摘要

自 20 世纪 90 年代以来，以因特网（Internet）为主的 IP 业务以极其迅猛的速度发展着，数据业务已成为主导业务。Internet 的传送内容，提供的服务也已经进入电话网的领域，传统的电信网和计算机壁垒分明的界限已被打破。通信网正以日新月异的速度发展着，业务结构不断发生变化。NGN 就是在这种背景下孕育而生的。它以 IP 作为其网络层的核心技术，重用 Internet 已有网络技术，并借鉴传统通信网成熟的网络管理技术，将传统通信网和 Internet 进行互补，两者结合成可管理的 IP 网络。软交换是 NGN 的核心技术，此时，传统的网关被分成媒体网关（MGW）和媒体网关控制器（MGC）。这样，将控制智能集中到网络中来。媒体网关控制协议 H.248 协议因此于 2000 年诞生雏形，并不断发展，它是媒体网关和媒体网关控制器之间实时通信所依据的标准，媒体网关控制器通过该协议对媒体网关进行业务的控制。近两年来，国内各大研发中心都在积极地投身于 H.248 相关的研发工作，但是大多成果不够明显。

针对以上状况，本文在全面细致的阐述 H.248 的重要概念，诸如终结点、关联、命令、描述符与包、事务、消息等基础之上，就协议应用的关键点进行深入的分析和研究，并以作者重点参与研发的中兴 3GCN 项目为背景，介绍在下一代网络（CDMA2000 核心网）中实现 H.248 的具体设计方法。

在文本中，作者具体阐述了项目中实现时的软件模块架构，深入介绍了协议模块进程的设计，其中包含相应重要的数据结构、算法以及处理流程，通过介绍，读者可以清楚地理解协议的具体实现方法。在本文的第五章，将以作者在真实网络测试环境中获取的测试结果为依据，证明本套设计方案的最终成功。这一研究成果填补了同期国内 NGN 相关空白，它与其它研发的技术相综合，使中兴的软交换产品处于国内乃至国际领先的地位。于此同时，该研发的成功必将推动全球范围内 NGN 的研发，世界范围内必将进一步掀起对 NGN 应用的关注热潮。

关键词：H.248，媒体网关，媒体网关控制器，NGN，软交换

ABSTRACT

From 1990s, IP technology, especially Internet, has been developed rapidly and data service has become the dominant one in networks. Also, the transmission and service in Internet has stepped into that of in telephony networks, too. The barrier between traditional telecommunication networks and computer networks has become not so distinct. Communication networks have been developed very rapidly and contents of service have been varying continuously. NGN was put forward in this background. IP was utilized as its key technology in its network layer and some other technologies in Internet were applied too. In addition, it took the ripe network management technology in traditional telecommunication networks, thus integrating the advantages of traditional telecommunication networks and Internet together and making a kind of manageable networks. Soft-switching is the key technology in NGN. Here, traditional gateways are separated into Media Gateways（MGW）and Media Gateway Controllers（MGC）. Thus, the controlling is integrated into the networks. Media Gateway Controlling Protocol H.248 was born in 2000 under this background and developed step by step. It is the standard for the communications between MGC and MGW and MGC controls the services on MGW according to this protocol. Recent two years, several big domestic R & D centers were busy with the relevant implementing tasks with H.248. However, the outcomes were not so satisfactory.

Towards the above status, based on the full and detailed introduction of the important concepts of H.248, such as Termination, Context, Command, Descriptor and Package, Transaction and Message, this thesis discusses the key appliance points of the protocol deeply. Also, we introduce our specific design method of implementation of H.248 in next generation networks (CDMA2000 core networks) according to the “3GCN” project in ZTE.

In the thesis, we discuss the construction of software modules in details in the project and analyze the design of particular processes in H.248 module, including relevant important data structures, algorithms and processing procedures. From these, readers can understand the details about the design explicitly. In Chapter 5, based on the testing outcome the author attained in the real network circumstances, we can demonstrate the final success of the scheme. This accomplishment has filled the blank area in this field at home. In addition, with the combination of the other techniques, this success has made ZTE’s soft-switching products the leading ones in China and even all around the world. It will definitely improve the world’s research and design of NGN, and a new tide of concerning NGN’s appliance will emerge throughout the world.

Keywords: H.248, Media Gateway, Media Gateway Controller, NGN, soft-switching

图、表清单

注释表

AG	Access Gateway	接入网关
API	Application Programming Interface	应用程序接口
CA	Call Agent	呼叫代理
CC	Call Control	呼叫控制
CDMA	Code Division Multiple Access	码分多址
DB	Data Base	数据库
HLRe	Home Location Register Emulation	归属位置寄存器仿真
ID	Identifier	标识符
IP	Internet Protocol	Internet 协议
ISUP	ISDN User Part	ISDN 用户部分
M3UA	SS7 Message Transfer Part 3 (MTP3) User Adaptation layer	SS7 消息传递部分第三级(MTP-3)用户适配层
MEGACO	Media Gateway Controlling	媒体网关控制
MG/MGW	Media Gateway	媒体网关
MGC	Media Gateway Controller	媒体网关控制器
MID	Message Identifier	消息标识符
MRFP	Multi-Media Resouce Function Processor	多媒体资源功能处理器
MSCe	Mobile Switching Center Emulation	移动交换中心仿真
NGN	Next Generation Networks	下一代网络
OMC	Operating and Managing Control	操作维护模块
PSTN	Public Switched Telephone Network	公共交换电话网
RAN	Radio Access Network	无线接入网
RTP	Real-time Transport Protocol	实时传输协议
SCF	Service Controlling Function	业务控制功能
SCTP	Stream Control Transmission Protocol	流控制传输协议
SDF	Service Data Function	业务数据功能
SCPe	Service Controlling Point Emulation	业务控制点仿真
SIP	Session Initiation Protocol	会话初始化协议
TDM	Time Division Multiplex and Multiplexer	时分复用和复用器

承诺书

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

本人授权南京航空航天大学可以有权保留送交论文的复印件，允许论文被查阅和借阅,可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文。

(保密的学位论文在解密后适用本承诺书)

作者签名：_____

日　　期：_____

第一章　绪论

1.1　国内外通信网络的发展和下一代网络（NGN）概念的引入

自 1876 年贝尔发明电话以来，通信网始终是电话业务占主导地位，并以电话交换为主导技术。进入 20 世纪 90 年代，通信网业务结构发生了极大的变化。全球传统电话业务的增长率为 5%～10%，而数据业务的年增长高达 25%～40%，呈指数式增长，特别是自 20 世纪 90 年代中期以来，随着 WWW 业务的问世，以因特网（Internet）为主的 IP 业务每 6～12 个月翻一番，比著名的摩尔定律指出的 CPU 性能价格比每 18 个月翻一番的速度还要快 1.5～3 倍。至本世纪初，出现了电信业 100 多年来的一个历史转折点：全球的数据业务量已经超过电话业务量，从网络的承载流量来看，数据业务已成为主导业务。在此背景下，各国大力投资扩建 IP 网络，增加其带宽，有效地提高了 Internet 传送质量，反过来又促进了 Internet 业务的发展和用户数的增长。我国通信网的业务发展也遵循同样的规律。据统计，1999～2003 年 Internet 和电话业务量的年平均增长分别为 362%和 41.89%。与此同时，Internet 的传送内容也从 E-mail 的低速率普通文本，到包括语音和图像在内的中速率数据，直至网络游戏和影视等含活动图像的高速率信号，其提供的服务已经进入电话网的领域，传统的电信网和计算机壁垒分明的界限已被打破。近 3 年来，我国 IP 电话量已经超过全部电话量的 1/4，特别是对固定电话业务已形成很大的冲击。

由此可知，由于 Internet 的高速发展和应用的成功，不仅使数据业务量超过了电话业务量，而且使 IP 技术进入了包括语音和视频在内的传统通信网领域。因此，网络业务数据化发展趋势对电信网技术的发展必将带来深刻的影响。

现在可以肯定的是，未来的通信网络必将采用 IP 作为其网络层的核心技术，将重用 Internet 已有网络技术。但是业界也清醒地认识到，面向数据应用的 Internet 有其固有的缺陷，特别是缺乏传统通信网固有的高服务质量、高性能、完备的网络管理和网络智能等优异性能，这对于一个公众运营网络的可持续发展是不利的。在此背景下，下一代网络必将借鉴通信网成熟的网络管理技术，将通信网和 Internet 进行互补，两者结合成应该可管理的 IP 网络。然后再综合通信网络的智能网技术和计算机网络的分布计算技术，赋予该 IP 网络灵活提供增值业务的能力，最后构成一个可管理、可运营、可盈利的 IP 通信网。这样的

新一代网络称为 NGN。

虽然不同背景和专业的人士对于 NGN 尚有不同的理解和诠释，但是基本认同 NGN 是泛指以 IP 为核心、支持多媒体业务、智能化的、融合的、可管理可运营的全业务网络。NGN 是电信发展的重大战略性趋势，其实现应该是一个演进过程。NGN 的基本技术特征可以归纳为以下几个方面：

首先，网络层趋于采用统一的 IP 协议，要以 IP 协议为基础实现异质网络和异质终端的互联通信。

其次，链路层趋于采用电信级分组节点，采用边缘路由器和核心路由器相结合的网络结构。

第三，传送层趋于采用光传送技术，在物理层提供巨大而廉价的带宽，支持任何业务和信号的透明传送。

第四，接入层采用多元化的宽带接入技术。

第五，呼叫控制和业务层独立于承载层。为了确保 IP 通信网上的实时通信业务具有和传统通信网同样的业务性能，NGN 借鉴通信网成熟的交换技术，在 IP 网络层上设立一层交换控制层，该控制层和底层采用的具体承载技术无关，这就是所谓的软交换技术。

第六，管理平面提供运营网络各种所需的管理功能。

由此看来，NGN 的网络体系将是一个全新的结构。

1.2 本文的研究重点及意义

软交换是下一代网络的核心技术。在传统的网络中，网关（Gateway）不但要执行媒体格式变换，还要进行信令转换。此外网关还要控制内部资源，为每个呼叫建立网关内部的话音通路。网关的功能非常复杂。上述集所有功能于一体的集成网关结构对于 IP 通信系统的大规模部署具有相当的制约。鉴于上述考虑，人们考虑是否可以将传统的网关进行分解。这就引入了网关分离的概念。

在下一代网络中，网关被分解成媒体网关（MGW，Media Gateway）和媒体网关控制器（MGC，Media Gateway Controller）。媒体网关负责媒体格式的变换以及 PSTN 和 IP 两侧通路的连接，而媒体网关控制器负责根据收到的信令控制媒体网关业务的建立和释放。

上述分离网关结构的重要特点是将控制智能集中到网络中来，即少量的

MGC 中，其思路和电信交换网络类似。此时网关相当于终端设备，数量大而功能简单；MGC 相当于交换机，数量少而功能复杂，一个 MGC 可以控制多个网关。MGC 以信令的形式控制 MGW，而 MGW 负责媒体的变换和传送，这样控制和承载充分的分离，实现了软交换的核心。

人们开始探索，利用怎样的方法可以完成媒体网关和媒体网关控制器之间的实时通信，以及怎样利用媒体网关控制器与媒体网关的接口对后者进行业务的控制。2000 年 ITU-T 第 16 工作组提出了媒体网关控制协议 H．248，随着该协议的不断发展，以上问题将逐步得到解决。迄今为止，该协议还在不断的完善过程之中。

除了 H. 248 核心协议以外，人们还为软交换组织和设计了其他一些重要的标准或协议，诸如 SIP，H. 323 等等。软交换和 NGN 的各种其他标准也在不断被研究和提出，全球一些大型的科研基地正在积极地投身该领域的研究工作中。

由于 NGN 还处在应用研发的初期阶段，很多的科研中心都是处于自身的摸索开发阶段，根据协议标准研发的相应产品也存在许多问题，但是该项工作的应用价值和发展前景却是无限巨大的。

笔者非常幸运的在研究生期间进入中国顶尖的通信研发基地中兴通讯南京研发中心，参与下一代核心网协议 H.248 的研究和开发任务。由于该领域的全新性特点，几乎完全缺乏前人的研究经验，该项工作充满了挑战性。在整个研发过程中，作者的能力得到了充分的锻炼，也汲取了很多宝贵的经验。对于 H.248 协议的具体研究和在下一代核心网中的应用与实现将是本文的重点。针对国内 NGN 相关科研极不成熟的现状，该项工作具有很强的应用价值，这点是毋庸置疑的，事实上，本次研发最终获得了成功，这将对 NGN 的未来发展起到十分积极的作用。

1.3 论文的组织

由前述，在本篇论文中，我们将围绕 H.248 的具体研究和应用有条理的展开。具体安排章节内容安排如图 1.1 所示。

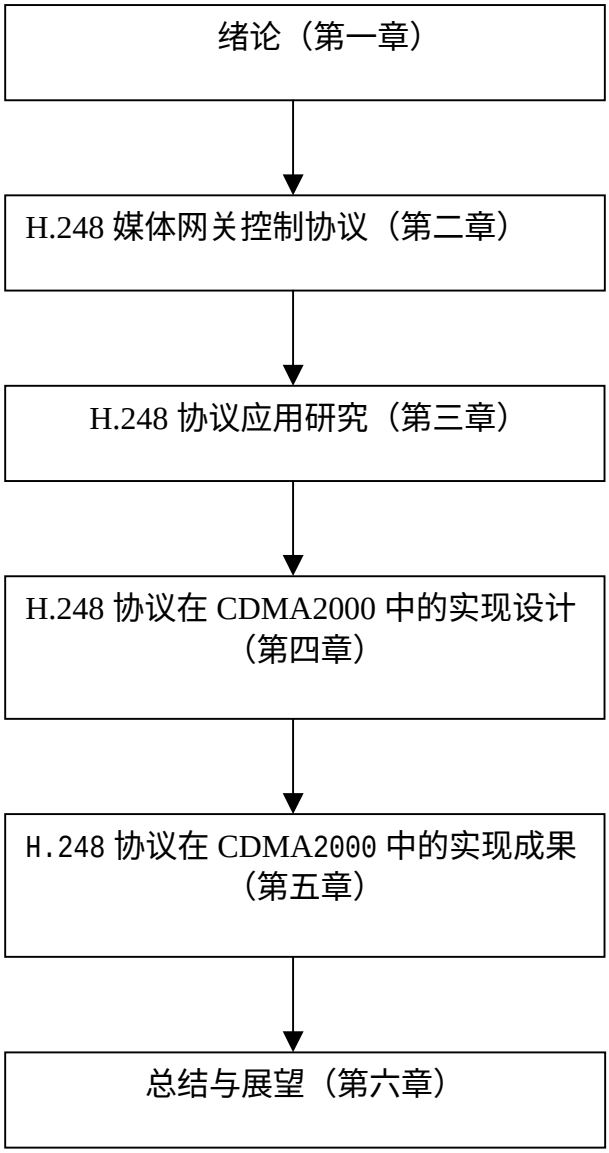

图 1.1　全文组织结构

第一章 本课题的概述和文章结构的安排。

第二章 以笔者对协议的深层次研究为基础，对协议关键的概念及其语法、功能进行细致的介绍。内容上，由浅入深依次引入终结点、关联、命令、包与描述符、事务、消息等非常重要的概念，这些概念构成了协议的主体。

第三章 在第二章介绍的基础上，继续深入拓展协议的具体应用，重点放在：第一，协议应用的具体编解码方式；第二，协议应用时所需注意的网络问题和应采取的相应措施；第三，协议的典型应用实例。

第四章 以作者重点参与的中兴 3GCN 项目的科研为背景，介绍 H.248 协议在下一代核心网中的具体实现方法。为了让读者能深入浅出的理解设计的思想，作者先从软件模块入手介绍系统的框架，之后深入到系统进程内部介绍协议相应模块的研发设计思想，并通过具体的实例说明协议的最终实现方式。

第五章 根据笔者在真实网络设备中实际测试得到的结论，证明本次研发设计的最终成功。

第六章 总结本课题，充分肯定本次研究的应用价值，并展望未来 NGN 的发展。

第二章　H.248 媒体网关控制协议

2.1　H.248 媒体网关控制协议及其基本概念

H.248协议是下一代网络中的接口协议之一，它应用于下一代网络中媒体处理和信令控制分离后所产生的控制接口，标准规定了媒体网关控制设备（媒体网关控制器/软交换设备）和相应的媒体处理设备（网关/媒体服务器/IP 智能终端等）之间，进行通信时的协议要求。协议在网络中的位置如图2.1所示：

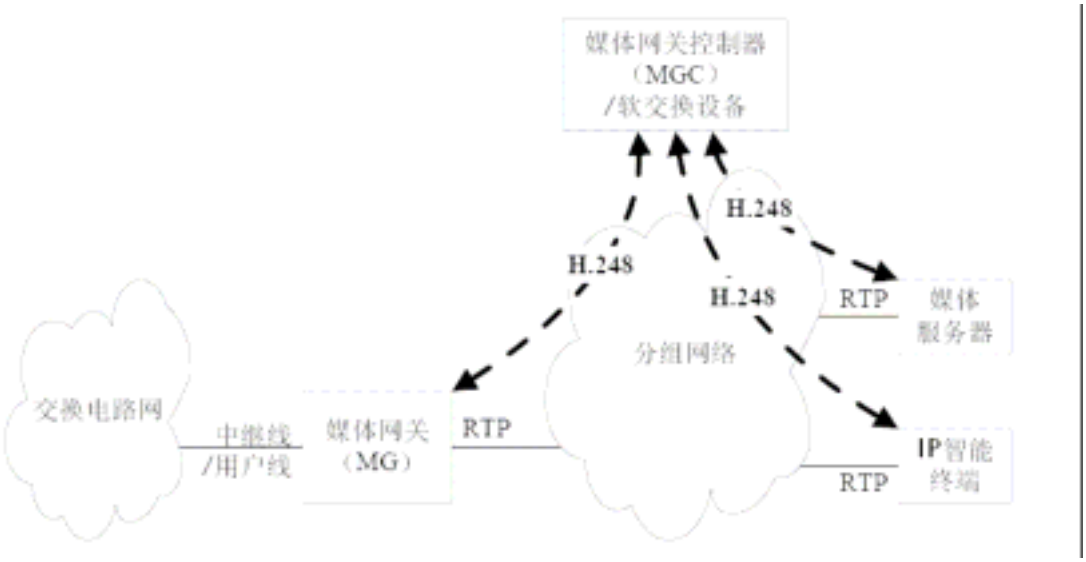

图 2.1　协议在网络中的位置

H.248协议涉及以下的基本概念：

（1）媒体网关（MG或MGW）

将一种网络中的媒体转换成另一种网络所要求的媒体格式的网络设备。一般地，MG 能够完成电路交换网的承载通道和分组网的媒体流之间的转换。MG 可以处理音频、视频或者T.120，也可以具备处理这三者任意组合的能力。MG 能够进行全双工的媒体转换，播放视频/音频消息，实现其它IVR 功能，也可以进行媒体会议。在本文后续部分中，将媒体网关简称为网关。

