中山大学

硕士学位论文

# 基于CSTA标准的CTI中间件的研究与实现

姓名：顾敏

申请学位级别：硕士

专业：计算机应用技术

指导教师：陈有青

20050508

# 摘要

计算机技术在电信领域的应用产生了 CTI（Computer Telecommunication Integration）技术。经历了多年的发展，CTI 技术的应用越来越广泛，从最开始的计算机电话、交互式语音应答、自动总机、智能呼叫路由安排、智能网络、统一消息，发展到语音处理、传真处理、视像会议、因特网电话、呼叫中心、文本转语音、语音识别、语音消息等，并且随着计算机和网络技术的进步还在不断地向前发展。

本文基于香港纬视通信技术有限公司的产品项目——企业级统一商务通讯平台。该平台是将电子传真、短信、电子邮件、计算机电话集成、呼叫中心等技术融合于一体的统一通讯平台。

要开发这样的平台，CTI 中间件是解决计算机与电话交换机之间通信的一项关键技术。由于目前主流交换机，如西门子、爱立信、华为、阿尔卡特等品牌均使用了 ECMA（Eoupean Computer Manufacture Association）的 CSTA（Computer Supported Telecommunications Applications）协议标准，因此本项目采用 CSTA 作为交换机和计算机间通信的协议。目前，CTI 中间件大都由国外的交换机厂家提供，价格昂贵，本项目自行开发基于 CSTA 标准的 CTI 中间件，一方面可降低开发企业级统一商务通讯平台的成本，另一方面可为其它应用开发提供底层支持，即开发者可根据不同需求方便地开发出不同的业务应用系统。

本文主要研究 CSTA 协议，包括其功能结构、操作模型、语法描述和服务，特别是状态报告服务、交换功能服务和计算功能服务，结合项目需求设计并实现了一个企业应用级的 CTI 中间件，本人主要实现了 CTI 中间件 CSTA 解析层，将 CSTA 呼叫模型按照 ASN.1 (Abstract Syntax Notation number One)的描述，转化为交换机可以处理的字节流，并为上层应用提供应用程序接口。本文的成果，已经使用在企业级统一商务通讯平台中，获得了满意的效果。

关键词：CSTA 协议，CTI 技术，CTI 中间件，ASN.1

# Abstract

With computer technologies applied in the field of telecommunication, a new technology which is called CTI (Computer Telecommunication Integration) has been occurred. Furthermore, the related applications for CTI are getting more and more popular nowadays.

This paper is based on a Communication Platform for Enterprises. The platform gets together E-Fax, SMS, E-Mail, CallCenter, etc. There's a key problem concerned how to communicate the computer with the telephony exchange when developing the platform. The CTI middleware can solve the problem well. At present, the CTI middleware, which is mostly provided by the exchange producers abroad, is too expensive for most of enterprises in our country. Therefore, the platform developing team considered trying to develop all CTI middleware themselves. This would help to reduce the cost for Enterprise to establish the Communication Platform. On the other hand, the CTI middleware intends to advance other CTI applications development.

In the first chapter of this paper, the function of the CTI middleware is presented. Then we generally introduce the CTI technology in the second chapter, including the concept and the categories of CTI middleware, and CSTA (Computer Supported Telecommunications Applications) protocol. In the third chapter, we further analyze the functional architecture of CSTA protocol, CSTA Operational Model, the Services CSTA provided, and ASN.1 (Abstract Syntax Notation number One). The fourth chapter describes the implementation of CTI middleware. Finally, in the fifth chapter, we show the applications of CTI middleware and the idea for further research on CTI middleware.

Currently, the CTI middleware has been applied in the Communication Platform for Enterprises and the results are quite satisfied.

**Key Words:** CSTA, CTI, CTI Middleware, ASN.1

# 第一章 引言

## 1.1 项目背景以及使用CTI中间件的意义

多媒体接入、统一消息、语音技术、联络中心等在新经济e时代的客户关系管理中被越来越多的企业所要求。本人参与开发的企业级统一商务通讯平台就是根据这种需要将电子传真、短信、电子邮件、客户关系管理、工作流、呼叫中心、计算机电话集成等于一体，把企业的多种媒体应用集中到一台PC服务器上完成，并可用电话、传真、手机、PC等通信设备处理相关信息，在有线、无线、互联网以及内部通信网之间架构起一个互联通道，突破原有的办公和客户服务方式的限制，向企业员工、客户提供各种的通信服务和信息服务。中小企业能够充分利用企业各种信息资源，建立全新的工作模式，从而全面提高服务能力和获得更多投入回报。

在本平台中使用CTI技术，采用交换机作为交换设备。为了使计算机能够通过交换机进行呼叫处理，早期是一种紧耦合的集成模式：交换机提供应用程序开发接口，应用程序通过对这些接口的调用直接操纵交换机。在这种集成模式下，存在着下述问题：第一，程序可移植性差，应用程序使用交换机专用接口，如果要使用新的设备，开发者必须在新接口上重新开发；第二，由于应用程序直接操纵交换机，要求开发者具有良好的测试能力保证交换机的其他功能不受影响。第三，由于封闭的交换机仅仅提供了呼叫控制的接口，不承担保存数据的任务，难于在应用程序之间共享与呼叫相关的数据。

为了解决上述问题，在应用程序和交换机之间引入CTI中间件，一方面作为应用程序的服务器，另一方面它又为客户机访问交换机提供服务。

CTI中间件是处于交换系统和计算机之间将两者功能进行集成的软件[1]。引入CTI中间件后，承载交换的底层设备对各个应用程序是透明的，CTI中间件在电信设备和计算机设备之间执行协议转换和映射的工作，它为各个应用程序屏蔽了各种交换机之间的差异，使应用程序可以调用底层交换设备提供的服务而无须了解其具体实现。同时，CTI中间件是用计算机上的软件实现的，相对于封闭的交换机来讲，它可以很灵活地如利用内存，文件系统，数据库等多种手段为应用程序

提供呼叫数据共享的机制,可以实现呼叫和大量数据在各个应用系统之间的同步转移。

## 1.2 自主开发CTI中间件的目的和意义

CTI中间件和交换机的联系非常密切,所以交换机厂商在开发CTI中间件方面比较有优势。目前主流的CTI中间件分为两种:通用的CTI中间件适合较多品牌的交换机,例如Genesys、CT Connect;专用的CTI中间件一般是交换机厂商开发的,只适用于自己的交换机,比如Avaya的CVCT,Siemens的ProCenter。这两种中间件各有优势:通用的CTI中间件可以帮助系统集成商在使用不同的交换机时不需要重复开发客户端软件,节省开发时间和成本;专用CTI中间件的优势是在它的专业上,能够提供更丰富的功能。

但总的说来,由于历史原因,CTI中间件大都由国外的交换机厂家或计算机厂家提供,控制在外国厂家的手里,价格昂贵。

因此,本项目组所用的CTI中间件是自主开发的,一方面可以节省开发成本,另一方面也可以为其它应用的开发提供底层支持,应用开发者可以根据不同的需求基于本CTI平台进行二次开发,快速简便地构筑呼叫中心、企业增值服务等。

## 1.3 本人所做的工作

企业级统一商务通讯平台项目组自去年8月成立以来,本人作为成员之一参与了CTI中间件的开发,实现交换机和计算机之间的通信,为上层CTI应用程序提供编程接口。我的主要工作是研究CSTA协议,熟悉协议提供的功能和服务,由于目前本通讯平台主要实现监视管理通信通道和操作控制电话功能,因此主要研究协议中的状态报告服务、交换功能服务和计算功能服务;由于CSTA消息结构由ASN.1来描述的,计算机与交换机之间传输的是字节流,需要将抽象的结构转化为机器可以处理的字节流,因此实现CSTA消息的编解码与传递也是本人的主要研究工作;最后本人还参与了提供应用程序接口的工作。

# 第二章 CTI技术与CTI中间件

## 2.1 CTI技术

CTI技术中的“T” 由开始的“Telephony”演变为“Telecommunication”，意味着目前的CTI技术不仅要处理传统的电话语音，而且要处理包括传真、图形图像等其它形式的数据信息媒体。CTI 技术跨越计算机技术和电信技术两大领域，目前提供的一些典型业务主要有交互式语音应答、呼叫中心系统、电信增值业务、IP电话等[2]。

### 2.1.1 CTI发展历程

计算机与电信技术结合可以追溯到“程控交换机”的出现，发展历程如下：

1．程控交换机的出现

电信网络的基础，就是电话系统，其中的负责线路转换的交换机经历了步进式、纵横式两代，一直很难取得满意的效果。计算机技术中的“存储转移”概念应用到交换机后，就出现了程控交换机，大大地提高了电信网络效率和处理能力。

2．呼叫中心业务

为了使电话系统为用户提供更好的服务，最初是由专门的话务员，根据自己的经验和记忆力，为打入电话的顾客进行咨询服务，由于是凭人工记忆，信息容量有限，服务能力也无法提高。将计算机数据库技术应用到呼叫中心以后，可以将与各种服务有关的数据存入计算机中，这样，顾客需要哪方面的服务，只需调出相应的数据库即可，而不必非由该方面的专家完成，提高了服务质量和效率。

3．计算机电话集成技术

通过电话语音卡及有关的软件，使得计算机也能够处理电话的声音及有关的信令信息，也可以提供诸如自动语音应答（IVR）、自动呼叫分配（ACD）、语音信箱等业务功能。

4．智能网技术

在呼叫中心技术的基础之上，出现了智能网技术，根据交换和控制相分离的思想，单独分离出一个计算机控制的电信业务平台，同时完成有关的管理功能，

大大地提高了电信业务的服务能力和水平。

5. IP电话/IP传真

计算机网络技术，也在涉足传统电信网络的语音通话等领域。对语音信号进行压缩和打包之后，通过数据传输网络，完成电话及传真等功能，提供了一种新的电信业务的传输方式。

### 2.1.2 CTI技术内容

目前，国外依然称CTI为计算机电话集成，但从CTI所包含的内容看（如呼叫中心、IP电话、智能业务平台等），已经远非仅仅局限在电话角度了。

1. 从提供电信业务的角度

CTI技术的起源，是源于市场的需求，其目标也是为了给用户提供高效和高质量的电信服务，因此，我们先从业务的角度进行如下的划分：

1）基于交换的方式

由于在现有的电信网络中，都是基于交换方式的，因此为了提供电信业务，一种最基本的思想就是从交换部分着手。在交换机部分，通过有关的计算机平台，提供各种电信业务。

这种情况也可以分为两类。一类是基于中心交换局的智能网方式，通过独立的业务平台、管理平台等；另一类，是基于PBX的，用户打入PBX的电话，可以通过CT Link、CT Connect等接口（API），以局域网方式转到有关的计算机平台，获得流程处理信息，提供电信业务。本论文研究的情况即属于第二类，主要用在企业内部。

2）基于语音板卡的方式

在电话端直接通过普通的电话线，将用户打入的电话接入到计算机系统中，这时需要通过语音板卡，对接入的用户电话语音及信令进行控制处理。

2. 从整个通信系统角度

由于CTI技术要涉及通信系统的每个环节，由此，可作如下的划分：

1）用户端技术

用户端技术包括电话机、头戴式耳机、语音板卡、网络板卡等产品内容，也包括语音识别、文语转换等技术部分。

2）接入技术

接入技术即是如何将通信网络中的信息最终传递到用户端的部分，包括XDSL、同轴电缆、光纤等接入技术。

3）传输技术

传输技术是解决信息如何传输的部分，包括：光缆技术、SDH技术、CDMA、GSM等内容。不仅考虑有线部分，还要考虑无线方面的传输部分，同时，也包括计算机网络中的协议内容。

4）交换技术

交换技术是为了解决信息的路由转接问题，包括数字交换机技术、ATM技术、路由器技术等。

总之，在通信系统中，包括信息从信源到信宿的全部环节，每个环节中都在引入计算机技术。

3．从计算机网络提供电信业务的角度

1）IP电话／IP传真

涉足传统的语音通信及传真功能，如今用户的通讯，可以完全不经过传统的电信网络，而是通过计算机网络就可以独立完成。但通话质量问题仍然有待进一步的解决。

2）电子商务

从提供附加值业务服务的角度，通过计算机网络，完成选择、购物、支付等个人采购过程；企业中可以包括原材料的采购、管理、销售等诸多环节等等。如此，可以大大地提高生产效率、降低成本、减少流通环节，为企业和个人带来效益。本项目组研发的企业级统一商务通讯平台将不断进行升级，以达到上述目标。

## 2.2 CTI中间件

### 2.2.1 中间件

中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。中间件在操作系统、网络和数据库之上，应用软件的下层，总的作用是为处于自己上层的应用软件提供运行与开发的环境，帮助用户

灵活、高效地开发和集成复杂的应用软件[3]。

中间件特点如下：

1) 满足大量应用的需要；
2) 运行于多种硬件和OS平台；
3) 支持分布式计算，提供跨网络、硬件和OS平台的透明性的应用或服务的交互功能；
4) 支持标准的协议；
5) 支持标准的接口。

## 2.2.2 CTI中间件

### 2.2.2.1 CTI中间件定义

CTI中间件在计算机网和电信网之间发挥桥梁和纽带作用。

下面来看看CTI中间件如何定义的。

CTI中间件运行在CTI服务器上，用来从呼入电话中找出至少三个信息：主叫号码(通过自动号码识别服务ANI)、被叫号码(通过被叫号码识别服务DNIS)、交互式语音应答系统中呼叫方输入的数字序列[4]。

CTI中间件有四个特征：用来在计算机应用程序和电话设备之间充当接口；允许两个或多个交换机之间互相交换数据，即使这些交换机来自不同的厂商；能把交换机中有关呼叫的信息传递给多种类型的电话设备，例如IVR系统、呼叫监控系统等；能把有关呼叫的数据和其它数据、呼叫中心的软件包（如计费系统、数据管理工具）进行共享。CTI中间件目前已经拓宽了应用范围，涉足其它一些需要交互式应用的场合，如回复接收的电子邮件信息、回应Web站点上访问者的申请表格或文字信息等。

