摘 要

传统电信网、计算机网和有线电视网络都是针对特定应用的信号特点、信号形式而设计的，因而在通用性上与现代要求存在一定的差距。对于电信运营商，由于软交换的出现，促成了电话网和互联网的融合。电话网通过改造升级适应了IP业务的发展，IP技术通过媒体网关和信令网关，能方便地提供话音和智能网业务，即实现电话网和互联网互联互通。其中，媒体网关主要负责媒体格式的转换及PSTN和IP两侧通路的连接，媒体网关由软交换核心设备来控制。它们之间通过H.248协议（媒体网关控制协议）完成通信。

H.248协议是软交换中的关键技术，它可以被看作是连接软交换系统的大脑（软交换核心设备）和四肢（媒体网关）的神经。论文以网络管理系统的开发为背景，全面地分析了H.248协议并结合网络管理系统的特点完成了网络管理系统中H.248消息解码模块的设计与实现。同时，针对网络融合过程中网络中数据量越来越大的特点，在网络管理系统数据处理流程中采用了一种快速处理数据的方法。

论文首先简述了VoIP（Voice Over Internet Protocol）技术以及软交换技术，通过对网关分离详细描述说明了媒体网关控制器与媒体网关的功能与作用，分析比较了两种媒体网关协议（MGCP与H.248）说明了H.248协议在VoIP中的重要作用。然后对H.248协议体系以及H.248文本消息编码规范进行了分析与研究，并在此基础上结合网络管理系统体系结构，提出了解码模块的具体设计方案。

在软件模块设计中采取分层次模型和模块化设计原则，使得整个软件具有良好的可移植性和可扩展性。论文着重阐述了H.248消息的解码模块的设计与实现，通过对文本消息的分析，并结合消息结构和构成机制给出了分层解码系统结构以及具体实现方式，直至实现H.248消息的解码。最后论文展示了网络管理系统中H.248协议解码模块的实际应用并对结果进行了分析，从而证实了该模块能够适应网络管理系统的要求，很好地为系统的呼叫跟踪，呼叫统计以及Remon等功能提供所需的信息。

关键词： VoIP，软交换，网络管理系统，H.248协议，解码

Abstract

Traditional telecommunications networks, computer networks and cable networks are directed at a particular designed for specific applications, other applications can not be good to complete the requirements. The telecom operators face how to integrate their own two network (telephone network and the Internet)to the one, but the emergence of softswitch contributed to this fuse: the upgrading of the telephone network to adapt to the development of IP services, IP provides carrier-class voice and intelligent network services. Both through the media gateway and signaling gateway to interoperability. Media Gateway is mainly responsible for media format conversion and the PSTN and IP connections on both sides of access, The media gateway is control by the core equipment from the softswitch, and communicate through H.248 protocol (Media Gateway Control Protocol)

H.248 protocol is the key technology in softswich, it can be seen as nerves which connect the brain of soft switching system (soft switch core equipment) and limbs (Media Gateway).The thesis use the exploiture of network management system as a background, and comprehensive analysise of H.248 protocol then combining the characteristics of network management system to complete the design and implementation of H.248 message decoding module in network management system. At the same time it introduce a fast processing data method in the growing amount of data network management system flow during network fuse.

Firstly outlines the VoIP (Voice Over Internet Protocol) technology and soft-switching technology in the paper, detailed description the function and role of media gateway controller and media gateway through the separating of gateway, through the analyzing and comparing of two media gateway protocol (MGCP and H.248) drawn to the conclusion that H.248 protocol is very important role in VoIP. Secondly do the analyzing and studying of H.248 protocol system and H.248 text messages coding standard, based on the analysis and combine the architecture of network management system to propose the decoding module design. The software module design using a hierarchical model and the modular design, which makes the software has good portability and scalability. Through the analysis of text messages and combined with message structure, the paper advance a method that can hierarchically decode H.248 message protocol. Finally showed the practical application of H.248 protocol decoding module in network management

system in the paper, then analyzed the result and make sure that it can meet the requirements of network management system, and provide the well required information for the system's function ,such as: call tracking, call statistics , Remon and so on.