（2）媒体网关控制器（MGC）

对MG 中与媒体通道连接控制相关的呼叫状态进行控制的网络设备。后续

章节，将其简称为网关控制器。

（3）软交换设备（SoftSwitch）

电路交换网向分组网演进的核心设备，也是下一代电信网络的重要设备之一，它独立于底层承载协议，主要完成呼叫控制、媒体网关接入控制、资源分配、协议处理、路由、认证、计费等主要功能，并可以向用户提供现有电路交换机所能提供的所有业务以及多样化的第三方业务。

（4）中继（Trunk）

两个交换系统间的一个通信通道。例如，T1 或E1 中的一个时系DS0。

（5）流（Stream）

在一个呼叫或者会议中，由一个MG 接收或发送的双向媒体流或者控制流。

（6）事件（Events）

MGC请求MG 检测的事件。可包括传真音、导通检测结果和摘机/挂机等。MG 检测到请求的事件后，缺省地，向MGC 发送一个通知报告。

（7）信号（Signals）

MG 产生的媒体，如信号音（Tone）和录音通知，以及线路信号（如Hookswitch）。复杂的信号可以包含一个简单信号的序列，加上对媒体或线路信号接收和分析，并以此作为信号产生的条件。信号也可以要求准备一些媒体内容来产生以后的信号。

（8）描述符（Descriptor）

协议中的一种语法元素，用来描述一组相互联系的属性。

（9）通配值（Wildcard）

协议语法中定义的特殊符号，有“ CHOOSE ”和“ ALL ”两种。“ ALL ”表示需要使用所有满足条件的取值，“ CHOOSE ”表示需要选择一个满足条件的取值。在没有特殊说明时，通配值往往特指“ ALL ”。

（10）终结点（Termination）

又称“ 终端 ”，是MG 上的逻辑实体，它发起和/或接收媒体和/或控制流。终结点可以用一些属性来描述，如媒体流、modem 和承载能力等属性，这些属性组成了一系列描述符。

（11）关联（Context）

又称“ 上下文 ”，是一些终结点相互联系而形成的结合体，比如一个呼叫中的两个终端相互联系形成的结合体。需要注意的是，关联中有一种特殊的类

型，称为空关联（Null），它包含所有那些与其它终结点没有联系的终结点。例如，接入网关中所有的空闲线路都被看作空关联中的终结点。

（12）命令（Command）

本协议定义的用于对协议连接模型中的逻辑实体(关联和终结点)进行操作和管理的语法结构。命令提供了本协议所支持的最精微层次的控制。

（13）事务（Transaction）

MG 与MGC 之间的一组命令构成的结构。一个事务可以由一个或多个动作（Action）组成，每个动作又由作用范围局限在同一个关联中的一个或多个命令组成。

（14）请求（Request）和响应（Reply）

MGC 和MG 之间进行的各种层次上的交互（如事务交互、动作交互和命令交互等）。每次交互包含一个请求和一个响应。请求由发送方发起；接收方对请求进行处理，并将处理结果包含在响应中返回给发送方。

以上只是先引入这些概念，给读者一个初步的映像，且方便读者在后续章节阅读时碰到这些概念时查找之用。在本章后面的部分中，将对其中重要的概念，即协议的核心部分，进行深入的研究和介绍。通过这些介绍，读者将对这些重要概念有深层次的理解。

2.2 关联和终结点

2.2.1 关联（上下文）

关联（Context）是一些终结点具有相互联系而形成的结合体。当这个结合体中包含两个以上终结点时，关联可以描述拓扑结构（谁能听见/看见谁），及媒体混合和（或）交换的参数。通常，关联与呼叫相对应。

关联具有ContextID（关联标识符），拓扑（Topology），关联优先级（Priority）以及紧急呼叫的标识符（Indicator for Emergency Call）等几个属性。其中拓扑（Topology)（谁能听见/看见谁）用于描述在一个关联内部终结点之间的媒体流方向。而关联优先级（Priority）用于MGC指示MG 处理关联时的先后次序。

一种特殊的关联称为空关联（Null），它包含所有那些与其它终结点没有联系的终结点。空关联中终结点的属性参数也可以被检查或修改，并且也可以检测事件。

一个关联中最多可以有多少个终结点是MG 的一个属性。只提供点到点连接的MG 中的每个关联最多只支持两个终结点，支持多点会议的MG 中的每个关联可以支持三个或三个以上的终结点。

当呼叫建立或释放时，本协议可用于创建和删除终结点之间所形成的联系，即关联，并且可以修改已存在关联的参数。本协议还定义了命令，可将终结点加入关联，从关联中删除终结点以及在关联之间移动终结点。

2.2.2 终结点（终端）

终结点是MG 上的一个逻辑实体，它发起和（或）接收媒体流和（或）控制流。MG在创建终结点时，赋予终结点一个唯一的TerminationID 来标识终结点。终结点用一些特征属性（Property）来描述，这些属性组成了一系列描述符（Descriptor）包含在命令中。

代表物理实体的终结点（称为物理终结点）具有半永久性。例如代表一个TDM 信道的终结点，只要MG 中存在这个物理实体，这个终结点就存在。代表临时性的信息流（例如RTP 流）的终结点（称为临时性终结点），只有当MG 使用这些信息流时，这个终结点才存在。

终结点用TerminationID 进行标识，TerminationID 的分配方式由MG 自主决定。对于物理终结点，其TerminationID 是在MG 中预先规定好的。这些TerminationID 可以具有某种结构。例如，一个TerminationID 可以由一个中继组号及其中的一个中继号组成。

对于TerminationID 可以使用一种通配机制。该通配机制使用两种通配值（Wildcard）：“ ALL ”和“ CHOOSE ”。通配值“ ALL ”用来表示多个终结点，“ CHOOSE ”则用来指示MG 必须自己选择符合条件的终结点，例如MGC 可以通过这种方式指示MG 选择一个中继群中的一条中继电路。

有时，一个命令是针对整个MG 的，而不是其中的某个终结点。在本协议中将其定义成一类特殊的终结点“ 根 ”（Root）。

本协议可用于创建新的终结点；也可以修改已存在终结点的属性值，包括向终结点添加或删除事件或信号。MGC 只可以释放它已经通过某种方式（如Add 命令）占用的终结点及该终结点代表的资源。

2.3 命令

本协议定义了8 个命令用于对协议连接模型中的逻辑实体(关联和终结点)进行操作和管理。命令提供了本协议所支持的最精微层次的控制。例如，通过命令可以向关联增加终结点、修改终结点、从关联中删除终结点以及审计关联或终结点的属性。命令提供了对关联和终结点的属性的完全控制，包括指定要求终结点报告的事件、向终结点加载的信号以及指定关联的拓扑结构（谁能听见/看见谁）。

本协议规定的命令大部分用于MGC 对MG 的控制，通常MGC 作为命令起始者发起，MG 作为命令响应者接收。但是， Notify 和ServiceChange 命令除外。Notify 命令由MG 发送给MGC；而ServiceChange 既可以由MG 发起，也可以由MGC 发起。本协议规定的命令及其功能如表2.1所示。

表 2.1 H.248 命令及功能

Add	使用Add 命令可以向一个关联添加一个终结点，当使用Add 命令向一个关联添加第一个终结点时，同时创建了一个关联。
Modify	使用Modify 命令可以修改一个终结点的属性、事件和信号。
Subtract	使用Subtract 命令可以删除一个终结点与它所在的关联之间的联系，并返回终结点处于关联期间的统计信息。当使用Subtract 命令删除一个关联中最后一个终结点与它所在的关联之间的联系时，同时就删除了这个关联。
Move	使用Move 命令可以单独地将一个终结点从一个关联转移到另一个关联。
AuditValue	使用AuditValue 可以获取终结点属性、事件、信号和统计的当前信息。
Auditcapabilities	使用Auditcapabilities 可以获取终结点的属性、事件和信号的所有可能值的信息。
Notify	MG 使用Notify 命令向MGC 报告MG 中所发生的事件。
ServiceChange	MG 使用ServiceChange 命令向MGC 报告一个或者一组终结点将要退出服务或者刚刚进入服务。MG 也可以使用ServiceChange 命令向MGC 宣布其可用性（即注册），或者向MGC 报告MG 即将开始或已经完成重新启动。 MGC 可以使用ServiceChange 通知MG 对其控制即将由另一个MGC 接替。MGC 还可以使用ServiceChange 命令通知MG 将一个或者一组终结点进入服务或退出服务。

每个命令都要指定命令所操作的终结点的TerminationID。TerminationID 可以取通配值。当一个命令的TerminationID 是通配值时，相当于在该通配值所匹配的所有终结点上重复该命令。

2.4 包与描述符

不同类型的网关上可以实现不同类型的终结点。本协议通过允许终结点具有可选的属性（Property）、事件（Events）、信号（Signals）和统计（Statistics）来实现不同类型的终结点。为了实现MG 和MGC 之间的互操作，本协议将这些可选项组合成包（Packages）。

包的定义由属性、事件、信号和统计组成，这些项以及它们包含的参数（Parameter）分别由一些标识符（Id）进行标识。标识符有特定的有效范围。例如，对每个包而言，属性标识符（PropertyId）、事件标识符（EventId）、信号标识符（SignalId）、统计标识符（StatisticsId）和参数标识符（ParameterId）都有独立的名字空间，同一个标识符可以用于它们中的每一个。而不同的包中的两个属性标识符也可以相同。

包是本协议重要的组成部分，许多应用涉及到包的使用，例如MGC对终结点呼叫过程中摘挂机的检测以及振铃回铃的控制都是通过包来实现的。由于协议所含包的内容十分繁多，限于篇幅，本文不可能尽数描述，如有兴趣，可参见[1]。

终结点可以用一些属性来描述。每个属性由一个PropertyID 标识。大部分属性有缺省值，其缺省值在本协议中或某个包中进行定义。在主协议中没有定义的属性都在包中进行定义。为了方便起见，相互有关的属性被组合成为描述符。

描述符是命令的输入和输出参数。当使用命令对某一个终结点进行控制或操作时，可以通过加入适当的描述符作为命令输入参数来设置可读写的属性值。有些情况下，描述符作为命令的输出参数在应答中返回。通常，如果上述命令中完全省略了某个描述符，对执行该命令的那个终结点，该描述符中的属性保持原值。另一方面，如果一个命令中省略了一个描述符中的某些可读写属性，对执行该命令的那个终结点，这些属性将被重置为它们的缺省值；除非在包中规定了其他处理。

描述符由描述符名称和一些参数项组成，参数可以有取值。许多命令中用到相同的描述符。协议中定义的描述符如表2.2所示。

表 2.2　描述符及其用处

Modem	标识modem 类型和属性
Mux	描述多媒体终结点的复用类型和形成Mux 的终结点
Media	媒体流属性的列表
TerminationState	与特定媒体流无关的终结点属性（可在包中定义）
Stream	对应于单个媒体流的remote/local/localControl 描述符的列表
Local	对MG 从远端实体接收到的媒体流进行描述的一些属性
Remote	对MG 发送给远端实体的媒体流进行描述的一些属性
LocalControl	MG 和MGC 之间的一些控制属性（可在包中定义）
Events	描述需要MG 检测的事件，以及当事件被检测到时作出的反应
EventBuffer	描述当EventBuffer 处于激活状态时，要由MG 检测的事件
Signals	描述向终结点加载的信号
Audit	可作为Auditvalue 和Auditcapabilities 命令的参数，定义需要审计的信息
Packages	可作为AuditValue 命令的参数，返回由终结点实现的包的列表
DigitMap	为MG 定义的号码采集规则，用于匹配拨号事件，使拨号事件按组而非单个上报
ServiceChange	可作为ServiceChange 命令的参数，描述何种业务发生改变以及业务发生改变的原因等等
ObservedEvents	可作为Notify 或者 AuditValue 命令的参数,报告被检测到的事件
Statistics	可作为Subtract、 Auditvalue 和Auditcapabilities 命令的参数，报告与终结点有关的统计数据
Topology	描述关联中终结点之间的媒体流流向
Error	定义了错误码和错误注释字符串，该描述符可作为命令响应及Notify请求命令的参数

描述符的文本格式如下：

DescriptorName=<someID>{parm=value, parm=value　.}

参数的值可以是：完全指定（Fully specified）、部分指定（Underspecified）或多余指定（Overspecified）的。

1) 完全指定：指定的参数具有唯一、确定的值。

2) 部分指定：使用通配值 “ CHOOSE ”，允许命令响应方为该参数选择任何一个它所支持的值。

3) 多余指定：参数具有多个可能的值列表，该列表的顺序指定了命令发起方对于这些值的优选权，命令响应方从该列表中选择一个值作为对命令发起方的响应。

如果一个描述符（除了Audit 描述符）没有在一个命令中指定，该描述符保持以前的取值不变。除了Subtract 命令，一个命令中没有指定Audit 描述符就等于指定了一个空的Audit 描述符。

2.5 事务

2.5.1 事务与事务的处理

MG 和MGC 之间的一组命令组成了事务(Transaction)。每个Transaction 由一个TransactionID来标识。

Transaction 由一个或者多个动作（Action）组成。一个Action 又由一系列命令以及对关联属性进行修改和审计的指令组成，这些命令、修改和审计操作都局限在一个关联之内。因而每个动作通常指定一个关联标识。但是有两种情况动作可以不指定关联标识符，一是当请求对关联之外的终结点进行修改或审计操作时，另一种情况是当MGC 要求MG 创建一个新关联时。事务、动作和命令之间的关系示意图如图2.2 所示。

事务分三种类型，TransactionRequest（事务请求）、TransactionReply（事务应答）以及TransactionPending。事务一般由TransactionRequest（事务请求）发起。对TransactionRequest 的响应放在一个单独的TransactionReply（事务应答）里面。在收到TransactionReply 之前，可能会先出现一些TransactionPending（事务处理中）消息。

事务保证对命令的有序处理。即在一个事务中的命令是顺序执行的。各个事务之间则不保证顺序处理，即各个事务可以按任意顺序执行，也可以同时执行。

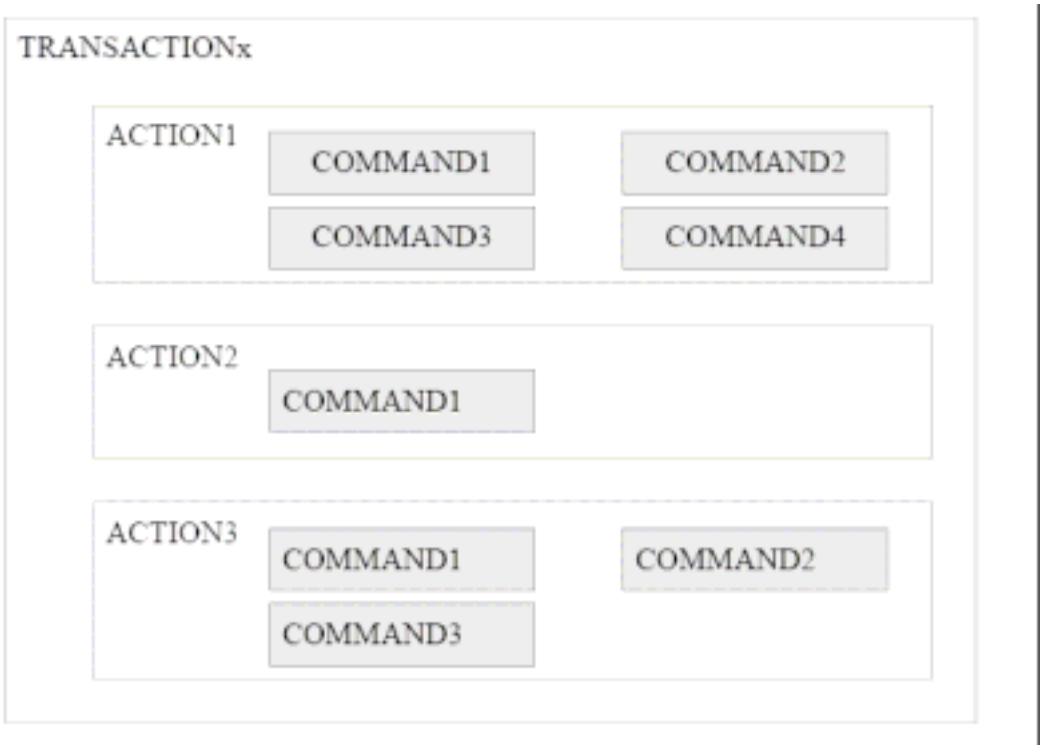

图 2.2　事务、动作和命令

如果一个事务中有一个命令执行失败，那么这个事务中的所有剩余命令都将停止执行。如果命令中包含通配形式的TerminationID，则对每一个与通配值匹配的TerminationID 执行此命令。TransactionReply 包含对应每个与通配值匹配的TerminationID 返回的一个响应；即使对其中一个或多个终结点产生了错误码。如果与通配值匹配的终结点在执行命令时发生了错误，则对此终结点之后的所有通配值终结点的命令将不再执行。

TransactionReply 包含相应的TransactionRequest 中的所有命令的执行结果，其中包括成功执行的命令返回值，以及所有执行失败的命令的命令名和Error描述符。

具体实现上，对每个事务都应该设置一个应用层定时器等待TransactionReply。当定时器超时后，应该重新发送TransactionRequest。当接收到TransactionReply 后，就应该取消定时器。当接收到TransactionPending 消息后，就应该重新启动定时器。具体介绍见下一章。

2.5.2 公共参数

2.5.2.1 Transaction ID

事务由TransactionID 标识。TransactionID 是由事务发起方分配并在发送方范围内的唯一值。如果TransationRequest 的TransactionID 丢失，TransactionReply 则带回一个Error 描述符指示TransationRequest 中的TransactionID 丢失，其中包含的TransactionID 填0。

2.5.2.2 Context ID

关联由ContextID 标识，ContextID 是由MG 分配并在MG 内唯一的值。在后继的与该关联相关的Transaction 中，MGC 应使用由MG 提供的这个ContextID。本协议规定，当涉及一个不在任何关联中的终结点时，MGC 可以使用“空关联”（Null）这个特定的值。

MGC使用通配值“CHOOSE ($)”来请求MG 来创建一个新的关联。

MGC 可以使用通配值“ALL”来寻址MG 中所有的关联。空关联不包含在“ALL”内。

MGC 不应该使用包含通配值‘CHOOSE ’或 ”ALL ’的部分通配的ContextID。

2.5.3 事务 API

本节给出一个应用程序接口（API）来描述本协议的事务。该API 只是说明事务及其参数，而不试图说明诸如模块功能调用的实现。它从一个更高的层面描述本协议的不同的Transaction 使用的输入参数和返回值。