### 2.2.2.2 CTI中间件分类

CTI中间件根据应用的层次和功能定位，一般可分为三种类型：Link层中间件、API层中间件和应用层中间件，分别实现不同的功能，并且前后紧密关联[5]。

三种不同类型的CTI中间件的功能和相互关系示意图如图2-1所示：

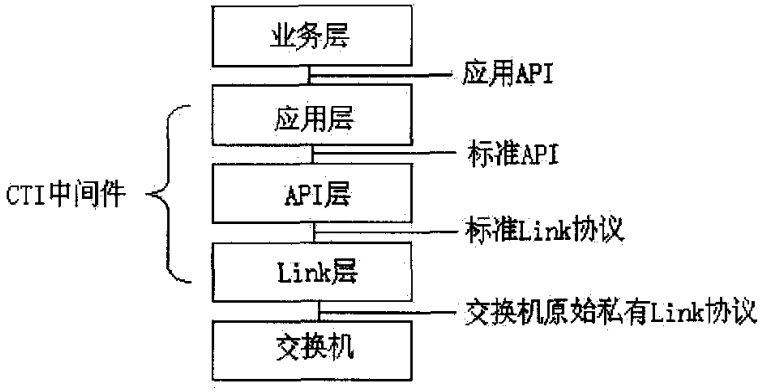



图2-1 CTI中间件的功能和相互关系示意图

1. Link层中间件：

实现把交换机的原始私有Link协议转换为标准的Link协议，如Nortelnetworks的Meridian Link、Avaya的ASAI Link，以及遵循工业标准CSTA Phase I、CSTA Phase II、CSTA Phase III的CSTA Link，从而可实现与Callpath、CT Connect、Gensys、CCM等CTI中间件连接。

2. API层中间件：

在Link层的基础上，实现把标准Link协议转化为标准的API，如CTC API、TAPI、JTAPI（2.2.2.3节中将有介绍）等，为CTI应用开发提供一套标准的API，屏蔽了不同交换机拥有的CTI协议和CTI接口，使应用程序员无需关心消息是如何在计算机和交换机之间传递，专注于客户端的应用开发，实现应用程序与交换机平台无关性。本论文实现的CTI中间件属于API中间件。

3. 应用层中间件：

在API层的基础上，通过API层提供的标准API，开发智能路由、去话呼叫管理、来话呼叫管理、呼叫监控、统计报表等应用层产品，与业务开发紧密相关。

### 2.2.2.3 相关标准

CTI Link是连接交换机和计算机的通信链路。主要包括两种解决方案：First Party（第一方控制）和Third Party（第三方控制）。

第一方控制是指呼叫由参与通话的实体控制的，如图2-2所示。图中的线路

信令是双向的。所有从交换机来的信号和应答信号，均可以通过计算机进行检测和解释。使用这些信息，电话呼叫可以通过第一方计算机进行控制，并通过计算机屏幕进行监视，但是无法实现对交换机的控制。

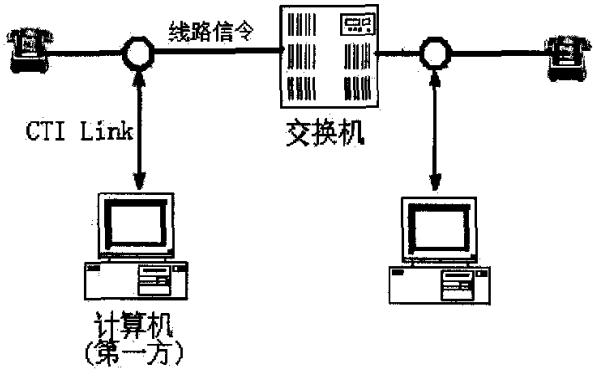



图2-2 CTI Link第一方控制

第三方控制中，呼叫可以由一个不参与通话的实体发起和控制，即存在一个实体，它不参与媒体会话，却可以知道呼叫的状态，影响呼叫进程。由于第三方控制提供了更高的灵活性和媒体访问功能，这种方案为很多应用所采用。CTI 中间件也是采用这种方式。图2-3显示了第三方控制的配置。

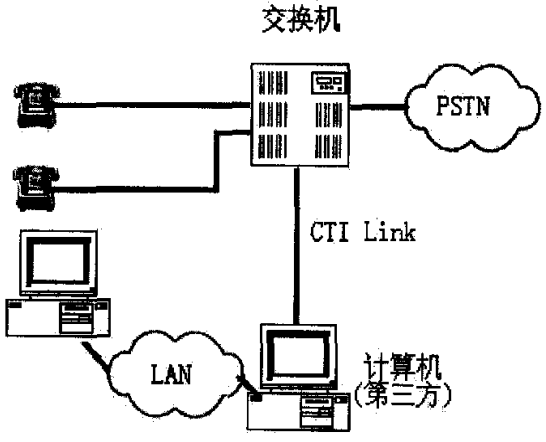



图2-3 CTI Link第三方控制

CTI Link 是 CTI 技术的基础，因此，CTI Link 协议的制定成为通信厂商和计算机厂商共同关注的焦点。从交换机发展历史上来看，通信领域的巨人（Lucent、北电、阿尔卡特、西门子、爱立信、HARRIS）纷纷制定了自己的CTI

Link 协议。

1988 年 ECMA 首先提出了计算机与交换机之间的标准协议：CSTA，是 OSI 应用层协议，其主要对象是基于交换机特别是专网电话交换机的中继线和排队机，主要技术包括连接、应答、转移、会议等呼叫控制服务；话机性能、音量控制等设备服务；消息的收发、事件管理和计费等内容。

美国国家标准学会（ANSI）于 1989 年制定了应用于程控交换机中虚拟交换网（Centrex）方面的标准：SCAI（Switching Computer Application Interface）实现了计算机和电话交换的集成。目前，这一协议的研究工作已停，其功能略逊于 CSTA，但它的定义更为严格。

ITU 曾致力于开发一种国际解决方案：TASC——用于交换机和计算机的电信应用，由于种种原因，TASC 行动于 1994 年被束之高阁。

随着交换机技术和计算机技术的发展，协议趋于统一，目前主流的 CTI Link 协议分成三大类，如表 2-1：

表 2-1 CTI 标准及支持厂商

| CTI Link 协议 | 支持的厂商 |
|---|---|
| CSTA | 西门子 |
| | 阿尔卡特 |
| | 爱立信 |
| | 华为 |
| ASAI | AVAYA |
| Meridian | 北电 |

从计算机软件发展历史上来看，计算机领域的巨人（Microsoft、IBM、Novell、Sun）在应用一侧，API 向应用程序提供一组过程调用或消息，并且允许实现所有支持的 CTI 功能（典型的包括 Dialogic 的 CT Connect 、Microsoft 的 TAPI、Novell 的 TSAPI、IBM 的 Callpath、HP 的 ACT、Sun 的 JTAPI）。目前国内较常用的 API 主要有三种，如表 2-2 所示：

表2-2 常用CTI应用程序接口

| API接口 | 开发厂商 |
| --- | --- |
| CT Connect | Intel(Dialogic) |
| TAPI | Microsoft |
| TSAPI | Novell |

CTI Link协议中，CSTA应用最为广泛； API接口则以CT Connect的应用为最多，已经成为事实上的标准。

## 2.3 CSTA简介

为了响应ECMA建立CTI标准的运动，专门成立了一个代表主要PBX制造商的委员会，包括AT&T、 NorthernTelecom、Alcatel、Siemens、IBM、HP及其他公司。该委员会制订了标准ECMA-179计算机支持的电信应用服务（CSTA）和标准ECMA-180——计算机支持的电信应用协议，该协议是计算机与电信网间的 OSI第7层通信协议。CSTA获得了大多数交换机制造商和呼叫中心开发公司的支持，目前的版本有CSTA I（1992），CSTA II（1994）和CSTA III（1998）。CSTA为集成计算机和电信网平台定义了总体结构、要求和协议。这项技术标准强调了计算和交换的灵活性、双向通信和分布模型。由于CSTA起初是针对专用网的，因此它关注的是基于交换的对象，比如电话、中继线和队列等。CSTA服务是独立于交换平台的，它并不知道交换机是如何完成CSTA服务请求的特殊细节。

第一版 CSTA是1992年颁布的，作了以下定义：

1) 呼叫控制服务，包括呼叫应答、清除连接、摘机和呼叫转移。

2) 设备服务，包括话机性能比如免打扰、转移和消息等待。

3) 状态报告服务。

4) 系统状态和退出服务。

第二版CSTA是1994年公布。第二版扩展各种服务的定义，包括：

1) 设备服务：麦克风和音量控制。

2) 语音设备服务：发消息和记录消息。

第三版相对第二版作了以下扩充：

1) 增加了新的服务和事件类别，如交换，计费和其它媒体的结合。

2) 增加了呼叫和设备控制的附加服务。

3) 加强了现有服务和事件的功能。

4) 基于组群的服务和事件管理（呼叫控制，设备控制等）。

5) 用连续“模板”描述服务和事件，包括初试/最终连接状态，状态迁移，事件监视序列等等。

第四版增加VoIP功能。

# 第三章 CSTA协议（第二版）分析

CSTA 提供了一种独立于物理实现的计算机和通信网络间的互操作机制，能够将计算机的计算能力和通信网络的交换功能进行集成。

CSTA协议（第二版）包括两个部分，一个是ECMA-217，为CSTA服务定义的说明；另一个是ECMA-218，为ASN.1语法描述的协议规格。

为实现计算机控制交换机工作，我们调用和控制各种不同的CSTA服务。CSTA服务由抽象语法标记ASN.1描述。计算机和交换机之间传输的是字节流。如何将ASN.1描述的CSTA消息转化为字节流，涉及到编码的问题。

下面就介绍CSTA功能结构、操作模型、CSTA提供的服务，简单介绍ASN.1的定义和编解码规则，并举例说明ASN.1描述的CSTA服务。

## 3.1 CSTA协议功能结构

### 3.1.1 分布式计算和交换功能

CSTA 从功能上可以分成三个部分，包括计算功能、交换功能、特殊资源功能，并且定义了各功能之间的逻辑交互关系。计算功能由位于计算机网络中的一台或多台计算机实现。交换功能由位于通信网络中的一台或多台交换机完成。特殊资源功能可以位于交换机或计算机内部。

CSTA 应用中每个功能实体划分为三层:处理层,通信层和底层的网络层，如图 3-1 所示。处理层通过 CSTA 服务定义与对等功能实体进行交互,调用对等层提供的服务。但它不涉及怎样具体与对等方通信,比如为了发起一个呼叫,需要发送哪些参数,可能会返回什么结果等。这些功能是由下面的通信层来实现。通信层与对等方按照CSTA 协议发送CSTA 数据单元,负责应用实体之间联系的建立、释放。CSTA 协议中对网络传输没有特殊的规定,即CSTA 协议与网络协议是无关的,开发者可以根据需要选择支持一种或多种网络协议。

尽管功能是分布在不同网络上的,但 CSTA 的应用呈现给用户应该是在单一网络上的单个应用,而不是在两个不同网络上的不同应用。

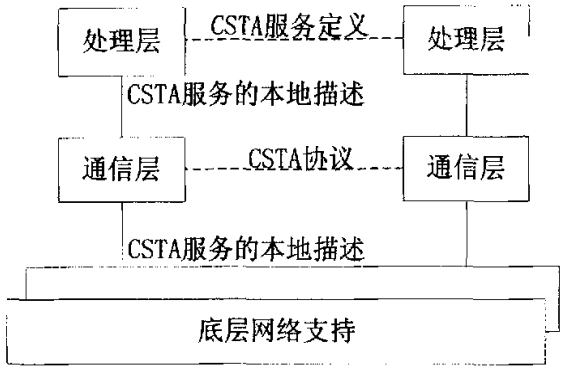



图 3-1 CSTA 分层模型

### 3.1.2 CSTA 服务

CSTA 中的“服务”指应用层为对等应用层提供的服务。“服务”的概念在 OSI 参考模型中是指某一层给高一层提供的利益（应用层除外）。在 CSTA 中，“服务”指的是在特定的参考点下，通过网络提供给用户的应用层功能，是由一端的应用层提供给对端的对等层的。计算、交换和特殊资源各自独立，都隐藏了内部功能的实现方法和细节。图 3-2 显示了 CSTA 服务，是对等层的交互，为水平方向的服务。

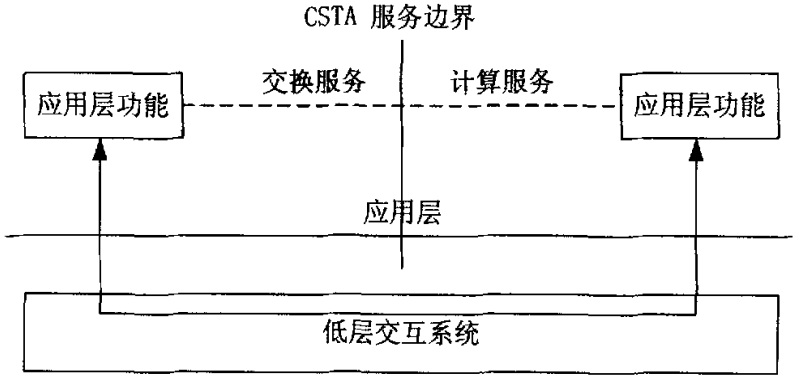



图 3-2 CSTA 服务

### 3.1.3 客户/服务模型

支持CSTA应用的模型为客户/服务(Client/Server)模型。服务（Service）

的请求方为客户端（Client），服务（Service）的提供方为服务端(Server)。客户端（Client）通过与服务端(Server)的通信调用该服务（Service）。

