系统。各种专用网管系统相互独立而并存的现象导致了目前的网管存在一些急需解决的问题。

1) 专业网网管系统之间缺乏有效的互操作[1]

电信运营商现有的专业网之间并不是独立的，而是相互关联的，其关系主要体现在两个方面：一是承载关系，二是互通关系。然而，各专业网的网管系统却相互独立，管理范围只限于本专业网，从而对各专业网之间存在的关系无法管理，

同时也无法了解到其他专业网中与之相关的信息数据。这就导致了网络运营商对全网资源没有一个统一全面的认识，也不能进行全网资源的调度和支配，从而大大降低了整个网管系统的管理效率，也降低了资源利用率。

2) 网管界面缺乏统一全局视图[1]

各专业网的网管系统仅有一个属于自己的网管系统界面，该界面只涉及了本专业网的管理功能，而缺乏对该专业网与其他专业网关系的管理，因此造成了网管系统界面的分割与多样化。

3) 网管数据的定义缺乏一致性[1]

网管数据的定义与网管系统紧密相关，各专业网由于都有一个独立的网管系统，所以它们也都有与之相对应的网管数据。由于各专业网的情况互不相同，所以对它们的管理要求、管理内容不一样，因此所建的管理对象类也不一样，最终获得管理数据也不同。

4) 网管系统缺乏综合管理功能[1]

由于各专业网之间存在承载或互通的关系，而这种关系也应该在网管系统中得到体现，换言之，网管系统应该具有跨各专业网的管理功能，管理各专业之间协调统一的运行，而这种综合的管理功能是任何一个专业网管系统所不具备的。

5) 网管系统的建设和运营费用高[1]

多个专业网都独立建设和运营，因此不可避免的将会出现重复建设。

1.2 综合网络管理系统的必要性和研究目标

从上述这些类型的网络以及存在的网管问题中可以发现，目前对网络的管理一般建立在独立的子网管理功能之上。但是，由于它们大多采用各自的管理协议，互不兼容，这样导致了即使是一个企业内部的网络中也有许多个不同管理功能和服务设施的子网管理系统的共存。不同的子网管理系统互不兼容使网络管理问题更加复杂，管理人员需要同时管理每个子网，而且各子网之间的信息交换仍需要通过管理员手工完成。

这样的综合网络管理方式，由于被管网络环境复杂，设备供应商多样化，其中多数的网络管理系统仅对某一厂商的网络较为合适，换言之，多数的网管系统未能具有良好的开放性，不能很好的解决多厂商，多技术网络环境给网管系统带来的问题。因此，使网管系统具有的良好的可互操作性是十分重要的，只有这样才能使网管系统达到对整个网络的管理目标。

由于传统网管系统在处理复杂网络环境时表现出的问题和缺陷，目前，大型的网络管理系统正朝着综合化、分布化和智能化的方向快速发展。网络规模的扩大、网络设备数量和类型的剧增促使网管系统必须能够高效管理大规模混合网络。

同时，用户对网络业务的关注程度日益提高，这表明网管系统的发展不仅要做到对网络业务的保障，更应该以用户为中心，使网络管理功能高度集成且简单易用，努力做到及时与用户的需求同步，提高网络管理系统的分析效能。

为了达到综合网管的目标，在设计综合网管时就必须有良好的统一构架，为实现混合网络的综合管理打下坚实的基础，使通信网络信息资源得到合理利用，实现各专业网管系统协调移植的工作。在完善专业网管接口时，加强了网管接口的可扩展性和可移植性。

本文主要针对混合网络及综合网管的现状，分析综合网络管理系统的必要性和主要需求，在综合网管系统的底层设计和实现关键技术方面进行了细致的实践。

1.3 论文所做的主要工作

本文的课题来源是“十五”预研项目《海军综合网络管理系统的研究》，该预研项目目前已完成总体鉴定。本论文的研究工作主要集中在综合网络管理整体设计要求，网元管理层与网络管理层接口定义及网元管理层关键技术、设计和实现等问题。

本文首先在认真研究目前综合网管系统总体现状的基础上，结合综合网管目标网络环境的特点以及海军综合网管的需求，从以服务为核心的角度出发，对综合网管系统的功能定制、系统整体结构和用户界面设计等各方面提出了整体设计思路。

在研究综合网管分布式基础设施的接口定义时，本文简单介绍了目前几个典型的用于实现分布式网管架构的中间件，分别从接口定义方式，通信能力、带宽占用、可靠性、灵活性等方面对这些中间件进行了比较，并且分析了不同的混合网络环境中适用于网元管理层与网络管理层分布式计算的有效中间件。在设计网元管理系统(层)的内部结构时，本文认真研究了电信管理论坛提出的网络管理系统实现框架－MTNM 多技术网络管理。借鉴其设计理念，本文提出了一种可行的构建综合网管网元管理层的代理模型 TMFAgent，并对 TMFAgent 中的设计模式应用和分布式接口规划做了简要介绍。

网元管理层是大型综合网管系统的构建基础，其主要功能在于对实际网络进行抽象建模，为网络管理层提供一个设备技术无关、生产厂商无关的统一网络管理基础平台。在混合网络环境中，网元管理层面对的是由多技术多厂商设备组成的十分复杂而庞大的网络结构。本文通过二类对象，抽象网元模型，Hibernate 等一系列技术方法解决了综合网管中复杂多变的具体设备信息转化为可供管理、便于管理的形式的问题。最后，本文提出了结合 TMFAgent 代理技术以及相关网管技术的综合网管网元管理系统实现方案。

1.4 论文的内容安排

论文共包括六章内容。

第一章介绍了综合网络管理系统研究的背景以及本文所做的主要工作。

第二章详细阐述了以服务综合为核心的综合网管系统在功能定制，系统架构，用户界面等方面的整体设计思路。

第三章简要介绍了几种分布式网络管理系统中广泛使用的中间件技术，全面比较了它们在不同混合网络环境中的适用性；介绍了 TMF 多技术网络管理技术概要，并从设计模式的角度分析了基于多技术网管的 TMFAgent 抽象代理技术。

第四章全面论述了综合网管系统中网元管理层涉及的关键技术及其应用方式。

第五章详细描述了综合网管系统网元管理层的设计结构和各个功能模块的具体实现。

第六章对全文进行总结，概括文中介绍的关键技术应用特点及网元管理层设计和实现要点，并指出了系统中仍有待进一步完成的工作。

第二章 综合网管系统的整体设计

通过前一章中对现有网络环境的简要介绍以及对综合网管的目标分析可以发现，网络管理服务的综合是综合网管系统的核心内容，一个综合网管系统的设计应该围绕着这个核心进行，使其为网络运营的各个方面提供全面的服务。因此，在设计一个综合网管系统时，要从网络管理的综合能力出发，以此为目标选择适当的系统架构，设计完善的用户界面。本章将从网管服务、系统架构和用户界面等角度详细阐述如何设计以网管服务综合为目标的网管系统。

2.1 核心功能设计

2.1.1 系统功能定制

服务的综合是网管的核心。所谓综合服务，不仅需要横向地实现各管理功能域的操作，同时，纵向上应当做到各层次功能完善，并且需要在面向用户的功能层次上做到可定制化。

1. 资源配置管理

综合网管资源管理需要实现多专业或全网的资源配置管理[3]。在资源管理方面，逐步实现对各专业资源和全网资源统一管理、统一调度，并加强对资源的使用情况分析，从业务的角度分析资源占用情况，支持业务的成本评估，能够为网络成本预算提供依据。综合资源管理还应该包括网络备品备件的管理，为合理配置备品备件、科学调配提供基础和依据。这一目标的实现需要依靠不同专业网络对各自网络资源的管理，在综合网管系统中将这些局部数据进行有效整合，最终达成全网的资源整体监控。

2. 故障告警管理

综合故障管理的目标是以客户和业务为中心进行预警，实现端到端业务的监控管理，将告警信息关联到电路和客户，提高业务保障能力[3]。其主要功能是基于不同专业的监控系统实现集中和综合的故障管理，即将现有的专业网管系统所产生的告警信息进行综合管理，提供跨专业平台的网络集中告警功能，在一个统一的网络管理平台上实现多专业综合告警的实时集中呈现，能够在众多相关联的告警中迅速判断出产生告警的真正原因，加强故障的统计、分析、评估等管理功能。

综合故障管理根据告警的级别、类型以及对故障处理的要求进行故障告警的过滤，根据运行维护的实际需要人工进行门限设置和告警等级设置；通过故障档案建立故障案例，用于网络性能的品质分析，为网络优化调整提供依据。

综合化集中告警功能定位于跨专业告警相关性分析。通过事件相关性分析以提高故障定位的准确性，实现专业定位；基于运行维护经验的积累与分析，实施告警的相关性分析，界定一些较为模糊的故障归属，提高故障处理能力和效率。

3．性能流量管理

综合网管的主要功能之一是跨专业网络性能流量监视，要求综合网管对各专业网络关键性指标进行实时监控，并能够对各专业网管采集到的性能数据进行统计，从全网角度进行分析[3]。通过网络数据采集对各专业网管的关键性能进行实时监视，使得事后维护向事前维护转变、补救性维护向预防性维护转变，并对需要进行干预的专业网管进行配置或调整，以优化整体网络性能；分析和评估接收到的各专业网络和网元的数据，重点是对间歇的多种网络资源混合使用情况的总体性能退化进行分析处理；对网管性能数据进行长期储存，进行数据挖掘、分析，加强性能分析的深入性、综合性，着重于发现影响网络质量的关键因素，逐步建立分析型的网管系统，为网络调优、网络扩容或新建提供依据。

4．服务质量管理

主要通过客户业务监控，关注客户的业务保障质量，面向用户服务，通过综合网管系统体现网络服务的新理念与新的服务方式[3]。针对 SLA 服务等级管理协定用户，为用户提供应用服务监控报表、性能统计分析报表、历史数据趋势分析报表、网络流量流向分析报表、SLA 报告以及针对 SLA 等级的用户管理。与此同时，需要将设备故障信息与对业务和客户的影响分析关联起来，做到在设备发生故障的同时，第一时间通知客户，并对影响该业务和客户的设备故障进行及时的处理，以提升客户的满意度。

2.1.2 系统结构规划

综合网络管理系统的功能服务需求决定着整体系统所要采用的体系结构，另一方面，系统的结构将成为影响网管服务实现的重要因素。合理的系统结构不仅能够提高系统自身的工作效率，增强其健壮性，同时还能有效改善不同系统或子系统之间的连接和互操作能力。同时，网管系统部署对网管系统的成功运行十分重要，网管系统部署的方案是根据系统的体系结构确定的。要设计良好的系统结构，使综合网管功能得以全面的发挥效用。目前，最广为人知的基本网络管理体系结构主要是集中式体系结构和分布式体系结构。

集中式管理体系是较早的一种结构。由于管理协议和实现手段等因素的限制，早期的网络管理系统大多采用这种结构进行管理。集中式结构的网管系统建立在一个计算机系统上，该计算机系统负责所有网络管理任务。采用集中式管理结构，网络管理系统的实现简单，管理关系简单、清晰。

然而，对于一个大规模综合网络，集中式管理的结构显然无法满足这样的要

求，即先对各专业网进行独立管理，再由综合网管系统统一整合。集中式管理单一的结构势必会限制综合网管的扩展。因此，一般采用多客户端多服务器的 C/S 结构是综合网管系统的基本要求。在 C/S 结构中，各服务器完成对被管网络的直接访问控制，客户端和服务器是服务请求者和服务提供者的关系。尽管两者采用平等的模式，但是仍有大量逻辑处理放在客户端(前端)，例如一些表达服务和业务规则实现就放在客户端应用程序中。

在采用 C/S 结构的分布式网络管理系统中，客户端程序和服务器端程序都比较庞大，各自都含有较多逻辑组件。当升级应用程序的时候，用户需要重新安装配置整个系统(包括客户端和服务器端)。这显得较为复杂和繁琐，用户也不希望看到。与之相对具有瘦客户端的是基于 Web 技术的浏览器/服务器(B/S)结构。

在基于 Web 技术的 B/S 结构中，把实现业务规则和提供数据服务的角色分配给 Web 服务器端，管理人员可以将很多的计算与存储任务转移到 Web 服务器上，从而可以使客户在客户机平台上访问它们。Web 浏览器承担表达服务的流水型角色，它能够把用户所需要获取的网管信息灵活地显示，给用户一个便于接受的、容易使用的界面。相对于单纯的 C/S 架构，基于 Web 的系统安装、配置、升级、维护等工作都将被大大简化[15]。

尽管目前的网管系统开发中多种设计框架并存，但可以说并不存在所谓的“最好”的体系结构，每种类型都具有在一定网络环境下工作良好的特定功能。随着用户不断提出的新需求以及各种新技术的应用，总体上朝着结构分布化、界面 Web 化的趋势发展。应当选择一种最适合实现已设计网管服务功能要求的网络管理体系结构。

2.1.3 图形界面设计

综合网管系统的图形用户界面同样应该服务于既定的综合网管功能，要针对综合的服务设计用户界面。综合网管系统不同于普通的网络应用程序，它是网管人员重要的管理工具，是网络运营商的运营支撑系统。通过综合网管的用户界面，网管人员可以监视网络运行情况，适时调整网络运行的各种参数；运营商可以及时分析网络业务的优劣，规划商业化的网络服务。因此，一个成熟的综合网管系统仅仅能够提供友好的用户界面是不够的，综合网管图形用户界面的设计必须考虑已定制的资源配置管理，故障监测管理，性能流量管理，网络业务管理等多个方面的功能，做到综合网络管理服务的展示。

从资源管理方面考虑，完整准确地显示各专业网的网络情况，突出各专业网之间相互的承载关系；通过与资源管理系统、客户资料系统集成，提供客户组—客户—资源等多层拓扑视图；能够根据用户需要，以定制的方式为网管人员提供所需的网络视图，帮助网管人员明晰全网资源拓扑，整个网络的连接情况，实施

网络资源关联分析。

故障管理功能模块同样应该具备良好的图形化配置界面，并与资源管理系统紧密结合，提供故障相关性、时间相关性、事件相关性、业务相关性、客户相关性、地理位置相关性等故障关联模型的分析、显示。实现网络故障闭环管理体系，完善报障及排障流程，提供电子工单处理，服务于日常的生产、管理工作，加强故障处理流程的跟踪管理。

同时，在性能流量统计方面，网管系统应当能够分层次的显示网络的流量分布，实时、准确地统计网络各项运行参数。用户界面对性能参数的表示应当直观多样，可以采用历史曲线、条状图、饼图、表格等多种方式为用户提供网络性能分析，进行网络业务开发提供有效参考，这是差异化网络服务保障手段。同样，性能管理中也可以增加个性化重定义监控图等功能。