2.5.3.1 TransactionRequest

TransactionRequest 由事务发起方发送，每发起一个请求后就有一个事务与之对应。一个请求包含一个或者多个动作，其中每个动作都指定了它的目标关联以及对目标关联作用的一个或者多个命令。TransactionRequest的文本格式可简略表示为

TransactionRequest(TransactionID {
    ContextID {Command … Command},
    . . .
    ContextID {Command … Command } })

其中，TransactionID 参数必须指定一个值，用该参数可以将TransactionRequest 与以后接收方发出的TransactionReply 或者TransactionPending 关联起来。

ContextID 必须指定一个值，该值用于随后的所有命令，直到指定下一个ContextID 或TransactionRequest 结束。

Command 即在2.3 节中描述的其中一个命令。

2.5.3.2 TransactionReply

TransactionReply 由事务的接收方发送，作为对TransactionRequest 的一对一响应。一个TransactionReply 包含一个或者多个动作，其中每个动作都必须指定动作的目标关联，以及对应每个关联的一个或者多个响应。当事务的响应方完成了TransactionRequest 的处理后，就会发送一个TransactionReply。

当出现以下两种情形之一时，就认为接收方已完成TransactionRequest 的处理：

（1）TransactionRequest 中所有动作已处理完毕；

（2）TransactionRequest 中的一个非可选命令处理失败。

当命令中的所有描述符都已处理完毕，就认为这个命令处理完毕。

对于信号描述符，如果该描述符语法描述正确，接收方支持信号描述符所指定的信号类型，并且指定的信号已经置于等候加载的队列中，就认为信号描述符处理完毕。

对于事件描述符，如果该描述符语法描述正确，接收方能够检测事件描述符中所指定的事件，能产生事件描述符中嵌套的信号，能识别事件描述符中嵌套的事件类型，并且MG 已处于检测事件发生的状态，就认为事件描述符处理完毕。

TransactionReplyt的文本格式可简略表示为

```
TransactionReply (TransactionID {
    ContextID{Response … Response},
    . . .
    ContexID {Response … Response}})
```

TransactionReply 中的TransactionID 参数必须与相关的TransactionRequest 中的TransactionID 相同。

ContextID 参数必须指定一个值，该值适用于动作中所有命令的响应。ContextID 可以指定为确定的值、“ ALL ”或“ NULL ”。

Response 参数代表命令返回值，或者命令执行失败时的Error 描述符。失败命令之后的其他命令将不执行，也不产生命令响应。但有一个例外。当Transactionrequest 中的一个标记为“ Optional ”（可选）的命令执行失败时， 事务中该命令后的命令将继续执行并返回命令响应。

2.5.3.3 TransactionPending

TransactionPending 由接收方发送，它表示一个Transaction 正在被处理，但是处理尚未完成。当对一个Transaction 的处理还需要一些时间来完成时，发送这个消息用来防止发送方认为相关的TransactionRequest 已丢失。

TransactionPending的文本格式可简略表示为

TransactionPending (TransactionID {})

TransactionPending 中包含的TransactionID 参数应与相关TransactionRequest 中的TransactionID 相同。根终结点的“ NormalMGExecutionTime ”属性由MGC 设置，用于指示MGC 期待从MG 接收到事务的TransactionReply 时间间隔，另一属性“ NormalMGCExecutionTime ”属性由MGC 设置，用于指示MG 期待从MGC 接收到事务的TransactionReply 时间间隔。事务发送方可能会接收到一个事务的多个TransactionPending 消息。当处于Pending 状态的接收方接收到一个重复的TransactionRequest，接收方可以立即发送一个重复的TransactionPending 消息，或者等待定时器超时后发送另外一个TransactionPending 消息。具体介绍见下一章。

2.6 消息

一个消息由多个Transaction 组成。每个消息都有一个消息头，其中包含标识消息发送者的标识符。可以将消息发送者的名称（如域地址/域名/设备名）作为消息标识符MID（MessageIdentifier）。在一对MGC 和MG 具有控制关系期间，一个H.248实体（MG 或MGC）在它作为发起方发送的消息中必须始终如一地使用同一个消息标识符MID。

消息包括一个版本字段用于标识消息所遵从的H.248协议版本。版本字段为1位或2 位数。目前已有的协议版本有版本1，版本2和新出的版本3。不同的版本，

协议的一些细微地方会略有不同。

消息中所包含的Transaction 是各自独立处理的。消息不规定任何顺序，也无所谓消息的应用程序或对消息的应答。例如， 消息X 包括TransactionRequest A ，B 和C，对它的响应可以是：由消息Y 包含对TransactionRequest A 和C 的应答，由消息Z 包含对TransactionRequest B 的应答。同样，消息L 包括TransactionRequestD ，消息M 包括TransactionRequest E，可以由消息N 同时包含对TransactionRequest D 和E的应答。

2.7 小结

本章重点学习和研究了H.248协议的重要概念和基本结构。H.248的基本结构基于终结点或终端，上下文或关联、命令、描述符、事务、消息等。这些都是非常抽象的概念，通过对这些概念的深入研究，为协议的应用研究打下了坚实的基础。在下一章，将就H.248的应用作深层次的研究。

第三章 H.248 协议应用研究

上一章对H.248协议及其消息的组成元素作了一定深度的介绍。但是在实际应用中，采用什么样具体的方式将上一章中介绍的一些组成元素编码成H.248消息，是首先需要解决的问题。其次，关于H.248协议的传送机制，是我们需要特别关注的一个对象。协议的传送机制应该支持对在MG 和MGC 之间的所有Transaction 的可靠传输，且传输应当与协议中需要传输的特定命令无关，可适用于所有的应用程序状态。如果是在IP上传输本协议，特别是UDP承载时，要保证可靠传输，应注意很多潜在的问题。在本章中，将在上一章介绍的事务API的基础上，首先引入H.248协议消息的文本编码方法，其次对IP/UDP上承载H.248协议消息的机制展开深入探讨。最后，为了说明协议的具体应用，将在本章末尾分别给出一个典型的呼叫建立和呼叫释放流程的例子。

3.1 H.248 文本编码

所谓的H.248文本编码方式指的是将H.248协议定义的消息的组成元素，按照特定的文本编码要求对其进行语法编码使之成为完整的规范的并且可以被唯一识别或辨认的H.248消息的方法。

在真实的应用之中，网络实体会将每条具体的H.248消息进行文本编码，并经IP分组网络传输至对端的对等实体，由于双方均采用特定的文本编码方式，故对端可以识别出相应的H.248消息，之后便完成消息所要求的相应的操作。

考虑到实际应用的广泛性，我们这里只考虑含单事务的H.248消息的编码方法。

在上一章2.5.3中，已经对事务的文本格式作了一简单的介绍。这里将深入。我们将通过一个经过文本编码后得到的H.248消息的例子，来说明H.248文本编码的具体方法。参见如下H.248文本编码消息：

```
MEGACO/1 [123.123.123.4]:55555 Transaction = 10003 {
    Context = $ {
        Add = A4444,
        Add = $ {
            Media {
```

```
                Stream = 1 {
                    LocalControl {
                        Mode = ReceiveOnly,

                        nt/jit=40 ; in ms
                    },
                    Local { v=0 c=IN IP4 $ m=audio $ RTP/AVP 4
a=ptime:30 v=0 c=IN IP4 $ m=audio $ RTP/AVP 0
                    }
                }
            }
        }
    } }
```

以上这条消息是一条典型的 H.248 文本编码消息，该条消息实现的功能是将一个物理终结点伴随一个 RTP 终结点加入一个新创建的关联。

分析该条消息的具体编码方法。

首先编码的是消息头部分。它主要包含协议识别字段和 MID 字段。该条消息的协议识别字段中，“ MEGACO ”，即“ Media Gateway Controlling ”，表明这是一条 H.248 消息。“ / ”后接上的数字，表示该条消息依据的 H.248 协议的版本号（不同版本的 H.248 协议内容会略有不同）。协议识别字段的编码方式一般是这样固定的，但有的时候会采用“ ! ”来代替“ MEGACO ”。MID 字段中，这里采用的方式是 IP 地址加端口号的方法作为消息发送者的标识符，中间用“ ：”隔开，即“ [123.123.123.4]:55555 ”。也可以采用其他的 MID 表示方法，如域名方式，当然编码的结果会有相应的产生变化。

第二步是事务部分的编码。 首先出现的是“ Transaction ”字段，其表明该条消息下面进入事务部分。“ = ”出现在“ Transaction ”后，其后添上网络实体给该条事务分配的唯一的 TransactionID。以上这些都是必不可少的。“ { ”后换行。

第三步是 Action 部分的编码。由于 Action 是属于事务的，所以准确的说这部分应该还是属于事务部分的编码。但是为了清晰理解，这里单独列出了 Action 部分的编码，把它作为编码的第三步。在这部分中，首先在文本中添上“ Context ”

字段，这是标识消息已经进入 Action 部分了。“ = ”出现在“ Context ”之后，并带上该关联的 ContextID。这些都是必不可少的。这里由于是 MGC 向 MG 申请新创建的关联，MG 还未对其 ContextID 进行分配，故其 ID 是“ $（表示‘ choose ’） ”。 “ { ”后换行。

第四步是命令部分的编码。类似于 Action 对事务的从属关系一样，命令也是包含在 Action 中的。但是这儿便于理解，也单独列出。首先编码的是命令名，这里是“ Add ”，之后是“ = ”，“ = ”后便是该条命令的执行对象，即对应于一个特定终端的 TerminationID。这些都是必不可少的。如果该 Action 中含有多条命令，在编码一条命令后，应立即换行编码下一条命令，该条消息就是这样的。“ A4444 ”表示的是物理终结点的 TerminationID，而“ $ ”表示的是还未分配的 RTP 终结点的 TerminationID。对于其中任意一个 TerminationID，如果需要对其属性进行修改，即消息包含描述符，则应在相应 TerminationID 后加上“ { ”，之后换行。

第五步是描述符的编码。即将命令中出现的描述符名称及其所带参数依次编码。上述消息包含的主要是对“ Media ”描述符的编码。描述符部分的编码相对复杂一些，这里就不作深入细微的介绍了。具体可以参见[1]。

描述符部分编码完成后，整条消息的文本编码便结束了。

对于实现不同功能的各种 H.248 消息，它们的编码方式是完全一致的，即按照以上介绍的特定的语法规则，以标准的 “ 消息——事务——动作——命令——描述符 ”的顺序依次伸入编码，最终实现一条完整的 H.248 文本消息。

H.248消息还可按照二进制的形式进行编码。但是其编码复杂，编码的可读性差，不利于真实应用。这里就从略了。

3.2 H.248 消息在 IP/UDP 上的传输

3.2.1 At-Most-Once 功能

当H.248协议消息在UDP 上进行传输时，可能会发生消息丢失。当发送的消息命令没有及时得到响应时，命令必须重复发送。大多数命令重复执行后产生的结果与单次执行不相同。MG在重复执行一条命令的情况下，比如多次执行了 Add 命令后，状态会难以预料。因此，传送程序应该提供“ At- Most-Once ”（最多处理一次）机制。

协议中的对等实体双方应当在各自的内存中保留两个列表：一个用来记录执行完最近接收Transaction 后所返回的TransactionReply ； 另一个用来记录当前未处理完的Transaction。当协议实体接收到一个TransactionRequest 消息时，它将接收到消息的TransactionID 与具有相同MID 的已发出的TransactionReply 的TransactionID（即第一个列表的记录）相比较，如果发现有匹配的TransactionID，则接收方不执行接收到的Transaction，而只是简单地重复发送响应消息（TransactionReply）。如果没有发现匹配的TransactionID ， 则接收方将该接收到的TransactionRequest 消息与未处理完的Transaction 列表相比较， 如果在列表中发现匹配的TransactionID ， 则表明此TransactionRequest 消息为重复发送，接收方不执行此Transaction（见3.2.4节关于发送TransactionPending 的程序）。

以上处理如图3.1所示。

在这个处理过程中采用一个定时器——长定时器（ LONG-TIMER）。LONG-TIMER 设定的时间值应该大于一个Transaction 的最大处理持续时间，应将Transaction 的最多重传次数、重传定时器的最大值以及分组在网络中的最大传输时延等计算在内。

当协议实体发出响应消息后，如果LONG-TIMER 超时，或者接收到了来自对等实体包含“ Response Acknowledgement ”参数的响应证实消息，则协议实体将从列表中删除该响应消息的拷贝。对于得到响应证实消息的Transaction，协议实体还应当保留其TransactionID的拷贝，直到LONG-TIMER 定时器超时。

通过以上机制，协议实体可以检测并忽略网络中产生的重复TransactionRequest。

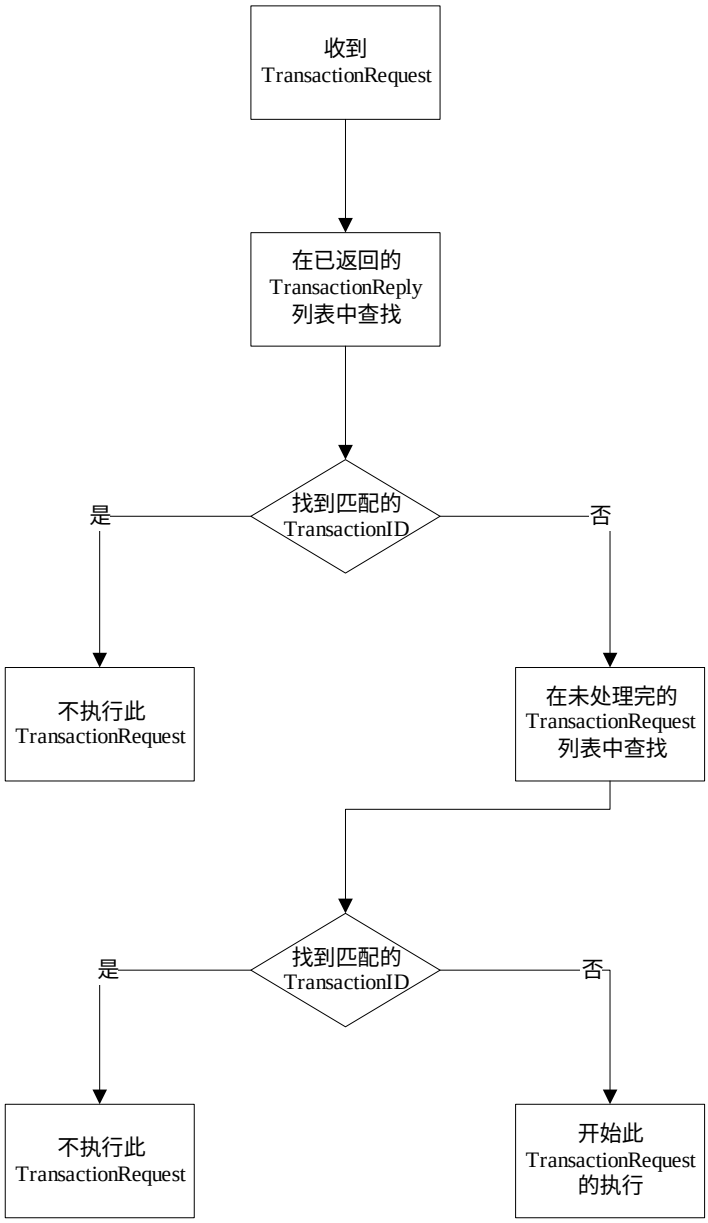

图 3.1　At-Most-Once 处理

3.2.2　TransactionID 和三次握手机制

3.2.2.1　TransactionID

事务标识（TransactionID）是一个32 比特的整数，MGC 可以为其控制范围内的每一个MG 预留一个特定的数值空间，或者为属于同一组（组可以任意划分）的所有MG 预留同一个数值空间。MGC 可以创建多个独立进程来实现对

一个大型MG 的操作管理，这些进程将共享一个TransactionID 编号空间。共享方式有多种，例如可以采用TransactionID 统一分配，或者采用为不同的进程预先分配互不重叠的数值空间。具体实现时应确保一个逻辑MGC（消息的MID 相同）产生的每一个Transaction 使用唯一确定的TransactionID。通过这种方式，MG 只需检查TransactionID 和MID 就可以检测到重复的事务（Transaction）。

3.2.2.2 三次握手机制

所有消息都可以携带“响应证实参数”（Response Acknowledgement Parameter），该参数携带了一组“已被确认的TransactionID 范围”（confirmed transaction-id ranges）。协议实体应当删除TransactionID 属于“已被确认的TransactionID 范围”的那些Transaction 的响应消息拷贝，在LONG-TIMER 超时前再收到TransactionID 属于这些范围的Transaction 应该直接丢弃。

如果在“已确认TransactionID 范围”中的MG 向MGC 发送的最后一个响应消息（TransactionReply）的LONG-TIMER 定时器超时，或者MG 重新启动，则“已确认TransactionID 范围”不再有效。此后，当接收到包含这些TransactionID 的TransactionRequest 消息时应该接受并处理。

以上处理可形象地用图3.2表示。

包含“Response Acknowledgement”参数的消息可以按任何顺序进行发送。协议实体应将接收到的“已确认TransactionID 范围”保留至其中最后一个响应消息的LONG-TIMER 超时。