客户/服务模型提供双向通信。如图3-3所示，即计算功能和交换功能都有可能是Server，也都有可能是Client，视具体的服务而定。

交换功能服务：由交换功能提供服务，计算功能作为客户端设备。典型的服务如Make Call Service。

计算功能服务：由计算功能提供服务，交换功能作为客户端设备。典型的服务如Route Request Service。

特殊资源功能服务：由特殊资源功能提供服务，计算功能或交换功能作为客户端设备。典型的服务如Play Message Service。

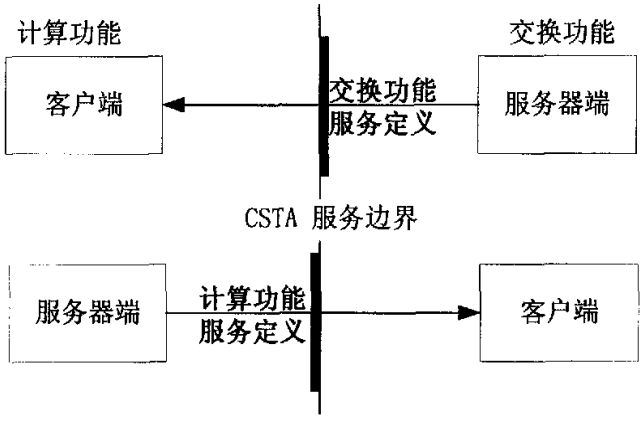



图3-3 客户/服务模型

### 3.1.4 服务和对象

由服务器对客户机提供的服务包括监视和/或操作服务器可以访问的对象。这些对象及其行为在CSTA操作模型中有定义。当一个呼叫在交换子域中可见后，交换子域为呼叫分配一个呼叫标识符，用于在呼叫存在过程中唯一识别该呼叫。呼叫状态由CSTA记录相关数据，包括：Connection（连接，表示一个呼叫和CSTA设备的关系，由呼叫标识和设备标识两部分组成）、Call Event Reports(呼叫事件报告，由交换功能产生)、Call Status（CSTA呼叫状态，由一组与呼叫相关的设备的连接状态表示）。

下面一节详细介绍了CSTA操作模型。

## 3.2 CSTA操作模型

CSTA域包括交换域、计算域和特殊资源域，一个CSTA应用至少包括两个域。一个应用程序可以从计算功能、交换功能和特殊资源功能获得服务。这些计算功能、交换功能和特殊资源功能组成的集合可以定义一个CSTA域。图3-4是CSTA域的例子。图中，CSTA域包含计算域、交换域和特殊资源域，粗线划成的三个区域，左边是计算域，右边是交换域，下方是特殊资源域。计算域包含计算功能：C1，C2 和C3，交换域包含交换功能S1，S2 和S3，特殊资源域包含特殊资源功能SR1，SR2 和SR3。组成某个CSTA应用的功能都被包含在该应用的视图中。CSTA应用包含至少两个不同的子域，下图中的应用域可以表示CSTA应用。

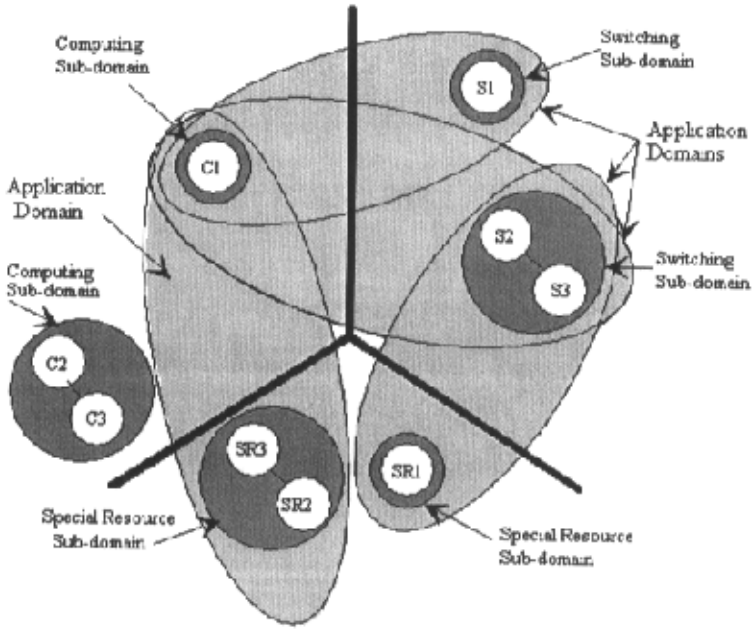



图3-4　域和子域

（该图摘自参考文献[6]）

### 3.2.1 交换子域模型

交换子域模型中定义了交换功能的抽象视图和交换子域对象。交换子域模型中可操作和观察的对象有设备、呼叫和连接。通过CSTA可以观察和控制的设备

叫做CSTA设备，设备具有各自的属性和标识符DeviceID，用于监视和操作。当一个呼叫在交换子域中可见后，交换子域为呼叫分配一个呼叫标识符 CallID，用于在呼叫存在过程中唯一识别该呼叫。连接表示一个交换子域内呼叫和 CSTA 设备的关系。连接可同时被观察和操作。观察和操作连接是呼叫控制服务（比如清除连接、呼叫应答等）的基础。连接属性主要包括连接标识符和连接状态。

交换功能为每一个连接分配一个唯一的标识符。一个呼叫和每一个所连接的设备分配一个连接标识符。每个连接包括一个设备标识符 DeviceID 和一个呼叫标识符 CallID。对一个呼叫来说，连接标识符的数量与该呼叫相联系的设备的数量是一样的，对于一个设备来说，连接标识符的数量与该设备相联系的呼叫的数量是一样的。在一个子域内并且是经过单个服务边界，连接标识符是唯一的。当有一个新的呼叫产生时或是一个新的设备加入到一个呼叫中，交换功能就会产生一个连接标识符。图 3-5 显示了呼叫、设备和连接三者之间的关系。

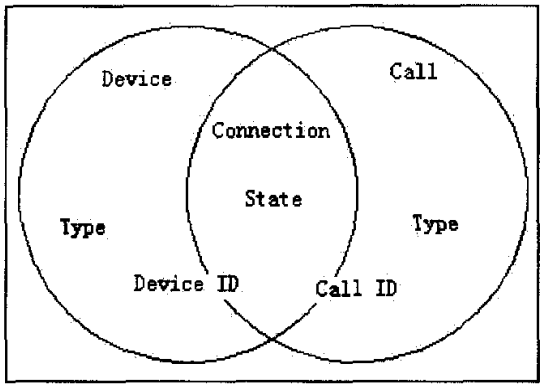



图 3-5 呼叫、设备和连接之间的关系

（该图摘自参考文献[10]）

连接状态包括空闲、呼入（振铃）、失败（忙音）、保留、连接、排队、初始化。连接状态通过“快照”（snapshots）操作获取。空闲状态表示设备和任何呼叫均无关系；振铃意味着呼叫尝试和某个设备建立连接；连接是设备参与到一个呼叫时的状态，包括逻辑参与和物理参与；保持状态说明某个设备被动地参与一个呼叫，逻辑上参与，物理上悬挂；排队时等待接入时的状态；初始化相当于拨号状态。

连接状态的变化以事件报告的形式通过底层信令系统向外界报告。

## 3.2.2 建立联系

### 3.2.2.1 联系控制服务元素（ACSE）

ACSE(Association Control Service Element)，联系控制服务元素，负责联系的建立和释放。所谓联系是指两个应用实体之间的连结，其中一个实体称为发起者（initiator），另一个称为响应者（responsor）。CSTA协议定义的服务，可以通过ACSE建立联系（前提是交换机支持ACSE时，才可以使用ACSE建立交换功能与其他功能的联系）。一旦联系建立起来，交换功能便可以接收CSTA服务消息[18]。

在应用层，不用连接的概念，因为建立连接意味着面向连接的服务，而联系的概念比连接要广泛得多，它包括了很多应用层的语义，建立了应用联系就意味着构造出一种应用平台。ACSE所提供的服务包括：

（1）联系建立。

（2）联系的有序释放，采用这种方式可以避免被传送信息的丢失。

（3）用户或ACSE服务提供者所发起的联系异常释放，采用这种方式可能造成被传送信息的丢失。

ACSE提供了四种服务原语，如表3-1所示。

表3-1　ACSE使用的原语及提供的服务

| 服务原语 | 类型（原语数） | 说明 |
|---|---|---|
| A_ASSOCIATE | 证实型（4条） | 建立一条应用联系 |
| A_RELEASE | 证实型（4条） | 有序释放联系 |
| A_ABORT | 非证实型（2条） | ACSE服务用户发起的异常释放联系 |
| A_P_ABORT | 非证实型（1条） | ACSE服务提供者发起的异常释放联系 |

两个对等的ACSE实体之间利用表示层的服务原语向对方传送信息，这些信息就是ACSE协议数据单元（ACSE APDU）。它们和来自服务原语的其他参数一起，作为相应表示服务原语的用户数据参数，传递到对等的ACSE实体。也就是说，每一ACSE服务原语被一对一地映射到ACSE APDU，然后，每一ACSE APDU被映射到一个表示服务原语。

#### 3.2.2.2 ACSE建立CSTA联系

建立CSTA联系，是由计算机发起建立联系的请求，在计算机跟交换机之间通信的消息，称为CSTA消息。CSTA消息一般包含下面的参数：

1) CSTA版本信息——指明支持CSTA协议的版本号。如果两个交互的系统有不止一个版本，使用其中最高版本的CSTA协议；

2) 功能需求——指明功能所需要的CSTA服务和事件报告；

3) 功能提供——指明功能所提供的支持的最高版本CSTA服务和事件报告。

建立联系的初始化序列，如图3-6所示。首先，计算功能提供参数发出ACSE请求；然后，交换功能发出ACSE应答；接着，交换功能发送一个系统状态信息，Enabled或者Normal；最后，计算功能确认，交换功能收到来自计算机的确认消息后，就完成了初始化序列必须执行的部分 。

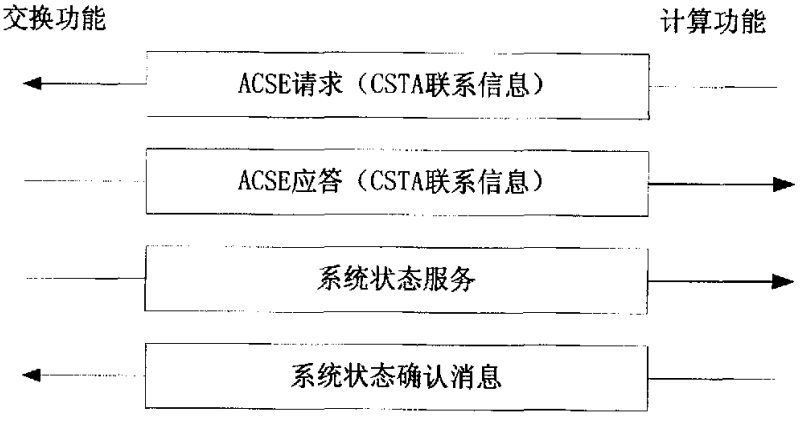



图3-6　建立CSTA联系序列

## 3.3 CSTA提供的服务

CSTA提供的服务有交换功能服务（21项）、状态报告服务（6类）、计算功能服务（5项）、双向服务、输入输出服务（10项）、语音单元服务（12项）。

本论文目前主要用到交换功能服务、状态报告服务、以及计算功能服务。

交换功能服务：这类服务使CTI中间件能够通过交换机完成电信网中呼叫建立、释放、保持、转移等功能，从而使应用程序可以操纵电话设备和声讯设备(如语音卡等）。交换服务不仅能够完成基本的呼叫处理，还可以实现一些特殊的功

能。如“预拨号”可以为工作组发起呼叫。当被叫设备接通后交换机在工作组中选择一个空闲的设备并将呼叫分配给它，这样相当于程序自动建立语音通道，大大减轻了话务员的工作量。

状态报告服务：如果应用程序不知道电话设备的状态,那么它可能向交换机发出错误的指令。比如说,试图从一个正被使用的电话终端为起点发起一个呼叫。为了让应用程序了解设备的状态及呼叫处理过程,CSTA 定义了状态报告服务。状态报告服务采用消息驱动的方式将各个设备上发生的事件通知应用程序。比如说,当一个新的呼叫到达时,物理设备会振铃,同时交换机将会发送一条消息告诉应用程序有呼叫到来。不仅物理设备可能触发事件通知,逻辑设备也可以产生事件,如话务员(Agent) 可以产生登录、注销、暂停分配来话等事件；排队队列(Queue)可以产生呼叫排队等事件。

计算功能服务：CSTA 规范提供了开放的路由接口允许用户定制路由策略根据主叫号码或者用户选择的服务类别来确定路由。交换机维护着 CSTA 路由逻辑设备,应用程序可以监控此对象。呼叫进入交换机时首先到达路由点,交换机向应用程序发出“查找路由”请求,请求中包含呼叫的主叫号码等信息作为路由的依据,应用程序根据这些信息按某种规则如查询数据库等计算出目的地,交换机根据计算结果将呼叫路由到相应的目的地如排队队列、工作组、人工坐席、自动语音播放设备等,这样可以实现基于主叫号码的智能路由来提供个性化的服务。

## 3.4 基于 ASN.1 的 CSTA 语法描述

CSTA 协议处于 OSI 参考模型中的应用层上,在 OSI 参考模型中,应用层的应用实体间通过应用协议数据单元APDU来交换信息,传输中的APDU 可以用多种方法表示,CSTA 协议的定义采用了抽象语法描述 ASN. 1 并选择了基本编码规则 BER (Basic Encoding Rules)。作为一个在很多领域特别是通讯领域得到广泛应用的国际标准,ASN. 1 具有很多优点。首先，它能够在较高的抽象层次上描述数据结构，与具体的平台和编程语言无关；第二,当需要在计算机网络里面传输数据结构信息时，ASN. 1 提供相应的编解码规则,通过相应的位模式来传递数据结构信息。第三,ASN. 1 很好地解决了可扩展性问题,利用 ASN. 1 开发的某个产品的最新版本可以很好的兼容早期版本。