2.2 辅助功能设计

2.2.1 系统容错能力

不论什么网络应用系统都或多或少的存在一些不稳定因素，某些时候会引起系统无法正常运行。此类问题的存在对网管人员和网络运营商来说都会造成许多损失，轻则暂时无法顺利控制网络运行状态，重则完全丧失对网络的监控能力。因此要使网管系统在出现问题的情况下仍能保持正常的工作状态，保证以往的网络数据不丢失是系统容错能力的基本要求。

综合网管系统除了能够监控网络以外，同时也应当对系统自身可能会出现的错误状态进行监测及定位。系统对错误的主动侦测和定位，能够帮助网管系统使用人员快速找到问题的源头，有助于及时解决出现的问题，使由于系统错误而造成的损失降到最低。一些常见的由网络故障引发的问题，如服务器和客户端连接超时，数据库访问失败，由程序原因引起的服务器或客户端运行异常，只要能够及时发现并有效应对，都不会对系统造成严重影响。

另外，系统面对错误状态应该提供自动从错误状态恢复的机制，最基本的要求是为使用人员恢复系统功能做好充分的准备，为网管人员快速恢复系统减少不必要的机械操作。及时备份数据是这一方面的常见工作，有效的数据备份处理能提供运行网管系统所需数据的快速恢复。

2.2.2 系统升级方案

被管网络会不断变化，网管人员和网络运营商会对综合网管系统的功能，性能，容错等各方面能力不断提出更高的要求。这就需要开发人员尽早考虑系统的升级方案。好的升级方案可以在方便，快速的为系统增加功能的同时，不对原有

系统的功能和运行造成任何影响，使整个系统保持良好的连贯性。制定网管系统的升级方案需要从系统各个角度全面地去考虑，在设计网管系统初期就应当做好充分思想准备。大多数系统的升级都会按不同模块分步骤执行，因此网管系统的结构要做到适合各组件独立升级互不影响。系统的升级应当是扩展的方式，而非否定的方式。换言之，不应该将原有系统的设计实现全面丢弃，而应当采取逐步扩展，逐步替代，最终达到更新整个系统的目的。

网管系统是一个长期连续运行的网络应用，对网络数据的维护是网管系统的重要组成部分。因此，数据库更新是网管系统的另一项重要的更新内容。数据库的更新主要包括数据库软件版本升级，网管数据库表格的优化设计，海量数据的定期备份等工作。这些内容都需要开发人员在设计综合网管系统初期就能有充分的准备。

2.3 本章小结

本章主要强调综合网管系统的设计应该以网络管理服务的综合为核心，无论是系统体系结构，用户界面设计都应当以此为目标。本章介绍了网管综合服务在各管理功能域的具体体现，分析了实现服务综合所需的系统架构及用户界面要求。同时，本章还简要介绍了设计综合网管系统时对系统容错和系统升级等方面的辅助功能需求。

第三章 综合网管系统中的基础设施

3.1 综合网管中的分布式对象中间件

3.1.1 网络管理与分布式计算

网络管理软件本质上属于网络应用软件。传统的网络管理模式是一种集中式的管理者一代理模式。一般是在一个网络组织中，设置一个网络管理系统，作为管理者的角色，该管理系统负责收集网络中的各种信息，监视网络的运行情况，并且控制网络运行的行为。这种模式在网络规模相对较小的时候是可行的。然而，随着网络规模的增大，网络管理系统的负载也将急剧增加，进而由于网络管理而引起的通信开销也大大增加，最终可能导致网络本身性能的下降。系统的升级能力总是有限的，并且网络管理系统最合适的安装位置往往也不是网络管理人员所处的位置。这个时候较为合适的解决方案就是改变网络管理系统的部署方式，通过分散网络管理目标或功能，让多个网络管理子系统负责管理网络的不同部分。换言之，可以按照管理的层次、规模或者功能特性将网络管理系统划分成不同的模块，模块之间松散耦合，独立运行。按照这种思路构建的网络管理系统就是分布式的网络管理系统。随着网络管理技术的演化和发展，分布式网络管理技术已经越来越成为主流。对于大型的综合网管系统来说，由于网络的规模之大，采用的技术之多，使得分布式的网络计算方式成为实现综合网管的首选。

如何把网络中运行在各种异构的软硬件平台上的网络管理程序集成起来并实现新的网管功能是一个非常现实而困难的问题。为了解决这一问题，中间件的引入为网管开发提供了良好的思路。中间件是位于平台和应用之间的通用服务，这些服务具有标准的程序接口和协议，支持分布计算，可以为应用程序提供跨跃网络、硬件和操作系统平台的透明交互能力。 由于标准接口对于可移植性和标准协议对于互操作性的重要性，中间件已成为许多标准化工作的主要部分。对于应用层的软件开发，中间件远比操作系统和网络服务更为重要，它为应用程序的开发提供了一个相对稳定的高层应用环境。

中间件向上可以提供不同形式的通讯服务，包括同步、排队、订阅发布、广播等等，在这些基本的通讯平台之上，可构筑各种框架，为应用程序提供不同领域内的服务，如事务处理监控器、分布数据访问、对象事务管理器等。中间件通信平台为上层应用屏蔽了异构平台的差异，而其上的框架又定义了相应领域应用的系统结构、标准的服务组件等，用户只需告诉框架所关心的事件，然后提供处理这些事件的代码。用户程序不必关心框架结构、执行流程、对系统级 API 的调用等。当事件发生时，框架会调用用户的代码，所有底层工作由框架负责完成。

因此，基于中间件开发的应用具有良好的可扩充性、易管理性、高可用性和可移植性。

在分布式网络管理系统的构建过程中，如何使分布运行的子系统和组件成为一个谐调统一的有机整体，为用户提供对整个网络的透明管理是一个至关重要的问题。而中间件技术正是连接分布式网络管理系统，提供透明性支持的最好选择。使用成熟的中间件技术来构建网络管理系统可以充分的利用中间件技术的优势，在异构的网络环境中方便的衔接网络管理系统的各个部分，并当系统的局部进行调整时，不影响其他部分的正常工作。

多数综合网管系统都存在于典型的异构网络环境中，存在多种硬件设备、操作系统、数据库平台和网络协议。为了解决好分布式计算环境中不同模块之间的通信问题，综合网管的实现必须有中间件技术的支持。考虑到基于对象的软件开发模式，目前综合网管可选用合适的对象中间件平台作为构建网管系统的基础。对象中间件是面向对象技术与分布式计算技术相互结合形成的分布式对象计算技术，是当今网络软件开发技术的主流方向。在开发网络管理的运行系统时，对象中间件平台将为网管系统内部功能单元间的交互提供通信支持；为不同的网管系统之间的互操作提供接口；为管理系统和被管资源间的通信提供接口。

3.1.2 分布式对象中间件

目前较为流行的对象中间件是 CORBA 和 Ice，它们各有特色，优点。

1．公用对象请求代理体系 CORBA

CORBA(Common Object Request Broker Architecture，公用对象请求代理体系)是由 OMG 对象管理组织制定的中间件技术规范，其核心是一套标准的语言、接口和协议，以支持异构分布式应用程序间的互操作及独立于平台和编程语言的对象重用[5]。它为可移植的、面向对象的分布式计算应用程序提供了不依赖于平台的编程接口和模型。其不依赖于编程语言、计算平台、网络协议的这一特点，使得它非常适合于开发分布式应用系统和实现遗留系统集成。

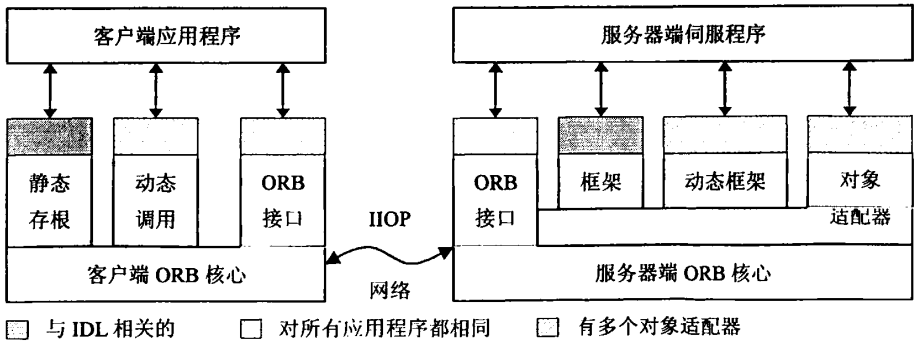

图 3.1 CORBA 体系结构

CORBA 组件

1)对象请求代理

CORBA 这一体系结构的核心部分是对象请求代理(Object Request Broker，ORB)，它作为一个"软件总线"来连接网络上的不同对象。客户只要遵循服务对象的对外接口标准向服务对象提出业务请求，由 ORB 在分布式对象间建立客户－服务器关系。在客户端，ORB 提供一个发送操作调用的接口；而在服务器端，ORB 提供一个将操作调用传送到一个合适的伺服对象的实现。

2)对象适配器

ORB 是通过一个对象适配器(Object Adapter，OA)将操作调用传送给伺服对象的。对象适配器是 ORB 的一部分，它介于 ORB 核心和对象实现之间，负责伺服对象的注册、对象引用的创建和释放、服务进程的激活和挂起、伺服对象的激活和挂起以及客户请求的分发等任务。

3)IDL 语言

IDL 是 CORBA 体系中的另一重要概念，它提供了一组类似于 C++的语法，供 CORBA 应用的开发者描述服务对象的接口。IDL 的好处是使高层的设计人员不必考虑具体的实现细节而只需关心接口功能的描述。由于 IDL 语言只具有描述性，而不具有可执行性，因此 CORBA 规范中定义了 IDL 语言到具体编程语言的映射。IDL 的引入使 CORBA 做到了与编程语言无关。

4)静态存根和静态框架

在静态调用方式下，IDL 编译器将用户的 IDL 描述文件生成对应编程语言的一个客户端存根程序和一个服务器框架程序。存根程序负责把用户的请求进行编码，发送到对象实现端，并对接收到的处理结果进行解码，把结果或异常信息返回给用户。框架负责发送一个操作调用给一个真正提供此操作的实现程序，它对用户请求进行解码，定位所要求的对象的方法，执行该方法并把执行结果或异常信息编码后发送回客户。

5)GIOP 和 IIOP

GIOP(General Inter-ORB Protocol，通用 ORB 间协议)定义了不同 ORB 之间的通信协议，它是一种通用协议，在不同的网络上需要有不同的实现。目前最广泛使用的便是 Internet 上的 GIOP，称为 IIOP(Internet Inter-ORB Protocol，因特网 ORB 间协议)，它基于 TCP/IP 协议。

CORBA 服务

CORBA 服务规范是对 CORBA 核心规范的扩充，它定义了一系列大多数分布式对象处理所需的公共服务，如名字服务、生命周期服务、事件服务、通知服务、交易服务以及安全服务等。网管系统中常用的服务主要有名字服务(Naming Service)和事件服务(Event Service)

1) 名字服务

CORBA 名字服务通过为每个对象分配一个唯一名称，来管理从名字到对象引用的映射关系。名字服务相当于一个目录服务，给定一个名称，该服务返回一个存储在该名称下的对象引用，之后便可利用该对象引用进行相应的操作。

2) 事件服务

事件服务通过对事件(由对象产生并且传送给其他对象)进行封装，提供了基本的消息传递功能，在事件发送接收对象相互不很了解的对象之间建立起一条宽松耦合的通信信道。

2. 网络通信引擎 Ice

网络通信引擎(Internet Communications Engine，Ice)是由 ZeroC 的分布式系统开发专家实现的一种较新的高性能对象中间件平台。它提供完善的分布式系统解决方案，适合所有的异构网络环境[6]。像 CORBA 一样，Ice 也提供了客户端与服务对象的完全分离，客户端不需要了解服务对象的实现过程以及具体位置。Ice 采用软总线机制，使得在任何情况下、采用任何语言开发的软件只要符合接口规范的定义，均能够集成的分布环境中去。

Ice 是一种面向对象的中间件平台。从根本上说，这意味着 Ice 为构建面向对象的客户/服务器模式的应用提供了工具、应用程序接口和库支持。

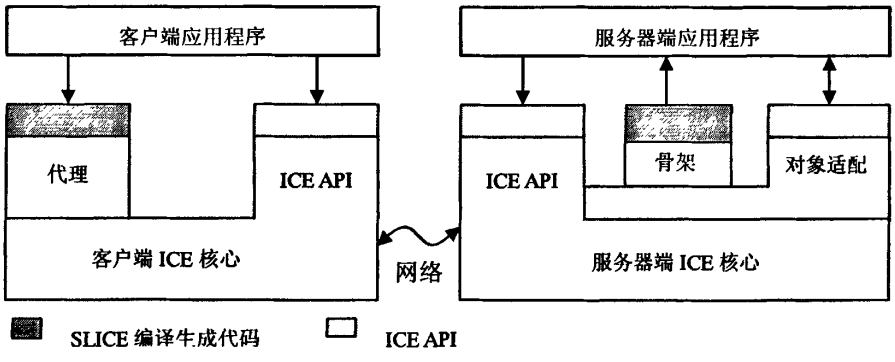

图 3.2 ICE 体系结构

Ice 组件

1) Ice 核心

Ice 处于核心地位的是由大量的链接库实现的对象总线，为客户端和服务器的远程通信提供运行支持，在 Ice 中被称为通信器(Communicator)。它主要关心的是线程，字节序和其它一些网络细节及相关事务，并将应用程序与这些底层事务隔离开。Ice 核心定义了异构环境下对象透明地发送请求和接收响应的基本机制，是建立对象之间的客户端/服务器关系的核心组件。

2) 对象适配器

对象适配器是专用于服务器端的 Ice API 的一部分，只有服务器才使用对象适配器。对象适配器有若干功能：

- 对象适配器负责创建传给客户端的Ice代理。对象适配器知道每个对象的类型、标识，以及传输机制的详细信息。
- 对象适配器可以与一个或多个传输端点关联在一起。如果与某个适配器关联的传输端点不止一个，就可以通过多种传输机制到达在该适配器中的伺服对象。
- 对象适配器把来自客户端的请求映射到服务器端特定对象的特定方法上。对象适配器会跟踪在内存中的伺服对象，记录其对象标识，实现适配请求的功能。

3) Ice 代理与 Ice 骨架

代理代码是由用户定义的 Slice(Specification Language for Ice)文件经过编译后生成的。一个客户端要想与一个 Ice 对象建立联系，必须持有该 Ice 对象的代理。代理是存在于客户端地址空间中的远地 Ice 对象的代表。代理主要有两个功能：