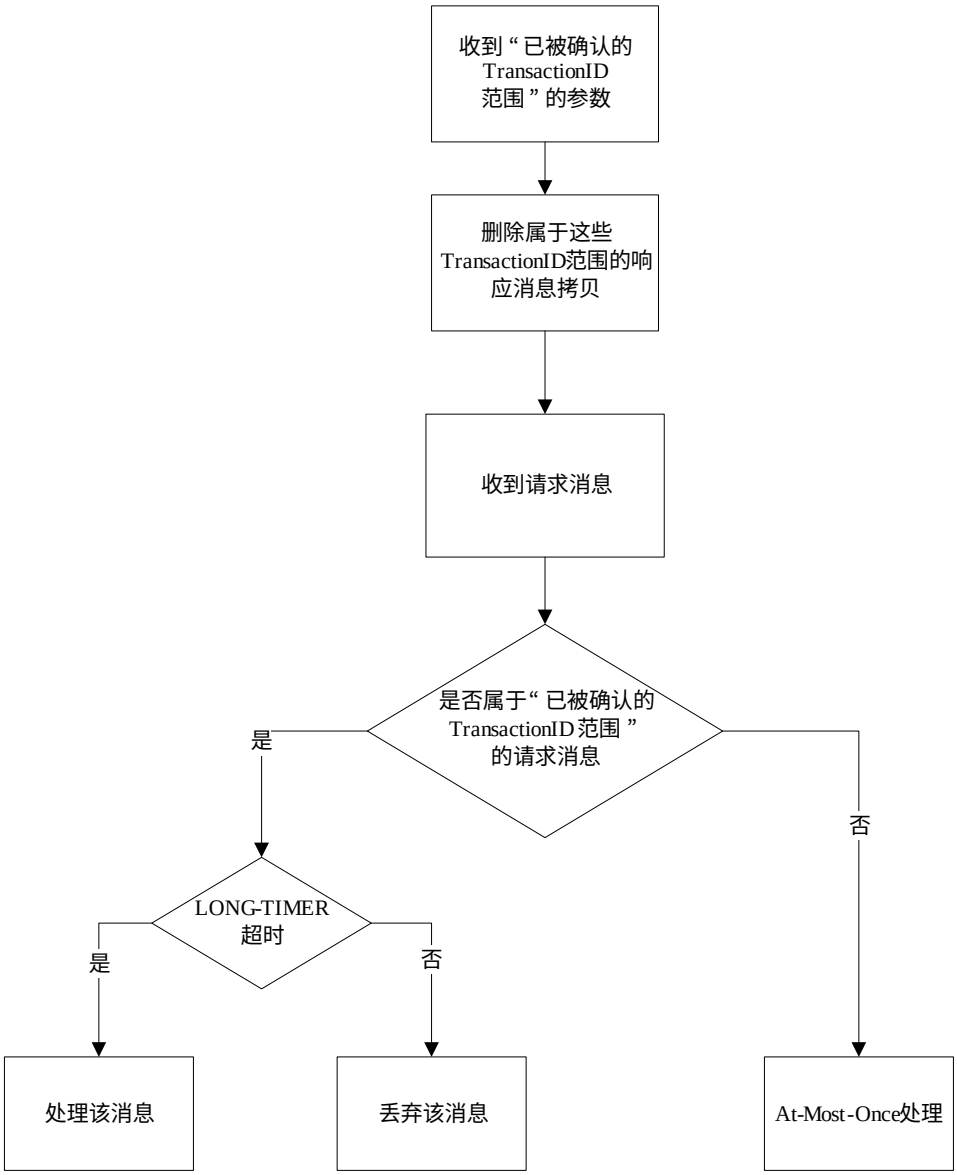

图 3.2　三次握手机制

3.2.3　重传机制

发起事务请求的实体应当为所发送的Transaction 启动一个重传定时器。当定时器超时，如果实体仍未接收到事务响应消息（TransactionReply），则应当重复发送该Transaction。当重复多次发送的Transaction 仍无法得到响应时，并且从第一次发起事务请求开始LONG-TIMER 超时，那么发起事务请求的实体应当采用某种替代服务和（或）拆除现有或即将建立的连接。

通常重传定时器的数值与特定的网络状况有关，一般应当通过测量发起事务请求消息与接收到响应消息之间的时间间隔来估算重传定时器数值。并且，发生第一次消息重传后，计算重传定时器的算法应对此后发生的每一次重传的重传定时器数值按指数递增方式进行修正。

3.2.4 临时响应

某些Transaction 的执行可能需要较长的时间，从而可能会导致Transaction执行与基于重传定时器的重传进程发生冲突。执行时间太长可能导致Transaction重传多次，或者导致重传定时器数值设得过大而降低传输效率。如果协议实体能预见某一Transaction 需要较长的执行时间，则它可以先发送一个临时响应消息TransactionPending 表示Transaction正在处理。当协议实体接收到重复的TransactionRequest 消息，而该Transaction 正在处理时，也应该发送TransactionPending 消息。

接收到TransactionPending 消息的实体应当转换到一个不同的重传定时器来重传相应的TransactionRequest 消息。根终结点Root 有一个属性叫作“ProvisionalResponseTimerValue”（临时响应定时器取值），该属性可以被设置以指定接收到TransactionRequest 消息和发TransactionPending 消息之间的最大时间，单位为毫秒。如果发起TransactionRequest 的实体在接收到TransactionPending 消息之后，接收到了最终的响应消息TransactionReply，则应立即向对端实体发送一个确认消息，此后应重新使用正常的重传定时器。发送了TransactionPending 临时响应消息的实体，应在随后发送的最终响应TransactionReply 中包含一个“immAckRequired”（需要立即确认）字段，表示希望立即得到一个确认消息。对于同一个TransactionRequest，实体在接收到最终TransactionReply 之后，应忽略接收的TransactionPending 消息。

通过At-Most-Once，三次握手，重传以及临时响应的传输和执行机制，有效地保证了在IP/UDP上，H.248消息的可靠传输。

3.3 H.248 协议的典型应用

在下一代固网和移动通信网络中，H.248协议的最重要功能是完成MGC对MG侧终结点业务（最常见的是呼叫）的控制。如何利用第三章介绍的命令以及对终结点属性进行修改的描述符，完成MGC对MG侧业务的控制，将是本节讨论

的重点。这里，将以流程图的形式，介绍MGC和MG之间反复交互的H.248消息对一种常见呼叫的建立和释放过程所进行的控制。通过这两个例子，读者将对H.248协议及其应用有更深入、更形象的认识。

3.3.1 不同接入网关设备上的终端之间呼叫建立的流程

H.248消息对不同接入网关设备上的两个终端之间呼叫建立的控制流程如图3.3所示。

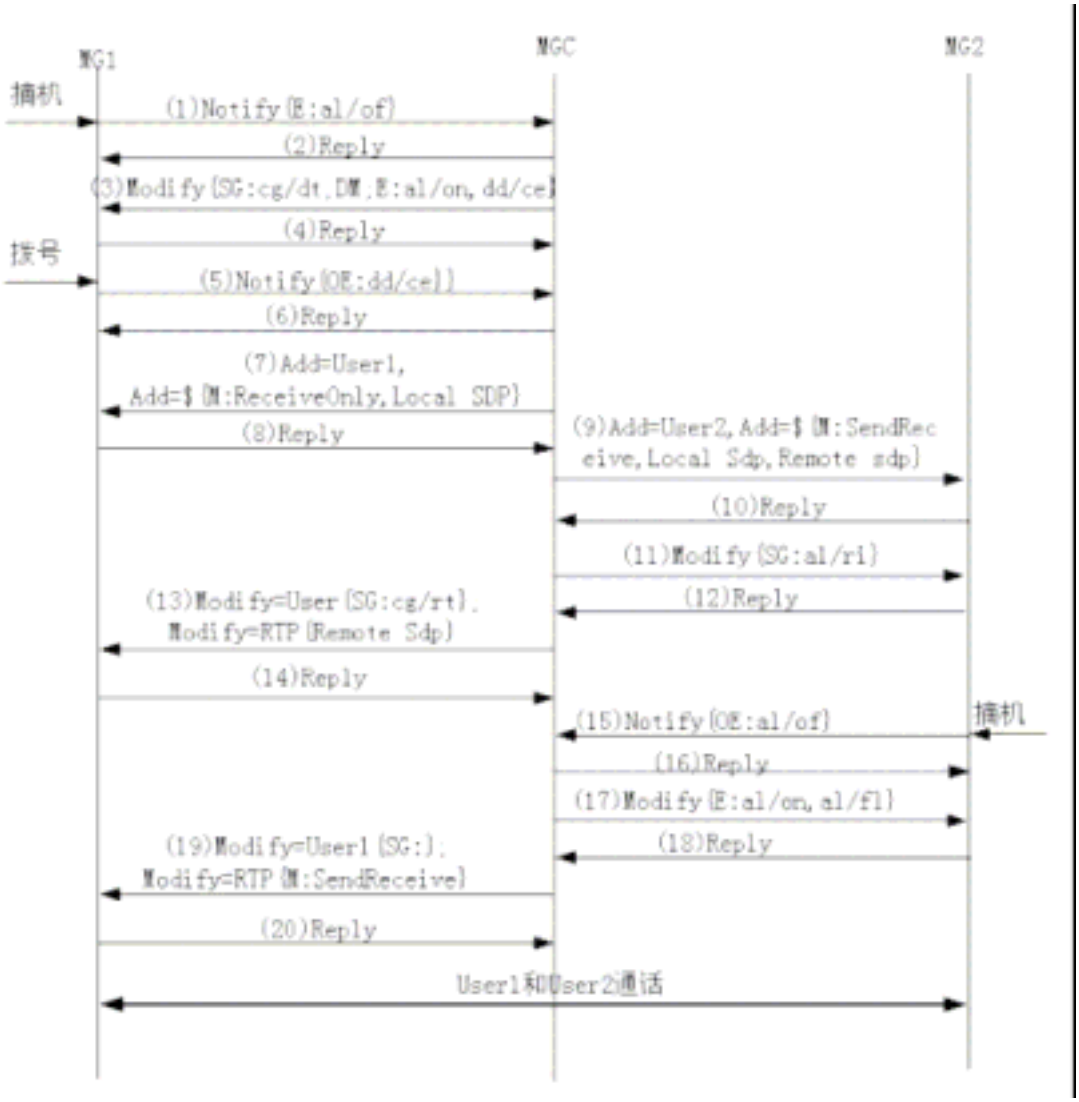

图 3.3 AG 终端呼叫建立流程图

图中采用了简略的H.248文本编码方式描述了MGC和MG之间每一个命令的

发起和响应。

流程说明如下：

1) MG1 检测到用户User1 的摘机，将此摘机事件通过Notify 命令上报给MGC；

2) MGC 向MG1 返回Reply；

3) MGC 向MG1 发送Modify 消息，向MG1 发送号码表（Digitmap）；请求MG1 放拨号音（cg/dt）；并检测收号完成（dd/ce）、挂机（al/on）、拍叉簧（al/fl）事件；

4) MG1 向MGC 返回Reply；

5) MG1 上的用户User1 拨号，MG1 根据MGC 所下发的号码表进行收号，并将所拨号码及匹配结果用Notify 消息上报MGC；

6) MGC 向MG1 返回Reply；

7) MGC 向MG1 发送Add 消息，在MG 中创建一个新context，并在context 中加入用户User1 的 termination 和RTP termination，其中RTP 的Mode 设置为ReceiveOnly，并设置语音压缩算法；

8) MG1 为所需Add 的RTP 分配资源RTP1，并向MGC 应答Reply 消息，其中包括该RTP1的IP 地址，采用的语音压缩算法和RTP 端口号等；

9) MGC 向MG2 发送Add 消息，在MG2 创建一个新context，在context 中加入用户User2的termination 和RTP termination，其中Mode 设置为SendReceive，并设置远端RTP 地址及端口号、语音压缩算法等；

10) MG2 为所需Add 的RTP 分配资源RTP2，并向MGC 应答Reply 消息，其中包括该RTP2的IP 地址，采用的语音压缩算法和RTP 端口号等；

11) MGC 向MG2 发送Modify 消息，MG2 向被叫送振铃音（al/ri）；

12) MG2 向MGC 应答；

13) MGC 向MG1 发送Modify 消息，让User1 放回铃音，并设置RTP1 的远端RTP 地址及端口号、语音压缩算法等；

14) MG1 向MGC 返回Reply；

15) MG2 检测到用户User2 的摘机，将此摘机事件通过Notify 命令上报给MGC；

16) MGC 向MG1 返回Reply；

17) MGC 向MG2 发送Modify 消息，让MG2 检测User2 的挂机（al/on）、

拍叉簧（al/fl）事件；

18) MG2 向MGC 返回Reply；

19) MGC 向MG1 发送Modify 消息，让User1 停回铃音signal{}，并设置RTP1 的Mode 为SendReceive；

20) MG1 向MGC 返回Reply；User1 与User2 正常通话。

如果是同一网关上两个终端的呼叫，可以省略对User1和User2的RTP资源的分配，以及创建RTP termination的操作。因为在同一网关上的终结点建立链路，无需分配RTP资源便可实现连接。

3.3.2 不同接入网关设备上的终端之间呼叫释放的流程

依然是不同接入网关设备上的两个终端，其呼叫释放过程如图3.4所示。

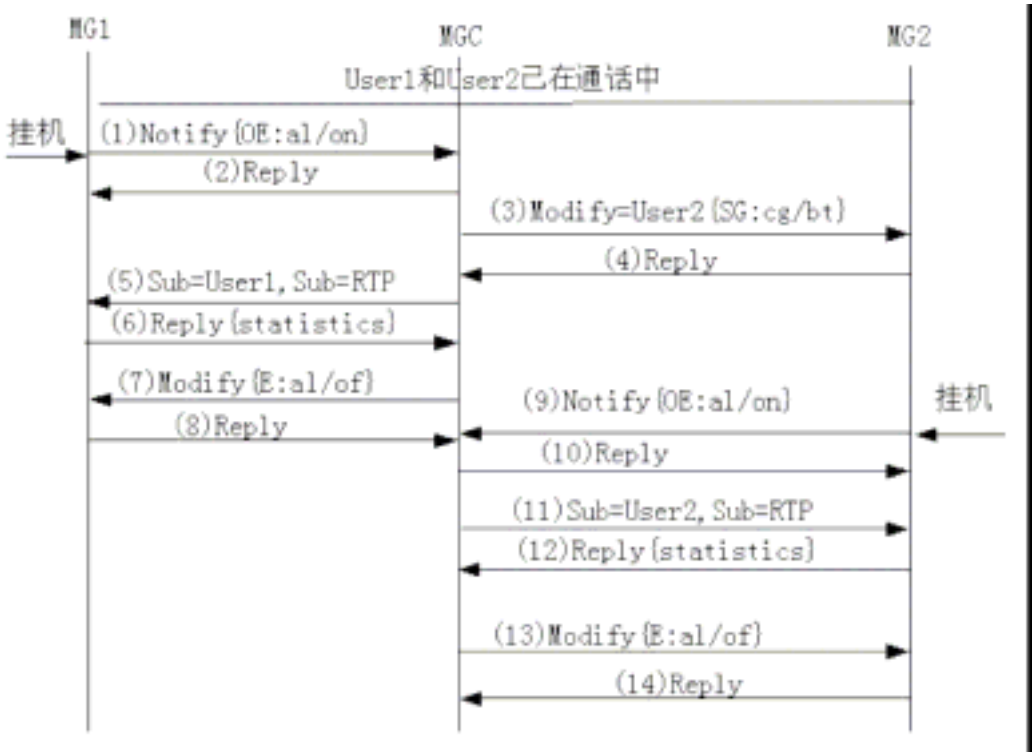

图 3.4 AG 终端呼叫释放流程图

流程说明：

1) MG1 检测到用户User1 的挂机，将此挂机事件通过Notify 命令上报给MGC；

2) MGC 向MG1 返回Reply；

3) MGC 向MG2 发送Modify 消息，让MG2 对User2 放忙音（cg/bt）；

4) MG2 向MGC 返回Reply；

5) MGC 向MG1 发送Subtract 消息，释放User1 和RTP1；

6) MG1 向MGC 返回Reply，释放资源，并向MGC 上报呼叫的媒体流统计信息；

7) MGC 向MG1 发送Modify 消息，让MG1 检测User1 的摘机（al/of）；

8) MG1 向MGC 返回Reply；

9) MG2 检测到用户User2 的挂机，将此挂机事件通过Notify 命令上报给MGC；

10) MGC 向MG2 返回Reply；

11) MGC 向MG2 发送Subtract 消息，释放User2 和RTP；

12) MG2 向MGC 返回Reply，向MGC 上报呼叫的媒体流统计信息；

13) MGC 向MG2 发送Modify 消息，让MG2 检测User2 的摘机（al/of）；

14) MG2 向MGC 返回Reply。

对于同一网关上两个终端的呼叫释放过程，由于无需创建RTP终结点，所以以上流程中RTP资源的释放操作可以省去。

除了正常的呼叫建立和释放业务外，H.248 协议还支持异常的呼叫控制，如久不拨号，空号，错号，后挂方久不挂机等。另外，H.248 协议还可完成对许多新型业务和智能业务的控制，如呼叫前转，主叫号码显示，呼叫等待，区别振铃，三方通话，会议电话，智能放音，实时传真数据业务等等，它们其中一些是通过 H.248 扩展协议包来实现的。限于篇幅，这里都从略了，有兴趣的可以参考相关的扩展协议。由此可见，H.248 协议在下一代核心网中提供了各种业务的实现方案，随着新型业务的开发和拓展，H.248 协议也将同步的不断发展。

3.4 小结

本章，作者主要就 H.248 的具体应用进行了深入的研究。首先，我们探讨了真实应用中，协议采用的编码方式。事实上，在国外，自协议诞生之日起，这点就一直存在争议。这里，作者介绍了现在相对较为认可的也是最为简单的编码方式，即文本编码方式。其次，由于协议是承载在 IP 上的，尤其是如果承载基于 UDP，协议传输需要提供相应的机制确保传输的成功。这点在实际设计时

非常关键。丢失或重复执行某条指令都可能会给系统带来不可挽回的后果。于是作者深入介绍了 At-Most-Once 功能、三次握手机制、重传机制和临时响应，这些方法可以确保系统消息传输和执行的可靠性。在本章的最后一节，作者给出了两个具体应用的实例，通过对实例的研究，可以对协议的应用有更全面，更高层次的理解。

第四章　H.248 协议在 CDMA2000 中的实现设计

前面我们重点介绍了 H.248 协议的内容和应用。在本章中，我们将探讨本文的核心内容，即如何在真实的下一代网络系统中实现 H.248 协议。这里将以作者重点参与开发的中兴公司 3GCN-MSCe 项目为背景，具体介绍在中兴 CDMA2000 系统核心网元 MSCe 中实现 H.248 的方法。

4.1　3GCN 组网基本介绍

3GCN 网络架构如图 4.1 所示。

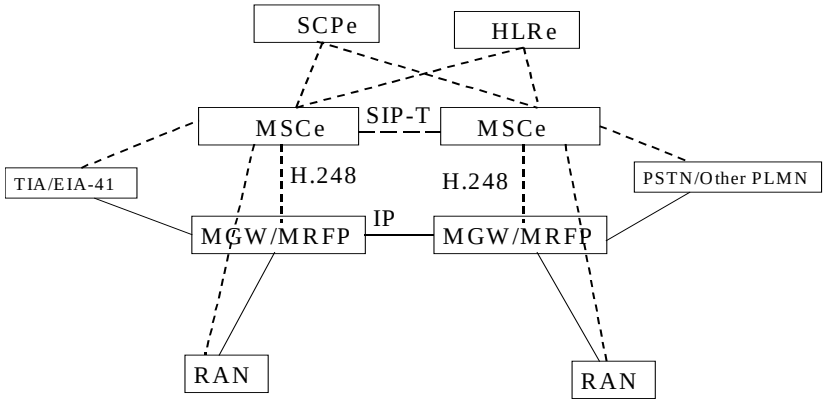

图 4.1　3GCN 网络结构图

系统主要的网元：

移动交换中心仿真 MSCe（Mobile Switching Center Emulation） 3GCN 网络的核心控制网元，完成与 MGW、无线接入网 RAN、传统 PSTN 以及其他 PLMN 网之间信令的互通。在整个系统中，MSCe 提供对包括呼叫在内的各种业务的相关控制。MSCe 与 MSCe 之间依据 SIP-T 协议通信， MSCe 与 MGW 之间依据 H.248 协议通信。在与 MGW 交互时，MSCe 实现了软交换系统中媒体网关控制器 MGC 的功能。