Keywords: VoIP, soft switching, network management systems, H.248 protocol, decoding

第一章 绪 论

1.1 研究背景

随着电话网、计算机网和有线电视网趋于融合，以软交换技术为核心的 NGN 将三大网络合并在一起，使网络可以基于 IP 在各自数据应用平台上提供多媒体信息服务，在网络层面上实现网间的互联互通，在业务层面上实现各种业务互相渗透和交叉，承载多种业务，让已经具有基本能力的各种网络系统进行适当的业务交叉，充分发挥各类网络资源的技术潜力，实现网络资源最大程度的共享。近年来随着数据通信和 IP 业务的迅速发展，以分组交换为基础的 IP 网络由于其简单和开放，得到了越来越广泛的应用。已有专家预测，未来的各项电信业务将统一在 IP 网络上。传统电话网将不可避免地过渡到以数据业务特别是 IP 业务为中心的融合的 NGN(下一代网络)。NGN 将以 IP 网络为核心，通过以 TCP/IP 为基础的分组交换网络，承载起包括话音在内的所有通信类业务。

1.2 VoIP 概念

VoIP（Voice Over Internet Protocol），即 IP 上传送语音，实现了语音在 IP 上的实时传送，为了有效地利用 IP 带宽资源，通常在传送之前先要对语音数据进行压缩处理。VoIP 的基本原理是：通过语音的压缩算法对原始的语音数据进行压缩处理，然后把这些压缩过的语音数据按实时传输的要求进行打包，经过 IP 网络把数据包送至接收地，再把这些语音数据包按原来的时间次序进行串行化处理，并将数据包中的语音数据进行解压缩处理，恢复出原始的语音信号，从而实现在 IP 上实时传送语音的目的。

1.2.1 基本原理

传统的电话网是以电路交换方式传输语音，所要求的传输宽带为64kbit/s。为了充分地利用网络带宽资源，VoIP通信中通常根据实际使用的要求采用各种压缩算法对原始的语音数据进行压缩处理，常用的有G.723.1、G.729等；然后才用网络技术将压缩后的语音数据进行打包处理，在运输层采用无连接的UDP的方式，其主要目的是为了保证语音数据的传输的实时性，然后将UDP数据报交由IP分组网络来进行传送；在将压缩数据传送至UDP之前，先用RTP/RTCP协议对压缩数据进行处理，RTP协议用以传送语音数据，而RTCP协议用以传送语音数据的控制信息。如图左侧所示。接收侧收到数据后，其处理过程与发送侧相反，如图右侧所示。

通常每个网关，既要发送语音数据，又要接收语音数据，所以包含图1.1中的所有功能。实际上，网关的基本功能就是完成语音数据的压缩、解压缩和打包、解包处理。实际的传输网络可能是非常简单的局域网，也可能极其复杂的广域网，传输途径中包含各种网络设备，如网络交换机、路由器、ATM交换机、SDH等。