- 为客户提供了一个向下调用的接口。以代理为中介，客户端发起的调用最终会调用到服务器目标对象上相应的操作。
- 提供编码和解码。编码是将复杂的数据结构串行化，把数据转换成适于传送的标准形式。解码是编码的逆过程。

骨架(skeleton)代码也是由用户定义的 Slice 文件经过编译后生成的，其中的内容与 Slice 中定义的对象和数据的类型是对应的。它提供了向上调用的接口，允许 Ice 把控制线程转交给服务器端的应用程序代码。

4) Slice 语言

在 Ice 中，协议是通过使用 Slice 语言描述的相关应用程序接口来定义的。Slice 使 Ice 做到语言无关，使高层设计人员不必考虑实现细节而只需关心功能描述。设计 Slice 的过程也是设计对象模型的过程，是编写 Ice 应用的第一步。

5) Ice 协议

Ice 协议可以建立在多种流行的通信协议之上，定义了客户端和服务器端的 Ice 核心之间的通信机制。Ice 协议的设计简单、开销很小、同时又具有广泛的适应性和可扩展性，以适应不同网络。Ice 协议的引擎是可以扩展的，开发人员可以通过增加 API 插件来增加新的底层传输协议，而不用修改原代码。Ice 的 SSL 传输就采用了这种插件结构。Ice 协议定义由三个主要部分组成：

- 数据编码规则：确定各种数据类型的串行化方式。
- 协议状态机制：规定客户端和服务器如何交互不同类型的信息。
- 版本控制：确定客户与服务器怎样就特定的协议和编码版本达成一致。

Ice 服务

除了核心功能之外，Ice 提供了大量这样的服务。这些服务被实现成 Ice 服务

器，应用程序充当这些服务器的客户端。使用这些服务可以大大的简化分布式应用程序的开发。这些服务作为 Ice 平台的一部分，使开发人员可以专注于应用程序的开发，而不必事先构建许多基础设施。

1) IcePack

IcePack 是一个完善而复杂的服务器配置和激活工具。它的主要功能是提供定位服务。在 IcePack 中记录并维护了对象适配器名和服务器地址的对应关系的映射表。通过检索这张表，客户端就能获取服务器端当前的地址信息。通过这种间接代理和 IcePack 相结合的方式，服务器的迁移将不会影响到客户端的正常工作。

IcePack 还提供服务器激活服务：当客户端发起请求时，服务器可能处于未运行状态，IcePack 会在第一个客户请求到达时，随需启动服务器。通过 IcePack，开发人员可以轻松地配置包含若干服务器的复杂应用。

2) IceStorm

IceStorm 是一种消息的发布/订阅服务，能够消除消息的发布者和订阅者之间的耦合关系。IceStorm 充当发布者与订阅者的中介。当发布者准备好分发一个新消息时，它只需要简单的向 IceStorm 服务器发出一个请求，而无需对订阅者有任何了解。由 IceStorm 服务器全权负责把消息递送给订阅者，IceStorm 还负责处理由于订阅者的行为有问题或订阅者不存在所造成的异常。发布者可以专注于其应用程序特有的职责，而不是管理订阅者这样的琐事。订阅者也可以集中精力解决对消息的处理，而不用关心其它琐事。

3.1.3 中间件的比较与选择

CORBA 与 Ice 在各个方面存在许多差异，因此有必要在比较它们各自的特性的基础上，讨论不同网络环境中这两种中间件的适用性。

1．中间件的比较

1) 中间件核心

CORBA 的核心被称为对象请求代理(Object Request Broker，ORB)，Ice 的核心则称为通信器(Communicator)，两者作为对象中间件的核心有着许多共同点。无论是 ORB 或是 Communicator 都作为对象总线，为客户方发出的请求寻找对应的服务，并管理服务方与客户方之间的连接。这样的总线型组件在对象之间建立客户/服务器关系的中间层，使客户端可以透明地调用通过网络连接的某个服务器的方法[4]。同时，两个核心组件分别为 CORBA 和 Ice 提供了底层线程调度功能，通过创建和管理对象适配器，为客户端的请求或服务器的响应作出合理分配。因此，CORBA 和 Ice 的核心组件，在整体基本功能上是类似的。但 Ice 晚于 CORBA 设计和开发，在其核心部分另外提供了一些有用的附加功能，如日志记录对象，统计对象和插件管理等。

2) 对象适配器

对象适配器是中间件核心和服务器端伺服对象之间的纽带，主要负责对象引用的创建，确保目标对象都有伺服对象实例化，传送请求给相应的伺服对象。CORBA 与 Ice 在对象适配器的设计和实现上有着较大的差异。在 CORBA 2.2 以后的版本中，可移植的对象适配器 POA(Portable Object Adapter)取代了原有的 BOA(Basic Object Adapter)，实现了不同厂商 ORB 之间的互相调用。POA 有着复杂的策略体系供用户定义 POA 的性能，多个策略机制可以组合使用，已达到不同的效果。同时，POA 还有多层的分层创建方式，最终形成 POA 的一个分层结构。相比之下，Ice 的对象适配器实现的相当简洁。在 Ice 中对象适配器被简单的命名为 Adapter。Ice 对象适配器没有层次化的结构，也没有复杂的策略机制。Ice 的 Adapter 没有另外的对象进行管理。可以说，Ice 将 CORBA 中较为复杂的对象适配器创建和管理机制进行了分解和取舍，简化了对象适配器的设计和实现。在分布式计算环境最小需求的前提下实现了足够功能的对象适配器。

3) 接口定义语言

对象中间件解决的是异构网络环境中分布式对象通信的问题[4]。分布式对象(或设备)之间的通信需要一定的协议和约定，这些通信规定是对象之间调用的接口。在 CORBA 和 Ice 中都有各自的接口定义语言，即 IDL 和 Slice，它们为开发者提供了使用自己的描述语言或一种公认的标准来定义设备之间需要使用的协议的机会。使用 IDL 和 Slice 这类语言的好处是使高层的设计人员不必考虑具体的实现细节而只需关心接口功能的描述。这两种语言所关注的焦点是分布对象之间的接口，接口所提供的操作，以及操作可能引起的异常。它们所描述的各种数据类型和对象接口与具体的实现语言无关，所以客户端应用程序和服务器端应用程序可以使用不同的编程语言。目前，IDL 和 Slice 都提供了向高级编程语言的映射。

尽管 IDL 和 Slice 都是中间件的接口描述语言，有着上述的共同特征，但是根据设计的意图和适用的条件，它们之间仍有许多不同的特征。IDL 的大多数语法特性和 C++的语法规则相似，它提供了基本和用户定义的数据类型，接口定义和异常声明等一系列功能。IDL 也提供了面向对象的特性，支持接口的多继承。Slice 语言取消了 IDL 中的一些特性，也增加了其他的一些特性。Slice 支持传递类的引用语义，这在 CORBA 的 IDL 中是不支持的。另外，Slice 去除了 IDL 中的属性，双向参数，上下文等概念。IDL 中复杂的 Any 类型在 Slice 中也用其它的机制替代了。

除了以上的区别外，Slice 还增加了其他一些特性，如：定义名值对的字典类型，支持实现继承和接口继承的类类型，对异常继承的支持，支持继承和聚合等。这些增加的特性使得 Ice 的故障恢复功能更强大，使得使用 Ice 开发的应用程序变得更高效，更安全，更容易设计和实现。

4) 分布式对象通信协议

CORBA 在通信协议上采用了 GIOP 和 IIOP 协议，而 Ice 则采用了私有协议。两种中间件的通信协议存在以下一些不同。

- 编码方式的差异

Ice 采用了紧凑的编码方式，无须字节对齐；而 CORBA 规范使用 CDR 编码方式编码，CDR 的特点是需要字节对齐，如 short 类型数据要在 2 字节边界对齐，int 类型数据要在 4 字节边界对齐等等。

- 协议消息的差异

Ice 使用了五种协议消息：请求、批请求、答复、验证连接、关闭连接；CORBA 规范规定的协议消息种类有请求、答复、取消请求、定位请求、定位答复、关闭连接、错误消息和分片消息等，内容都与 Ice 有所出入。

可以看出，Ice 编码规则简洁紧凑。所有的数据都用固定的字节序，采用 byte 对齐，不作任何边界填充，保证了编码体积最小化，带宽利用率很高。而 CORBA 需要字节对齐和填充，使得它在网络利用率上浪费了一些带宽，降低了编码速率。从另一方面看，尽管在内存中对齐字节会造成一定冗余，但是有可能提高 CPU 访问内存读取数据的效率。

CORBA 的 IIOP 协议有着完善的协议消息机制，Ice 的协议状态机制则相对简单，只有 5 个消息类型。但是，IIOP 缺少对请求的封装，不利于类型化信息的传送。Ice 支持在线路上进行压缩，支持 UDP，且构建了高效的事件转发机制。在诸多方面都进行了仔细设计，使 Ice 在性能和效率上都有很大程度的提高。同时，Ice 在协议的设计上也考虑了穿越防火墙的问题。由于 CORBA 地址信息在 IIOP 中的位置不固定，在穿越防火墙时带来了一些问题。Ice 的协议构建是在充分认识到这一点的基础上进行的相应设计，并解决了这一问题。

5) 服务的比较

CORBA 和 Ice 都提供了基本的对象定位服务和消息服务。CORBA 的 Naming Service 和 Ice 中的 IcePack 都提供了通过对象名字来查找、定位以及连接对象的功能。这两项服务都实现了类似对象(适配器)名称到伺服对象服务地址的映射。在消息传递方面，CORBA 提供了事件服务和通知服务，后者比前者在服务质量上更为可靠，事件通道是两种服务都要使用的服务中介；Ice 的 IceStorm 服务已经具有了基本数据分发服务(Data Dispatch Service：DDS)的能力，通过发布/订阅(Publisher/Subscriber)模式实现消息的传递。两种中间件分别用不同的技术实现了消息发送者和消息接收者之间的去耦。

2. 中间件的选择

通过以上对 CORBA 和 Ice 的简要介绍，以及对两者在中间件核心、对象适配器、接口定义语言、分布式对象通信协议和提供服务的比较，下面将从网络环

境和网管应用需求等角度来分析：怎样在综合网络管理系统中选择合适的中间件平台。

在以光纤通信为主的骨干网络中，通信环境宽松，带宽充足，线路冗余度高。采用有线通信，信息的安全性可以得到相应的保障。这样的网络中拓扑结构相对稳定，不会频繁出现大规模设备移动的情况。在网络设备方面，骨干网以具有丰富计算资源的大型网络设备为主。这些情况下，在可利用资源相当丰富的网络环境中，网络管理系统运行速度和网络数据获取能力都可以得到充分的满足。人们对网管系统的要求是稳定、有效的监控。

另一方面，有些特殊的网络(如海军岸海网)，这样的网络以无线通信为主，通信条件较为恶劣，可利用的通信频率非常有限，带宽也十分不足。同时，无线通信的信息保密难度较高，通信安全存在一定隐患。无线环境的另一个特点就是网络拓扑的不确定性。拓扑结构不固定，网络中的设备会因为各种情况而产生移动，使得原本的网络连接关系不复存在[14]。因为频繁的移动，使得网络的特征数据随之改变，网管系统必须重新获取相应的数据以管理这些网络对象。相比于骨干网，无限通信网中的设备多为内存小，计算资源稀少的嵌入式设备。要在这些设备上运行多种网管程序的代价和难度相对比大型设备多出许多。多变而不稳定的网络环境使得网管系统的实时性监控要求凸现出来。

CORBA 是 OMG(对象管理组织)提出的中间件标准，其核心 ORB 具有广泛的兼容性，可以使多种 CORBA 产品实现互通。对象适配器 POA 的策略多样，可以提供多种伺服对象实现方式。IIOP 提供了基于 TCP/IP，面向连接的可靠通信传输。CORBA 的各类服务十分全面，几乎涵盖了分布式计算的绝大部分应用领域。CORBA 各项标准的提出都经过了大量协商，所以有一定的通用性。但是，由于 CORBA 的规范日益庞大，运行 CORBA 所需的软硬件平台要求也随之水涨船高。

Ice 不同于 CORBA，它具有简单的对象模型和类型系统，小而强的运行 API，简单的语言映射，紧凑高效的协议，丰富的调用和派遣模式，完善的安全解决方案，多种高效实用的服务和工具。同时，Ice 中也借鉴了许多 CORBA 的优秀理念。Ice 的优势在于简单紧凑的结构，和高效的通信性能。比 CORBA 简洁的体系框架降低了运行 Ice 的软硬件平台的门槛。所以，相对于 CORBA，Ice 更适合那些对性能要求较高的分布式应用程序开发。

所以，CORBA 的优势在于全面的服务以及广泛的兼容性。基于 CORBA 的网络管理能够解决传统网络管理技术所面临的一些问题如：支持大规模网络管理任务的灵活分布；为异构平台及遗留系统提供集成；软件总线机制为系统的扩展和升级提供了一定空间等等。但是 CORBA 技术也并非尽善尽美的，兼容性也有其较高的代价。由于考虑到多厂商产品之间的兼容性，CORBA 规范的定制把大量的精力放在“达成一致”上，复杂的规范带来的直接问题就是其实现产品结构复

杂、冗余度高、效率低。Ice 则突出了它高效的通信能力和执行效率。对于通信条件苛刻的网络环境，Ice 能够充分发挥其运行代价小，简洁、高效、易于使用的特点，使得网管系统之间的数据交换做到低成本，高效率。

3.2 多技术网络管理代理设计

3.2.1 网络管理发展历程

商业网络管理系统的发展与网络技术、计算机技术、网络管理需求、市场环境的发展和变化等密切相关，不同的外部环境下形成了不同的网络管理思想。按照网络管理体系架构的差异以及不同的时间段，网络管理系统的发展历经了以下几个阶段：第一阶段主要是厂家专用和私有的运行维护管理系统；第二阶段是利用 SNMP 和 CMIP 这样小型管理者－代理模式建立的简单网络管理方式；第三阶段则是以 ITU-T 的 TMN 为代表的大型网络管理体系；第四阶段，对网络简单管理功能逐步转向全面的网络运营支撑发展，以 TMF 的 NGOSS 为代表。

1．简单网络管理[2]

上世纪 80 年代早期，设备市场基本处于垄断环境中、竞争压力较小、服务业务单一、管理需求相对简单。整个行业习惯于效率较低的网络维护和管理方式，引入新技术、新业务缓慢。到 80 年代后期，市场管制放松，垄断被打破，竞争日益激烈。在网络管理行业，出现了管理多厂家设备，实现网络运行、管理、维护自动化的业务需求，产生了一些通用的接口标准来保证这些相对复杂的管理需求的实现。这一阶段以 SNMP 和 CMIP 为代表，并采用面向对象的软件开发技术。