媒体网关 MGW（Media Gateway） 完成无线接入网 RAN、传统 PSTN 以及

其他 PLMN 网之间媒体的转换和互通。来自源端点的各种业务媒体流经一个或多个 MGW 处理，传输到目的端点。

多媒体资源功能处理器 MRFP（Multi-Media Resouce Function Processor） 主要完成媒体流的混合输入（如多方会议）、多媒体流的发送（如多方广播）以及多媒体流的处理（如语音编码转换、媒体分析）。

归属位置寄存器仿真 HLRe（Home Location Register Emulation） 系统的中央数据库，管理用户的业务特征以及位置和可接入性信息，为 MSCe 和 MGW 与 MRFP 提供建立呼叫所需的路由信息等相关数据。与 2G HLR 相比，其增加了 IP 信令的接口。

业务控制点仿真 SCPe（Service Controlling Point Emulation） 负责执行业务逻辑所指定的操作，实现业务控制功能 SCF 和业务数据功能 SDF，同时接受业务管理系统 SMS 的管理。

从网络结构中可以发现，3GCN 与传统的 CDMA 网络相比，最大的一个变化是采用 All-IP 的架构，融入软交换的设计理念，将 MSC 分成 MSCe 和 MGW 与 MRFP，成功地实现了信令和承载的分离，是典型的 NGN 网络。

4.2 3GCN—MSCe 项目 H.248 软件模块分析

在 3GCN 项目中，MSCe 同 MGW 之间基于 H.248 协议进行实时通信。H.248 协议可以直接基于 SCTP/IP 传输，也可以基于 M3UA/SCTP/IP 传输。H.248 协议承载基于的协议栈如图 4.2 所示。其中，SCTP 或 M3UA/SCTP 部分，主要完成 MSCe 和 MGW 之间链路的连接和维护，保障两个网元之间某个连接的正常通信。

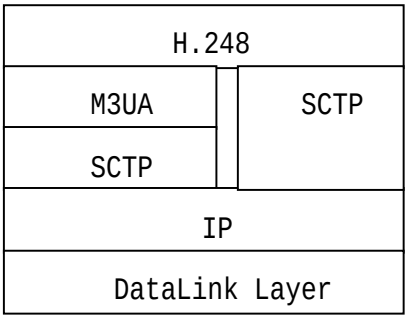

图 4.2 H.248 承载基于的协议栈

在 MSCe 中，H.248 模块完成 H.248 协议的处理，使 MSCe 在网络中成为

MGW 的控制实体。它将业务信令的逻辑关系映射成为 H.248 协议的描述，控制 MGW 实现以呼叫为核心的各种业务。

经方案的设定，其功能具体包含：

1.在 MSCe 内部，实现 H.248 协议的消息格式及信令的逻辑关系与内部消息之间的转化；

2.实现 H.248 协议中 MSCe 侧的全部功能；

3.对操作维护模块和网管模块提供管理接口，使其支持包括信令跟踪与告警在内的中兴公司统一网络管理平台工具的使用。

需要指出的是，“ 信令跟踪 ”功能虽然跟协议本身联系不大，但是它却是系统中很重要的一部分。除了业务功能的真实实现以外，利用网管平台上获得的跟踪信令，也可以帮助判别本模块程序设计最终是否成功。在本文的第五章中，就重点运用了这个工具。

从软件模块的角度上来看，在 MSCe 中，首先，将 H.248 模块置于呼叫业务模块 CC 的下一层。CC 模块是整个系统中所有可实现的实时业务的汇总，系统中所有的业务，均是从 CC 模块发起和控制的。利用这样的结构，H.248 模块可以直接将 CC 模块的业务控制消息，转化成 H.248 协议消息，向网络下层实体（M3UA/SCTP 或 SCTP）发送，以完成对 MGW 上终端间各种基本呼叫业务的实时控制。

其次，将 M3UA/SCTP 或 SCTP 对应的软件模块置于 H.248 模块的下层，它们负责接收 H.248 协议消息，并与它们的下层模块协同工作，完成 H.248 消息至 MGW 的传输。

第三，设定 H.248 模块与专门为 3GCN 项目设计的数据库模块之间的接口，以实现重要数据的读写。

最后，设定 H.248 模块与操作维护模块和网管模块之间的接口。

我们将 H.248 模块以及与之有直接交互的相关模块，它们之间的关系以图 4.3 表示出。图中，OMC 表示操作维护模块，DB 为数据库模块。

从图 4.3 中，可以发现，H.248 模块的软件代码，在设计时，主要将其分为两大基本进程，即 H248 TTM 和 H248 CA（以下简称 TTM 和 CA）。其中，TTM 进程位于 H.248 模块软件的下层，它直接处理来自 MGW 的 H.248 消息，并将消息转化成 H.248 模块的内部消息发送至 CA 进程；相反地，TTM 进程也接收 CA 进程发送来的模块内部消息，并将之映射成 H.248 文本消息，发送给 MGW。

CA 进程位于 H.248 模块软件的上层，它直接接收 CC 模块下发的业务消息，并将该业务消息转化成 H.248 模块的内部消息发送给 TTM 进程，TTM 进程对该消息进行相应的 H.248 协议编码，并将之发送给 MGW；相反地，CA 进程也接收 TTM 进程发来的模块内部消息，并将其转化成相应的业务消息发送给 CC 模块。

CC 模块与 CA 进程之间的接口，我们称之为 Veinu 接口，如图 4.3 所示。它们之间传输的业务消息，我们称之为 Veinu 消息。

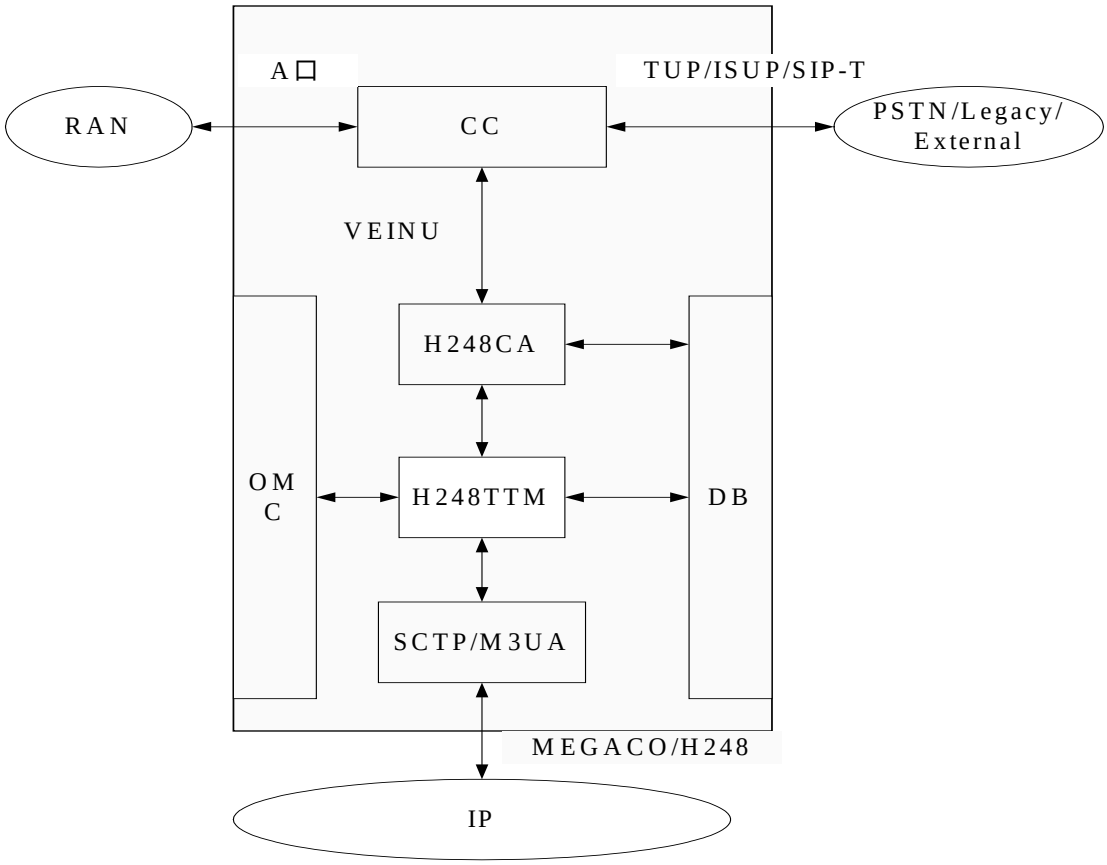

图 4.3　H.248 软件模块相关结构图

各进程及模块之间的接口设计，采用如下的方法：

TTM 进程同 SCTP 模块之间采用函数调用的方法进行交互。

TTM 进程同 M3UA 模块之间采用消息接口进行消息的传递。

TTM 进程同 CA 进程之间采用消息接口进行消息的传递。

TTM 进程同 OMC 模块之间采用消息接口进行消息的传递。

TTM 进程同 DB 模块之间采用函数调用的方法进行交互。

CA 进程同 DB 模块之间采用函数调用的方法进行交互。

CA 进程同 CC 模块之间采用消息接口进行消息的传递。

4.3 TTM 进程的设计

作为 MSCe 中 H.248 模块的下层，从协议栈的角度，TTM 进程是 MSCe 和对等实体（MGW）交互 H.248 消息的窗口。TTM 进程不但要完成和 CA 进程以及 M3UA 或 SCTP 模块之间消息结构的转化，还要保存与 MGW 交互的事务队列，以完成 At-Most-Once、三次握手机制以及临时响应的处理，保证协议消息 IP 承载下的传输和执行的正常。

4.3.1 TTM 进程的内部消息队列结构的设计

为了实现 H.248 协议基于 IP 的传输机制，在 TTM 进程中，我们设计了两组重要的消息队列，它们成为 At-Most-Once、三次握手机制以及临时响应处理实现的重要保证。

需要指出的是，在实际实现中，为了简便，我们将系统中产生的 H.248 消息均设置成单事务消息。

设计的两组消息队列，其中一组消息队列称之为 MSCe 消息队列，它包含两种消息队列，一种用来存放发送给 MGW 的事务请求消息，另一种用来存放发送给 MGW 的事务应答消息，我们分别称之为 MSCe 事务请求消息队列和 MSCe 事务应答消息队列；另一组则称之为 MGW 消息队列，它只包含一种消息队列，用来存放来自 MGW 的事务请求消息，我们称之为 MGW 事务请求消息队列。

准确说来，设计的队列是用于依次存放属于该队列的消息数据区（见 4.6.4 节），这些数据区中保存了相应消息的重要信息。在本节的介绍中，为了描述简单，就称消息队列中依次保存了相应的消息，同样对于消息队列的操作，比如某条消息对应的数据区在相应队列中的加入和删除操作，就称在该队列中加入或删除此消息。关于队列中消息数据区及队列操作具体的介绍，详见 4.6.4 节。

4.3.2 TTM 进程接收 MGW 消息的预处理设计

TTM 进程通过网络接收到 MGW 传送来的 H.248 事务消息后，首先应进行事务分离，并获得相应的事务类型。其中，依据前述协议的内容，所获得的事

务类型将会是以下四种中的一种，即事务请求（TransactionRequest）、事务应答（TransactionReply）、临时响应（TransactionPending）以及应答确认（TransactionResponseAck）。TTM 进程获得事务类型后，应按照不同事务类型的处理方法对其进行处理。

在具体的预处理的设计中，采用如下的流程：

如果事务的类型是 TransactionRequest，首先进行 At-Most-Once 处理。TTM 进程先在 MSCe 事务应答消息队列中查找，如果找到了收到的事务请求消息所对应的事务应答消息，表明此事务请求消息已经处理过，并且已经给 MGW 回送过相应的事务应答了；如果没有找到相应的事务应答消息，则在 MGW 事务请求消息队列中查找，如果找到相应的事务请求消息，表明此事务请求消息正在处理；如果在 MSCe 事务应答消息队列和 MGW 事务请求消息队列均未找到匹配的消息，则认为接收到的是一条新的事务请求消息。

根据查找的三种不同结果，按以下步骤分别对其进行处理：

第一种，即此事务已经处理过，并且已经给 MGW 回相应的事务应答了。TTM 进程将认为此事务请求是 MGW 发送来的一条重传的事务请求，此时由于某种原因（网络繁忙或是网络丢包），MGW 可能还没有收到 MSCe 发送的 H.248 事务应答消息，TTM 进程将重传此事务应答消息，但收到的事务请求消息不进行处理。

另一种，即此事务正在处理中，因为事务的处理需要涉及到 H.248 模块以及其他可能需要交互的模块，所以会需要一定的时间。如果对端 MGW 的 H.248 事务重传定时器时长设置的较短，可能会出现此种情况。此时将直接丢弃此事务请求，或给对端的 MGW 发送一个临时响应，以此抑制 MGW 对此事务请求的重发。

第三种，即此事务请求是一个新的事务，此时 TTM 进程将此事务放入相应的 MGW 事务请求消息队列中，并保留其中的关键参数，例如 MGWID、TransactionID 等等，之后进入了 TTM 进程的事务请求消息的处理流程。

如果事务的类型是 TransactionReply，说明此事务对应 MSCe 发起的一个事务请求消息。TTM 进程首先在 MSCe 事务请求消息队列中查找对应的事务请求，如果没有找到，此事务应答丢弃。如果找到对应的事务请求消息，则终止为此事务请求消息设置的重传定时器，同时将此事务请求从相应的 MSCe 事务请求消息队列中删除。如果对端需要三次握手，则给 MGW 发送此事务应答的

TransactionResponseAck。之后进入了 TTM 进程的事务应答消息的处理流程。

如果事务的类型是 TransactionPending，表明此时收到了 MGW 的临时响应消息，可能的原因是 MGW 比较繁忙。此时，需在 MSCe 事务请求消息队列中查找与此对应的事务请求，找到之后，终止为此事务请求而设置的重传定时器，并重新设立一重传定时器。

如果事务的类型是 TransactionResponseAck，表明对端的 MGW 已经收到了 TTM 进程发送的 TransactionReply，此时将相应 MSCe 事务应答消息队列中保留的事务应答消息删除，并将其 TransactionID 保留至长定时器超时。

以上流程参见图 4.4 所示。

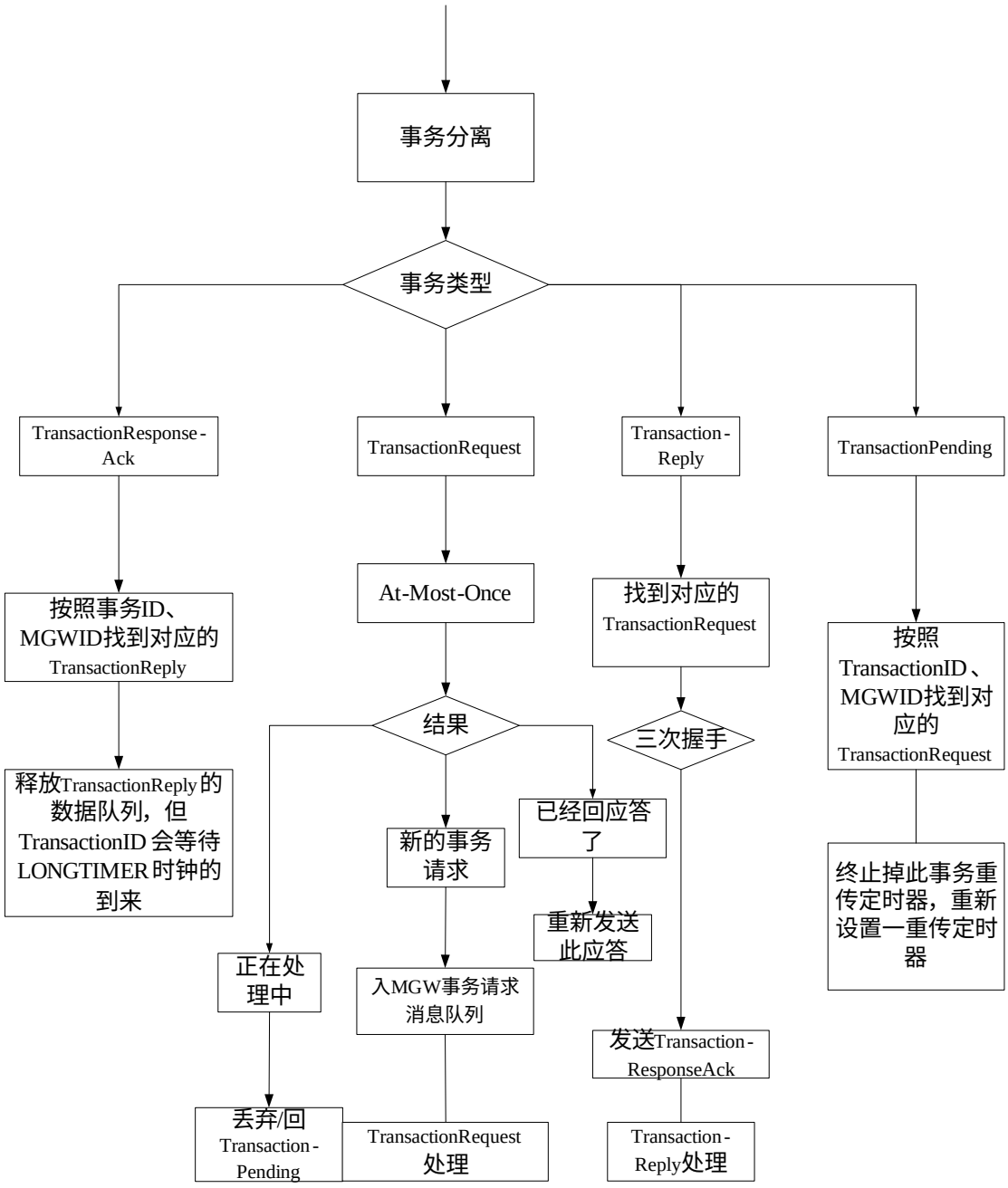

图 4.4　TTM 进程接收 MGW 消息的预处理

4.3.3　TTM 进程关于 MGW 事务请求消息处理的设计

对于 MGW 传送的事务请求，在预处理结束后，TTM 进程将调用解码函数，

依次进行 Actions、命令、描述符的解码，如果解码失败，则会给 MGW 反馈相应的 Error 描述符。如果解码成功，则为此事务请求分配数据区，之后以命令为单位，将该消息中重要信息依次填写该数据区，并将填写后的数据区的首地址发送给 CA 进程。该地址即作为 CA 进程从 TTM 进程获取消息的接口。