下面介绍一下ASN.1的语法。

### 3.4.1 ASN.1——抽象语法表示方法

ASN.1有严格的BNF定义，具有简洁、精确和无二义性的特点。它有两种用途：一是用于如电子邮件等应用语法；二是用作定义特定协议实体PDU结构的一种手段[18]。

ASN.1类似于高级程序设计语言的数据描述部分，它提供若干语言构件用以定义类型和值。类型和值是任何数据都具有的两个重要的属性，类型对应结构，值对应内容。如果给定一种类型，则该类型的一个值就称为该类型的一个具体实例。但是与其他程序设计语言不同的是，ASN.1的类型不需要由机器实现，例如ASN.1中的整型INTEGER允许使用所有的整数作为其值，这样一种类型在实际机器中是不可能表示的。

ASN.1的基本构件是模块，一种抽象数据类型可以用ASN.1定义成一个模块，这个模块描述了抽象数据类型的抽象语法。模块可以用名字来引用，模块名也是它定义的抽象语法的名字。当应用实体把协议数据单元交给表示服务时，同时要说明这个协议数据单元的抽象语法名[18]。

#### 3.4.1.1 基本类型

ASN.1定义了若干种简单类型，也提供了由简单类型构造复杂数据结构的手段。在研究ASN.1的类型定义之前我们先介绍ASN.1的词汇规则。ASN.1中用的词汇是区分大小写的，这一点与FORTRAN、PASCAL等语言不同。具体地说，ASN.1由以下关于词汇的约定：

（1）所有的名字标识符都由大小写字母、数字和连至符（-）组成，长度不限。

（2）内部类型名和保留字全部用大写字母表示。

（3）用户定义的类型或模块名的第一个字母要大写。

（4）其他标识符（例如值的名字、字段的名字等）的第一个字母要小写。

ASN.1中定义的类型除类型名外还有一个标签（tag），标签由一个保留字和一个非负整数组成，它的作用与编码规则有关，在稍后部分解释。ASN.1中定义

了6种基本类型，如表3-2所示。

表3-2　ASN.1的基本类型

| 基本类型 | 中文名称 | 标签 | 值的集合 |
|---|---|---|---|
| BOOLEAN | 布尔型 | UNIVERSAL 1 | TRUE，FALSE |
| INTEGER | 整型 | UNIVERSAL 2 | 零和任意长度的正负整数 |
| BITSTRING | 字符串 | UNIVERSAL 3 | 0个或多个比特序列 |
| OCTETSTRING | 八位位组串 | UNIVERSAL 4 | 0个或多个八位位组序列 |
| REAL | 实型 | UNIVERSAL 9 | 所有实数 |
| ENUMERATED | 枚举类型 | UNIVERSAL 10 | 一个整数值的表，每个整数可以有一个名字 |

布尔型有两个值，这两个值的名字是ASN.1内定的，所以全用大写。

每一个整型的值可以有一个名字，以表示特别的意义，例如 sunday（1），monday（2）等。

比特串是0个或多个比特的有序集，其中的每个比特都可以赋予名字。比特串的值可以用二进制或十六进制数字串表示，数字写在两个单引号之内，后跟字母B（二进制）或H（十六进制）。为了避免歧义，十六进制数字表示的比特串只能用于整数个字节的比特串。

八位位组串类型的值也可用二进制或十六进制数表示。

实数值一般用科学计数法表示为{M,B,E}，其含义是：$M*B^E$，其中的尾数 M 和指数E可以是任何正负整数值，基数B可以是2或10。

枚举型的值由直接列举的整数以及它们的名字组成的表来表示，例如

ENUMERATED {red(0),green(1), blue(2), white(3), black(4)}

这样的带名字的整数表也可以认为是整数类型，即

INTERGER {red(0),green(1), blue(2), white(3), black(4)}

可以认为是同样的类型。

应该说明，ASN.1把枚举型的值直接与整数相联系，即名字后面的各整数不一定在一个连续的整数范围内。这一点和通常的程序设计语言（如 PASCAL）不相同。这种做法使得枚举值与具体的机器表示无关，也即不依赖于具体的编译程序。

ASN.1 还定义了几种字符串类型（CHARACTER STRING），这些字符串类型都是 OCTET STRING 类型的子类型。每一种字符串类型的值都是取自该字符集中的字符串，字符集包含 G 集（图形符号集）和 C 集（控制字符集）。例如 ASCⅡ码字符集中，G 集含有 ASCⅡ编号从 33 到 126 的字符，C 集含有 ASCⅡ编号从 0 到 31 的字符。空格字符（编号为 32）和删除字符（编号为 127）包含在两个字符集中。ASN.1 定义的字符串型列于表 3-3 中。

表 3-3 ASN.1 的字符串类型

| 类型 | 标签 | 值的集合 |
|---|---|---|
| NumericString | UNIVERSAL 18 | 数字 0 到 9，空格 |
| PrintableString | UNIVERSAL 19 | 所有大小写字母、数字、标点符号和空格 |
| TeletexString | UNIVERSAL 20 | 由原 CCITT T.61 建议定义的字符集 |
| VideotexString | UNIVERSAL 21 | 由原 CCITT T.100 和 T.101 建议定义的字符集 |
| IA5String | UNIVERSAL 22 | 国际字符集 5 （等价于 ASCⅡ） |
| GraphicString | UNIVERSAL 25 | 由 ISO 8824 定义的字符集 |
| VisibleString | UNIVERSAL 26 | 由 ISO 646 定义的字符集，取自 IA5 的图形字符 |
| GeneralString | UNIVERSAL 27 | 通用字符集，所有标准化的字符集的组合 |

#### 3.4.1.2 构造类型

在 ASN.1 中，序列和集合是用以上简单类型构造复杂数据结构的手段。表 3-4 列出了 ASN.1 的构造类型（也称为结构类型）。

表3-4　ASN.1的构造类型

| 构造类型 | 名称 | 标签 | 主要特点 |
|---|---|---|---|
| SEQUENCE | 序列型 | UNIVERSAL 16 | 取值为多个数据类型的按序组成的值 |
| SEQUENCE OF | 同类序列型 | UNIVERSAL 16 | 取值为同一数据类型的按序组成的值 |
| SET | 集合型 | UNIVERSAL 17 | 包含多个数据类型的0个或多个组成元素的无序集合 |
| SET OF | 同类集合型 | UNIVERSAL 17 | 包含同一数据类型的0个或多个组成元素的无序集合 |
| CHOICE | 选择型 | 无 | 可选择多个数据类型中的某一个数据类型 |

SEQUENCE值是包含一个或多个组成元素的有序列表，每一个组成元素是一种ASN.1类型。为提高可读性，用标识符（小写字母开头）形式表示的名字常常与组成元素有关。标识符后面跟有类型描述符，以描述组成元素的类型。每一组成元素的类型描述符后面可以跟OPTIONAL或DEFAULT保留字。OPTIONAL表示在SEQUENCE值中，该组成元素不一定出现。DEFAULT表示了类似的意义，但如果该组成元素的值没有出现时，它将分配一事先确定的默认值。下例给出了使用SEQUENCE构造类型的一个PDU的定义，该PDU用于FTAM（File Transfer Access and Management）中。

```
SEQUENCE {
stateResult        StateResult DEFAULT success,
actionResult       ActionResult DEFAULT success,
ATTRIBUTES         SELECTATTRIBUTES,
sharedASEinform    SharedASEinform OPTIONAL,
diagnostic         Diagnostic OPTIONAL  }
```

SEQUENCE OF类型是SEQUENCE类型的特例，它的组成元素必须是同一个ASN.1类型（有点象一个数组）。例如：SEQUENCE OF VisibleString。

SET类型类似于SEQUENCE类型，但组成元素是无序排列的。也就是说接收方和发送方的元素的顺序可以不一致。例如

```
SET {
name            [0]     VisibleString OPTIONAL,
organization    [1]     VisibleString OPTIONAL,
countryName     [2]     VisibleString OPTIONAL  }
```

上面例子中的[0]、[1]、[2]是为了消除歧义性而采用的上下文有关的标签。

SET OF类型的元素也是无序的，但要求具有同一的ASN.1类型。

CHOICE类型包含一个可供选择类型的列表。CHOICE类型的每一个值都是该组成元素（已知类型）的一个值。当描述的值在不同环境下可能有不同的类型，而且所有这些可能性事先都能知道的话，那么CHOICE类型是很有用的。例如说明ROSE（Remote Operations Service Element）协议使用的4种APDU的CHOICE类型如下：

```
CHOICE {
roiv-apdu               ROIVapdu,
rors-apdu               RORSapdu,
roer-apdu               ROERapdu,
rorj-apdu               RORJapdu  }
```

此外，ASN.1还定义了几种数据类型。如ANY类型（任意型，在定义数据时还不知道其类型的情况下使用，以后可以被任一 ASN.1 类型置换），OBJECT IDENTIFIER（说明对象标识符），NULL（只取空值的类型，用于尚未获得数据的情况下）等。

#### 3.4.1.3 模块定义、类型分配与值分配

ASN.1模块的一般格式如下：

```
<moduleIdentifier> DEFINITIONS TagDefault ::=
                    BEGIN
                            EXPORTS
                            IMPORTS
                            AssignmentList
                    END
```

其中模块标识符moduleIdentifier是模块名，后面可以有选择地跟一个标识模块的对象标识符，模块名的第一个字母必须大写。TagDefault结构与标签（tag）有关，我们将在后面解释。符号“::=”称为赋值号。EXPORTS结构规定了模块中某些定义是从其他模块移植来的。IMPORTS结构用于定义其他模块可以移植的类型或值。除非moduleIdentifier包括了对象标识符，否则不可以使用IMPORTS或EXPORTS。模块中的AssignmentList中含有类型分配（type assignment）、值分配（value assignment）和宏定义（MACRO）。类型分配语句的格式为：<类型名>::=<类型定义>，同样，值分配（对标识符赋一个值）也采用以上格式的语句。比如定义一个模块的例子如下：

```
ModuleExample DEFINITIONS ::=
BEGIN
TypeA ::= INTEGER
TypeB ::= BOOLEAN
valueA TypeA ::= 10
valueB TypeB ::= TRUE
END
```

该模块名为ModuleExample，模块定义从赋值号后开始，模块体在保留字BEGIN和END之间定义。它含有两个类型分配（即TypeA和TypeB），以及两个值分配。我们来看一个类型分配语句的例子：

```
Fadu ::= SEQUENCE {
node            NodeDescriptor,
data            DU OPTIONAL,
children        SEQUENCE OF Fadu  }
```

该例取自FTAM标准，定义Fadu为SEQUENCE类型。这里假定NodeDescriptor类型已经在别的地方做了定义。从该例可以看出，ASN.1允许递归类型的定义。再举几个值分配和类型分配的例子：

```
(1) bitA BITSTRING ::=  '10110' B          -- 5bits
    bitB BITSTRING ::=  '1A45D' H          -- 20bits
(2) pi REAL ::= {31415927, 10, -7}
```

```
(3) IntColor ::= ENUMERATED { red(0), blue(1), green(2), white(3) }
    colorX IntColor ::= green                    -- or, same as next line
    colorX IntColor ::= 2
```

上面三个例子的意义是明显的，需要说明的是双连字符(--)的作用是引入注释，可用另一个双连字符结束，或者以该行的结束为注释结束。

#### 3.4.1.4 标签

在传送的序列或集合中，如果缺少了某些任选项，接收端就可能无法区分已有的项（前面讲过的协议数据单元中有很多字段就是任选参数）。例如由6个字段组成的序列中有2个任选项散布其间，如果6个字段的类型都相同（比如同为INTEGER），那么，当有些任选项出现，有些任选项不出现时如何区分它们呢？

ASN.1解决这个问题的办法是加标签。前面介绍的几种内部类型都有ASN.1规定的通用类标签（UNIVERSAL）。标签有两个分量，一个分量是类（class）标识符，另一个分量是非负整数。标签的作用就是保证译码时能够无二义性地得到原来的数据类型。任何一类标签都需放在类型名前面的方括号中。标签共划分为以下四类（class），并使用括号中的标识符：

（1）通用类（UNIVERSAL）：由ASN.1分配给它所定义的最常用的一些数据类型，它和具体的应用无关。除了CHOICE和ANY类型外，所有的基本类型和构造类型都分配了一个通用类标签，在上述的几个表中已经列出了部分通用类标签以及所对应的ASN.1类型。

（2）应用类（APPLICATION）：OSI标准中所定义的类型，如X.400 MHS、FTAM等。在一个特定的标准或建议中，一个应用类标签只指定给一种类型，从而在其作用范围内保持唯一性。

（3）上下文类（CONTEXT）：上下文所定义的类型，它属于一个应用的子集。其作用范围仅限于特定的上下文，例如在一个序列或集合结构内部有效。这种标签不需要类标识符，仅由一个非负整数组成。

（4）专用类（PRIVATE）：不由ISO标准或ITU-T建议分配，而是保留给一些厂家所自己定义的类型。

标签与编码规则有关。当传送一个字段时，对其类型、长度和值都要进行编

码以便传送。如果一个字段加上了标签，并且接收方知道该标签所代表的类型，则不必发送类型信息而用标签的编码代替之。这时可把标签说明为 IMPLICIT，表示类型是隐含的。如果接收方不知道加标签字段的类型，则必须把标签、类型、长度和值都进行编码发送。可见隐含标签可产生较短的编码，当有大量的数据结构要传送时，使用 IMPLICIT 可以节省相当数量的编码字节，只要可能就应尽量采用。注意，虽然标签没有传输，但接收方可以从 DCS（即已定义的上下文集）中推导出标签的信息。

ANY 和 CHOICE 类型不能标签为 IMPLICIT。如果将它用于 ANY 型，它就不可能区分是用哪一种 ASN.1 类型替代。如果将它用于 CHOICE 型，它也不可能区分是哪一个元素被选用。下面列举几个标签的使用例子：

```
Exuniv ::= [UNIVERSAL 2] INTEGER
Exappl ::= [APPLICATION 0] INTEGER
Expriv ::= [PRIVATE 1] INTEGER
Excont ::= SET {
        type1  [0]  INTEGER OPTIONAL
        type2  [1]  INTEGER OPTIONAL  }
```