CMIP(Common Management Informaiton Protocol 公共管理信息协议)和 SNMP(Simple Network Management Protocol 简单网络管理协议)都是以管理者－代理模式运作的网络管理方式。管理者根据管理用户的需求向运行在被管设备上的代理程序发出请求，代理程序查询或设置设备中的 MIB－管理信息库以获得所需数据，然后将其返回给管理者。Manager-Agent 方式的管理结构如图 3.3 所示。CMIP 是 OSI 公共管理信息服务的协议部分。由于 CMIP 追求成为适用广泛的网络管理协议，使得它的规范过于庞大，造成现有的系统中很少有能轻松地运用最终实现出来的 CMIP 协议。因此，CMIP 的推广受到很大限制。相对于复杂的 CMIP，简单网络管理协议 SNMP 由于其实现较为简单，使其开发者从与 OSI 兼容的限制中解脱出来。随着 SNMP 的发展进程越来越快，SNMP 在不同厂商的产品内得到推广运用，并在 Internet 广泛流行。

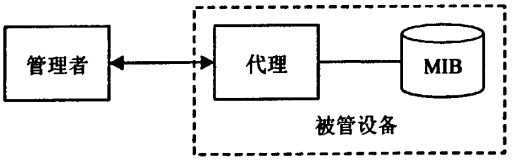

图 3.3 管理者-代理结构

2. 电信管理网

TMN(Telecommunication Management Network)是 CCITT(现更名为 ITU-T 国际电信联盟)在上世纪八十年代推出了一系列电信网络管理的技术方案[1]。它对管理业务、接口描述方法、管理功能、通用信息模型等进行了描述和规定。TMN 的提出将网络管理从集中式的管理体系发展到了分层体系结构。TMN 逻辑分层模型如图 3.4。从图中可以看出，TMN 将网络管理分为五个层次，每一层完成相应的功能，并为其上层提供实现高级应用所需的基本服务。

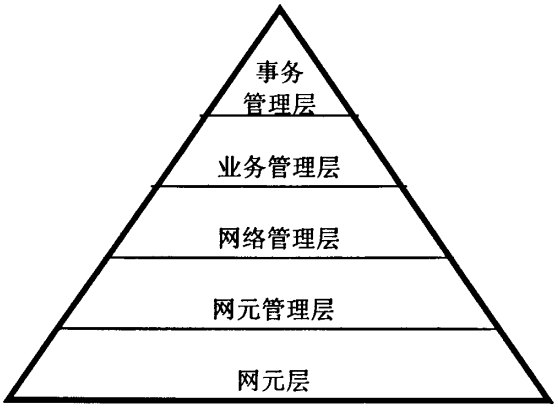

图 3.4 TMN 逻辑分层模型图

由于历史原因 ITU-T 采取了自底向上的视点(从网元管理层向事务管理层)的管理信息建模。从标准化工作的程度看，ITU-T 制定的一系列 TMN 标准仅仅覆盖了一个狭义的网络管理范围。大量工作关注于被管网络本身，而对网络上的业务管理以及更高抽象级别的商务管理仍然有很大的发展空间。TMN 的信息体系结构也不能完全支持分布式的管理环境。随着分布式计算技术的不断发展和网络管理需求的新变化，诸多不足限制了 TMN 在新一代网络的分布管理环境中的应用。

3. 新一代网络运营支撑系统

新一代运营支持系统(New Generation Operations Systems & Software，NGOSS)是在 2000 年由电信管理论坛(Telecom Management Forum，TMF)提出的，用于迅速、灵活地集成和整合原有 OSS，以及指导新一代 OSS 的设计、开发、实现、测试和部署的体系框架和分析方法[7]。

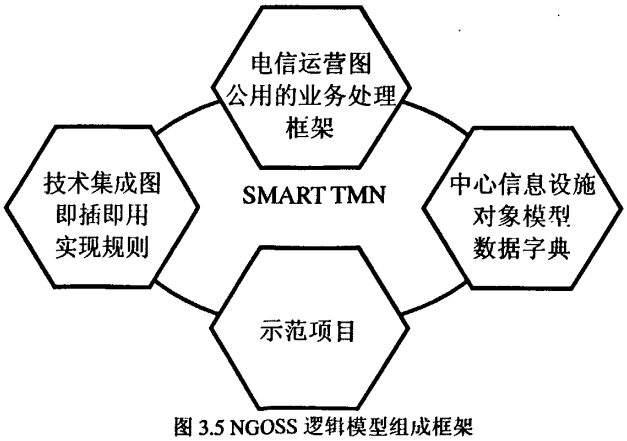

图 3.5 NGOSS 逻辑模型组成框架

虽然 NGOSS 不是专门的网络管理标准协议，但是可以看到网络管理系统与电信运营系统目前面临问题极为相似。网络管理为更高效地实现电信运营提供支持，它们之间的界限已经不十分清楚，网络管理系统已开始向运营支撑系统靠拢。

1) NGOSS 逻辑模型

TMF 针对 TMN 的弱点，从另外一种视点对 TMN 模型的高层管理信息建模和组织管理功能，从而丰富了整个 TMN 的管理能力。NGOSS 解决方案的组成框架按照“面向客户”的原则，以 TMN 的层次结构为指导，采取自顶向下的视点来规划和实施。其逻辑模型的基本框架见图 3.5 所示。

组成框架的核心是“SMART TMN”，表示 NGOSS 的指导思想来自 TMN。核心的周围包括四个重要的组成部分：电信运营图、中心信息设施、技术集成图和示范项目。

2) 电信运营图－TOM[8]

对网络业务应用来说，网络有哪些可用资源，哪些被占用资源，网络资源实现业务数据传输的性能如何才是最关心的事情。因此，综合网管系统需要在已有网管系统的基础上做进一步抽象，把对网络资源统一调度，统一监控的功能提取出来，同时屏蔽网络的具体实现技术，最终实现运营系统的自动化。

TOM 的层次结构模型和 TMN 的层次模型有很多相似之处，存在一种逻辑映射关系，如图 3.6 所示。

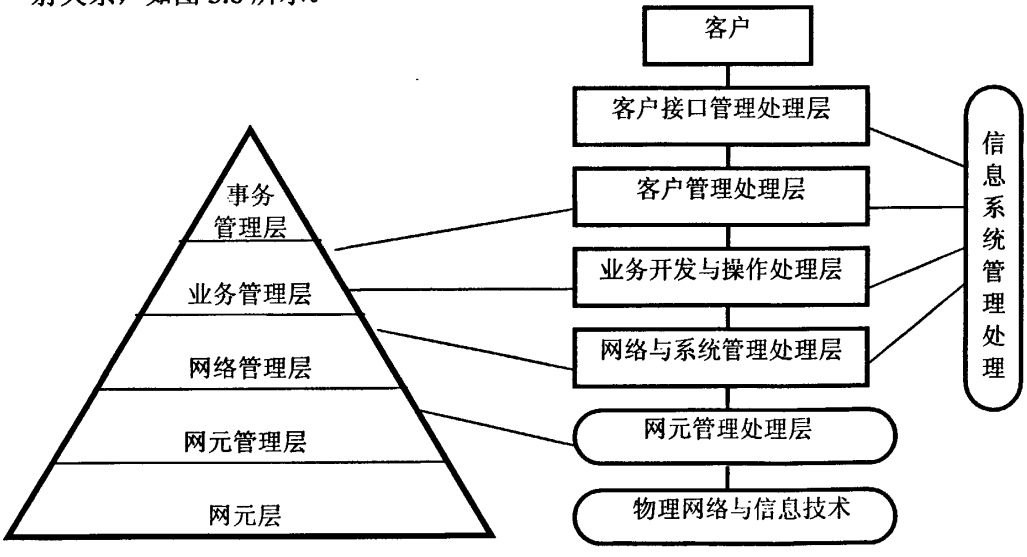

图3.6 TMN管理层次和TOM的逻辑映射

TOM 的视点主要集中在 TMN 的业务管理层和网络管理层。参考 TMN 的分层原则，TOM 将商务处理过程分成以下几个层次：

客户接口管理处理层

客户接口管理处理层是客户访问 TOM 业务处理过程的接口，它将客户的请求转换成内部业务处理过程可识别的事件，触发相关的业务处理过程的执行。

客户管理处理层

客户管理处理层的一组业务处理过程用于和客户直接交互，以实现业务的供应、维护、使用、报告等操作。

业务开发与操作处理层

业务开发与操作处理层的一组商务处理过程用于业务的后台实现和管理，其中有些功能只能被执行一次，其它功能在一个业务的生命周期内可以多次被执行。

网络与系统管理处理层

网络与系统管理处理层构成了业务管理层与网元管理层之间的桥梁，它从网元管理层采集信息，经过综合、关联以及归纳将相关的信息传递给业务管理系统。

网元管理处理层

混合网元管理是 NGOSS 的基础。对于网元管理层在 TOM 中没有给出详细定义。但是 TMF 的多技术网络管理组针对多技术混合网元管理提出的多技术网络管理规范(MTNM)对于网元管理层的构建有很强的指导意义。这一层管理功能的实现是本文描述的重点，下一节将主要介绍 MTNM。

3.2.2 MTNM 多技术网络管理

随着网络技术的发展，管理由多技术网元构成的下一代网络将变得越来越普遍。构成网络的网元由不同制造商提供，多厂商设备之间的互通复杂性上升到管理层面。同时，随着业务提供商利用传送网设备技术的进步，为了提供新业务和优化网络资源利用率，部署下一代的多技术混合网元将成为普遍现象。以前那种只注重单一技术的网络管理解决方案是不够的。现实中需要的是功能丰富、经济、可扩展和不面向单一技术的网络管理解决方案，该方案中多技术、多厂商网络管理系统可以在开放的体系结构环境中进行交互。

MTNM(Multi-Technology Network Management)是 TMF 下属的多技术网管研究组 MTNMT(Multi-Technology Network Management Team)根据 TOM 提出的 EMS 和 NMS 之间的 NML-EML 管理需求接口[9]。NML-EML 接口可以支持一系列的管理功能，它实际上是一个针对多种技术，多种厂商设备并存的网络管理的标准，为简化端到端的管理和多厂商设备并存网络的预配提供了新的机会，旨在完成多技术混合网元的管理。目前，这一多技术网网络管理体系的商业规范已经提出了 3.0 版本。

MTMNT 的目标就是提供一个单一通用方案，用于管理多种传送网络技术和支持多种技术的混合网元。MTNMT 在 TMF608 v2.0 和 TMF814 v2.0 中分别定义了该接口的信息模型和基于 CORBA 的接口定义。

在图 3.7(a)中可以看到如果采用多个技术接口管理多技术混合网元对于系统的扩展将十分不利。如果要添加新的网元类型，在 NMS 上必须添加新的接口用于和新的 EMS 制造商的产品通信，NMS 的内部实现也要进行较大改动，扩展几乎是不现实的。但是，如果采用图 3.7(b)中的单一通用接口管理多技术混合网元，优点是显而易见的。只要 EMS 制造商实现通用接口，NMS 无需任何改动。这种良好的扩展性避免了在网络管理系统建设中普遍存在的重复投入的问题。

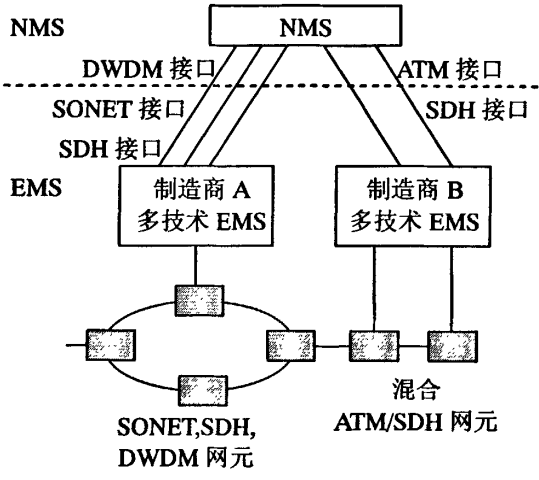

图 3.7（a）采用多个技术接口管理多技术混合网元

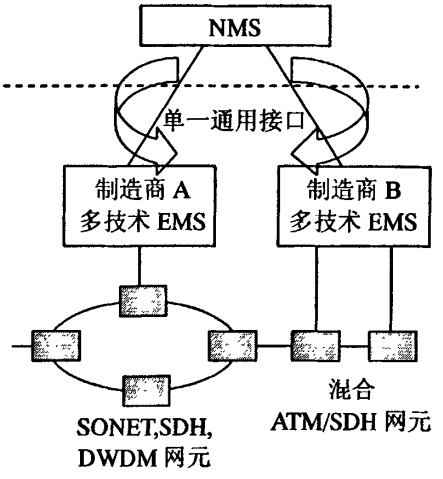

图 3.7（b）采用单一通用的接口管理多技术混合网元

MTNM 就是针对通信领域存在的这种现实的需求，提供的一个单一通用的解决方案[9]。采用单一通用的接口给业务提供商、用户和制造商带来了很多好处。最主要的好处是为多厂商、多技术网元的管理提供一个灵活可扩展的解决方案，完成对多技术混合网络的管理。MTNM 的核心思想是对被管网络的物理资源和逻辑资源进行抽象建模，模型集中体现的是同类被管资源的共性，以及不同类被管资源之间的从属关系。MTNM 采用粗粒度的 CORBA 管理器接口对所有被管资源进行统一管理。无论是被管资源模型还是管理器都具有很好的可扩展性。

MTNM 在 TMF 模型中是位于网络管理层和网元管理层之间的 NML-EML 管理需求接口。它是 TMF 模型中为实现电信运营图中的网络和系统管理流程而设计的体系结构，所以网络和系统的管理流程就是 MTNM 的业务需求。该流程涉及到的管理功能跨越了网元管理层和服务管理层，管理的基本行为是组织来自网元管理系统的各类信息，将它们集成、使之相互关联并加以总结，然后将服务管理系统需要的相关信息传送到上级组件，或者根据信息处理结果直接对网络进行管理。

3.2.3 TMFAgent 多技术网管代理

TMFAgent 是基于 MTNM 核心思想及对象模型的一种具有通用接口的网管代

理和 EMS(Element Management System 网元管理系统)，它位于 TMN 逻辑分层结构中网络管理层和网元管理层之间。TMFAgent 的主要功能是：