以上设计流程参见图 4.5 所示。

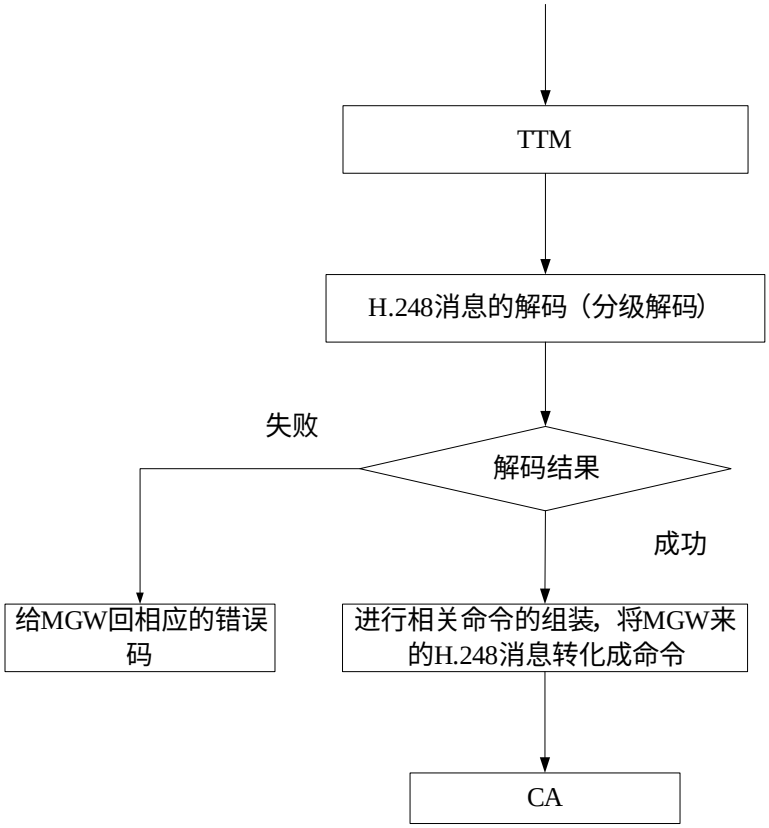

图 4.5　TTM 进程对 MGW 事务请求消息的处理

4.3.4　TTM 进程关于 MGW 事务应答消息处理的设计

对于 MGW 传送的事务应答，在预处理结束后，TTM 进程同样的调用解码函数，依次进行 Actions、命令、描述符的解码，如果解码中发现有 Error 描述符，则给 CA 进程发送相应的命令，并带上相应的错误码。如果解码中未发现 Error 描述符，则为此事务应答分配数据区，并将解码后的命令描述符填入该数据区，之后将填写后的数据区的首地址发送给 CA 进程。

以上设计流程参见图 4.6 所示。

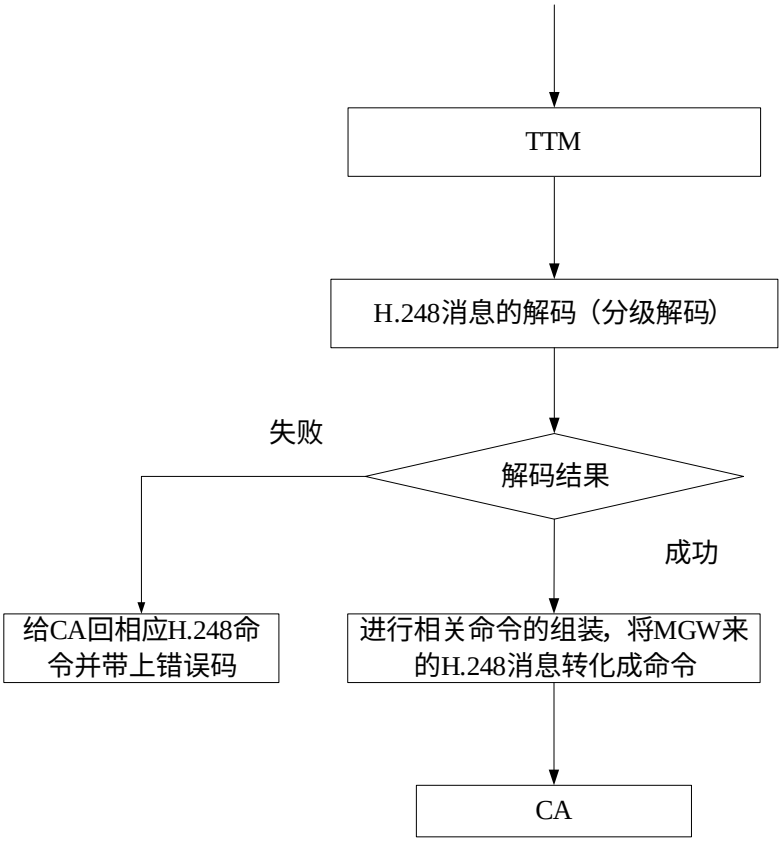

图 4.6　TTM 进程对 MGW 事务应答消息的处理

4.3.5　TTM 进程关于 MSCe 事务消息处理的设计

对于 MSCe 向 MGW 传送的事务消息，TTM 进程首先通过与 CA 进程之间的消息接口，接收 CA 进程传送来的 H.248 协议命令及参数。TTM 进程根据收到的事件号，判断该命令对应的事务将会是 TransactionRequest 还是 TransactionReply。如果事件号指示该事件是请求事件，则依次进行命令、Actions、Transaction 的编码，之后将生成的事务请求放入相应 MSCe 事务请求消息队列中，同时设置此事务请求消息相应的重传定时器，最后将该事务发送给 MGW。如果事件号指示该事件是应答事件，则依次进行命令、Actions、Transaction 的编码，并将生成的事务应答放入相应 MSCe 事务应答消息队列，同时设置此事务应答相应的长定时器，之后将此事务应答消息发送给 MGW。

以上设计流程参见图 4.7 所示。

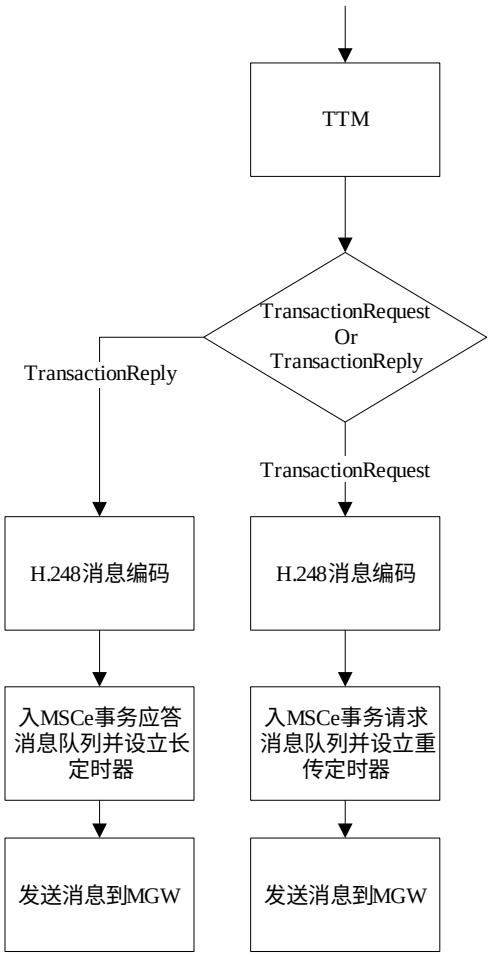

图 4.7　TTM 进程对 MSCe 事务消息的处理

4.4　编解码模块的设计

TTM 进程一个非常重要的功能就是 H.248 协议消息的编解码处理。考虑到其在 H.248 软件模块中的不可或缺性，我们将其相关的设计提取出来，单独介绍。为了方便起见，这里，把 H.248 软件模块中相应的部分独立称作为编解码模块。

1．编码的设计

在 3.1 节中，我们已经介绍了 H.248 消息的文本编码方法，编解码模块编码部分的设计完全采用这种方法。

2．解码的设计

首先考虑一个基本的加法公式 A１ ＋B１ ＝C１，可以分解出该公式中最小的独立单元有“A1”、“＋”、“B1”、“＝”、“C1”。类似的，对于符

合 H.248 协议的文本编码消息，也可以分解出其中最小的独立单元，在 H.248 协议之中我们称之为限定符。

以 3.1 节给出的 H.248 文本编码消息为例，在第一行中，限定符有“ MEGACO ”、“ / ”、“ 1 ”、“ [”、“ 123 ”、“ . ”、“ 123 ”、“ . ”、“ 123 ”、“ . ”、“ 4 ”、“] ”、“ : ”、“ 55555 ”、“ Transaction ”、“ = ”、“ 10003 ”、“ { ”。

由于 3.1 节描述的 H.248 文本编码方式的规范性，在一条 H.248 文本编码消息中，某个限定符出现的位置将会有特定的要求。

在编解码模块解码部分的设计中，采用如下的思想，即设立一个位置变量，用于保存当前解码的限定符所处消息中的位置信息，在消息限定符依次解码的过程中，TTM 进程相应的改变位置变量的值。当位置变量处于某值时，根据 H.248 文本编码规则，如果发现希望出现的限定符未出现，或拼写出现错误，则认为接收到的 H.248 文本消息码流在该处发生了语法错误，TTM 进程将报错，此时解码失败，解码过程结束。

对于上面的公式例子，在“ A１ + B１”之后理应出现“ = ”，但如果在该位置，未能出现该符号，则认为该公式在此处出现语法错误。同样的，对于 3.1 节介绍的例子，如果在“ MEGACO ”后未出现“ / ”，或“ / ”后未出现代表协议版本的数字（或是在其他位置出现违反编码规则情况的），则认为该消息在相应位置出现了语法错误。

对于接收到的 H.248 文本消息，如果位置变量指示至消息中最末尾的一个限定符，并且在整个解码过程中未出现违反编码规则的语法错误，则该消息解码成功，解码过程结束。之后，TTM 进程完成对消息解码信息的保存。

4.5 CA 进程的设计

作为 MSCe 中 H.248 模块的上层，CA 进程是 H.248 模块和 CC 模块交互的窗口。H.248 模块对 MGW 发送的各种 H.248 呼叫控制消息均是来源于 CC 模块对 CA 进程发送的相应的业务控制消息。如果系统需要提供某种业务（比如创建一个呼叫），CC 模块则会向 CA 进程发送相应的一条业务控制消息，包括 CA 进程在内的 H.248 模块对接收到的消息进行一系列处理，并将其转化为一条 H.248 文本编码消息，向 MGW 发送，以对呼叫进行控制。因此，在呼叫业务进

行过程中，CA 进程直接和 CC 模块交互，主要完成业务消息的接收和处理任务。

考虑到 H.248 模块整体的功能是基于 TTM 进程和 CA 进程组合来实现的，设计 CA 进程，使其实现以下几个具体功能，以完成 H.248 整个模块的功能要求：

1.接收 CC 模块的业务消息，并将其转化为 H.248 协议的命令及参数；

2.完成上下文数据区和终端数据区的创建和管理，以保护和管理呼叫；

3.通过同 TTM 进程之间的消息接口与 TTM 进程交互，将从业务消息中获得的命令及参数通过地址传送的方式发送给 TTM 进程；

4.接收 TTM 进程传送来的 H.248 协议命令及参数，并将其转化为 CC 模块可以识别的业务消息，发送给 CC 模块；

5.与 DB 模块交互，以同步 MGW 的状态。

由 4.2 节中的定义，我们将 CC 模块与 CA 进程之间的接口称之为 Veinu 接口，它们之间交互的业务消息称之为 Veinu 消息。

4.5.1 CA 进程两个重要数据结构的设计

对于基于呼叫的业务，CA 进程需要根据 CC 模块传送来的业务消息或 TTM 进程传送来的 H.248 协议命令及参数，实时存储即将或已经建立呼叫的用户终端信息和上下文拓扑信息，这是 H.248 模块实现控制和管理呼叫的重要一部分。

为了有效的存储这些重要的信息，这里专门设计了一个终端数据结构和一个上下文数据结构。对于某一特定的终端或上下文，相应的数据结构将在系统中占用一段内存单元，我们将其称之为终端数据区或上下文数据区。终端数据结构（或上下文数据结构）用于保存某一特定终端（或上下文）相关的 H.248 协议参数、进程交互和处理参数以及相应数据区自身的索引参数等。

对于终端数据结构，包含的参数主要有：

```
BYTE                bFlag;
BYTE                bCellIndex;
BYTE                bRelFlag;
BYTE                bCmdType;
WORD                wCrntEvent ;
PH248_PID_CSN       tRuler;
PH248_TID_ASSOCI    tTidTerm;
MC_D_SDP            tLocalSdp;
```

它们依次表示该终端数据区的存在标志，终端数据区索引，终端数据区释放标志，当前对该终端操作的命令类型，当前事件号，当前 H.248 模块对应的进程号信息，终端名信息以及本地的媒体描述。

终端数据结构名为 PH248_CELL。

对于上下文数据结构，包含的参数主要有：

```
BYTE                bFlag;
BYTE                bTidNum;
BYTE                bCreateFlag;
WORD                wEvent;
WORD                wContextIndex;
WORD                wCsEvent;
DWORD               dwContextId;
CSN                 tH248Csn;
PH248_CELL          tH248Cell[MAX_TID_IN_CTX];
WORD                wPrev;
WORD                wNext;
```

它们依次表示该上下文数据区的存在标志，该上下文所含终端的数目，该上下文数据区建立的标志，当前事件号，该上下文数据区的序号，当前业务事件号，ContextID，当前 H.248 模块对应的进程号信息，该上下文所包含的终端的相关信息以及 Hash 队列中前后上下文数据区的序号（关于 CA 进程 Hash 队列的介绍见 4.6.3 节）。

显而易见地，终端数据结构是上下文数据结构中的一部分。

上下文数据结构名为 PH248_CONTEXT。

4.5.2 CA 进程接收 CC 模块 Veinu 消息的处理设计

CA 进程收到 Veinu 消息后，首先根据设计的 Veinu 消息结构的规范性，判断该消息是否是一条合法的业务控制消息。如果该消息是合法的，则对该消息进行解析，以提取该消息所指示的业务类型和业务操作。

如果该 Veinu 消息指示，要让 MGW 上的终端创建呼叫，且该终端是此呼叫中第一个被加入的终端，CA 进程则相应的创建上下文数据区，由前述，终端数据区是包含在上下文数据区之中的，因此此时也创建了终端数据区。在呼叫

未释放之前，数据区将一直保存在系统内存之中。之后 CA 进程根据上下文和终端数据结构，将该 Vienu 消息带进的一些重要的参数写入相应数据区的内存单元之中。

如果该 Veinu 消息指示，要让 MGW 上的终端创建呼叫，但 CA 进程已经为该呼叫创建了上下文数据区（即已为对应呼叫完成了一个或若干个终端的加入操作），CA 进程则为此条消息操作的终端对象创建终端数据区，并将其加入已经创建的对应该呼叫的上下文数据区中。之后将消息带进的一些重要参数写入数据区对应的内存单元之中。

如果该 Veinu 消息指示，要对 MGW 上已经创建呼叫的终端的属性进行修改或进行呼叫的释放，则将该 Veinu 消息带进的一些重要的参数写入该呼叫创建时创建的终端和上下文数据区中对应的内存单元之中。

经过以上处理之后，CA 进程将该 Veinu 消息带进的参数依次转化为 H.248 协议的命令和参数，并将其保存在内存中的某一部分，最后将其首地址发送给 TTM 进程，TTM 之后接收这些信息并进行处理（见 4.3.5 节）。

4.5.3 CA 进程接收 TTM 进程消息的处理设计

CA 进程接收到 TTM 进程传送来的接口消息首地址后，首先找到该消息存储的数据区，并识别该条消息对应的 MGW 传送给 TTM 进程的 H.248 协议命令。如果命令名指示此消息来源于 MGW 的一条事务请求消息（如含 Notify 命令的事务请求），则将该消息中的参数取出，经过适当的变换，将其转换为 Veinu 消息的结构，发送给 CC 模块。如果命令名指示此消息来源于 MGW 对 TTM 进程发送的事务应答消息，首先同样的将该消息中的参数取出，经过适当的变换，将其转换为 Veinu 消息的结构，发送给 CC 模块；之后，CA 进程还要根据不同的命令名，进行相应的处理。比如，如果命令名是 Add，则将已经创建的上下文数据区加入上下文 Hash 队列之中（关于 Hash 队列的介绍见 4.6 节）；如果是 Subtract，则将命令操作的终端对象对应在 CA 进程的已创建的终端数据区删除，如果该终端是呼叫中最后一个被释放的终端，则还要删除 CA 进程对应该呼叫的上下文数据区，并将其从上下文 Hash 队列中删除。

4.6 Hash 队列算法在 H.248 模块设计中的应用

对于多呼叫并存的网络状态，CA 进程相应的也需要建立多个上下文数据区，

用于保存对应每一个呼叫的参数信息。如何方便高效的查找及操作上下文数据区以提供对呼叫的控制，是我们设计时需要关注的问题。对于 TTM 进程中 MGW 事务消息和 MSCe 事务消息的查找和操作，也存在同样的问题。考虑到多方面的因素，这里选择采取 Hash 队列算法对其进行相应的设计。

4.6.1 直接查找技术

设表的长度为 m，如果存在一个函数 i=i(k)，对于表中的任一元素的关键字 k，如果满足

（1）1≤i≤m ；

（2）对于任意的 k1≠k2，恒存在 i(k1)≠i(k2)。

则此表为直接查找表。表中函数 i=i(k)称为关键字 k 的映像函数。

由直接查找表的定义可以看出，直接查找表中各元素的关键字 k 与表项序号 i 之间存在一一对应的关系。因此，对直接查找表的查找，只需根据映像函数 i=i（k）计算待查关键字项目在表中的序号即可。

4.6.2 Hash 表

在直接查找技术中，要求映像函数能使得表中任意两个不同的关键字其映像函数值也不同，即在直接查找表中，不允许元素冲突的存在。但在一般的实际应用中，这一条是很难做到的，即往往会出现这样的情况，对于 k1≠k2，而 i(k1)=i(k2)，这就产生了冲突。Hash 表就是基于这种情况被引入的。

设表的长度为 m，若存在一个映像函数 i=i(k)，对于表中任一元素的关键字 k，满足 1≤i≤m ，则称 i(k)为关键字 k 的 Hash 码。

由这个定义可以看出，在 Hash 表中允许元素冲突的存在。如果在 Hash 表中没有冲突的存在，则 Hash 表就成为直接查找表。

由于 Hash 表中存在元素的冲突，采用什么样的方法妥善解决元素之间的重复问题成为设计时关注的对象。另外，在查找关键字为 k 的元素时，计算 Hash 码 i（k）的工作量又是影响查找效率的重要因素。