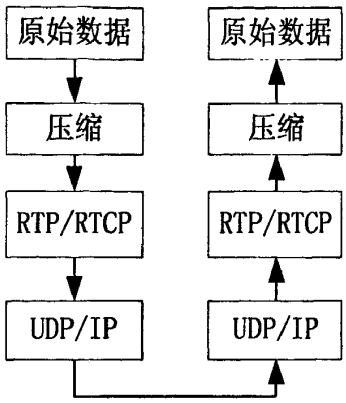

图 1.1 语音数据处理示意图

1）语音的压缩编码和解码

模拟的语音信号首先要进行模拟－数据转换，也就是对模拟语音信号进行量化处理，可以使用各种语音编码方案来实现，目前通常采用的语音编码标准主要有ITU-T G.711（含G.711A和G.711 μ 两种）。这种语音编码标准的数据速率为64kbit/s，为了充分地利用IP网络的带宽资源，通常会根据具体的应用场合采用合适的语音压缩编码，将原始的语音数据进行压缩处理。最常用的压缩编码有ITU-T G.723.1和ITU-T G.729等，其中ITU-T G.723.1的数据速率为5.3kbit/s或6.3kbit/s，而ITU-T G.729的数据速率为8 kbit/s，可见，经过压缩处理可以在很大程度上提高网络带宽的利用率。在接收端，有一个相应的语音解压缩的处理过程，为语音压缩的逆过程。最后将解压缩后的语音数据经数字－模拟转换等处理后输出。

2）RTP/RTCP

经压缩的语音数据采用RTP/RTCP封装后，再利用UDP来进行传输。RTP/RTCP为音频、视频等实时数据提供端到端的传递服务，可以向接收端传送恢复实时信号必须的定时和顺序信息，并向收发双方提供QoS检测手段。RTP/RTCP实际上包含两个协议：RTP（实时传送协议－Real-time Transport Protocol）和RTCP（实时传送控制协议－Real-time Transport Control Protocol）。其中RTP本身用以传送实时数据，接收端可以利用RTP包含的信息来正确地重组原始信号，如实现压缩编码识别、排序、丢包补偿等功能；而RTCP用以传送实时数据传送的质量参数，提供QoS监视机制。但RTP/RTCP本身并没有保证QoS的机制，它必须借助于资源预留协议等手段。

3）传送

RTP 封装后语音数据的经由 UDP/IP 来进行传输。由于 UDP 提供的是无连接的服务，容易产生网络丢包等问题，并且不能保证包传输的次序，这些问题可以利用 RTP/RTCP 协议部分得以解决。接着数据包通过数据链路层和物理层封装，并经网络设备送至目的端。目的端接收后进行源端的逆过程，最后输出语音信号。

IP 电话网总体框架[2]包括 IP 电话网关、IP 承载网、IP 电话管理层面以及电路交换网接入部分构成总体结构如图 1.2 所示。

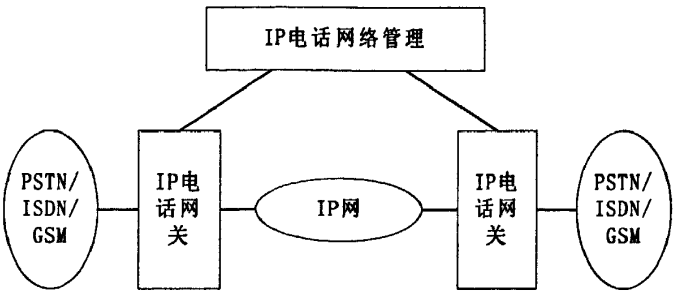

图 1.2 IP 电话网总体框架

- IP 承载网络：用于传送 IP 电话的承载网，其可以是公网也可以是专用网络。
- IP 电话网关：主要功能是完成来自 PSTN 的语音业务流的编码和解码并将压缩编码的话音业务流打包，通过 IP 承载网络传送给目的地网关，它位于 IP 网与传统的电路交换网之间，实现电路交换到分组交换的双向转换。即可以实现两个传统电话用户之间经过两次 IP 电话网关经过 IP 网络来实现通信，同时也可以实现多媒体 PC 用户与传统电话用户之间进行通信，这是仅仅需要经过一个 IP 电话网关即可。因此 IP 电话网关就相当于协议转换以及数据转换。
- IP 电话网络管理：负责用户的接入认证、地址解析、计费和结算等。