- 访问被管网络物理或逻辑资源
- 对所获得的网络结构和数据进行抽象建模
- 通过 CORBA 接口为网络管理层的应用提供相应服务

正如前文 TMN 和 TOM 中介绍的一样，TMFAgent 同样提供网络管理的五个功能域的全面实现。不同于其他管理层次的功能域，在网元管理层网络管理的各个功能域强调的是对设备的监控和管理，并为其上层(即网络管理层)的功能实现提供相应的综合服务。下面将从设计模式和接口描述的角度介绍 TMFAgent。

1. 模式应用

代理模式

代理技术是综合网管领域最常用的软件实现方法之一。从软件设计模式角度分析，代理技术亦可称为代理模式，它是系统的结构模式。代理模式给某一个对象提供一个代理对象，并由代理对象控制对原对象的引用[12]。它能为 TMFAgent 提供的主要优势包括：

- 加强封装－对于综合网管系统这样的网络应用，代理可以将网络的细节隐藏起来。
- 安全可靠－可以为综合网管系统提供运行时对用户有关权限进行检查，给不同的用户提供不同级别的使用权限。
- 性能优化－通过代理控制网管系统根据需要创建一个资源消耗较大的对象，使得此对象只在需要时才会被真正创建，优化系统对象加载过程
- 同步保护－为多个网管用户操作的同一个目标结果提供临时的存储空间，使几个用户能够同时使用一个对象而没有冲突。

TMFAgent 作为实际网络环境中各种设备的抽象代理封装了真实的网络环境，优化了网管人员访问网络设备方式，使网络管理操作具有更高的安全系数。

Facade 模式 (门面模式)

TMFAgent 的系统接口部分采用了设计模式中的 Facade 门面模式。门面模式要求系统外部与一个子系统的通信必须通过一个统一的门面对象进行[12]。换言之，门面模式提供一个高层次的接口，使得子系统更易于使用。门面模式中主要包含两个角色：

- 门面角色－客户端可以调用这个角色的方法。此角色知晓相关的子系统的功能和责任。
- 子系统角色－子系统是一个类的集合，子系统不知道门面的存在。

在 TMFAgent 抽象代理系统中，系统接口表现为门面角色；TMFAgent 核心则表现为子系统角色。

由于 TMFAgent 是具有综合网管功能的代理系统，在各管理功能域上都具有复杂多样的功能接口。利用门面模式可以为这样一个复杂的综合网管系统提供一个简单接口。首先，综合网管系统的接口往往会因为不断演化而变得越来越复杂，使用门面模式可以使得子系统更具可复用性。同时，为了保持 TMFAgent 系统的独立性，将其与它的客户端以及其他的子系统分离，可以提高它的独立性和可移植性，使其独立于其他系统组件开发与维护。另外，门面模式在层次化 TMFAgent 系统时，减少了客户层与服务层之间是相互依赖的，限定仅通过 Facade 进行通信，从而简化了层与层之间的依赖关系。

门面模式是软件设计原则中迪米特法则的体现，它将客户端所涉及的属于一个子系统的协作伙伴的数目减到最少，使得客户端与子系统内部的对象的相互作用被门面对象所取代。

2. 接口概述

在 TMFAgent 中，NML-EML 接口是根据 MTNM 的协议规范定义和扩充的。接口定义采用了 Ice 的 Slice 语言，规定了多技术网络管理体系中网元管理层向网络管理层提供的可用接口。接口分别按照网元管理接入控制，拓扑管理，故障管理，性能管理，以及配置管理等方向展开。

(1) 系统管理接口：用于管理 TMFAgent 系统的标识和配置信息，向用户提供系统的管理范围信息。

(2) 拓扑管理接口

- 子网管理接口：管理层子网资源，提供对子网配置、子网范围的管理，管理子网连接资源(SNC)的分配、开通、删除等。
- 被管网元管理接口：管理被管网元及其内部端点(Termination Point)对象、交叉连接(Cross Connection)对象以及连接被管网元的拓扑连接(Topological Link)对象。

(3) 故障管理接口：向用户提供指定故障监控对象，设置故障监控策略以及开始和停止轮询的命令等。

(4) 性能管理接口：向用户提供设置性能监测点和监测内容，设置性能告警门限的方法，按照用户定制的策略采集性能数据并提供给用户，实现性能监测告警。

(5) 配置管理接口

- 设备清单管理接口：管理设备(Equipment)及设备容器(Equipment Holder)的清单、状态、安装、配置，进行设备预分配。
- 保护管理接口：向用户提供网络的保护配置信息，执行用户发出的保护操作命令，提供保护交换的数据信息。
- 维护管理接口：向用户提供系统可以支持的维护管理操作，并执行用户发出的维护管理命令。

3.3 本章小结

本章主要对综合网管系统中分布式通信架构采用的基础设施做了介绍和分析。首先本章讨论了分布式中间件技术在网络管理中的应用，分析了流行的CORBA 和 Ice 分布式对象中间件各自的特点和不同的适用性。另外，在介绍综合网管代理技术时，简要介绍综合网络管理技术的演化过程，并根据 MTNM 多技术网络管理提出了 TMFAgent 抽象代理技术。

第四章 网元管理系统中的关键技术

4.1 网元数据建模－抽象网元模型

目前的网络环境正朝着大规模，多技术，多厂商的方向发展。在这样的网络中，要对被管的网络对象进行数据建模并非一项容易的工作，在网管数据的统一上存在交大难度。在 TMF 定义的 MTNM 规范中，对网络对象建模提出了二类对象的概念，其中基本网元(ManagedElement)模型是该综合网管系统中的基本数据类型，但其只试用于描述 SDH 传输网络和 ATM 宽带业务中的网元。本节中在 MTNM 规范网元模型的基础上，根据软件设计原则和设计模式对这一模型进行了扩展和改进，提出了抽象网元模型，并用 Java 实现了该模型。

4.1.1 基本网元模型

MTNM-EMS 中的基本网元模型为描述单一技术的网络结构提出了较好的解决方案，但对于多技术、多厂商的网络环境，它的表现能力就显得较为薄弱。

1. MTNM 中的二类对象

二类对象是指在 MTNM-EMS 中，用以描述被管网络资源(包括物理资源、逻辑资源)的 Java 对象。二类对象类型众多，如表示子网的 Subnetwork 对象、表示网元的 ManagedElement 对象、表示逻辑连接的 TopologicLink 对象等。

MTNM-EMS 中二类对象的设计，采用了基于 CORBA 的网管接口粗粒度建模方法。所谓粗粒度接口是指在 MTNM -EMS 与上级 NMS 的接口中，用以描述网络资源的仅仅是 Ice 接口中传递的一些数据结构[10]，而不是 CORBA 对象。二类对象继承自这些从 CORBA 数据结构映射到 Java 的基本对象，因此二类对象除具备已有的数据成员外，还可以具有特定的方法来描述网络资源的行为。MTNM-EMS 通过二类对象具体属性、类型和状态以及二类对象具有的相应方法来呈现被管网络的状态和行为，从而在 MTNM-EMS 中构建起由二类对象组成的一个虚拟网络。

2. ManagedElement 对象

ManagedElement(简称 ME)是二类对象中核心成员之一，TMF 规范中将其定义为具有相同位置物理资源的抽象集合。换言之，ME 对象描述的是所有独立的网元设备。在多技术网络管理系统中 ME 对象可能代表许多种设备，如以太网中的普通主机，无线局域网中接入点(AP)，接入控制设备(AC)，路由器，SDH 网络中的分插复用设备等。尽管这些设备的结构都各不相同，TMF 规范对它们的共同点进行了抽象，得出了 ME 对象。图 4.1 中给出了 ME 对象的 UML 类图。

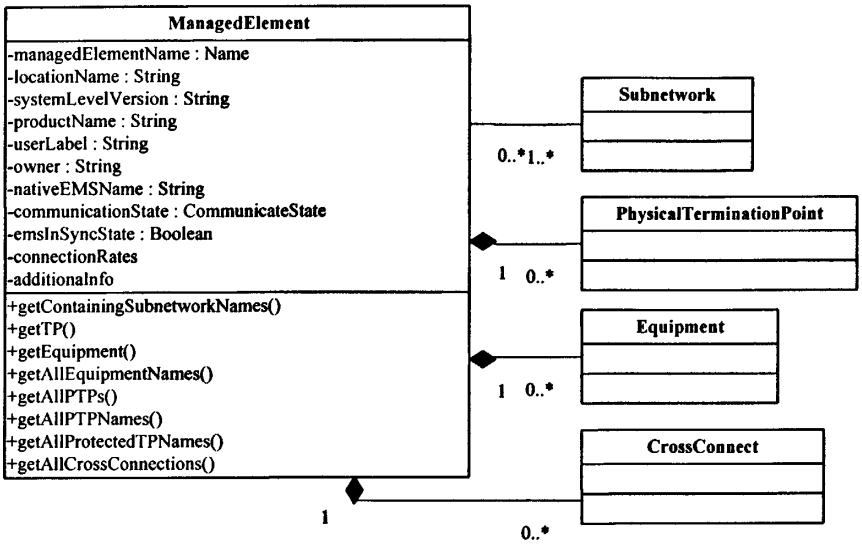

图 4.1 ManagedElement 对象 UML 类图

从 UML 类图中可以看出，ME 对象的属性涵盖了大多数普通网元的通用属性，且不涉及网元的具体技术特性。在 ME 对象与其他二类对象的关系方面，一个 ME 对象可以根据不同的属性，如层速率、设备类型、设备制造商等关联不同的子网(Subnetwork)。在 ME 内部可以存在多个设备(Equipment)或设备容器(EquipmentHolder)、物理端点(PhysicalTerminationPoint)、交叉链接(CrossConnect)。ME 对象作为独立节点需要具备相应的成员变量和方法，来保存和修改所有这些二类对象。

虽然，ME 对象内容较为丰富，但仅用这样的结构无法适应多技术、多厂商的网络环境。因为在技术多样的环境中，一些简单的通用属性不仅无法准确描述具体的设备信息，而且会给网管系统的修改和扩展带来巨大的困难。系统开发人员会发现，在综合的网络环境中，哪怕仅仅是对 ME 对象做一些很小的修改，都会给整个系统造成很大的影响。

4.1.2 抽象网元模型

为了开发适用于复杂多变网络环境的综合网管系统，抽象网元模型以 MTNM 规范中定义的二类对象被管网元 ManagedElement 为蓝本，继承并扩展了这一基本网元模型。根据软件设计原则，合理应用软件设计模式，利用抽象网元、类型接口以及适当的继承实现机制优化了原有的综合网络体系中网元的描述方式[16]。抽象网元模型能较好适应变化的多技术、多厂商网络环境，降低过快的网络环境变化给开发综合网管系统带来的难度，使网管软件自身的维护变得更加方便。

抽象网元模型＝< 抽象网元，类型接口，二级继承类型实现机制 >

1．抽象网元

抽象网元将原来的 ManagedElement 类声明为抽象类(Java abstract class)。它仍用来定义网元的通用属性和行为。不同的是，根据 Java 的特点，抽象网元可以实现部分网元的通用行为，但是抽象网元不能被实例化。换言之，抽象网元不能直接被用于描述实际网络中的网元设备，它只用来定义通用的网元模板，提供网元类型的部分实现，这部分实现的方法将成为所有网元对象的缺省行为实现。在抽象网元中定义而未实现的方法称为抽象方法，它们将由抽象网元的子类实现。

2. 类型接口

类型接口是根据各种技术网元的不同行为特征定义的网元行为接口(Java interface)，它是某种特定网络技术网元的行为集合。类型接口当然来自于网元的具体行为，但它只描述了这些行为的特征，而没有实现这些行为。因此，当这些方法在不同的地方被实现时，可以使拥有相同接口的对象具有完全不同的表现。采用类型接口的另一个目的是，在 Java 中一个类至多只能有一个超类，但同时可以实现多个接口。这样的单继承多实现的规定，十分符合实际情况。利用这一点，当一个网络中的网元集多种功能于一身时，只需要让描述这一网元的类型实现多个类型接口就可以达到目标。

3. 二级继承类型实现机制

抽象网元模型中最基本的是 TMF 规范中的 ManagedElement 类，它是所有网元类型的基类。在模型中它被定义为抽象类，也可以称为一级抽象网元。利用对一级抽象网元 ManagedElement 的继承，对不同类型接口的实现，抽象网元模型中得到了二级抽象网元类型的定义。在这一层次的抽象类型中，多技术类型的网元得到了区分。这些二级抽象网元继承了根网元中对网元通用属性的定义，并且加入了用于描述特定网络技术的网元属性。同时，由于实现了特定的类型接口，因此二级抽象网元还拥有了特定的网元行为描述。

二级抽象网元和根网元一样，也不能被实例化。这些二级抽象类型定义了特定网络技术中网元类型的模板，它们包含通用网元属性，特定网元属性以及未实现的网元行为，即网元模型中声明的抽象方法和类型接口中定义的网元接口。

抽象网元模型中可实例化的网元类型继承自二级抽象网元，这些具体的类必须实现其超类中声明的抽象方法来完成对一个网络实际网元的描述。图 4.2 中给出了抽象网元模型的实现示意。

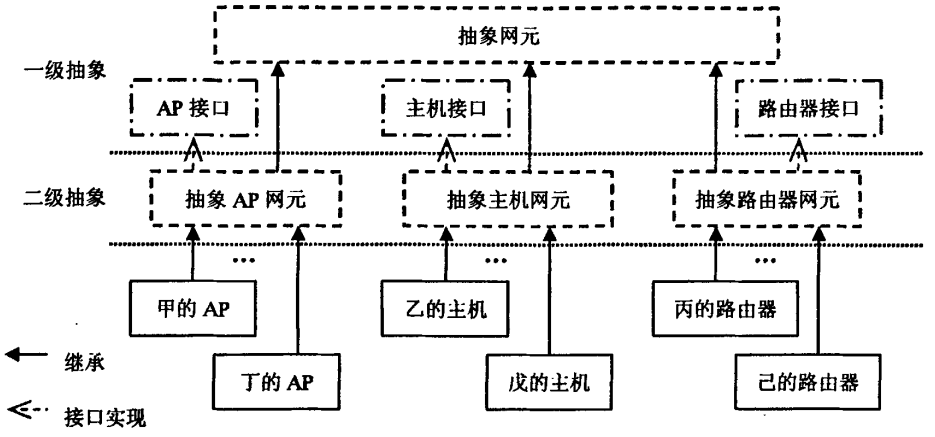

图 4.2 抽象网元模型的实现示意图

4．遵循的软件设计原则

为了解决综合网管系统可维护性差，复用效率低的问题，抽象网元模型中遵循了如下的软件设计原则。

“开－闭”原则：一个软件实体应对扩展开放，对修改关闭

在设计网元类型时，应该在不修改原有代码的情况下，改变网元模型的行为。要做到这一点，必须扩展已有的网元模型，使其提供新的行为，从而使变化中的网管系统有一定的适应性和灵活性。同时，对已有的网元模型，尤其是最重要的抽象层网元尽量不能修改，这样才能保证网管系统的稳定性和延续性。