由以上分析，在实际构造 Hash 码时，要综合考虑如下两方面：

1.使表中的各关键字尽可能均匀的分布在 Hash 表中，以便减少冲突发生的机会；

2.使 Hash 码的计算要简单。

以上两个方面在实际应用中往往是矛盾的，即为了保证 Hash 码的均匀性比较好，其 Hash 码的计算就要复杂；反之，如果 Hash 码的计算比较简单，则其均匀性就比较差。在实际构造 Hash 码时，我们需要根据具体的情况，选择一个较为合理的方案。例如，当 Hash 表放在慢速的二级存储器中时（文件系统中往往是这种情况），由于存取一项所需的时间较长，因此，在这种情况下，应侧重于尽量减少 Hash 码的冲突，而 Hash 码的计算稍微复杂一些是无关紧要的；当 Hash 表放在内存时，则应使 Hash 码的计算尽量简单，虽 Hash 码冲突机会稍多一些，但在整体上考虑还是划算的。

由于 Hash 码的构造在很大程度上依赖于各关键字的特性，因此，一般很难给出一个普遍适用的方法，只能根据具体的情况设计和构造 Hash 码。常见的 Hash 构造方法有截断法、分段叠加法、除法、乘法等等，具体参见[41]。

4.6.3 Hash 队列在 CA 进程设计中的应用

由前所述，在 CA 进程的设计中，需要考虑多呼叫并行处理的网络状态。即在内存中，将会同时存在若干个已经创建的上下文数据区。如何方便的查找及操作 CA 进程的上下文数据区，是我们这里关注的重点。

在这里，我们参照 Hash 表算法的思想，引入 Hash 队列的概念。即对于 Hash 表中每一个索引引出一个队列，依次存取属于该索引对应队列的上下文数据区结构。

考虑到已经定义了上下文数据结构，为了方便起见，在设计 Hash 表时，表中每个索引对应的结构都为该结构。由于我们在结构中定义了 wPrev 和 wNext，因此可以方便的存取该上下文数据区对应的索引队列中前后上下文数据区的序号，通过这种方式，表中每个索引便引出了一串上下文数据区的队列，我们称之为上下文 Hash 队列。

对于某一特定的上下文数据区，我们可以在确定其属于哪个索引队列后，利用该数据区的序号、wPrev 和 wNext 信息，方便地在其所在的队列进行查找。另外，还可以对该队列进行数据区的插入及删除操作。

这里，正如前述 Hash 表中 Hash 码的构造方法一样，确定 Hash 表索引的方法是需要精心设计的。考虑到呼叫建立时的重要参数，以这些参数作为索引的计算关键字比较合理。经过选择，采用 ContextID 和网关节点号作为计算关键字，并采用除法构造方法构造 Hash 码。具体地，用 ContextID 加上网关节点号再除

以一个常数得到相应的 Hash 队列索引。这个常数根据对网络上并发呼叫统计的经验值得出，其为 Hash 表的长度。

当上下文创建时，CA 进程通过计算索引，将该上下文对应的数据区加入相应的 Hash 队列中。当某一呼叫结束时，CA 同样地计算出索引，并在相应的队列中找到相应的上下文数据区，将其从队列里删除。

通过这种方式，CA 进程有效地解决了并发呼叫时上下文数据区的操作。

4.6.4 Hash 队列在 TTM 进程设计中的应用

类似 CA 进程的上下文数据区，对于 TTM 进程中的 MGW 事务消息与 MSCe 事务消息，也采用 Hash 队列对其数据区进行保存。

在 TTM 进程中，构造三种 Hash 队列，用于保存进程需要保留的所有实时 MSCe 事务请求消息、MSCe 事务应答消息和 MGW 事务请求消息。类似于在 CA 进程中为上下文定义一个数据结构那样，在 TTM 进程中，也为每一种不同类型的消息定义一个数据结构，用于保存消息中的重要信息。对于一个对应特定消息类型的 Hash 队列，每一个 Hash 索引对应的结构为该类型消息的数据结构。类似上下文数据结构，通过在消息数据结构中定义这样的变量成员，即该消息相应队列中前后相邻消息数据区的序号，进而构造成一串逻辑上的消息队列。

同样的，对于某一特定的消息数据区，我们可以在确定其属于哪个队列、哪个索引后，利用该数据区的序号、前后消息的序号，方便地在其所在的队列进行查找。另外，还可以对该队列进行数据区的插入及删除操作。

在 Hash 码的构造上，参照 CA 进程的构造方法，采用 TransactionID 和网关节点号作为计算关键字，并采用除法方法构造 Hash 码。具体地，用 Transaction ID 加上网关节点号再除以一个常数得到相应的 Hash 队列索引。该常数依然依据经验值设定。

TTM 进程接收到来自 MGW 的事务请求消息后，如果发现这是一条新的事务请求，则计算队列索引，之后将该消息对应的数据区加入相应的 Hash 队列中。当 H.248 模块处理完该条消息并给 MGW 回事务应答后或 Pending 总时长超时，将其从相应消息队列中删除。MSCe 事务请求消息和事务应答消息，操作依此类推。

通过 Hash 队列的构造，成功地解决了 TTM 进程和 CA 进程中关键数据区

的查找和操作。

4.7 一个例子——同一 MGW 内终端基本呼叫建立的实现流程

在本节中，我们将以同一 MGW 内两个终端之间呼叫的建立为例，以系统处理流程的眼光，综合前面 TTM 进程和 CA 进程的设计介绍，完整的描述 MSCe 中 H.248 模块对呼叫过程的控制。通过这个例子，读者将对整个设计有全面系统的认识。

设需要建立呼叫的终端名分别为 T1 与 T2，它们分别连接在同一个 MGW 上。呼叫模型如图 4.8 所示。由图所示，当呼叫建立后，T1、T2 将位于同一个上下文中。这里我们假定 T1 对应着主叫基站的一个用户； T2 对应着被叫基站的一个用户。

处理呼叫建立的流程如下（为了简洁描述，以下默认主叫已摘机并拨号成功）：

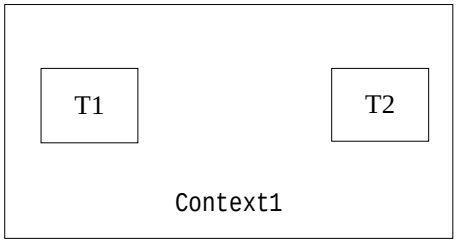

图 4.8 MGW 内终端基本呼叫模型

首先，CC 模块请求 MGW 为终端 T1 创建呼叫，此时 CC 模块发送相应的 Veinu 消息给 CA 进程。CA 进程收到此消息后会创建 CA 进程的上下文数据区以及该终端对应的终端数据区，之后将该消息转换为 H.248 协议命令及参数发送给 TTM 进程，其中命令名为 Add，包含的参数有：$（ContextID），T1（TerminationID），SendReceive（T1 的模式），以及相关的承载信息。TTM 进程接收相应的命令和参数后，进行 H.248 文本编码操作，将其转化为 H.248 文本消息，发送给 MGW。

MGW 给 TTM 进程回事务应答消息后，TTM 进程进行解码操作，将接收到

的事务应答消息转换为 H.248 协议命令和描述符，传递给 CA 进程。CA 进程接收后，将已经建立的上下文数据区加入相应的 Hash 队列中，并给 CC 模块回相应的 Veinu 响应消息。至此对 T1 终端的呼叫创建操作完毕。

T2 终端的呼叫创建过程同 T1 类似。首先，CC 模块请求 MGW 为终端 T2 创建呼叫，此时 CC 模块发送相应的 Veinu 消息给 CA 进程。CA 进程接收到相应的 Veinu 消息后，为 T2 创建终端数据区，并将该数据区加入到 T1 终端对应的上下文数据区中。之后将该消息转换为 H.248 协议命令及参数发送给 TTM 进程，其中命令名为 Add，包含的参数有：Context1（ContextID），T2（TerminationID），ReceivevOnly（T2 的模式），以及相关的承载信息。TTM 进程接收相应的命令和参数后，进行 H.248 文本编码操作，将其转化为 H.248 文本消息，发送给 MGW。

MGW 给 TTM 进程回事务应答消息后，TTM 进程进行解码操作，将接收到的事务应答消息转换为 H.248 协议命令和描述符，传递给 CA 进程。CA 进程接收后，给 CC 模块回相应的 Veinu 响应消息。

接下来，CC 模块发送让 T2 放震铃音的业务控制消息给 CA 进程。CA 进程接收到相应的 Veinu 消息后，依然将该消息中的一些参数信息填入已经创建的上下文数据区中。之后将该消息转换为 H.248 协议命令及参数发送给 TTM 进程，其中命令名为 Modify，包含参数有：Context1（ContextID），T2（TerminationID）。TTM 进程接收相应的命令和参数后，进行 H.248 文本编码操作，将其转化为 H.248 文本消息，发送给 MGW。

MGW 给 TTM 进程回事务应答消息后，TTM 进程进行解码操作，将接收到的事务应答消息转换为 H.248 协议命令和描述符，传递给 CA 进程。CA 进程接收后，给 CC 模块回相应的 Veinu 响应消息。

而后，CC 模块又发送让 T1 放回铃音的业务控制消息给 CA 进程，CA 进程接收到相应的 Veinu 消息后，将该消息中的一些参数信息填入已经创建的上下文数据区中。之后将该消息转换为 H.248 协议命令及参数发送给 TTM 进程，其中命令名为 Modify，包含参数有：Context1（ContextID），T1（TerminationID）。TTM 进程接收相应的命令和参数后，进行 H.248 文本编码操作，将其转化为 H.248 文本消息，发送给 MGW。

MGW 给 TTM 进程回事务应答消息后，TTM 进程进行解码操作，将接收到的事务应答消息转换为 H.248 协议命令和描述符，传递给 CA 进程。CA 进程接收后，给 CC 模块回相应的 Veinu 响应消息。

被叫摘机后，CC 模块发送业务控制消息给 CA 进程，使 T2 的模式更改为 SendReceive。CA 进程接收到相应的 Veinu 消息后，仍然将该消息中的一些参数信息填入已经创建的上下文数据区中。之后将该消息转换为 H.248 协议命令及参数发送给 TTM 进程，其中命令名为 Modify，包含参数有：Context1（ContextID），T2（TerminationID），SendReceive（T2 的模式）。TTM 进程接收相应的命令和参数后，进行 H.248 文本编码操作，将其转化为 H.248 文本消息，发送给 MGW。

MGW 给 TTM 进程回事务应答消息后，TTM 进程进行解码操作，将接收到的事务应答消息转换为 H.248 协议命令和描述符，传递给 CA 进程。CA 进程接收后，给 CC 模块回相应的 Veinu 响应消息。

至此，主被叫可以通话，呼叫建立过程完成。

H.248 模块关于基本呼叫释放的控制，也是通过和 CC 模块及 MGW 的依次交互来实现的。相对于呼叫建立的过程，呼叫释放则显得简单很多，只需对两个终端依次进行呼叫的释放并从上下文 Hash 队列中删除相应的上下文数据区即可完成，即从 MSCe 到 MGW，总共两个来回的交互。对应的 H.248 命令名为 Subtract。具体流程介绍，限于篇幅，这里从略。

除了可以控制连接在同一 MGW 上终端的基本呼叫过程，H.248 模块还可以控制连接在不同 MGW 上的两个终端的基本呼叫过程。相对于前者，其处理过程要复杂一些，但控制的原理都是基于 H.248 协议的。限于篇幅，我们这里也不作相应的介绍了。

由上所述，MSCe 中 H.248 模块完整的提供了对 MGW 侧各类型基本呼叫过程的控制，充分将 H.248 协议对于呼叫业务的控制应用到实际的下一代网络中来。

4.8 小结

在本章中，作者根据自身在中兴 3GCN 项目所承担的 H.248 相关科研和开发工作，具体介绍了在 CDMA2000 核心网网元 MSCe 中实现 H.248 的具体方法。文中首先介绍了系统的架构方案和分层结构。接着，作者详细分析了系统软件模块设计的方案，细致介绍了各模块的功能和模块设计的思想，提供了各模块之间交互的接口。之后，作者就 H.248 模块的设计作了十分详尽的分析和描述，其中包含 TTM 和 CA 两大核心进程的设计。在设计过程中，我们将 At-Most-Once

功能、三次握手机制、重传机制和临时响应的设计放在了 TTM 进程，将控制呼叫的重要数据区（终端数据区、上下文数据区）设置在 CA 进程，消息的编解码放在了 TTM 进程。通过这种设计，使得消息处理流程显得条理清晰，便于分析和处理。为了完成重要的编解码操作，在 TTM 进程中特意设计了相应的模块。为了能够迅速查找和控制呼叫，在 CA 进程引入了 Hash 队列，用于分配和处理上下文和终端数据区。为了方便查找和处理 H.248 消息，同样在 TTM 进程引入 Hash 队列，用于保存 MSCe 和 MGW 消息。这些可以确保最终设计的成功。在介绍进程设计过程中，文中采用了很多进程处理流程的描述方法，这样便于读者理解设计的思想。在下一章中，将给出设计的最终测试成果。

第五章　H.248 协议在 CDMA2000 中的实现成果

在上一章中，我们阐述了 H.248 协议在中兴 CDMA2000 系统核心网元 MSCe 中软件实现的设计方法。在本章中，将给出程序编写完成后，作者在真实网络环境中得到的一些重要数据结果，这些结果将充分证明，在 MSCe 中，实现了 H.248 协议的功能，从而有力的说明设计的最终成功，这无疑将对下一代网络的发展起到十分积极的作用。

5.1　测试环境简介

硬件环境，如图 5.1 所示：

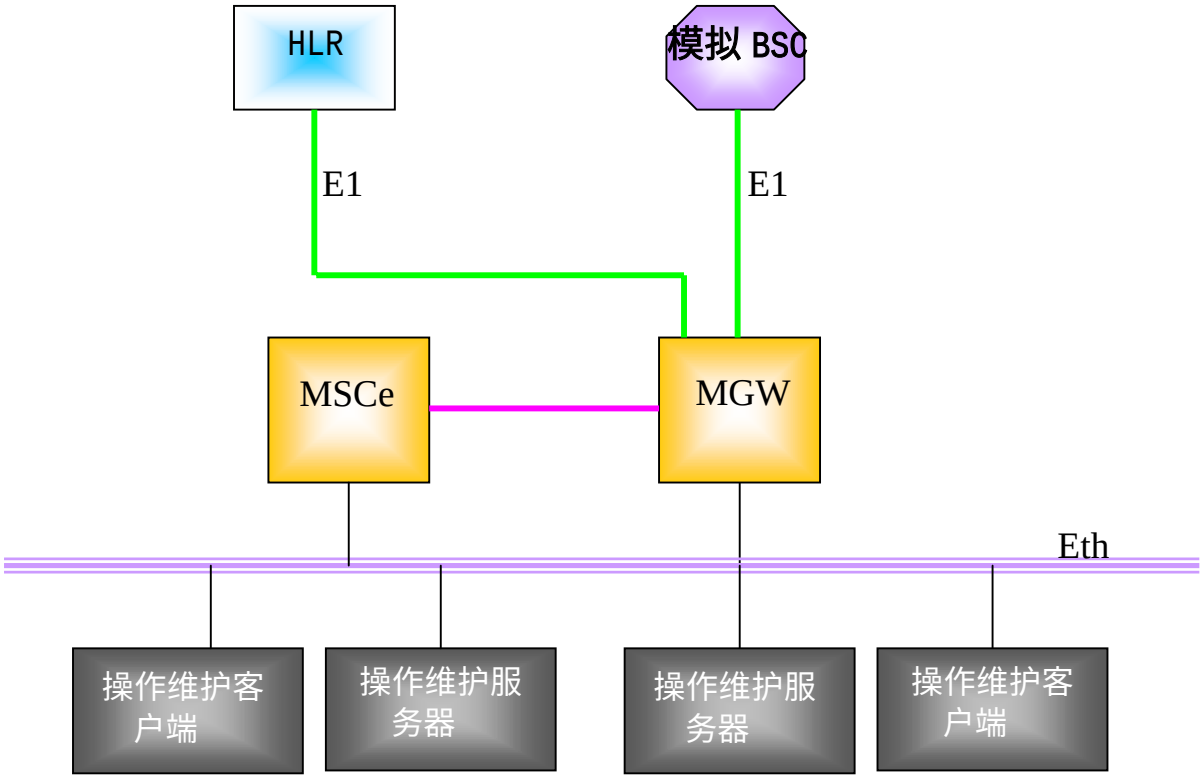

图 5.1　硬件测试环境

（1）被测对象——一个 MSCe 局

（2）一个 MGW 局

（3）一个 HLR 单模块局

（4）MSCe 对应的操作维护后台(一个服务器与一个客户端)

（5）MGW 对应的操作维护后台(一个服务器与一个客户端)

软件环境：

（1）VxWorks ——MSCe 和 MGW 的主处理器操作系统

（2）Windows2000 Advanced Server＋SP4 ——后台服务器操作系统

（3）SQL SERVER 2000＋SP3 ——后台数据库

5.2 后台信令跟踪结果

“信令跟踪”工具是在后台客户端上运行的“中兴公司统一网络管理平台”中的一个重要工具，是为开发和测试人员获取系统中实时处理的各种协议的消息文本而设计的。由于 H.248 模块设计时提供了对网管界面的接口，支持了包括“信令跟踪”在内的工具的使用，所以，利用“信令跟踪”，可以提取系统中实时的 H.248 文本编码消息。

可以人为的设计测试的用例，通过测试时从后台界面上获取的相应 H.248 文本编码消息，辅以实时具体的网络呼叫状况，来判断本模块软件设计和代码编写的最终成功性结果。

在信令跟踪工具中，只需要简单的设置一些模块和终端参数，伴随着呼叫的建立，即可获取各种终端（如固定电话，移动电话等）之间呼叫的 H.248 文本编码消息。

选择以下的集成测试用例，即连接在同一 MGW 局中的两个真实 TDM 终端之间呼叫建立和释放的过程。经过测试，在后台客户端上获得的信令跟踪界面如图 5.2 所示：