因为 Excont 类型的两个组成元素是同一类型（即整型），都是任选的，所以当 Excont 类型的值仅含有一个组成元素时，就需指明该组成元素值是来自哪一个组成元素。这样采用上下文类标签（即[0]和[1]），就可以消除歧义性。需要说明的是，标签后的类型本质上是一种新类型。例如 Expriv ::= [PRIVATE 1] INTEGER 不同于 Expriv ::= INTEGER。

#### 3.4.1.5 ASN. 1 的宏定义功能

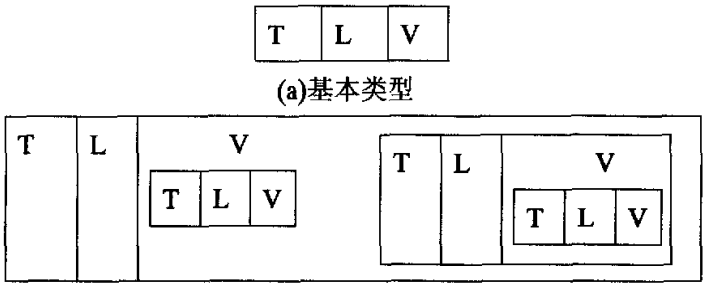
ASN. 1 有一个有用的特性：宏定义功能。远程操作表示法利用 ASN. 1 宏定义应用上下文的数据部分，包括应用上下文中使用的应用服务元素，在两个通信应用实体间交换的联结/ 去联结信息，在应用联系中使用的抽象文法等。远程操作表示法定义了 6 种 ASN. 1 宏，宏 Bind/ Unbind 用于定义在连接/去连接过程中两个应用实体调用间交换的信息。宏 OPERATION 用于定义远程操作；宏 ERROR 用于定义在执行操作时可能发生的错误；宏 APPLICATION - SERVICE

-ELEMENT 用于定义应用服务元素；宏 APPLICATION-CONTEXT 用于定义应用上下文。

### 3.4.2 BER——基本编码规则

基本编码规则 BER 中有多种方法，用来把 ASN.1 的抽象值编码为字节串，这个字节串就是抽象值的传送语法形式。编码的基本原则是类型—长度—值的三段式结构，简称 TLV（Type—Length—Value）结构。任何编码的第一个字段是类型（T）字段，表明了关于标签和编码格式的信息；长度（L）字段定义数值的长度（按八位位组计）；值（V）字段表示实际的数值。每一个字段是按八位位组对齐的，并且遵循靠左对齐的原则。

BER 编码的值部分是递归的，即它本身也可以是 TLV 结构，这样就具备了表达复杂数据的能力。图 3-7 (a)和(b)分别表示了编码的基本型结构和递归型结构。




(a)基本类型

(b)结构类型

图 3-7 ASN.1 编码格式

在类型字段中，前两位用于标识四种标签，00 代表通用类，01 代表应用类，10 代表上下文类，11 代表专用类。第三位用于区分简单类型和构造类型，0 代表基本类型，1 代表构造类型。剩余的五位用于对标签的数值部分进行编码。如果标签的值大于 30，则这五位为全 1，实际的标签值表示在后继字节中。如图 3-8 所示。

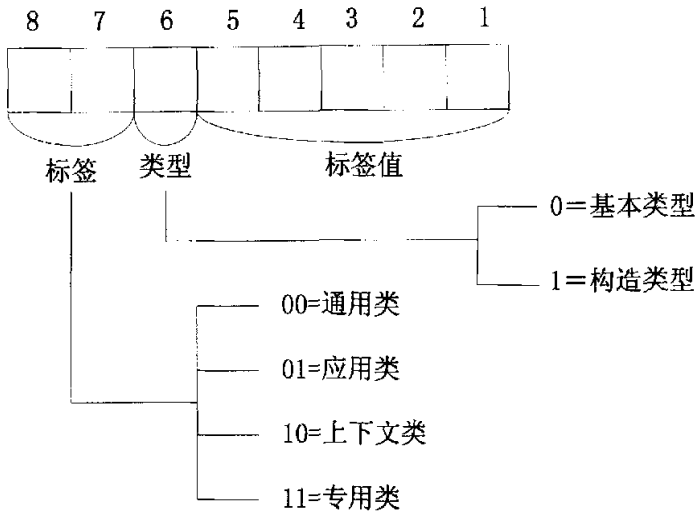



图 3-8 TVL 编码的第一个字节

我们先以标签[APPLICATION 23]为例，编码后的类型字段由一个八位位组“01110111”B 组成。最左边两位（01）表示应用类，接下来以为（1）表示编码格式为构造类型，后继的五位（10111）表示标签号为 23。再看一个标签为[PRIVATE 42]的例子。42 的二进制表示是 101010，多于五位，因此要用两个八位位组编码。编码后的标签的二进制格式为 11111111 00101010，关于扩充字节的方法在后面再介绍。

标签的引入增加了编码的长度，为了节约编码空间，在 ASN.1 模块定义中可以使用 TagDefault 结构。假如将 TagDefault 置为 IMPLICIT，那么只有用户定义的标签（而不是预先定义的标签）类型才需编码。但如果在用户定义的标签后跟有保留字 EXPLICIT（显式），或者在模块定义中没有用 TagDefault 结构，那么用户定义的标签和预先定义的标签均需编码。

下面先介绍几个简单编码的例子，其中的数值如果没有注明都是十六进制数。

1）BOOLEAN 类型有两个值 TRUE 和 FALSE，都用一个字节表示，TRUE 是 FF，FALSE 是 00，布尔型是基本类型，标签为 UNIVERSAL 1，因此，值 TRUE 和 FALSE 的编码分别为“01 01 FF”H 和“01 01 00”H。

2）十进制整数-2 的编码为“0201FE”H，其中“-2”的二进制补码是 FE。

3）对于 BITSTRING 类型的值，因为它可能不是字节对齐的，所以在编码的

第三个字段（即值字段）中，用起始的第一个字节来表明值字段的最后一个字节中未使用的位数，这个数必须在0到7之间。例如‘0011 1010 1010 1010 1010’B的20比特的BITSTRING值，编码为“03 04 04 3A AA A0”H，其中03是编码类型字段，第一个04是编码长度字段，第二个04是未使用的位数，3A AA A0是编码值。

4）OCTETSTRING类型的值总是占用整数个字节，所以不需说明未使用的位数，只用长度字段就可以表明位组串的值了。例如八位位组串ACE的编码为“04 02 AC E0”H。

5）空值NULL的编码为“05 00”H，仅含有类型（05）和长度（00）两部分。对于构造类型的编码稍复杂一些，其类型字段的第3位要置为1。另外，值部分也是有一定结构的，也按TLV规则编码，每一个TLV结构表示其中的一个字段。我们接着给出几个例子。

6）SEQUENCE {

```
name        OCTETSTRING,
place       INTEGER {room1(0), room2(1), room3(2) }
}
```

该类型的值{name ‘1AA2FFGH’, place room3}的编码是：30 09 04 04 1A A2 FF GH 02 01 02。按照序列的结构可展开如下：

30 09

04 04 1A A2 FF GH

02 01 02

其中，30是编码后的类型字段，09是编码后的长度，其余部分给出其两个组成元素的编码。

7）集合类型{length INTEGER, soft BOOLEAN}的值{length 5, soft TRUE}编码为

31 06 02 01 05 01 01 FF

因为集合类型的元素可以是无序的，所以也可编码为

31 06 01 01 FF 02 01 05

由于该集合类型的两个元素类型不同，所以可以根据其类型进行区分。

下面的例子说明标签的使用情况。

假设设计一个文件安全协议，这里定义的类型其有效范围仅限于我们面对的这个应用，因此可以用APPLICATION标签来标识当前定义的类型。我们把口令定义为下面的类型：

PASSWORD ::= [APPLICATION 27] OCTETSTRING

这样，利用八位位组串定义了这个应用中的一个新类型 PASSWORD。如果字符串“Sesame”属于PASSWORD类型，则可以编码为

7B 08 04 06 53 65 73 61 6D 65

从中可以看出，应用标签和原来的标签都被编码了。为了减少编码长度，当在这个应用范围中标签值27代表的原来类型为已知时，可使用隐含标签，例如重新定义PASSWORD类型为：

PASSWORD ::= [APPLICATION 27] IMPLICIT OCTETSTRING

相应的“Sesame”编码为

5B 06 53 65 73 61 6D 65

由于这里只有一种类型信息，所以应认为是简单类型，第一字节的第三位置0。

我们再说明一个关于上下文类标签的编码例子。设有下面的集合类型：

```
Parentage ::= SET {
   subjectName   [1]  IMPLICIT IA5String,
   motherName    [2]  IMPLICIT IA5String OPTIONAL,
   fatherName    [3]  IMPLICIT IA5String OPTIONAL
                    }
```

这个集合类型中有三个相同类型的名字，其中两个是任选的，唯一能区分它们的是上下文类的标签。根据这种标签三个字段编码的第一个字节分别是 A1、A2和A3。

最后介绍BER扩充字节长度的方法。首先要扩充的是编码中第一个字节的后五位。当指定的标签数值部分大于31时这五位无法表示，需要扩充。另外一个要求扩充的是编码的长度字节，当抽象值的长度超过255时，一个字节也无法表示，同样需要扩充。

对标签数值表示的扩充方法如下：用五位表示0—30的编码，当标签数值大

于等于 31 时，这五位置全 1，实际的数值编码表示在后继字节中。后继字节的左边第一位表示是否为最后一个扩充字节，只有最后一个扩充字节的左边第一位置 0，其他扩充字节左边第一位置 1。这样，每个扩充字节实际只用了七位表示标签数值的编码。

对长度字节的扩充方法是：小于等于 127 的数用长度字节的右边七位表示，最左边的一位置 0；长度大于 127 的数用后继的若干字节（最多 126 个）表示，原来的长度字节的第一位置 1，其他七位表示后继的用于表示长度的字节数。例如十进制 255 可表示为 1000 0001 1111 1111。注意，对长度字节数的限制是 126，而不是 127，也就是说开头字节最大为 1111 1110，而不是 1111 1111，这个值是为以后的扩充保留的。

### 3.4.3 ASN.1 描述的 CSTA 协议

CSTA 是通过 ROSE 规范来定义计算机与交换机之间所有需要交互的业务单元的。在 ROSE 中用 ASN. 1 定义了 BIND、UNBIND、OPERATION、ERROR 这几个宏，CSTA 中所有的业务都是用这几个宏来定义的。

#### 3.4.3.1 远程操作服务元素（ROSE）

在分布式的交互应用环境中，一个应用实体可能要调用另一个远程应用实体的操作，这称为远程操作 RO（Remote Operation）。这种远程操作的调用往往采用请求/响应的工作方式，通常把这种工作方式叫做远程过程调用 RPC（Remote Procedure Call）。由于许多分布式应用系统要使用远程操作，所以在 OSI 应用层中就定义了一个公用服务元素 ROSE[18]。

ROSE 用来发起和管理远程应用实体间的交互式远程操作，涉及两个应用实体。请求 ROSE 服务的应用实体称为调用者（invoker），接受请求的应用实体则被称为执行者（performer）。调用者请求远程实体（执行者）执行某个操作，并把所要进行的远程操作名和携带的参数传递给执行者。

ROSE 定义了 5 类操作：

(1) 同步，需要返回结果或出错信息。

(2) 异步，需要返回结果或出错信息。

(3) 异步,只需要返回出错信息,成功的话不用返回。

(4) 异步,只需要返回结果,出错的话不用返回。

(5) 异步,不需要返回任何结果或出错信息。

其中,同步的意思是请求方必须得到被请求方的响应后才能发出下一请求,异步则不需要。

ROSE 提供 5 种服务原语,它们都是非证实型的。各种原语结合不同的参数值可以提供丰富的远程操作服务。表 3-5 列出了 ROSE 的 5 种服务原语。

表 3-5　ROSE 的 5 种服务原语

| 服务原语 | 类型 | 说明 |
|---|---|---|
| RO_INVOKE | 非证实型 | 调用远程操作 |
| RO_RESULT | 非证实型 | 返回操作的结果 |
| RO_ERROR | 非证实型 | 操作失败,返回否定应答 |
| RO_REJECT_U | 非证实型 | 服务用户拒绝执行操作 |
| RO_REJECT_P | 非证实型 | 服务提供者拒绝执行操作 |

CSTA 协议通过 OPERATION 宏定义时可能有 ARGUMENT、RESULT、ERROR 3 个参数,通过这 3 个参数的具体类型不同来表示不同的业务,通过这 3 个参数的有无来判断不同的操作类型。例如如果仅有 ARGUMENT 参数,而没有 RESULT 和 ERROR 参数,就属于第五类操作。

### 3.4.3.2 CSTA 协议中操作的定义

ECMA-218 应用 ASN.1 描述了 CSTA 服务。CSTA 协议中的操作一般定义如下:

```
CSTA - operation - name {对象标识符}
DEFINITIONS : : = BEGIN
Operation - Name OPERATION
ARGUMENT OperationArgument
RESULT OperationResult
ERRORS {universalFailure}      END
```

其中,OPERATION 和 ERROR 是远程操作表示法中定义的宏,OperationArgument 指明执行这一操作所需的参数,OperationResult 定义了操作返回的结果,universalFailure 则表示操作可能出现的错误。

# 第四章 基于CSTA的CTI中间件的实现

CTI 中间件一方面作为客户机访问交换机提供的服务,另一方面它将 CSTA 服务在网络上分布,为应用程序提供服务。这样，我们抽象出 CTI 应用的逻辑层次,将 CTI 应用分为业务层、消息处理层（CTI 中间件）以及交换层。消息处理层又可以分为 CSTA 解析层和 API 层。CSTA 解析层用来解析与交换机间传递的消息；API 层为提供给上层应用的编程接口。下面首先介绍一下 CTI 应用的网络结构，然后稍具体地介绍一下 CTI 应用的逻辑层次，接着阐述 CTI 中间件的实现包括 CSTA 解析层的实现和 API 的提供，最后介绍调用 CTI 中间件提供的接口可以实现的 CTI 功能。