里氏替换原则：一个软件实体如果使用一个基类的话，那么一定适用其子类，而且它根本不能察觉出其基类对象和子类对象的区别

里氏替换原则是抽象网元模型遵循的另一个重要的设计原则。二级抽象网元和具体网元类型的设计都应尽可能的符合这一点。只有当具体网元可以替换掉抽象网元，且系统的功能不会受到影响时，抽象网元才能真正被复用，而具体网元也才能在抽象网元的基础上扩展新的行为。

依赖倒换原则：要依赖于抽象，不要依赖于具体

在网元模型的应用中，高层次的网管功能模块应尽量根据一级抽象网元实现，使之与网元模型建立抽象的耦合关系。这样使得高层次的网管功能无需依赖与具体的网元类型，具体网元类型的变化对于网管功能是透明的，提高了重要的网络管理逻辑的适应性。

5．模型中的设计模式

网元的创建－工厂方法模式

工厂方法模式是类的创建模式，其用意是定义一个产品对象的工厂接口，将实际创建工作推迟到子类中[12]。将工厂方法模式应用到抽象网元模型中，首先定义一个工厂方法接口，其返回值类型是抽象网元类型。对每一种具体网元类型都

应创建一个具体工厂类来实现这个工厂方法。具体工厂类对工厂方法的实现根据不同的网元技术类型而各有差异，因此创建出的各种网元也不尽相同。但是，它们终将以抽象网元的形式交付上层网管功能使用，从而达到依赖于抽象编程的目的。对于新的网元类型，开发人员不必修改现有的网元创建方式，只需创建新的工厂类就可以做到对修改关闭，对扩展开放。

网元的缺省行为－缺省适配模式

缺省适配模式是类的结构模式。缺省适配模式为一个接口提供缺省实现，这样子类型可以从这个缺省实现进行扩展而不必从原有接口进行扩展[12]。抽象网元模型中存在两级抽象网元，一级是通用抽象网元，二级是技术抽象网元。根据 Java 语言的特点，允许在抽象网元类型中实现一些方法或接口。这些实现将由其子类，也就是具体网元类型继承。这样，在抽象网元中实现的网元行为就会成为某一类技术，甚至是全体网元的缺省行为。若某一具体网元类型与缺省行为存在着差异，那么只需覆盖实现其超类即抽象网元的缺省实现即可。

网元的行为模板－模板方法模式

模板方法模式是类的行为模式。准备一个抽象类，将部分逻辑以具体方法以及具体构造器的形式实现，然后声明一些抽象方法来迫使子类实现剩余的逻辑[12]。不同的子类可以以不同方式实现这些抽象方法，从而对剩余的逻辑有不同的实现，这就是模板方法模式的用意。抽象网元作为所有网元的超类定义了网元某些行为的模板。抽象网元无需实现所有的行为，它可以留下一些抽象子行为强迫其继承者去实现。作为抽象网元子类的具体网元必须根据自己的技术类型实现这些抽象子行为，帮助抽象网元实现未完成的行为。各种技术网元提供各不相同的子行为，对于上层网络管理逻辑将会反映出不同结果。

4.1.3 合理应用抽象网元模型

网元模型作为综合网管系统中基本核心的组成部分，在高级网管功能中，如拓扑管理、配置管理、性能管理、故障管理等，起着举足轻重的作用。只要合理地应用抽象网元模型，就能有效降低开发这些网管功能的代价，使大型综合网管系统的维护和扩展变得容易。

通用抽象网元和特定技术的抽象网元是一对多的继承关系，类似的，特定技术抽象网元与每一种该技术类型的具体网元也是一对多的继承关系。在此类继承关系中，网元的共同行为逻辑应该尽量向抽象网元移动。这样不仅可以提高代码的复用率，同时，在代码发生改变时也减少了设计人员修改代码的工作量。

与通用行为逻辑移动的方向相反，各种网元模型中数据的移动方向应该是从通用到特定，从抽象到具体，也就是从继承结构的高端向低端移动。一个网元模型中的数据不论是否使用都将占用资源，因此将技术相关性强的数据尽量放到具

体网元类型中就可以做到非必要的数据不占内存，节省了系统资源。

一级抽象网元是整个模型中的根元素，它应当包含网元继承体系中对整个系统来说重要的战略性行为逻辑，是必然性的体现；而具体网元类型只含有一些次要的与实现相关的算法和逻辑，以及战术性决定，具有相当的偶然性。因此，抽象层的网元模型在设计时就应当充分考虑系统运行时的需求，保证其代码的稳定性和适应性，以提高代码的复用率和可维护性。

4.2 网元数据持久化－Hibernate

4.2.1 Hibernate 的概念

常见的网络管理软件系统，经常会遇到数据库的应用开发的需求，如网络拓扑数据、性能数据等。网管系统中的数据库开发一般会采取两种策略：一种是采用基于面向对象的数据库，另外一种是采用传统的基于关系型的数据库。目前的系统，绝大多数采用面向对象编程，对象化的数据从直观上来看应当使用基于面向对象的数据库。但是，由于面向对象的数据库技术还不是十分成熟，因此大多数开发还是采用基于关系型数据库的策略。因此，在关系型数据库处理对象化数据时就有必要运用持久层实现面向对象数据库的操作。实现这一持久层的难点在于需要处理关系数据库中各个表之间的错综复杂的关联关系。所以，合理设计持久层是一个至关重要的问题。

在网元管理处理层和关系数据库之间增加一个持久层，可以实现拓扑对象和关系数据库之间的映射。利用这个映射框架的机制，对象与关系数据库之间的转换就可以透明地进行，而不用去关心数据库连接、并发、事务等问题。网元管理系统直接读取或存贮的就是清晰的拓扑对象，中间的转换过程就交给映射框架来处理。

Hibernate 是由 Gavin King 开发的、目前十分流行的一种持久层框架。程序员无须关心对象之间的关系如何在数据库中存储，只要按照最普通的方式使用对象，Hibernate 就自然会把各种关系处理好。Hibernate 的出现给数据库的开发带来了一种崭新的活力，它是一个纯 Java 的 O/R(对象/关系)映射框架，为开发者提供了一个强大而易用的 O/R 映射机制。

Hibernate 作为一个 Java 环境的对象/关系映射器，涉及到把一个数据表示由对象模型映射到关系模型，即 SQL-based 结构的一些技术策略。Hibernate 不仅包含 Java 类到数据表的映射，也提供数据查询和获取的便捷方法，同时缩短了数据处理的时间。Hibernate 是一个优秀的开放源代码的 Java 对象持久层轻量级封装框架，它既可以用来在 Java 应用程序中取代大部分 JDBC 代码，也可以整合到 J2EE 系统中作为持久层框架。

4.2.2 Hibernate 的组成框架

Hibernate 的组成框架如图 4.3 所示，从中可以大致了解一下 Hibernate 的运作方式。Hibernate 的基本框架包括 Hibernate 核心程序，持久化对象，Hibernate 配置和 XML 映射组成。

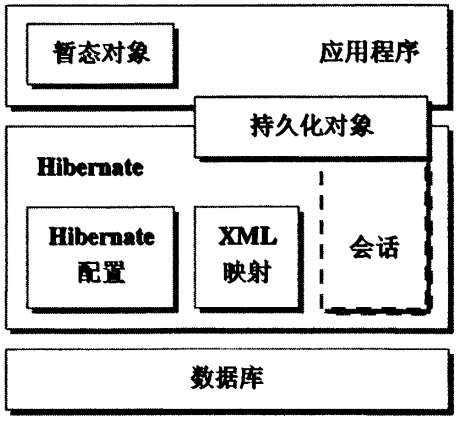

图 4.3 Hibernate 体系框架

Hibernate 的核心是一个会话工厂(Session Factory)，它是一个线程安全的组件，用来创建客户程序与数据库连接的会话。会话(Session)是一个单线程，生命周期相对较短的对象，负责应用程序(对于数据库来说应用程序即是客户程序)和持久存储器(数据库)的对话。它封装了 JDBC 的连接，并维护着持久化对象的数据缓存。核心程序的其他一些可选部分，这里不作赘述。

持久化对象是应用程序中暂态对象和关系型数据库中数据记录纽带，是 Hibernate 使用过程中用户定义部分的关键之一。持久化对象一方面反映着应用程序暂态对象的属性和对象的包含或继承关系；另一方面，它也可以看作关系型数据库中相应的表格设计。创建持久化对象，同写一般的对象类一样，为每一个数据库表写一个类，其中包含表的每一个字段的定义和一些操作的方法，主要是与之对应的 get 和 set 方法。字段变量和方法的定义根据表之间的关联性复杂度而有所不同。持久化对象定义的合理与否对 Hibernate 的使用有着十分重要的影响。

Hibernate 的配置主要依靠配置文件来实现。在配置文件中，用户可以指定数据库相关参数，包括驱动程序，地址及端口，用户名和密码等。同时，用户还可以指明使用 Hibernate 时的一些性能参数，如数据库连接池大小，数据库缓存大小等，这些性能参数的设置对 Hibernate 的实时运行状况都有很大作用。

4.2.3 Hibernate 的 XML 映射

在使用 Hibernate 实现对象持久化的过程中，编写对象与关系表的映射文件至关重要。Hibernate 向用户提供了 XML 文件的编写格式。这一 XML 映射是一种易于阅读，可以直接手工改写的结构化映射文档。XML 映射的描述方式是面向持久化对象而不是关系型数据库表格的。因此，每一个 XML 映射文件都对应一个持久化对象，并以该对象名来命名映射文件，以 hbm.xml 作为映射文件后缀。

对象类映射的 XML 文件定义了 O/R 映射的全部规则。映射文件的根节点是<hibernate-mapping>，根节点下是一个<class>节点，用于描述需要映射的类。最简单的情况下，<class>节点应该有 name 和 table 两个属性，分别用来描述对象所

属的类和它映射到的数据表。在<class>的节点下，描述类中需要映射的各个 field，其中，主键应该用<id>节点来描述，该节点下用<generator>节点声明主键的生成策略。Hibernate 提供了 4 种主键生成策略：Hi/Low、UUID、Identity 和 Assigned，前三种都是利用不同的方法由数据库自动生成主键；而 Assigned 则是用程序代码将主键值赋予对象，存入数据库。对于类中的每一个普通的 field 用一个<property>节点来描述。一般情况下<property>节点有 4 个属性：name、length、column 和 type，name 为必须的属性。XML 映射的强大功能还不在于此，它能更简单的处理对象之间的关联关系。对象之间的一对一、一对多、多对多等关系，只要在映射文件里做一点简单的修改设置(增加几个节点)，就可以实现。当然，有时也要在表对象中增加几个简单的实现方法。

尽管，许多 Hibernate 用户手工定义 XML 映射文件，但是目前有许多很好的工具可以用于自动生成映射文件。这些工具为开发人员带来了很大便利。

4.3 本章小结

本章主要介绍了网元管理系统中使用的两项关键技术：抽象网元模型和面向对象数据持久化技术。根据软件设计的一些重要原则提出的抽象网元模型，用于改善综合网管系统中网元模型的设计和开发，提高综合网管系统的适应性，可复用性以及可维护性。该模型的主要特点是：遵循软件设计原则，合理运用设计模式，网元模型结构灵活，适应性强，复用率高，可维护性好。面向对象数据持久化技术 Hibernate 为基于面向对象开发的网管系统数据保存工作带来了很大的便利，大大提高了网管系统的开发效率。

第五章 抽象代理 TMFAgent 的设计与实现

5.1 系统整体框架

抽象代理 TMFAgent 整体框架如图 5.1 所示。整个系统可以分成以下几个组件：分布式管理接口模块，网管功能管理器模块，二类对象模块以及数据访问模块。

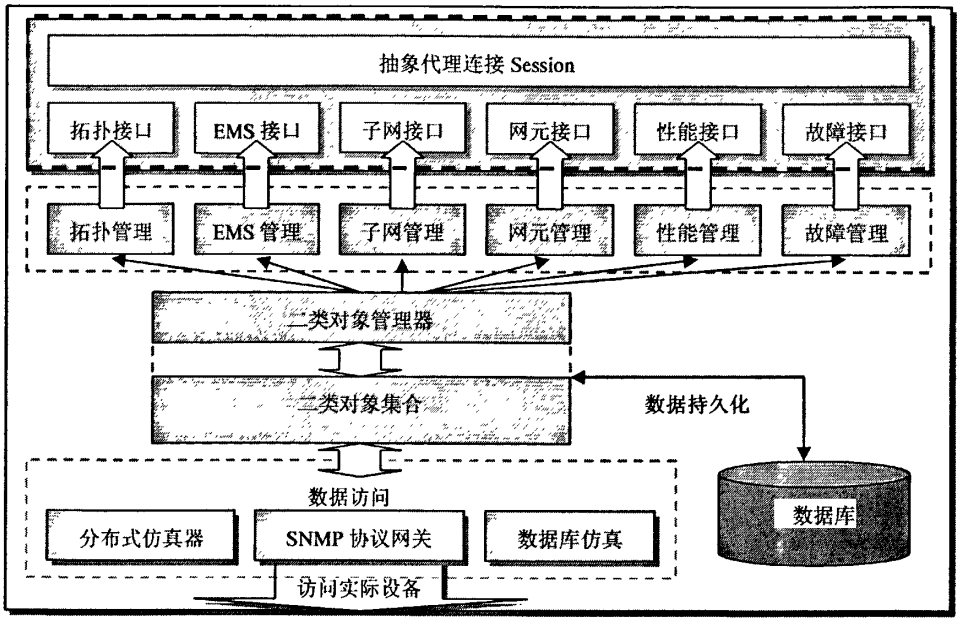

图 5.1　TMFAgent 整体框架

1．分布式管理接口

根据网管系统分布式运行的需求，TMFAgent 定义了一系列系统接口和网络管理功能接口。根据网络环境的不同，可以采用 CORBA 或者 Ice 来实现这些接口的定义和映射。通过第三章中对中间件在各种网络环境中适用性分析可以知道，在海军综合网络环境下，Ice 是较为合适的中间件。在所有的接口中：

- 系统接口负责客户端的接入和管理。
- 网管功能接口负责接收管理操作并返回相应的运行结果。

2．网管功能管理器

TMFAgent 通过各个功能管理器实现五大网络管理功能，系统中的管理器继承自上述网管功能接口，实现了分布式接口定义的所有操作。这些管理器实际上就是分布式对象中间件在服务器端的伺服程序。