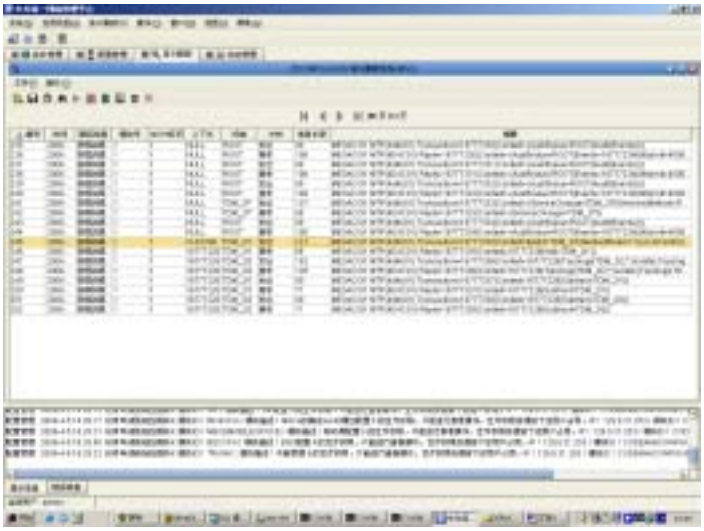

图 5.2　信令跟踪结果界面

双击该界面中的任意一行，即可得到该行对应的一条具体的 H.248 文本编码消息。

按呼叫的过程，将界面中几条核心的 H.248 文本编码消息依次摘取出来，并作相应的分析。

（1）

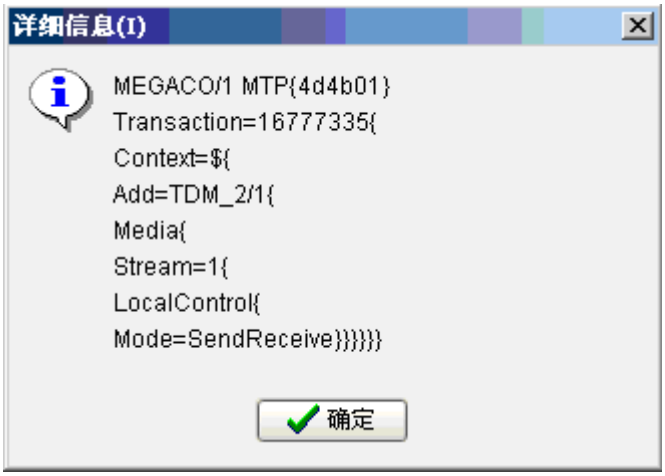

图 5.3　MSCe 请求主叫创建上下文的 H.248 文本编码消息

该条消息为，当 MGW 检测到主叫摘机并拨号成功上报 MSCe 后，MSCe 向 MGW 发送的请求主叫创建上下文（即主叫创建呼叫）的 H.248 文本编码消

息。其中主叫终端为 TDM_2/1，ContextID 为$（初次创建上下文），模式为 SendReceive，TransactionID 为 16777335。

（2）

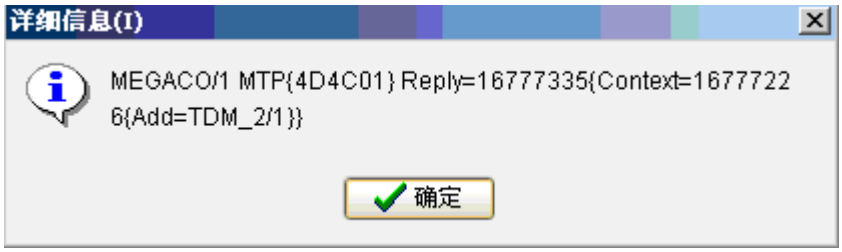

图 5.4　主叫创建上下文后向 MSCe 应答的 H.248 文本编码消息

该条消息为，MGW 收到 MSCe 请求主叫创建上下文的 H.248 协议消息后的应答消息。其中终端为 TDM_2/1，ContextID 为一有效整型值 16777226，表明 ContextID 已由 MGW 分配，应答事务的 ID 同上为 16777335。

（3）

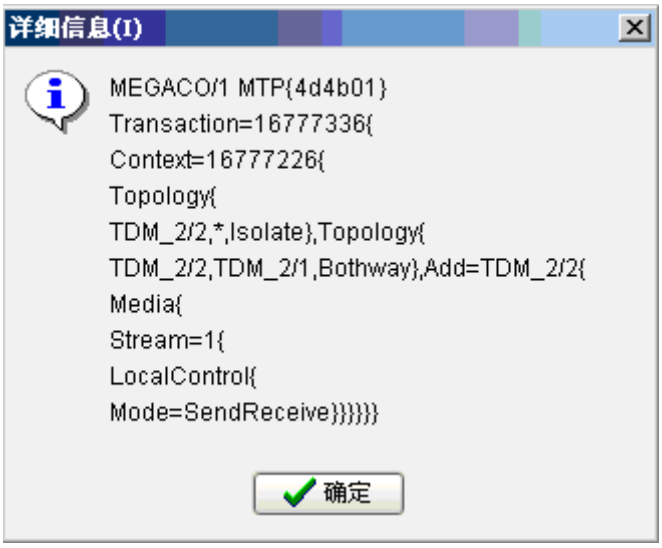

图 5.5　MSCe 请求被叫加入该上下文的 H.248 文本编码消息

该条消息为，MSCe 向 MGW 发送的请求被叫加入主叫所在上下文（即被叫创建呼叫）的 H.248 文本编码消息。其中被叫终端为 TDM_2/2，ContextID 为主叫所在上下文的 ContextID(16777226)，模式为 SendReceive，TransactionID 为 16777336。

（4）

图 5.6　被叫加入上下文后向 MSCe 应答的 H.248 文本编码消息

该条消息为，MGW 收到 MSCe 请求被叫加入主叫所在上下文的 H.248 协议消息后的应答消息。其中终端名为 TDM_2/2，ContextID 仍为 16777226，应答事务的 ID 同上为 16777336。

（5）

图 5.7　MSCe 请求主叫释放呼叫的 H.248 文本编码消息

该条消息为，当 MGW 检测到主叫挂机事件并上报 MSCe 后，MSCe 向 MGW 发送的请求主叫释放呼叫的 H.248 文本编码消息。其中终端名为 TDM_2/1，ContextID 为建立呼叫时分配的 16777226，TransactionID 为 16777337。

（6）

图 5.8　主叫退出呼叫后向 MSCe 应答的 H.248 文本编码消息

该条消息为，主叫退出呼叫后 MGW 向 MSCe 应答的 H.248 文本编码消息。其中终端名为 TDM_2/1，ContextID 为 16777226，应答事务的 ID 同上为 16777337。

（7）

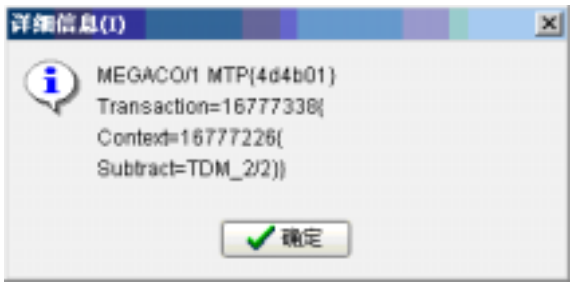

图 5.9　MSCe 请求被叫释放呼叫的 H.248 文本编码消息

该条消息为，MSCe 向 MGW 发送的请求被叫释放呼叫的 H.248 文本编码消息。其中终端名为 TDM_2/2，ContextID 为 16777226，TransactionID 为 16777338。

（8）

图 5.10 被叫退出呼叫后向 MSCe 应答的 H.248 文本编码消息

该条消息为，被叫退出呼叫后 MGW 向 MSCe 应答的 H.248 文本编码消息。

其中终端名为 TDM_2/2，ContextID 为 16777226，应答事务的 ID 同上为 16777338。

伴随着信令提取的过程，我们观察到两个 TDM 终端间也同步成功地完成了呼叫的建立、通话的过程以及呼叫的释放。将以上获取的 H.248 文本编码消息联系起来，经过分析，信令中的重要参数，如 MID、TransactionID、ContextID、TerminationID、终端模式等，伴随着整个过程完全正确。从获取消息的编码格式上来看，是完全符合 3.1 节所介绍的 H.248 文本编码规则的。由此，从这些方面，我们可以判定，在 MSCe 上，H.248 模块的软件成功实现了 H.248 协议的核心，即关于 MGW 上呼叫业务的控制。

5.3 小结

本章重点描述了网络测试环境，测试工具和最终得到的测试结果。利用信令跟踪工具可以使测试者非常方便的得到测试系统中实时的 H.248 文本消息，结合测试当时系统的呼叫状况，我们可以判别本模块软件设计是否成功。分析作者在该环境中得到的若干条 H.248 文本消息，伴随着呼叫建立、通话过程以及呼叫释放整个流程的成功，我们判定在 MSCe 上，H.248 模块的软件成功实现了 H.248 协议。

第六章　总结与展望

在本文中，作者首先介绍了下一代网络（NGN）中媒体网关控制器（MGC）和媒体网关（MGW）的概念，接着顺其自然的引入了本文的核心所在，即作为媒体网关控制协议的 H.248 的研究和应用工作。在国内这是一个全新的领域，在国际上，相关的研发也仅仅出于初期阶段，所以该项课题充满了实际应用价值和前景。

从本文的第二章，作者由浅入深，有条理的引入了 H.248 的重要概念，其中包括终端、上下文、命令、描述符与包、事务、消息等。这些都是很抽象的概念，通过对这些概念的深入分析和研究，对协议本身我们有了很层次的理解，这对于实际应用的开发起到至关重要的作用。

协议本身是抽象的，应用却又是另一层面上的事。基于这点，作者在第三章就协议的应用关键点做出了细致的研究。首先是应用协议时采用的编码方式，在国外，自协议诞生之日起，这点就一直存在争议。作者这里提出了现在相对较为认可的也是最为简单的编码方式，即文本编码方式。事实上，在实际实现协议过程中，正是使用了这样的方式。事实证明，这样的选择是成功的和有效的。其次，由于协议是承载在 IP 上的，尤其是如果承载基于 UDP，协议传输需要提供相应的机制确保传输的成功。这点在实际设计时非常关键。丢失或重复执行某条指令都可能会给系统带来不可挽回的后果。于是需引入 At-Most-Once 功能、三次握手机制、重传机制和临时响应，事实证明，这些方法可以确保系统消息传输和执行的可靠性。

有了这些应用的准备后，在真实网络中实现该协议的工作从容展开。首先需针对网络的特点设计相应的软件模块。研究的背景是作者从事的 ZTE 3GCN-MSCe 项目。根据网络设计的层次，软件模块被分解成 CC、CA、TTM、OMC、DB 等进程模块，其中 H.248 模块包含 CA 和 TTM 两大进程。在设计过程中，将 At-Most-Once 功能、三次握手机制、重传机制和临时响应的设计放在了 TTM 进程，将控制呼叫的重要数据区（终端数据区、上下文数据区）设置在 CA 进程，消息的编解码放在了 TTM 进程。通过这种设计，使得消息处理流程显得条理清晰，便于分析和处理。为了完成重要的编解码操作，在 TTM 进程中特意设计了相应的模块。为了能够迅速查找和控制呼叫，在 CA 进程引入了 Hash

队列，用于分配和处理上下文和终端数据区。为了方便查找和处理消息，同样在 TTM 进程引入 Hash 队列，用于保存 MSCe 和 MGW 消息。这些确保了最终设计的成功。在设计的过程中，需要和其他模块进行交互，我们设计了许多接口，以便于完成相应的交互。从整个设计的过程来看，考虑十分周密，以确保最终设计的成功。这些设计在第四章都进行了详尽的描述。

本文的第五章给出了作者在真实网络环境中测试得到的 H.248 文本消息。通过分析所得的消息，伴随着呼叫建立，通话过程以及呼叫释放整个流程的成功，我们可以证实在 3GCN-MSCe 中成功实现了 H.248 协议的功能。

这是一个极为振奋人心的结果！这一研究成果填补了同期国内 NGN 相关空白，它与其它研发的技术相综合，使中兴的软交换产品处于国内乃至国际领先的地位。于此同时，该研发的成功也将推动全球范围内 NGN 的研发，世界范围内必将掀起新一波关注 NGN 应用的热潮。由于 H.248 是下一代网络的核心协议，其本身也必将随着时间的推移和业务的不断更新，同步地发展，相应的应用发展前景十分广阔。目前，虽然还有许多基于 H.248 的特殊业务和未来可能出现的新业务的开发还有待研究人员去实践和完成，但是我们完全有理由有信心的展望，在不久的将来，这些问题会逐步得到解决。对 NGN 的研究和应用必将推动全球网络的全面变革和发展，NGN 的发展必将会给我们的生活带来全新的体验！

参考文献

[1] C. Groves, RFC3525，"Gateway Control Protocol"，2003 年 6 月

[2] ITU-T Recommendation H.248.1 Gateway Control Protocol: Version 3，2005 年 10 月

[3] 中国电信集团，中国电信 H.248 标准，2003 年 12 月 31 日

[4] RFC1819，Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+，2000 年 4 月

[5] Rose, M. and D. Cass, RFC 1006, "ISO Transport Service on top of the TCP, Version 3", STD 35, May 1987

[6] Brander, S., RFC 2026, "The Internet Standards Process – Revision 3", BCP 9, October 1996

[7] Bradner, S., RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, March 1997

[8] Crocker, D., Ed. and P. Overell, RFC 2234, "Augmented BNF for Syntax Specifications: ABNF", November 1997

[9] Handley, M. and V. Jacobson, RFC 2327, "SDP: Session Description Protocol", April 1998

[10] Kent, S. and R. Atkinson, RFC 2402, "IP Authentication Header", November 1998

[11] Kent, S. and R. Atkinson, RFC 2406, "IP Encapsulating Security Payload (ESP)", November 1998

[12] Postel, J., RFC 768, "User Datagram Protocol", STD 6, August 1980

[13] Postel, J., RFC 791, "Internet Protocol", STD 5, September 1981

[14] Postel, J., RFC 793, "Transmission Control Protocol", STD 7, September 1981

[15] Simpson, W., Ed., RFC 1661, "The Point-to-Point Protocol (PPP)", STD 51, July 1994

[16] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, RFC 1889, "RTP: A Transport Protocol for Real-Time Applications", January 1996

[17] Schulzrinne, H. and G. Fokus, RFC 1890, "RTP Profile for Audio and Video Conferences with Minimal Control", January 1996

[18] Kent, S. and R. Atkinson, RFC 2401, "Security Architecture for the Internet Protocol", November 1998

[19] Deering, S. and R. Hinden, RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification", December 1998

[20] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, RFC 2543, "SIP: Session Initiation Protocol", March 1999

[21] Greene, N., Ramalho, M. and B. Rosen, RFC 2805, "Media Gateway Control Protocol Architecture and Requirements", April 2000

[22] R. Sparks, RFC 3515, “The session Initiation Protocol (SIP) Refer Method”, April 2003

[23] A. B. Roach, B. Campbell, RFC 4662, “A Session Initiation Protocol (SIP) Event Notification Extention”, August 2006

[24] J. Rosenberg, P. Kyzivat, RFC 4596, “Guildlines for usage of the Session Initiation Protocol (SIP) Caller Preferences Extention”, July 2006

[25] A. Johnston, Avaya, RFC 4579, “Session Initiation Protocol (SIP) Call Control – Conferencing for User Agents”, August 2006

[26] J. Rosenberg, RFC 4575, “A Session Initiation Protocol (SIP) Event Package for Conference State”, August 2006

[27] H. Sinnreich, RFC 4504, “SIP Telephony Device Requirements and Configration”, May 2006

[28] K. Morneault, RFC 4666, “Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) – User Adaption Layer (M3UA)”, September 2006

[29] G. Sidebottom, RFC3332, “Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) – User Adaption Layer (M3UA)”, September 2002

[30] L. Ong, RFC3286, “An Introduction to the Stream Control Transmission Protocol (SCTP)”, May 2002

[31] Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation (X. 680)，1997 年

[32] Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER) (X. 690)，1994 年

[33] 糜正琨，《软交换组网与业务》，人民邮电出版社， 2005 年 9 月

[34] 陆立，《NGN 协议原理及应用》，机械工业出版社，2004 年

[35] Andrew S. Tanenbaum，Computer Networks (Third Edition)， 2004 年 3 月

[36] Comer / D.E.，Internetworking With TCP/IP Vol Ⅰ:Principles，Protocols，and Architectures Fourth Edition，2001 年 5 月

[37] Internetworking with TCP/IP—Design, Implementation and Internals (Volume 2)，2002 年 1

月

[38] Kopajtic, O., Lusa, R. ， H.248-implementation and interoperability issues ， Telecommunications, 2003. ConTEL 2003. Proceedings of the 7th International Conference on, Volume 2, 11-13, June 2003: 677 - 680 vol.2

[39] de Souza Pereira, J.H., Guilherme, J., Rosa, P.F.，Development of MGs in a next generation network with MEGACO/H.248 support，Networks, 2004. (ICON 2004). Proceedings. 12th IEEE International Conference on, Volume 1, 16-19, Nov. 2004: 239 - 243 vol.1

[40] Liao Wanjiun, Chang Jen-Chun, Li V O K. Application-Layer conference trees for multimedia multipoint conferences using megaco/H.248, Multimedia, IEEE Transactions on, Volume 7, Issue 5, Oct. 2005: 951 – 959

[41] 徐仕良，朱明方，《软件应用技术基础》，清华大学出版社，2000 年

致谢

在本论文即将完成之际，首先要感谢我的导师仰枫帆教授，能够在仰老师的指导下完成研究生的学习，是我今生十分幸运的事。他为人正直，学识渊博，平易近人，但是给我最深印象的，也是我终生受益的地方，就是他对待学术研究时一丝不苟、踏踏实实的精神，这种难能可贵的精神深深地感染着我，熏陶着我，成为我获得研究生学习成功的最大支柱。在这两年多的时间里，在通信领域，无论是传输技术还是网络技术，仰老师都给了我很多非常宝贵的指导，他渊博深厚的专业知识使我研究生阶段的学习无比充实。除此之外，仰老师看待人生，看待世界的观点更是这么多年来任何人都不能带给我的，在他的指导和关心下，我明确了人生的取向。仰老师教会我的一切将是我终身的财富，我将在以后的学习和工作中，努力拼搏，为人类的科学技术发展贡献出自己的一份力量！

其次，我想感谢中兴通讯有限公司所有帮助过我，和我一起工作战斗过的领导和同事，和他们共事让我学到了很多书本上没法学到的东西。

论文的完成还离不开南京航空航天大学信息科学与技术学院通信与信息系统专业的全体老师和同学们的大量帮助。特别感谢他们对作者在学习上的指导和生活上的帮助。

在论文撰写过程中，父母在生活上给予了我巨大的关怀，使我没有后顾之忧，得以全身心地投入到研究工作之中去。他们对我的关爱更多地体现在行动和目光中。在此向我的父母表示衷心的感谢。

还要感谢参加我论文评审和答辩的专家和老师，感谢所有默默无闻的帮助我，为我创造良好学习环境和科研环境的善良的人们，正是你们的工作才造就了我的成长。

最后，祝南航信息科学与技术学院的所有老师、同学们一帆风顺。

陈　鑫

2006年12月

在学期间的研究成果及发表的学术论文

1．陈鑫，仰枫帆，“ Research and Implementation of H.248 ”,南京航空航天大学第八届研究生学术会议论文集（光盘），2006年11月