1.2.2 常见协议

为了实现上述的语音数据的通信过程，需要有呼叫控制规范来实现呼叫的接续。VoIP的各种协议就是实现呼叫控制的技术规范。

H.323协议是由ITU-T制定，并已经在业界得到广泛的应用。H.323协议是一个协议族，包含RAS、Q.931、H.245等一系列的协议，RAS协议用于呼叫接入控制等功能，Q.931协议用于实现呼叫控制，而H.245协议用于媒体信道控制。H.323协议采用的是类似于传统电信系统的层次架构，容易实现系统的分层。协议的体系结构非常完整，但整个系统比较复杂，功能比较强大，但实现呼叫的功能不灵活。

SIP协议是另一个主要的VoIP体系。与H.323协议不同，SIP协议采用的是客户机/服务器（C/S）结构，定义了各种不同的服务器和用户代理，通过和服务器之间的

请求和响应来完成呼叫控制。SIP协议的呼叫流程比较简单灵活，系统的可扩展性较好。

MGCP协议实际上是一个补充的协议。在H.323协议和SIP协议中的网关设备，不仅要执行媒体格式的变换，如压缩和解压缩、RTP打包与解包等功能，而且还要进行信令的转换，在IP网络侧执行H.323或SIP协议，在PSTN侧执行电路交换的信令。这样网关的功能变得非常复杂，限制了每个网关设备的容量；而且，随着应用的不断普及，将有更多的网关终端进入用户，这类网关设备的由于成本较高，并存在网络安全等问题，严重影响VoIP系统应用。MGCP协议就可以解决这些问题，其基本思想就是将媒体变换功能和网关控制功能相分离。使网关只承担简单的媒体变换功能，称为媒体网关（MG：Media Gateway），复杂的网关控制功能则由网关之外的独立的控制实体来执行，该实体称为呼叫代理（CA：Call Agent），两者之间的接口就采用MGCP协议来进行交互。

H.248 协议与 MGCP 类似，也是一种媒体网关控制协议，与 MGCP 协议相比，H.248 协议可以支持更多类型的接入技术，并支持终端的移动性，除此以外，H.248 协议最著名之处在于比 MGCP 所允许的规模更大，更具灵活性，H.248 协议通过增加许多 Package 的定义来对协议的功能进行扩展。H.248 协议是在 MGCP 协议的基础上发展而来的，随着 H.248 协议的不断完善，将逐渐取代 MGCP 协议成为媒体网关控制的主要标准。

1.3 电话网与互联网融合的纽带——软交换技术

软交换的概念最早起源于美国企业网应用。在企业网络环境下，用户可采用基于以太网的电话，再通过一套基于 PC 服务器的呼叫控制软件（Call Manager、Call Server），实现 PBX(Private Branch Exchange，用户级交换机)功能（IP PBX）。对于这样一套设备，系统不需单独铺设网络，而通过与局域网共享来实现管理与维护的统一，综合成本远低于传统的 PBX。由于企业网环境对设备的可靠性、计费和管理要求不高，主要用于满足通信需求，设备门槛低，许多设备商都可提供此类解决方案，因此 IP PBX 应用获得了巨大成功。

同时，传统的电路交换网络的综合运营成本很高，运营商非常希望能够寻求到一种替代产品与技术。受到 IP PBX 成功的启发，业界提出了这样一种思想：将传统的交换设备部件化，分为呼叫控制与媒体处理，二者之间采用标准协议（MGCP、H.248），呼叫控制实际上是运行于通用硬件平台上的纯软件，媒体处理将 TDM 转换为基于 IP 的媒体流。于是，SoftSwitch（软交换）技术应运而生，由于这一体系具有伸缩性强、接口标准、业务开放等特点，发展极为迅速。电路交换网络与软交