3．二类对象

二类对象是 TMFAgent 的核心。TMFAgent 中定义了大量的二类对象，通过它们来描述实际网络的状态和行为。所有的这些二类对象由二类对象管理器负责管理，各个网管功能管理器则通过二类对象管理器访问二类对象数据。

TMFAgent 中对二类对象的持久化通过将其数据保存到数据库实现。同时，对网络的性能和故障监测所获得的数据也将保存到数据库中。

4. 数据访问

数据访问模块在这里是一个抽象的概念，它是 TMFAgent 和实际网络或网络仿真数据的沟通渠道。简言之，数据访问模块向二类对象提供了一个透明的网络数据来源。

5.2 系统接口定义与实现

5.2.1 分布式连接接口

TMFAgent 中分布式管理接口是按照 MTNM 规范定义的，主要负责完成与客户端或上级网管系统的交互。根据 MTNM 规范定义，TMFAgent 接口可以分为两部分：会话接口和管理器接口。两部分接口的关系如图 5.2。这里首先介绍一下会话接口的组成和作用。

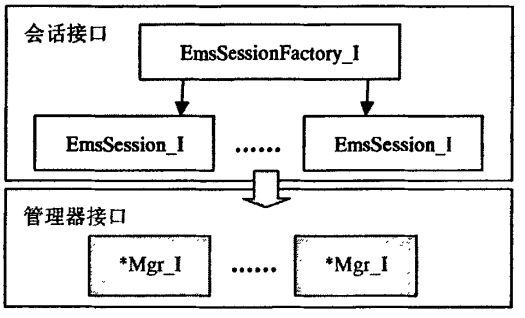

图 5.2 分布式接口

会话接口负责维护客户端(或上级网管系统)与 TMFAgent 之间的会话连接，并充当管理器接口工厂的角色。因为，一个 TMFAgent 可支持多个客户端，系统对于每个客户端都维护了一个 EmsSessionI 接口实例，以保存该客户端的一些连接信息。会话接口涉及的对象主要包括 EmsSessionFactoryI(会话对象工厂)，EmsSessionI(服务器/代理端会话对象)，NmsSessionI(客户端会话对象)。

EmsSessionFactoryI 是注册到名字服务(Ice 的 IcePack)的持久对象，也是整个系统的接入点对象。它监听客户端的建立连接请求，若请求连接的客户通过了安全认证，则为其创建一个 EmsSessionI 接口实例。EmsSessionI 接口为客户提供管理器接口的 Ice 代理，并保存对应客户的回调接口 NmsSessionI 的 Ice 代理。

5.2.2 管理功能接口与管理器实现

管理器接口主要定义了一系列网管操作，这些操作是抽象代理 TMFAgent 对网络管理系统的服务定义。网管功能管理器是这些接口的伺服程序，负责完成具

体的网络管理功能。管理器接口和管理器伺服程序可以按管理功能或管理目标对象的不同进行分类，管理器的类型和数量可以根据抽象代理 TMFAgent 的功能需求而定。常用的管理器包括：

- EMS 系统管理器，提供对本抽象代理 TMFAgent 所辖的第一级子网客体进行管理的操作；
- 多层子网管理器，对所有的子网对象进行统一管理的接口；
- 被管元素管理器，针对网元和网元内部的端点对象进行管理的接口；
- 性能管理器，提供对性能数据监测、采集等管理操作；
- 故障管理器，实现对网络中各种设备的轮询监视；
- 拓扑管理器，提供拓扑发现，拓扑创建，拓扑数据载入及保存等接口

TMFAgent 中对每种类型管理器仅维护一个实例。所有与客户端对应的 EmsSessionI 接口都可以访问到这些管理器接口实例。管理器接口实例作为 EmsSessionI 接口实现类的静态成员可以被所有 EmsSessionI 接口共享。换言之，同一个 TMFAgent 的所有客户端将共享此代理上的管理器。EmsSessionI 接口实现类维护了一个映射表，保存管理器接口的名字和伺服类程序之间的映射。客户端可以通过 EmsSessionI 接口中的 getSupportedManagers()方法得到 TMFAgent 所支持的管理器名，并通过 getManager(String managerName)方法得到相应管理器接口的对象引用。因同一代理中的管理器接口被所有的 EmsSessionI 接口共享，故管理器接口的伺服类程序在第一个 EmsSession_I 接口的实现类被实例化的同时创建。在此以后，其它 EmsSessionI 接口就可以直接访问这些已创建的管理器接口对象。这种连接方式实际上是 Facade 门面模式的实现，它为客户端提供了发现和查询抽象代理所提供服务的功能。

5.3 二类对象的实现

5.3.1 二类对象的概念

二类对象是指在 TMFAgent 中，用以描述被管网络资源(包括物理资源、逻辑资源)的 Java 对象。TMFAgent 中二类对象的设计，采用了基于分布式网管接口粗粒度建模方法。所谓粗粒度接口是指在 TMFAgent 与客户端(或上级网管系统)的接口上，用以描述网络资源的不是分布式对象(Slice 定义的 Interface)，而仅仅是分布式接口中传递的一些数据结构(struct)。这些数据结构仅用来描述网络资源的类型、属性和状态，却不反映网络资源的行为[13]。

二类对象继承自这些从 struct 中映射到 Java 的对象，因此二类对象除具有自己的数据成员外，还具有一定操作方法，可以被其它对象(包括分布式接口的伺服对象、Java 对象)调用。TMFAgent 通过二类对象具体属性、类型和状态以及二类

对象具有的相应操作来呈现所管网络的状态和行为，从而在 TMFAgent 中构建起由二类对象组成的一个虚拟网络。

采用分布式粗粒度接口模型，是为了减少服务器和客户端之间的通信给网络带来的额外压力。与粗粒度模型相对应的是细粒度接口模型。细粒度模型将每一个的网络资源都描述为一个分布式对象，这样的做法最大的缺点是对系统的开销较大。因为相对于简单的数据结构来说，分布式对象不仅具有属性，还具有方法。要在系统的服务器端和客户端分布式计算这样的对象，无疑加大了网络的压力。

面向对象中，每个对象都对应着一个事物，这个对象的实例有它自己的特征、状态，有它认识、知道的其它对象实例(关联)，有它和其它对象实例交互的通道(类操作)。TMFAgent 的核心，就是由很多二类对象实例通过各种关联关系织成的一张虚拟网络。二类对象的属性反映着实际网络的状态，二类对象的操作对应着网络的种种行为。

根据以上的思想，在设计二类对象的时候需要考虑的问题包括：

- 准确描述实际网络中对应网络元素的属性和状态
- ✓ 通过类成员变量(用 Java 中较为简单的数据类型定义的 Field，如 int，String，boolean 等)
- 具有实际网络中对应网络元素所具有的行为
- ✓ 通过类的方法(Java 中的 Method)
- 正确反映与其他网络元素之间的关系，在 TMFAgent 中反映为二类对象之间的关系
- ✓ 通过类成员变量(用 Java 中较为复杂的数据类型定义的 Field，如 ArrayList，HashMap，其他二类对象等)

二类对象类型众多，如表示子网的 Subnetwork 对象、表示网元的 ManagedElement 对象、表示设备的 Equipment 对象、表示物理接口的 PTP 对象、表示逻辑连接的 TopologicLink 对象等。

5.3.2 命名规则

在整个 TMFAgent 系统中有大量的二类对象被用来描述整个网络中的所有元素。这些二类对象之间存在多种关系，根据这些关系 TMFAgent 将重新构建出相应的网络拓扑环境。在真实的网络环境中，一些网络元素存在着包含关系。如一个子网中包含着各种网元，一个网元内部通常有许多设备，设备上可能有若干个接口等等。因此，在 TMFAgent 中对应这些网络元素的二类对象也需要能够准确地表示出这样的包含关系，同时也要保证二类对象的唯一性、明确性。

TMFAgent 中，可以利用二类对象的名字提供对这一需求的解决方法。根据 MTNM 规范定义，每一个二类对象的名字由两部分组成：name 和 value[11]。这两

个值都是字符串，被包装在名为 NameAndStringValueT 的数据结构中。其中，name 成员是二类对象的类型标识，表示二类对象的类型。其取值必须是以下字符串之一：

表 5.1 二类对象名字缩写表

EMS	表示 EMS 类型对象	PTP	表示物理端点类型对象
SN	表示子网类型对象	CTP	表示连接端点类型对象
ME	表示节点类型对象	SNC	表示子网连接类型对象
EQH	表示设备容器类型对象	LNK	表示拓扑连接类型对象
EQT	表示设备类型对象	CC	表示交叉连接类型对象

value 成员的取值作为映射二类对象的键值。键值用来反映对象之间的确切包含关系，还必须准确表示出这些对象的类型和实例名。这里采用“/”符来分割包含对象实例名和被包含对象实例名，对象实例名的格式是：[类型标识]_[实例名]。

多级名字结构是一种合理的名字结构，可以满足网络管理者对网络各种组织方式的需求。资源本身的名字独立命名，但资源完整的名字是多级结构，反映被管对象与其它对象间的从属关系。

5.3.3 存储管理方式

1. 二类管理对象管理需求

在 TMFAgent 中，大量二类对象的存在给整个代理系统对资源的管理造成了不小的考验。对二类对象的有效管理是整个 TMFAgent 系统顺利运行的基础。只有在保证所有二类对象都能够正确创建，修改，存储和删除的前提下，TMFAgent 才能顺利完成拓扑发现、创建，性能监测及故障管理的等各种网管功能。

要对二类对象进行高效的管理，就必须考虑采用什么数据结构保存大量的对象。同时，还必须设计合理的排序和查找算法，使得 TMFAgent 的其他模块可以快速准确地定位需要的资源。因此，有必要设计一个二类对象管理模块，即二类对象管理器。它的功能是维护二类对象的生命周期，向所有需要访问二类对象的其它对象提供方便的查找方法，对所有具有网管功能的管理器提供访问二类对象各种必要接口。由于二类对象管理是 TMFAgent 描述网络的核心和基础，所以它还必须保证所管理的二类对象的唯一性和正确性。

概括来说，二类对象管理器主要完成对二类对象的统一管理，主要工作包括：

- 注册新建的二类对象
- 注销已有的二类对象
- 查找指定的二类对象
- 更新已有的二类对象

2. 二类对象管理器

二类对象管理器实现了读写、查找二类对象的所有方法。其核心的成员变量

是使用 Java 的 TreeMap 对象定义的一棵对象树 ObjectsTree，它是管理器的私有静态成员，用以注册所有二类对象。

ObjectsTree 是一个两级映射表：

第一级映射表是关键字到注册同类对象的映射表的映射。键值的类型是字符串，表示被存储的对象类型，而被映射的对象类型是 TreeMap，是存储该类对象的映射表对象；

第二级映射表是对象名到对象实例的映射。键值的类型是名值对序列，表示对象实例的名字，而被映射的对象就是对应的二类对象实例。

可以看出 ObjectsTree 是由两级 TreeMap 构成的。TreeMap 这一数据结构是 Java 标准类库提供的，它保证 Map 中的对象按关键字升序排列，且对 TreeMap 执行查找，插入以及删除等操作的时间复杂度均为 log(n)。TreeMap 的这些优良特性为实现二类对象的高效管理提供了良好的基础。

存储

二类对象由名字唯一确定，采用 java.util.TreeMap 映射表对其进行存储。对象在映射表中按类型存储，其结构如图 5.3：

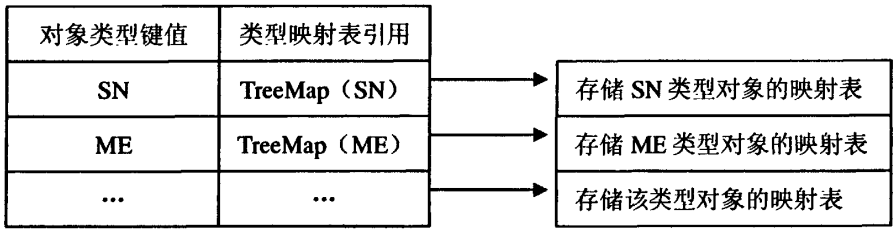

图 5.3 二类对象存储结构图

同类对象存储在同一个 TreeMap 中，TreeMap 对存储的对象按其键值(字符串)的 ASCII 码顺序排列。

排序

二类对象在映射表中是按其名字的字典顺序存储的，在对象被插入到映射表的时候，TreeMap 会自动对其键值进行比较，并将其插入到正确的位置。这样排序的结果是，名字前缀越接近的对象其存储位置也越接近。例如，具有如图 5.4 所示包含关系的对象在映射表中的存储顺序是：

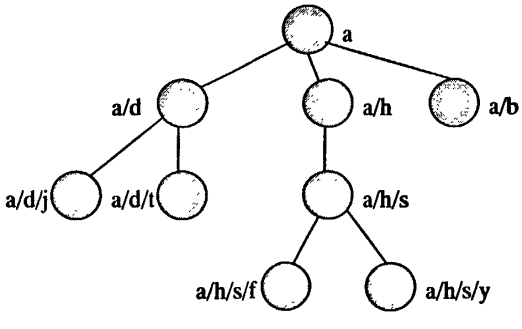

图 5.4 二类对象树状排序图

{a; a/b; a/d; a/d/j; a/d/t; a/h; a/h/s; a/h/s/f; a/h/s/y}

查找

对二类对象进行查找的目的主要是获得位于某指定节点下的其它对象，也就是获得二类对象包含树的某一分枝(子树)。TreeMap 对象提供了一个获得子映射表的方法：subMap(Object formKey, Object toKey)。此方法返回从键值 formKey(包括)开始到键值 toKey(不包括)结束的子表。利用它就可以取出用户需要的分枝。

例如，要获得对象 a/d 下的所有对象，需先插入开始标志键"a/d/*"(字符"*"小于所有字母和数字)和结束标志键"a/d/|"(字符"|"大于所有子母和数字)，这时映射表的结构变为：

{a; a/b; a/d; a/d/*; a/d/j; a/d/t; a/d/|; a/h; a/h/s; a/h/s/f; a/h/s/y}

然后调用操作 subMap("a/d/*", "a/d/|")，即得到子表：

{ a/d/*; a/d/j; a/d/t }

再将标志键"a/d/*"删除，就得到了需要的对象集合。

5.3.4 生命周期

1. 二类对象的创建

在 TMFAgent 运行过程中，出现以下几种情况时，需要创建二类对象：

- 系统初次运行，经过拓扑发现以后得到被管网络的网元信息，根据一定的映射规则，创建与这些网元对应的二类对象；
- 系统启动之后(并非初次运行)，从数据库中载入二类对象的状态信息，创建相应的二类对象；
- 管理器接口响应客户端(上级网管系统)的创建对象的请求，创建二类对象。