## 4.1 CTI 应用的典型网络结构及本 CTI 中间件开发环境

### 4.1.1CTI 应用的网络结构

基于企业局域网的第三方控制的 CTI 应用网络有三个主要组成部分：客户端、CTI 服务器、交换机。客户端的应用程序通过 API 与服务器相连，服务器通过 CTI Link 与交换机相连。

服务器的工作包括下面的功能：

1) 接收客户端应用程序发来的呼叫请求，将这些请求转化为交换机支持的协议消息并传给交换机执行。交换机应答，服务器将应答结果传给客户端；
2) 监测呼叫事件，并发送信息给向服务器发送请求的客户端；
3) 管理电话服务请求的流程以及服务器与应用系统间的状态信息。这样，应用程序开发者可以在程序中实现完整的呼叫路由、监测和呼叫控制功能。

本论文采用的网络结构如图 4-1 所示。

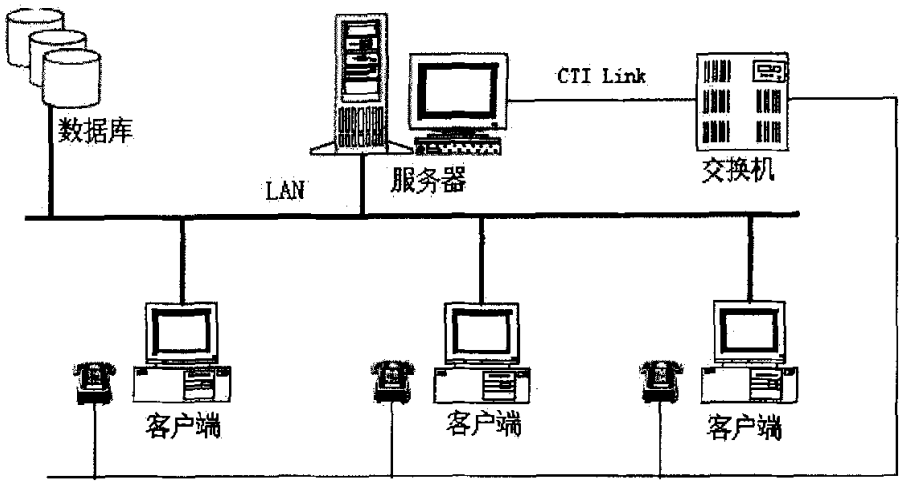



图 4-1 基于CSTA的CTI应用的网络结构

本论文采用的是西门子HiPath 3000交换机，该型号的交换机是中小型CTI产品开发商的标准平台，能方便应对新的业务需要，支持CSTA的最高版本是CSTA第三版。本论文所实现的CTI中间件安装于服务器，同时对客户端的应用程序提供API，起到第三方控制的作用。

### 4.1.2 本CTI中间件开发环境

操作系统：Windows2000

开发工具：Visual C++ 6.0

## 4.2 基于CSTA协议的CTI应用的逻辑层次

可以将CSTA应用抽象为计算机对交换机的控制。逻辑层次如图4-2所示，交换层为支持CSTA的交换机。所谓支持CSTA的交换机，是指具有ASN.1编解码层，并且具有传输层（使得交换机可以通过CTI Link与计算机通信）。消息处理层，实现对交换机、计算机之间传送的消息进行编/解码，并提供应用程序接口给上层的CTI应用程序使用。业务层，实现如呼叫中心等应用。本文所实现的CTI中间件，为API层中间件，处于消息处理层，它实现了CSTA解析层以及提供应用程序接口。CSTA解析层实现的是计算机与交换机之间的消息交互，API层为业务层实现对交换机的操作提供了接口。

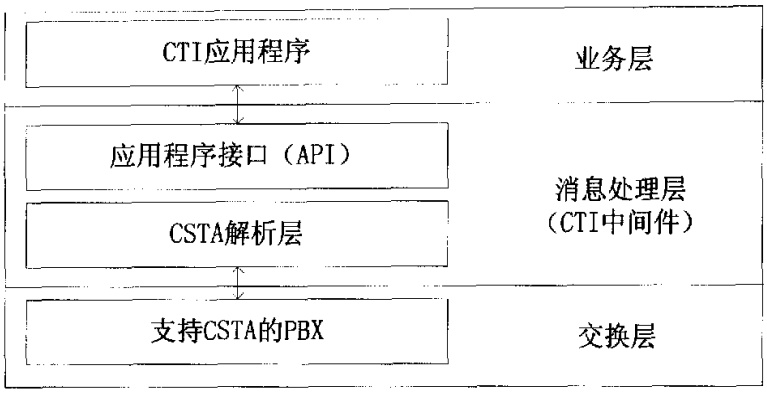



图 4-2 CSTA 应用分层结构

## 4.3 基于 CSTA 协议的 CTI 中间件的实现

### 4.3.1 CSTA 解析层的实现

所有的 CSTA 应用都基于 OSI 的 ISO 7 层参考模型。因此 CSTA 解析层是基于 ISO 7 层参考模型的。就本文研究的情况，对于 OSI 7 层参考模型中的第一层到第四层，由于服务器与交换机通过 TCP/IP 连接，所以网络节点通过基于 TCP 的 Socket 进行通信，传输层使用可靠 TCP 协议连接，因此第 5 层以及更上层可以认为下层的通信是可靠的、无差错的。传输层通过 Socket 建立连接以后，会话层在此基础上建立会话连接，接着应用层便可建立应用实体间的联系，这里应用实体指的就是交换功能与计算功能，联系建立以后，交换功能与计算功能便可以通信了，二者可以相互发送 CSTA 消息的编码。CSTA 解析层首先要解决的问题就是逐层建立与交换机的连接，使得计算机与交换机间可以进行 CSTA 消息的相互传送；接着就是将 ASN.1 描述的 CSTA 消息编码成字节流，传送给交换机，或者将交换机传送过来的字节流解码，从中获得 CSTA 服务中的参数，业务层可以通过 API 获取这些数据执行相关的操作。

下面就分别从建立连接、联系与 CSTA 消息的编解码两个方面阐述 CSTA 解析层的实现。

#### 4.3.1.1 协议层连接/联系的建立

- 传输层连接的建立

当计算机需要调用 CSTA 的交换功能、状态服务等功能时，交换机相当于TCP/IP服务器，它可以连接一个CSTA客户端（CTI服务器）。此时，CSTA客户端要提供的参数包括交换机的IP地址和TCP端口地址（7001），7001端口是回呼信号缓存管理端口（ callbacks to cache managers）。有了IP地址和TCP端口地址，传输层的连接便可以通过Socket实现。

➢ 会话层连接的建立

设置会话层的目的是管理用户应用进程之间的对话过程，即提供进程间的会话服务。所谓对话（dialogue）即是指本地系统的会话实体与远地对等的会话实体之间交换数据的过程。

会话服务三个阶段：连接建立、数据传送和连接释放。

应用层建立联系开始之前，第五层会话层必须已经建立。而在建立第五层之前，底层的连接必须已经建立。对于通过TCP/IP连接来说，通过SOCKET接口，将底层的连接建立起来。

只有计算机端可以发起建立会话的请求。当第五层的连接成功建立以后，第七层的数据就可以传输了。

当第七层的所有应用都结束的时候，会话才可以关闭。

会话层消息包结构如图4-3所示，包括消息头和数据。下面详细分析一下消息头的结构，并说明不同作用的消息包消息头各个字节的值如何设置。

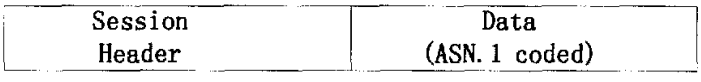



图4-3 会话层消息包结构

消息头的结构如图4-4所示：

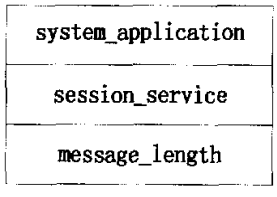



图4-4 会话层消息头结构

system_application、session_services、message_length各占一个字节。

1)system_application指明应用的类型，CSTA（第二版）应用为下面的值：

CTI_APPLICATION = 0x20

2)session_services指明所用会话层的服务的类型，可能为下面的几个值中的一个：

LOGIN_REQUEST = 0x00: Open session，发起建立会话层的请求；

LOGIN_RESPONSE = 0x01: Acknowledgment for LOGIN，确认请求消息；

LOGOFF_REQUEST = 0x02: Close session，关闭一个会话；

AINFO = 0x03: Transmission of data within this session，指明消息包为传输数据。

3)message_length指明消息头之后的数据长度。如果是AINFO，那么message_length是CSTA 消息编码的长度。

消息包的数据部分包括数据的名称，长度，以及数据的内容。

➢ 应用层联系的建立与释放

应用层为用户的应用进程访问OSI环境提供服务。OSI关心的主要是进程之间的通信行为，因而对应用进程所进行的抽象只保留了应用进程与应用进程间交互行为的有关部分。这种现象实际上是对应用进程某种程度上的简化。经过抽象后的应用进程就是应用实体AE（Application Entity)。对等到应用实体间的通信使用应用协议。CSTA采用ACSE建立两个应用实体之间的联系，也就是交换功能和计算功能之间的联系。本文在3.2.2节对ACSE已经做过介绍，下面介绍如何实现联系的建立、释放或异常中断。

1) 联系的建立：

首先，计算机向交换机发送一个协议数据单元(APDU)：ACSE-request (AARQ)。AARQ包含的信息有ACSE协议版本信息，应用上下文名称，以及用户数据信息。如下表4-1所示。

表 4-1 AARQ 包含的数据信息

| Field | Contents |
|---|---|
| Protocol version number | version1 |
| Application context name | iso (1)<br>identified-organisation (3)<br>icd-ecma (0012)<br>standard (0)<br>csta2 (218)<br>application-context-information (200)<br>All other optional fields are omitted. |
| User information | ACSEUserInformationForCSTA sequence |

其中，用户数据部分的序列在ECMA-218中定义如下：

ACSEUserInformationForCSTA ::= SEQUENCE

{cSTAVersion CSTAVersion,

cSTAFunctionsRequiredByApplication CSTAFunctionality,

cSTAFunctionsThatCanBeSupplied CSTAFunctionality }

也就是AARQ的用户数据部分包含CSTA的版本信息、AARQ发起者所需要的CSTA服务、AARQ发起者可以提供的CSTA服务。

然后，如果交换机以一个AARE消息确认，那么CSTA联系就可以建立起来。由交换机发给计算机的AARE消息包中，ACSEUserInformationForCSTA序列包含交换机所需要以及可以提供的CSTA服务。

联系建立起来以后，交换机便可以向计算机发送 CSTA 系统状态信息。联系的建立过程如图 4-5 所示：

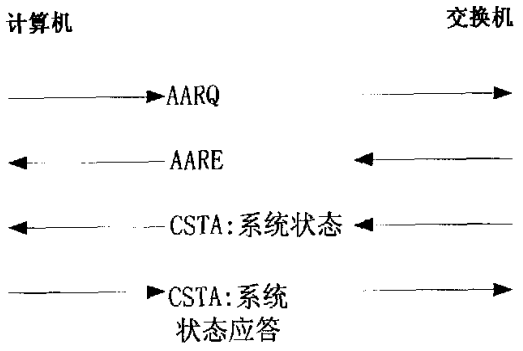



图 4-5 ACSE 建立 CSTA 联系

2) 释放联系：

CSTA 联系的释放可以从计算机和交换机两边发起。如图 4-6 和图 4-7 所示。对于交换机，当侦测到有主要问题时，它会发起释放请求。RLRQ 必须被确认。图 4-6 是计算机发起的释放请求的过程。图 4-7 是交换机发起的释放请求的过程。

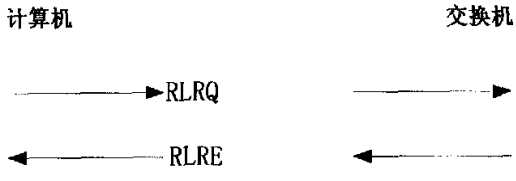



图 4-6 计算机发起的释放请求

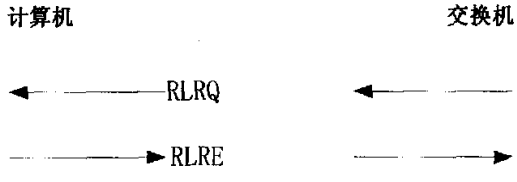



图 4-7 交换机发起的释放请求

3) 异常中断（不需确认的释放）：

在紧急情况下，CSTA 联系的释放可以通过 ABRT (ACSE Abort)来完成。这意味着不用确认，联系直接中断。如图 4-8 是计算机发起的释放

请求，图4-9是交换机发起的释放请求：


图4-8 计算机发起的释放请求


图4-9 交换机发起的释放请求

#### 4.3.1.2 编解码的实现过程

首先简单介绍一下从ASN.1描述的模块到可执行程序的一个过程。

通过适当的转换，ASN.1 可以在常用语言中使用 C/C++。SNACC 是 Sample Neufeld ASN.1 to C/C++ Compiler 的缩写，它可以将ASN.1描述的模块转换为C/C++语言描述的结构以及对数值的编解码函数，这些编解码函数严格按照 BER 规则进行编码。由SNACC产生的代码，主要包括编码、解码、打印和释放四个函数，而且每个类型都生成这四个函数。

图4-10显示了如何根据ASN.1描述的模块实现具体的应用的：

ASN.1文件主要是ASN.1描述的CSTA服务；

ASN.1编译器采用开源的SNACC编译程序；

C头文件是由ASN.1文件经过SNACC编译以后，生成的头文件，CSTA服务的消息结构用C语言来描述；

应用程序源码是指调用CSTA服务的应用源程序；

编解码函数库指SNACC的函数库，调用这些函数可以对ASN.1各种类型及其值进行编解码。

具体的做法是：

首先将ECMA-218协议中描述服务的各个模块放入SNACC中进行编译，每个服务的模块生成服务消息结构的C语言描述以及对该消息进行编解码的函数，包括头文件和函数定义C文件，应用程序使用某个CSTA服务时，需要包含头文件来调用该服务的编解码函数。另外，调用SNACC的库函数实现ASN.1类型的编码，