换网络如图 1.3 所示。

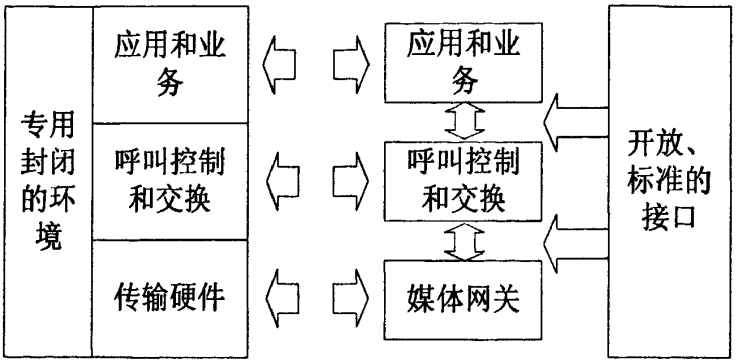

图 1.3 电路交换与软交换

软交换是 NGN 的控制功能实体，为 NGN 提供具有实时性要求的业务的呼叫控制和连接控制功能，是 NGN 呼叫与控制的核心，是电路交换网与 IP 网的协调中心，通过对各种媒体网关的控制实现不同网络之间的业务层融合[5]。软交换是基于“网络就是交换”的理念，是基于软件的分布式交换 / 控制平台，将呼叫控制功能从网关中分离出来，利用分组网代替交换矩阵，开放业务控制、接人和交换间的协议，从而实现异构网络的互通，方便网上多种业务的接入。

软交换的核心含义就是将呼叫控制功能从媒体网关中分离出来（即控制与承载分离），通过软件实现基本呼叫控制功能（包括呼叫选路、管理控制、连接控制和信令互通），从而实现呼叫传输与呼叫控制的分离，为控制、交换和软件可编程功能建立分离的平面，采取分布交换、分布业务智能、集中网络智能的体系结构，智能集中与网络，接入设备（媒体网关）只完成最基本的语音压缩编码等功能。

作为 NGN 网络的一项重要技术，软交换概念一经提出，很快便成为 NGN 最为活跃和热门的话题，软交换的引入具有重要的现实意义：

1) 从用户的角度

软交换可以实现跨域控制，可以对接入层面丰富多样的设备进行控制和管理，为用户提供话音、数据和视频业务，以及其它融合业务。

随着通信网络的发展趋势从技术驱动向业务驱动的转变，业务发展逐渐成为关注焦点。软交换向 ISV（独立软件开发商）、ISP、ICP、ASP（应用业务提供商）提供开放的业务接口 Open API，而 ISV/ISP/ICP/ASP 往往比软交换提供商更熟悉最终用户的业务需求，因此 ASP/ISP/ICP 提供的业务更贴近用户、贴近生活。此外，大量 ASP/ISP/ICP 参与竞争的结果，可加快推出多样、先进、经济和实用的业务。

2) 从运营商的角度

作为 NGN 核心部件，软交换可独立开发、生产和采购，其开发成本、生产成本和采购成本相对较低，从而使运营设备的投资相应降低。对于新运营商来说已无

法通过新建 PSTN 交换局同现有运营商进行市话业务竞争，通过软交换则可以轻易进入市话业务。

软交换的出现，可使三网在网络层面实现网间互联互通，在业务层面实现业务的渗透和交叉，使基于统一分组网络承载多种业务成为可能。运营商可终结三网分别投资的局面，通过一个融合的网络为用户同时提供话音、数据和多媒体业务，实现国际电联提出的“通过互联互通的电信网、计算机网和电视网等网路资源的无缝融合，构成一个具有统一接入和应用界面的高效率网路，使人类能在任何时间和地点，以一种可以接受的费用和质量，安全的享受多种方式的信息应用”的目标。

3) 从数据网络的角度

几年前的网络泡沫导致投资过度、带宽冗余，而软交换技术则可利用这些冗余带宽迅速部署 NGN 业务，同时软交换 Open API 还可为运营商提供强大的业务生成能力。