第一种情况，二类对象的创建由拓扑发现功能模块来实现，对象创建以后必须到存储该类对象的映射表中注册。

第二种情况，二类对象由二类对象管理类创建，二类对象管理类先读取数据库(在二类对象管理类中定义专门负责读写数据库的子类)，然后创建相应的二类对象，同时进行注册。

第三种情况，二类对象由管理器接口的实现类创建，在创建后必需进行注册。

总之，二类对象的创建位置是不固定的，在任何程序段中，只要需要就随时创建，但创建以后必须把新对象的对象引用注册到二类对象管理模块中。只有当二类对象的对象引用在 ObjectsTree 中存在以后，才标志着该对象是活着的。

2. 二类对象的删除

当一个二类对象对应的真实网络中的资源消失后，那么该二类对象就失去了存在的必要，可以对它进行销毁。在处理完与此二类对象相关联的那些对象后，通过在二类对象管理器中将其注销，就完成了对该对象的删除。

5.3.5 持久化方案

如上一章中介绍的，Hibernate 提供一种了简单的方法实现 O/R Mapping (对象关系映射)。它通过将 JavaBean 形式的 Java 类映射为 XML 文件，并在映射文件中配置类对象间的关系，使用这种配置 O/R Mapping 就可以轻松建立起来。同时，结合 Hibernate 的属性配置文件，在项目工程中，可以更便捷地实现数据对象的持久化。在程序中进行对象持久化时，不需要手工进行关系映射，而只需要执行代码操作 Java 对象即可以实现持久化工作。

1. Hibernate 的使用

- 在 Hibernate 的配置文件中配置 Hibernate 的属性以及数据库连接参数。
- 建立持久化对象类，所有类对象所要持久化的属性都以 JavaBean 的风格出现。
- 为持久化对象生成 XML 文件映射，并建立对象间的关系映射。
- 通过 XML 映射文件建立数据表。
- 在项目工程中实现持久对象的操作。

2. 表格设计

表 5.2 和表 5.3 是二类对象设计中与二类对象映射形成的关系型数据库表格，分别用于记录对象以及对象的管理关系.

表 5.2 ObjectsTable 二类对象主索引表

字段名	类型	键	其它	说明
id	bigint(20)	PRI	自动递增	自动编号
name	varchar(255)	UNI		二类对象 name 属性中的 value 域
nativeEmsName	varchar(50)			二类对象的 nativeEmsName 属性
userLabel	varchar(50)			二类对象的 userLabel 属性
type	smallint(6)			表示二类对象类型的编号
description	varchar(255)			附加说明信息(目前没有使用)
tmfagentID	varchar(255)			二类对象所属 TMFAgent 的标识 ID

表 5.3 RelationTable 二类对象关联关系表

字段名	类型	键	其它	说明
id	bigint(20)	PRI	自动递增	自动编号
relName	varchar(50)			关系名称
relType	varchar(50)			关系类型
relValue	varchar(255)			取值
objectIndexId	bigint(20)	MUL		对应主表的键值

3. 数据库

通过 Hibernate 的映射，目前 TMFAgent 在实现对象的存储上可以使用 MySQL，Orcale，Sybase，SQLServer 等大型专业的或是免费的数据库。数据库的选择需要根据系统运行的要求而定，对于管理大型网络的系统，通常采用性能

稳定、功能齐全的大型数据库，而对于运行条件苛刻的环境，则可以选择小型的数据库。

5.4 数据访问模块实现

5.4.1 SNMP 协议封装

常见的数据访问模块是各种网络协议的网关，这样的网关从实际网络中获取真实数据。二类对象可以通过从网关获取数据进行初始化以及与实际网络保持同步。各管理器则通过网关对实际网络实施影响。

对于一般的网元管理系统来说，最常用的网络协议就是 SNMP 协议。在这里，TMFAgent 对 SNMP 协议的访问方式做了一层封装。由于原始 MIB 库的对象标识由一长串数字组合而成，不易记忆，不易理解。这会给网络管理人员带来较大的麻烦。在长期的实践中，我们发现使用由 <MIB 库名：管理对象组名：管理对象名> 组成的三段式名字标识可以有效代替原本可读性很差的对象标识 ID。

SNMP 协议中的管理者可以通过大量的开源工具实现。选择好的开源 SNMP 实现对网管系统的性能会有很大提高。

5.4.2 其他的数据访问方式

在开发某些系统时，由于条件的限制，可能一时无法获得真实的网络数据，这时数据访问模块也可以是分布式的仿真器或者数据库中的数据。数据访问模块的有效性在于能否为二类对象集合提供透明的数据来源。

5.5 系统整体特性

通过对抽象代理 TMFAgent 整体框架的简要介绍，可以归纳系统的几个显著特点：

1. 组件化

首先，采用组件构成系统实现了功能的划分和系统整体的简化。系统由一系列边界明显的组件构成，主要包括故障管理组件，性能管理组件，网元管理组件，拓扑管理组件，子网管理组件，会话管理组件等，使网管系统的主要管理功能可相对独立的设计。

这些组件有些可单个独立部署，有些需要聚合后作为一个整体部署。部署方式的可选性使系统可灵活缩放，根据网络负荷或网络规模状态分散部署系统或集中部署系统，添加或减少组件。

2. 可扩展性。

采用外观设计模式(Facade Pattern)，应用组件在统一的会话外观框架中组织，

添加服务器应用组件对系统原有部分不需要做任何修改，也不会影响客户原有代码资源，而且也使扩展容易。

3．基于网元的粒度

TMFAgent 构成中，网络的逻辑和物理资源被表示为被管对象，与实际网络保持了管理视点所需要的对应映射关系。就结构而言，网络节点是被管对象的基本封装单位。这种方式有如下优点：

- 对于单个网络资源来说，所有相关信息得到集中和封装，是合理的信息组织方式。加上采用面向对象范型，针对不同资源的管理性能得到强化。
- 对于整个系统模型而言，相互间关系清晰简单，系统易于理解和维护。

对于运营需求而言，由于节点间的关系可以在粗粒度的级别上进行操纵，容易建立网络资源间的多重关系。例如，可以建立以运营范围为前提的子网划分，观察运营商的管理水平；可以建立以地理区域为前提的子网划分，考察地理因素对运营的影响；也可以建立以厂商为前提的子网划分，比较不同厂商设备的性能和故障率，等等，使管理者可以从多角度观察网络的运营状态。

5.6 本章小结

本章在前面介绍的相关技术的基础上，全面的介绍了综合网管系统中抽象代理 TMFAgent 的设计和实现。首先介绍了 TMFAgent 的整体框架，然后分别从分布式接口定义，管理功能接口定义及实现等方面介绍了系统的组成。同时，本章中详细叙述了二类对象的一系列概念和设计实现方法，并对数据访问模块做了简要的介绍。最后，本章对 TMFAgent 的系统整体特性进行了总结。

第六章 总 结

抽象代理系统的研究和实现是网络管理研究领域的一项重要技术实践，是实现高效综合网络管理的基础。混合的多技术网元管理系统能为上层网络管理系统屏蔽异构物理网络环境的巨大差异，提供一个透明的性能良好的底层网络管理平台。本文通过多种技术的研究、分析和比较，构建了整个综合网络管理抽象代理系统的实现框架，并为许多技术问题的解决提供了详细的实现。

本文所作的主要工作和贡献是：

从网管功能定制，系统架构规划，用户界面设计等方面阐述综合网管系统的整体设计思路与方案。该方案强调综合网管系统的设计应以服务的综合为核心，系统的所有组件都需从这一点出发。

对目前两种分布式对象中间件技术进行了较为详尽的比较和分析。分布式对象中间件是实现分布式网络管理系统的关键技术，是综合网络管理系统的底层系统交互平台，为分布的各网络管理系统组件提供软总线。对象中间件解决了综合网络管理系统中的系统分布问题，并提供了许多强大的基础服务。本文主要从中间件的结构、接口定义语言、通信协议、基础服务等多方面对 CORBA 和 Ice 两种较为流行的中间件进行介绍，并比较了它们在设计和实现上的优缺点，分析了不同网络环境中选择网络管理系统中间件平台的要求。

对电信管理论坛提出的网络管理相关文献进行了研究和分析。研究主要集中在简单网络管理协议，电信管理网，电信管理论坛的下一代运营支持系统、电信运营图和多技术网络管理等网管技术发展历程方面。电信管理论坛多技术网络管理体系规范中的许多理念和方法在 TMFAgent 多技术网元管理系统的 NML-EML 接口的设计和实现过程中得到了应用。

详细介绍了综合网管网元管理系统中网元数据建模和网元数据持久化的关键技术。抽象网元模型是根据多技术网管框架中基本网元模型，结合软件设计模式提出的。它能有效解决混合网络环境中网元模型一致性的问题，并大大提高网管系统的可维护性与可复用性。网元数据持久化同样是网管系统中不可缺少的重要技术组成。本文介绍了利用开源对象/关系映射工具 Hibernate，为对象数据和关系型数据库表格的建立高效的映射关系的方法。这一技术为网管开发人员带来了很大的便利。

详细地描述了 TMFAgent 综合网管网元系统已有的设计和实现情况。TMFAgent 现阶段完成的工作主要集中在网络管理层和网元管理层的接口定义，二类对象以及数据访问模块的实现。NML-EML 接口主要定义了网元管理层向网络管理层提供的服务接口，目前实现的网元管理系统功能服务由拓扑管理、故障

管理、性能管理、网元管理等模块组成。二类对象是 TMFAgent 实现的核心，网元管理系统通过二类对象的组合及关联关系复原实际的网络环境。文中通过对数据访问模块描述，简要介绍了网元管理系统的数据来源方式。

虽然，到目前为止综合网管抽象代理 TMFAgent 的研发已具有初步雏形并实现大部分功能，但仍有一些有待进一步研究和实现的内容，主要包括：

- 网管数据持久化的优化：简单的使用 Hibernate 可以提高网管系统的开发效率，但要真正做到通过 Hibernate 来改善网管系统的运行效率还有待对网管数据的进一步挖掘以及对关系型数据表格的合理设计，从而找到更为有效的 O/R 映射方式。
- 网元管理系统的伸缩性：对于中等规模的被管网络环境(5000 个网元左右)，TMFAgent 仍可以有效的进行管理。然而，当网元数量远大于一定数量时，如何使多个 TMFAgent 联合工作以达到良好的管理效果，这是一个十分值得研究的问题。

致 谢

我首先要感谢我的导师吴宇红副教授。感谢吴老师在项目实现和论文写作期间给予我的指导，以及在整个研究生学习阶段，对我学习、生活的关心和支持。吴老师严谨的治学态度、孜孜不倦的钻研精神、敏锐的学术洞察力、锐意进取的开拓精神和不断创新的作风给我留下了深刻的印象，是我学习的楷模。从论文的立题到撰写，吴老师不但从宏观上修正我的研究方向，还在一些具体的问题上给予我很多指导。

我还要感谢刘炯老师、张岗山老师、冯磊老师、周战琴老师在工作中和生活上给我的帮助和关心。其中我要特别感谢刘炯，张岗山和冯磊老师，他在课题研究方面给了我很多帮助，并教给我许多发现问题、分析问题和解决问题的方法，使我受益匪浅。

感谢我的各位同门，他们在我攻读硕士学位期间，给予我许多无私的帮助。感谢他们与我在工作学习上的探讨与鼓励，感谢他们对我的信任与支持。他们每个人身上都有着很多优秀的品质值得我去学习，与他们的相处丰富了我的人生。他们是王艳、齐芳、古玉洁、李富年、张跃宇、付松、王志军、付海涛、杜能功、王炎、王博、李全、周康、裘楷、陈琦、王鑫等多位硕士。

感谢我的家人，感谢他们在我成长过程中付出的一切。他们的理解与支持，是激励我在人生路上不断前进的永远动力。

感谢所有参加论文评审的各位专家、教授以及老师们。感谢他们对论文提出的宝贵意见。

参考文献

[1] 夏海涛、詹志强. 新一代网络管理技术. 版本. 出版地：北京邮电大学出版社，2003 年 5 月.页码

[2] 杨家海、任宪坤、王沛瑜. 网络管理原理与实现技术. 第一版. 清华大学出版社，2000 年 9 月

[3] 综合网管的建设思路 （2005-10-31）China 通信网

[4] Henning Michi . A New Approach to Object-Oriented Middleware. IEEE Computer Society. January~bruary 2004. 页码

[5] Michi Henning Steve etc. 徐金梧等译 基于 C++ CORBA 高级编程. 第一版. 清华大学出版社，2000 年 7 月

[6] Henning Michi、Spruiell Mark . Distributed Programming with Ice. Revision 2.0. 出版地：出版者，15 November 2004. 参考的页码。(http://www.zeroc.com)

[7] TMFC763 GB920v1[1][1].5, New Generation Operational Support Systems

[8] TMF. Telecom Operations Map v2.1[S]. TMFGB910. September 2000. 页码

[9] TMF. Multi-Technology Network Management Business Agreement v2.1[S]. TMF513. September 2002. 页码

[10] TMF. Multi-Technology Network Management Information Agreement v2.1[S] TMF608. August 2002. 页码

[11] TMF. Multi-Technology Network Management Solution Set Document v2.1[S]. TMF814. August 2002. 页码

[12] 阎宏. Java 与模式. 第一版. 电子工业出版社，2002 年 10 月.页码

[13] 齐芳、吴宇红、刘炯. 基于抽象信息模型的关系管理方法. 北京邮电大学学报. 2003 Vol.27. 页码

[14] 古玉洁、吴宇红、刘炯. 基于影子代理的战术网网络管理系统的研究. 北京邮电大学学报. 2003 Vol.27. 页码

[15] 李富年 基于 Web 的网络管理体系结构研究和实现. 硕士学位论文 页码

[16] 胡羽文、吴宇红 综合网络管理系统中抽象网元模型的设计与应用 电子科技

[序号] 作者 . 专著名称. 版本. 出版地：出版者，出版年. 参考的页码

[序号] 作者 . 文献名. 期刊名称. 年 ，月， 卷(期). 页码

作者在读研期间的成果

发表论文

《综合网络管理系统中抽象网元模型的设计与应用》发表于《电子科技》2006/2。作者：胡羽文、吴宇红。

参加科研项目

海军综合网络管理系统的研究

上海电信 WLAN 网络管理系统

基于 Windows CE 平台的 VoIP 软件电话 FreePP