应用程序编译后链接SNACC的库函数，即可以实现可执行的应用程序。

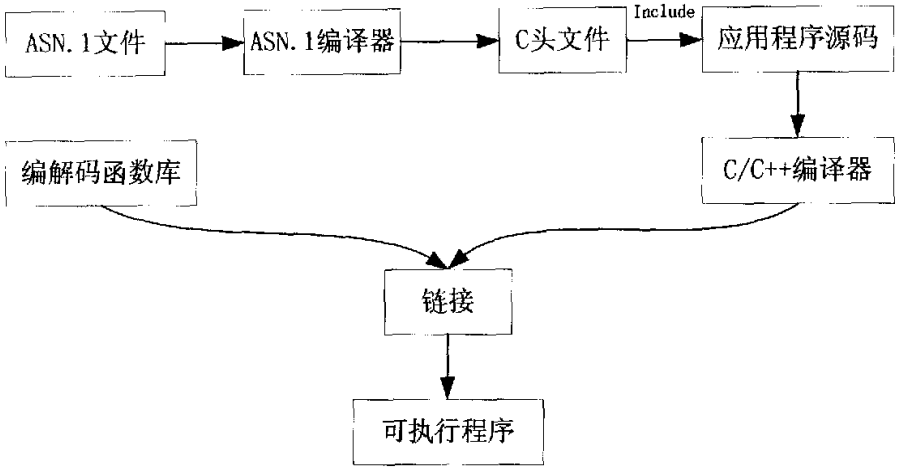



图4-1 ASN.1语法描述模块到应用程序的最终实现

下面以CSTA协议中的Make Call服务为例介绍如何由SNACC生成的编解码函数对服务进行编码的。CSTA应用通过ACSE建立联系以后,所有的CSTA服务都是通过ROSE APDU封装后发送的。

Make Call是由计算机向交换机发出服务请求，交换机返回结果。

每个CSTA服务有一个编号，用来标识不同的服务，比如Make Call服务编号值为local ： 10。如下所示：

```
makeCall OPERATION ::= {
        ARGUMENT   MakeCallArgument
        RESULT MakeCallResult
        ERRORS {universalFailure}
        CODE local : 10
}
```

计算机——〉交换机消息结构如表4-2所示：

表4-2　Make Call请求消息结构

| ROSE_INVOKE | | |
|---|---|---|
| InvokeID | INTEGER | |
| OPERATION | INTEGER | |
| SEQUENCE | DeviceID | IA5String |
| | CalledDeviceID | IA5String |

ROSE_INVOKE表明了该消息为ROSE调用，InvokeID用来标识异步操作中的不同应答和调用的对应关系，OPERATION的值表明CSTA服务类型。

交换机——）计算机消息结构如表4-3或者表4-4所示：

表4-3 Make Call服务的交换机应答消息结构

| ROSE_RESULT | |
|---|---|
| InvokeID | INTEGER |
| ConnectionID | INTEGER |

表4-3中，ROSE_RESULT表明了该消息为返回结果， InvokeID表明该应答对应的ROSE调用消息，ConnectionID标识连接号，表明连接已经建立。

表4-4 Make Call服务的交换机返回错误消息结构

| ROSE_ERROR | |
|---|---|
| InvokeID | INTEGER |
| UniversalFailure | INTEGER |
| Parameter | |

表4-4中，ROSE_RESULT表明了该消息为返回错误， InvokeID表明该应答对应的ROSE调用消息，UniversalFailure表明错误类型。

下面就介绍一下编解码的过程：

- **编码过程**

CSTA使用ROSE协议封装一般的消息头。编码需要遵循下面的步骤：

1）对CSTA基本消息类型进行编码；

2）CSTA消息编码完成后，将编码结果封装在一个ROSE消息结构中，然后再对这个ROSE消息进行编码，于是完成编码过程。

下面详细介绍一下上面两个步骤。

1）对一个CSTA消息进行编码。CSTA类结构的变量首先要赋予数值。这些CSTA类结构通常对应ARGUMENT或RESULT类型（ARGUMENT或RESULT类型是在CSTA中定义的供OPERATION类使用的模块）。

还是以makeCall操作为例。

例如，下面的模块定义了makeCall操作：

```
makeCall OPERATION ::= {
    ARGUMENT   MakeCallArgument
    RESULT MakeCallResult
```

```
    ERRORS {universalFailure}
    CODE local : 10
}
```

其中，CODE用来标识makeCall这个操作，其值为“local:10”。ARGUMENT定义了MakeCallArgument类型。MakeCallArgument被用来调用makeCall操作。RESULT定义了MakeCallResult类型，用来返回makeCall操作的结果。ERRORS定义了另一个模块，包含了makeCall操作过程中可能出现的错误。

对于每个操作，对操作进行编码的时候，用户要求设置操作CODE的值，并且对表中定义的Argument类型进行定义。解码时，用户检查操作码值，并且对相应的argument或者result进行解码。例如，调用makeCall操作，用户需要设置操作码值local:10，并且对MakeCallArgument类型进行编码。这样，MakeCallArgument被编码并且作为一个请求消息（或者称为ROSE调用）。接收端收到这个消息后，要求返回结果消息MakeCallResult类型或者是universalFailure模块中定义的一个错误。

下面介绍对MakeCallArgument进行编码：

```
MakeCallArgument ::= SEQUENCE
{
 callingDevice DeviceID,
 calledDirectoryNumber CalledDeviceID,
 deviceProfile DeviceProfile OPTIONAL,
 accountCode [0] IMPLICIT AccountInfo OPTIONAL,
 authCode [1] IMPLICIT AuthCode OPTIONAL,
 correlatorData [2] IMPLICIT CorrelatorData OPTIONAL,
 extensions CSTACommonArguments OPTIONAL
}
```

对应的C语言描述的结构：

```
typedef struct MakeCallArgument /* SEQUENCE */
{
    struct DeviceID* callingDevice; /* DeviceID */
    struct CalledDeviceID* calledDirectoryNumber; /*CalledDeviceID */
```

```
    struct DeviceProfile* deviceProfile; /* DeviceProfile OPTIONAL */
    AccountInfo accountCode; /* [0] IMPLICIT AccountInfo OPTIONAL */
    AuthCode authCode; /* [1] IMPLICIT AuthCode OPTIONAL */
    CorrelatorData correlatorData; /* [2] IMPLICIT CorrelatorData OPTIONAL
                                    */
    Struct CSTACommonArguments* extensions;
} MakeCallArgument;
```

对 MakeCallArgument 类型进行编码时的函数是：AsnLen BEncMakeCallArgument(BUF_TYPE b, MakeCallArgument *v)，其中参数 b 为存放编码结果的缓存，v 为指向 MakeCallArgument 结构实例的指针，该函数的作用就是将 MakeCallArgument 结构实例进行编码，编码结果放入缓存 b 中。BencMakeCallArgument 函数按顺序调用了下面几个函数：

- AsnLen BEncMakeCallArgumentContent (BUF_TYPE b, MakeCallArgument *v)；/*对 MakeCallArgument 内容进行编码,编码结果放入缓存 BUF_TYPE b 中,并返回编码长度*/
- BEncConsLen( b, len)；/*对内容长度进行编码*/
- BEncTagl (b, UNIV, CONS, SEQ_TAG_CODE)；/*对类型进行编码*/

2) 下面介绍如何封装成 ROSE 包：

一旦 argument 被实例化并且被编码，就必须得加 ROSE 头。ROSE 头发送一个调用消息需要包含 4 个部分：

a) Invoke ID：这是一个任意值的标识，作用相当于“句柄”，用来在消息传递过程中匹配请求和应答。

b) Linked ID：这是另外一个 Invoke ID，在当前操作中有一个子操作被初始化的时候用到 Linked ID。Linked ID 是父操作的 Invoke ID。

c) 操作码：定义了对接收端的操作。

d) 消息数据：开放类型。CSTA 编码的数据存放在这个类型中。例如 makeCall 操作对应 MakeCallArgument 类型。

经过以上两个步骤，就可以得到 Make Call 服务的消息编码结果如表 4-5 所示。左边部分是 ROSE 封装的消息结构，右边是对应的编码结果。

表4-5 Make Call 调用消息的编码

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ROSE_INVOKE, L=16 | A1 | 16 | | | | | |
| InvokeID(INTEGER, L=01, V=01) | 02 | 01 | 01 | | | | |
| OPERATION(INTEGER, L=01, V=0A) | 02 | 01 | 0A | | | | |
| SEQUENCE, L=0E | 30 | 0E | | | | | |
| DeviceID(L=04, "4711") | 80 | 04 | 34 | 37 | 31 | 31 | |
| CalledDeviceID (APPLICATION 2, L=06) | 62 | 06 | | | | | |
| DeviceID(L=04, "1860") | 80 | 04 | 31 | 38 | 36 | 30 | |

表4-5的第四行到第七行对应下面的ASN.1语句：

```
MakeCallArgument ::= SEQUENCE
{callingDevice DeviceID,
 calledDirectoryNumber CalledDeviceID,
}
```

其中第五行对应主叫设备标识的编码，‘4711’ 为主叫设备的设备号，字节序列中，右边四个字节为设备号的值，左数第二字节表示设备值的长度，左边第一个字节即值为80，表示数值类型为整型。

六七行为被叫设备的编码。第四行是对 SEQUENCE 进行编码，‘30’表示 SEQUENCE 类型。第三行是对服务操作码进行编码，第二行对 Invoke ID 进行编码，第一行封装ROSE头，标识所用ROSE服务。

- **解码过程**

解码是编码的逆过程，下面简单介绍对CSTA消息进行解码的步骤：

1) 从一个输入流读取要解码的消息；建立一个消息缓存，用来存放读入的要解码的消息；解析ROSE头。

2) 解析完ROSE头，继续解析消息头。首先解析出 Invoke ID 来找出该应答对应的请求。Invoke ID 部分的值，是一个随机唯一的数字，是在编码过程中设置的值。再解析操作码的值，对于 MakeCall 服务为“local:10”。对应于调用的请求，给出结果/错误的应答。

3) 再建立一个缓存，存放尚未解析消息(数据部分)。

4) 根据第2)步解析出来的操作码值，可得到CSTA服务类型，选择合适的解

码函数，进行解码，从而可以获得参数的值。

### 4.3.2 应用程序开发接口

CT Connect 是Dialogic 公司CTI 中间件，采用Client/Server 的结构，Server进行协议的转换，Client 向应用提供接口。CTConnect 是目前事实的标准。本论文开发接口参考CT Connect 接口。这样基于CT Connect 上开发的电话应用程序也可直接移植到本CTI 平台上。

本中间件提供的接口主要包括监视管理通信通道与操作控制电话功能。下面将列举出本中间件提供的接口：

- **监视管理通信通道接口**

为了接收一个设备的事件信息，需要监听被分配给这个设备的信道。事件信息包括每当信道上发生重要事件时设备当前状态的所有细节。如果设备的当前状态已知，则任意时刻电话的可用特征都能够被预测。例如，如果某设备上有活动呼叫，这时就可以为使用者提供传递这个呼叫的选项。

状态信息包含下列信息的组合：

1) 当前状态；

2) 大多数的最近事件；

3) 其他参与呼叫用户的身份；

4) 网络信息，例如，拨号标识服务(DNIS)或自动号码标识(ANI)。

这些信息可以用来建立和维护一系列呼叫的环境。例如，当使用者在活动呼叫和保持等待状态的呼叫之间进行切换时，或传递一个保持等待状态的呼叫时，可以跟踪与某台被监听的设备相关的一些呼叫。同时，也可以在监听信道上接收事件信息。一条监听信道是应用程序可以创建的用来监听多路设备的一条单独的逻辑信道，但不能用来控制的这些设备。通过分配一条监听信道，应用程序可以在这条信道上接收多个设备的呼叫数据、呼叫状态和参与呼叫用户的信息。应用程序可使用这些信息，例如，用来为特定的代理组记录统计信息。

下面介绍一组应用程序接口，可以用来控制使用者的应用程序和一个特定的电话设备之间的通信信道。每个接口的名称及功能如下：

1) ctcAssign 给一个设备分配一条通信信道，并且这条信道被唯一地标识

到应用程序中。

2) ctcAddMonitor 在监听信道上添加一个需要监听的设备。

3) ctcRemoveMonitor 停止对监听信道上的某个设备的监听。

4) ctcDeassign 从相关设备上取消被分配给它的一条信道，并且释放与这条信道相关的资源。

5) ctcGetChannelInformation 返回所分配的通信信道和相应的设备的信息。

6) ctcSnapshot 返回在设备上的或一个队列中的呼叫数目，并且查询这些呼叫的状态。

7) ctcSetAgentStatus 为一个 ACD 代理设置状态(登录或者退出)和工作模式(例如，“准备接收呼叫”)。

8) ctcSetCallForward 为一个设备设置允许转发，这样可以将输入呼叫转发到其他设备。

9) ctcSetDoNotDisturb 把一个设备设置为不被干扰(Do-Not-Disturb)，这样输入呼叫将不能在这个设备上振铃。

10) ctcSetMessageWaiting 将消息等待指示器设置为打开或者关闭。

11) ctcSetMonitor 将所指定设备的监听状态设置为打开或者关闭。该接口函数和 ctcGetEvent 接口函数一起使用可以接收与一个设备相关的呼叫的状态信息。

12) ctcSetRoutingEnable 允许或禁止所指定的路由节点的路由。当路由被允许时，交换机把指定路由节点的输入呼叫的路由请求传送给 CTC。应用程序可以通过使用 ctcGetRouteQuery 接口函数接收路由请求，并且可以使用 ctcRespondToRouteQuery 接口函数为输入呼叫指定一个新的目的地。

13) ctcSetRoutingEnable 当一个呼叫到达指定的路由节点时，显示交换机是否可以向 CTC 传送路由请求。

14) ctcGetAgentStatus 返回一个代理的当前状态信息。

15) ctcGetCallForward 返回呼叫转发的当前状态信息。

16) ctcGetDoNotDisturb 返回不被干扰(Do-Not-Disturb)状态的当前信息。

17) ctcGetMessageMonitor 返回消息等待指示器的状态。

18) ctcGetMonitor 返回所指定设备的当前监听状态信息。

19) ctcErrMsg 返回文本中一个条件值的细节信息。

20) ctcGetEvent 返回与所指定的设备的电话呼叫有关的信息：呼叫状态，例如初始化或者活动呼叫事件，例如应答或传递呼叫引用(呼叫的标识符)与这个电话呼叫有关的其他部分，以及网络信息，例如 ANI 或者 DNIS。