软交换是多种逻辑功能实体的集合，它提供综合业务的呼叫控制、连接和部分业务功能，是下一代电信网语音/数据/视频业务呼叫、控制、业务提供的核心设备。

软交换技术的主要设计思想为业务、控制、传送、接入分离，实体之间通过标准协议进行连接和通信，下面介绍软交换的功能：

1) 呼叫控制和处理

为基本呼叫的建立、维持和释放提供控制功能，包括呼叫处理、连接控制、智能呼叫触发检出和资源控制等；

接收来自业务交换功能的监视请求，并对与呼叫相关的事件进行处理，接收来自业务交换的呼叫控制相关信息，支持呼叫的建立和监视；

支持两方或多方呼叫控制功能，提供多方呼叫控制功能，包括多方呼叫特殊逻辑关系、呼叫成员的加入、退出、隔离、旁听和混音控制等；

识别媒体网关报告的摘机、拨号和挂机等事件，控制媒体网关向用户发送音信号，如拨号音、振铃音、回铃音等，满足运营商的拨号计划。

2) 协议功能

软交换是一种开放和多协议实体，采用标准协议与各种媒体网关、终端和网络进行通信，包括 H.248、SCTP 、ISUP、TUP、INAP、H.323、RADIUS、SNMP、SIP、M3UA、MGCP、BICC、PRI、BRI 等。

3) 业务提供

提供 PSTN/ISDN 交换机业务，包括基本业务和补充业务；

可与现有智能网配合，提供现有智能网所能提供的业务；

可与第三方合作，提供多种增值业务和智能业务。

4) 互通功能

可通过信令网关实现分组网与现有七号信令网的互通；

可通过信令网关与现有智能网互通，提供多种智能业务；

允许 SCF 控制 VoIP 呼叫，并对呼叫信息进行操作（号码显示等）；

可通过互通模块，实现与现有 H.323 体系的 IP 电话网的互通；

可通过互通模块，采用 SIP 协议实现与 SIP 网络体系的互通；

可实现与其他软交换的互通互连，采用 SIP、BICC 等协议；

可提供 IP 网内 H.248 终端、SIP 终端和 MGCP 终端之间的互通；

可通过综合媒体网关 UMG 实现与无线网络 PLMN 的互通。

5) 资源管理

提供资源管理功能，对系统中的各种资源进行集中管理，如资源的分配、释放和控制。

6) 计费功能

具有采集详细话单和复式计次的功能，可根据运营需求将话单传送至计费中心。同时具备 Radius 计费功能和智能计费功能。

7) 认证/授权

软交换与认证中心连接，可将辖区内的用户、媒体网关信息送往认证中心，进行认证与授权，防止非法用户或设备接入。

8) 地址解析

完成 E.164 地址至 IP 地址、别名地址至 IP 地址的转换，同时可完成重定向功能。

9) 语音处理

软交换控制媒体网关可选择语音压缩算法，包括 G.711、G.729、 G.723 等，软交换向媒体网关提供语音包缓存区，减少抖动对语音质量的影响。

1.4 本文主要研究的内容

不论在 NGN 网络测试仪器中还是在网络管理系统中要完成测试或者监测的功能都是建立在解码的基础之上的，因此协议的解码是实现这些功能的基础。本论文正是基于此在分析了 H.248 协议规范及编码之后，根据 H.248 消息结构重点介绍消息分层解码的方式设计和实现了 H.248 协议消息的解码。本论文主要来源于泰克公司网络管理系统。

全文共分为六章，各个章节内容安排如下：

第一章：概述 VOIP 和软交换技术以及他们之间的联系。并介绍了选题目内容。

第二章：简单说明了什么是网关分离，以及媒体网关的产生。介绍并比较了媒

体网关的两种协议 MGCP 和 H.248。重点介绍了 H.248 协议规范及其特点，分析了 H.248 消息的结构。

第三章：在该章节中介绍了网络管理系统，并根据网络管理系统数据处理流程网络管理系统中解码模块采用并行处理数据的方法从而提高效率。

第四章：阐述了解码模块采用分层的思想来设计，简要说明了分层带来的优点。详细描述了解码模块中各层的设计与实现。

第五章：解码模块的测试与应用。

第六章：总结了本文所做的工作，并探讨了进一步的研究方向。

1.5 本章小结

本章首先说明了论文的研究背景，网络融合必将以 IP 网络为核心，通过分组交换网络承载包括传统电信网、互联网和有线电视网的所有通信业务从而使得在网络融合过程中运营商面临自身两大网络传统的电信网与互联网的融合问题得以解决。然后介绍了 VoIP 技术的基本概念、原理以及常见的 VoIP 协议。接着介绍了软交换技术以及交换技术的引入对 VoIP 技术的影响。最后对论文主要研究的内容和组织结构进行了阐述。