21) ctcAssociateData 把一个呼叫与相关数据联系起来(例如，客户的引用信息)。

下面以 ctcAddMonitor 为例说明该接口的作用以及该接口所传递的参数。该接口的 C 语言格式：unsigned int ctcAddMonitor(ctcChanId monitorChannel,

ctcAssignData *assignData)

ctcAddMonitor 接口函数将一个设备与一条监听信道关联起来，并将这个设备设置为监听打开状态，这样在监听信道上就可以返回事件信息。一条监听信道是一个用来监听各路设备的单独的逻辑信道，这些设备包括电话、路由节点等。ctcAddMonitor 函数的参数说明如下：

- monitorChanel

数据类型：ctcChanId

访问方式：只读方式(read only)

传递机制：以数值方式(by value)

这个参数是一个包含信道标识符(channel ID)数值的 ctcChanId 数据类型，这个信道标识符数值是由 ctcAssign 接口函数为监听信道返回的。 使用这个参数来确定将被用来监听设备的监听信道。

- assignData

数据类型：ctcAssignData

访问方式：只读方式(read only))

传递机制：以引用方式(by reference)

这个参数包含一个固定格式结构的地址，可以将 ctcAssignData 数据类型存放在这个地址所指的内存中。

- **操作控制电话接口**

下面一组接口可以用来控制电话，每个接口名称及功能如下：

1) ctcMakeCall 从被指定了信道的设备上发出一个电话呼叫。
2) ctcAnswerCall 在具有免提功能的电话上应答输入呼叫。
3) ctcPickupCall 接收一个从其他分机打来的呼叫。
4) ctcHangupCall 清除指定设备上的活动呼叫。
5) ctcHoldCall 设置当前呼叫为磋商保持状态。
6) ctcConsultationCall 在指定设备上给第三个将要接收当前呼叫的用户发出一个呼叫，或者包括在一个会议呼叫中的所有参与呼叫用户。
7) ctcTransferCall 完成一个传递呼叫，并且切断与指定设备的连接。
8) ctcSingleStepCall 发出一个呼叫，并且正在没有让主叫用户保持等待的情况下传递这个呼叫(无监督传递)。
9) ctcCancelCall 切断一个磋商呼叫。
10) ctcRetrieveHeld 重新获得一个在磋商保持状念的呼叫。
11) ctcReconnecHeld 切断一个磋商呼叫，并获得保持等待状态的呼叫。
12) ctcSwapWithHeld 交换当前活动呼叫与磋商保持状态呼叫。
13) ctcDeflectCall 将一个拔入指定设备的呼叫转入另一分机。
14) ctcRespondToInactiveCall 通知一个忙的目的地设备有一个呼叫拨入，以便于当目的地设备完成。当前呼叫后，并且呼叫排在等待队列的第一位时，这个呼叫可以自动被连接。这就是所谓的自动等待功能。
15) ctcGetRouteQuery 给呼叫中心应用程序提供一个呼叫，以便于呼叫中心应用程序可以决定这个呼叫需要路由到哪个设备。
16) ctcRespondToRouteQuery 路由输入呼叫到一个由应用程序选择的目的地。
17) ctcMakePredictiveCall 允许在一个交换机上的虚拟参习用户代表一个用户来初始化呼叫。只有当被呼叫设备应答时(或者，例如，电话铃响了预先规定的次数)，这个呼叫才可以接通到这个使用者。注意这种情况可能需要额外的音频检测设备。
18) ctcSendDTMF 发送DTMF数字信号来模拟使用者在按键式电话上的按键。

下面以ctcMakeCall为例说明该接口的作用以及所需传递的参数。该接口的

C语言格式：unsigned int ctcMakeCall(ctcChanId channel,
ctcDeviceString calledNumber,
ctcApplString applicationData,
unsigned int *callRefId)

ctcMakeCall 从信道被分配给的设备上给任何一个交换机认为是有效的号码发出一个呼叫。可以使用 calledNumber 参数识别将要呼叫的设备。这个参数指定了这个设备的可以拨叫的号码。CtcMakeCall 所需的各个参数说明如下：

- channel

数据类型：ctcChanId

访问方式：只读方式(read only)

传递机制：以数值方式(by value)

这个参数是一个包含信道标识符(Channel ID)数值的 ctcChanId 数据类型，这个信道标识符数值是由 ctcAssign 接口函数为使用中的设备而返回的。

- calledNumber

数据类型：ctcDeviveString

访问方式：只读方式(read only)

传递机制：以引用方式(by reference)

这个参数是一个包含所呼叫设备号码的字符串的地址。这个 ASCII 码字符串可以包括数字 0 至 9 与字符*和#的任意组合。

- applicationData

数据类型：ctcApplString

访问方式：只读方式(read only)

传递机制：以引用方式(by reference)

可以使用这个参数把数据(例如，顾客引用信息或者帐户详细内容)与相应呼叫关联起来。这个参数是一个以 NUL 字符结束的字符串的地址。

- CallRefId

数据类型：整型数(unsigned)

访问类型：只读方式(read only)

传递机制：以引用方式(by reference)

这个参数是一个接收该呼叫的呼叫引用标识符的地址，为32位比特的整型数。

## 4.4 企业级统一商务通讯平台基于本CTI中间件实现的功能

通过基于CSTA CTI中间件的实现，为企业级统一商务通讯平台的业务处理系统提供了灵活、简单的开发接口，上层业务可以轻松实现计算机与电话的交互功能体现如下：

1) 自动号码识别(Automatic Number Identification)可以显示主叫的电话号码。在应答呼叫之前，坐席就可以从数据库中调出该呼叫的信息并且显示出来。
2) 被叫号码识别(Dialed Number Identification Service) 识别被叫号码。比如，客户使用不同的电话号码来发出订单或者报告错误。他们的电话可以路由到合适的设备，根据被叫号码在坐席端显示订单入口。
3) 呼叫转移（Call Transfer ）客户的一个呼叫被转移时，应答该呼叫的坐席的信息以及客户的数据一起被转移。比如，一个客户发出一个订单，然后想被转移到资金支付部门，那么客户的订单信息也可以被转移。
4) 自动拨号 （Auto Dialing）自动拨号又包括屏幕拨号、记录拨号和预先拨号等。也就是通过计算机的设备来存储，处理用户的数据，并由计算机按用户的设定要求自动拨号。计算机拨号也是CTI中用的最广的一种业务。
5) 智能路由（Call routing）通过相应的呼叫数据来决定一条最合适的通信路由。由计算机系统的应用软件分配路由，然后告诉交换机，将呼叫送至用户的电话，与此同时，计算机系统将在用户的计算机屏幕上显示呼叫的相关信息。

# 第五章 总结与展望

本文围绕CTI中间件的开发对CSTA协议进行了研究，通过SNACC，将ASN.1文件编译成C语言的文件，根据SNACC生成的编解码函数，对不同的CSTA服务消息进行编解码，实现了CSTA协议，并为上层应用提供监视管理通信通道和操作控制电话功能的接口。基于这些接口，可以实现基本的CTI功能,包括自动号码识别、被叫号码识别、呼叫转移、自动拨号、智能路由等，从而实现更多复杂的CTI应用。

本文所实现的CTI中间件为支持CSTA协议的交换机与计算机之间搭起一座桥梁，实现了交换功能与计算功能的融合。该中间件目前已经使用在企业级统一商务通讯平台中，将呼叫中心（Call Center）融合到了该通讯平台中。

CTI是目前和未来市场上最活跃的技术之一。CTI技术在国内已开始普及，大量的企业和从业人员开始涉及该领域。由于本中间件以CSTA协议作为交换机和计算机之间的标准，所以对于基于支持CSTA的交换设备的应用程序，应用开发者都可以根据不同的需求使用本CTI中间件进行开发，快速简便地构筑丰富多彩的CTI应用。

本项目所开发的企业级统一商务通讯平台所实现的CTI功能除了4.4节所介绍的功能外，还有交互式语音应答系统(Interactive Voice Response)、语音邮件（Voice Mail）等，这些是通过计算机侧的语音板卡来实现的，而不是本文目前开发的CTI中间件所提供接口实现的功能。同时，CSTA第三版增强了功能，第四版也增加诸如VoIP的功能。因此，为不断提升本CTI中间件的应用价值，论文进一步的工作将围绕两点：

1) 接下来将要继续进行的工作:继续研究CSTA的特殊资源部分（语音单元），为本CTI中间件提供语音处理的接口。
2) 未来不久要进行的工作：CSTA的高版本的实现。

# 参考文献

[1]杨涛,郑晓霞,刘锦德.基于CSTA规范的CTI中间件的研究与实现.计算机应用,Vol.21,No.10,2001:14-16.

[2]宋俊德,段云峰. CTI独步电信、PC之间.http://www.ctiforum.com/forum/forum0234.htm,2000.

[3]赛迪网：中间件技术专题.什么是中间件?.http://tech.ccidnet.com/pub/series/s64.html,2005.

[4]CTI论坛.什么是CTI中间件？.http://www.ctiforum.com/technology/ctmw/ctmw01_1201.htm,2001.

[5]CTI论坛.CTI中间件产品分类.http://www.ctiforum.com/forum/2003/12/forum03_1208.htm,2003.

[6]ECMA-217.Services for Computer Supported Telecommunications Applications (CSTA) Phase II, December 1994.

[7]ECMA-218.Protocol for Computer Supported Telecommunications Applications (CSTA) Phase II, December 1994.

[8]Intel.Intel® NetMerge™ Call Processing Software Introduction, September, 2002.

[9]Intel.Intel® NetMerge™Call Processing Software C Programming Guide, September, 2002.

[10]Thomas A. Anschutz, Lucent Technologies.A Historical Perspective of CSTA.IEEE Communications Magazine, April 1996:30-35.

[11]Dialogic Corporation. CT Connect Implementation and Mapping Guide for CSTA Phase II, April,2000.

[12]Objective Systems, Inc.CSTA C++ BER Encode/Decode API User's Guide, February,2003.

[13]Intel Telecom White Paper. Telephony Fundamentals:An Introduction to Basic Telephony Concepts.

[14]Intel Application Note. Computer Telephony Integration (CTI) in a SALT Environment.

[15]Siemens AG, Private Communication Group . CSTA Interface Manual-CSTA2 Link for Hicom 150 E Office 1. 0 and Office 2. 0 , 1999.

[16]ECMA Technical Report TR/68. Scenarios for Computer Supported Telecommunications Applications (CSTA) Phase II, December 1994.

[17]Computer Supported Telecommunications Applications (CSTA) Frequently asked Questions. http://www.ecma-international.org/activities/Communications/TG11/faq.htm.

[18]计算机通信网基础. http://ww.mhkj.com/JIAOCAI/homepage/index.htm.

[19]李爱振. CTI技术与呼叫中心. 北京：电子工业出版社，2002.

[20]戚英豪, 何为民, 黄佩伟. 基于 CSTA 协议的 CTI 应用. 通信技术，No.7, 2003:66－68.

[21](美) Kenneth D. Reed 著；孙坦，张学锋，杨琳等译. 协议分析. 北京：电子工业出版社，2002.

[22]张瑞. 透视CT中间件. http://www.ctiforum.com/factory/f01_02/www.qnuse.com/qnuse02_0901.htm, 2002.

[23]毛家华. 中间件在电信领域的两个关键应用. http://tech.ccidnet.com/pub/article/c1092_a212049_p1.html, 2005.

[24]《电脑变成技巧与维护》杂志社. Visual C/C++编程精选集锦——网络与通信分册. 北京：科学出版社，2003.

[25]余鹰, 范辉, 见春蕾. 基于BER的ASN.1编解码原理与设计实现. 计算机工程与科学, Vol.127, No.11, 2005:83-86.

[26]邓秀兰, 饶运涛. ASN.1的编解码规则与应用层网络协议开发. 微计算机信息, Vol.20, No.4, 2004:99-101.

[27]彭海, 吴安敏. CSTA协议分析及实现技术. 南京邮电学院学报(自然科学版), Vol.21, No. 3, 2001:89-94.

[28]Rob Wallters 著，宋俊德，段云峰等译. 计算机电话集成技术. 北京：人民邮电出版社, 2000.

[29]张云勇，张智江，刘锦德. 中间件技术原理与应用. 北京：清华大学出版社，2004.

[30]李棠之.通信网络技术.北京：科学技术文献出版社,2000.

[31](日)丸山修孝著，王庆译.通信协议技术.北京：科学出版社，2004.

[32]21 世纪计算机语音通信开发技术丛书编委会.计算机语音通信核心技术内幕——CT Connect：C程序设计指南.北京：北京希望电子出版社,2000.

[33]张曙光，李茂长.电话通信网与交换技术.北京：国防工业出版社，2002.

# 致谢

在中山大学读研究生的两年时间里，我的导师陈有青副教授给予了很多关怀与指导。陈老师的言传身教，使得我不但在理论的学习上更进一步，实践能力有了很大的提高，同时，也学会了很多做人的道理。感谢陈老师，也祝陈老师身体健康，万事顺意。

另外要感谢广州纬视公司的张黎明先生。他从百忙之中对我的毕业论文进行了很多无微不至的指导。

感谢软件学院的周小舟师弟，他一起参与了我的毕业设计，他的很多想法都给了我很多的启示。同时也要感谢我的各位同门：钱漫、钟宝静、姚辉武、熊胜华、吴毅敏、沈进平，在学习上和生活中，他们给予了我很多关心与帮助。

最后，还要感谢我的父母以及关心我的各位朋友对我的支持。

# 原创性声明

本人郑重声明，所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。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。本文完全意识到本声明的法律结果由本人承担。

学位论文作者签名：顾敏

日期：2005年6月6日