第二章 媒体网关协议

2.1 媒体网关

传统的网关既要支持媒体变换又要支持媒体控制和信令，功能过于复杂，对于IP电话系统的大规模部署有很大的制约。于是人们提出了网关分离的思想，即把原来自媒体和信令转换功能的网关分离为媒体网关和信令网关，两者通过媒体网关控制器联系即呼叫控制和媒体处理分别由独立的物理实体来完成，这样网络就可清晰地分为控制流层面和媒体流层面。将网关设备分解成媒体网关和媒体网关控制器就是这种趋势的体现。其结构如图2.1所示。

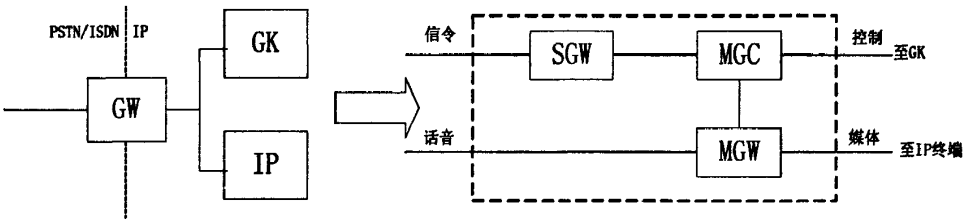

图2.1 网关分离模型

其中MG(Media Gateway)负责媒体格式的转换及PSTN和IP两侧通路的连接；SGW(Signalling Gateway)负责信令的底层转换，即从TDM电路传送转变为IP网络中的传送方式，从应用层的角度看，SGW对于信令仍是透明的；MGC(Media Gateway Controller)，又称SoftSwitch或Call Agent，负责根据收到的信令控制MG的连接建立和释放。MGC对信令消息进行分析和处理并进行应用层的互通变换。不同类型的网关可以支持不同类型的终端。

2.2 H.248协议

为了处理媒体网关控制器（MGC）与媒体网关（MG）这两个分离实体之间的通信，IETF(The Internet Engineering Task Force：互联网工程任务组)和ITU-T(ITU Telecommunication Standardization Sector：国际电信联盟远程通信标准化组)分别推出了对应的媒体网关控制协议即MGCP和H.248协议。H.248协议是2000年由ITU-T第16工作组提出的媒体网关控制协议，它是在早期的MGCP协议基础上改进而成。相对于MGCP协议H.248协议具有更多的优势。具体表现为：

1) H.248协议支持多媒体，MGCP不支持
2) 应用于多方会议时，H.248比MGCP更容易实现。
3) MGCP基于UDP传输，H.248基于TCP、UDP等传输。
4) H.248消息编码基于文本和二进制，MGCP基于文本。

5) MGCP 从功能角度定义 MGC 和 MG 的行为实现简单，但互通性和支持业务能力受限；H.248 支持业务能力强，不断有附件补充其能力，是媒体网关与软交换之间的主流协议。

Megaco/H.248 协议的使用范围广泛包括：

1) 物理上分离的媒体网关之间应用的的协议；
2) 用于包交换网络，包括 IP、ATM 和其他网络；
3) 接口支持多种电路交换网的信令系统（语音信令、ISDN、ISUP、QSIG 和 GSM）；
4) 适合大型网关、接入网关系统。

图 2.2 说明了 Megaco/H.248 协议在软交换系统中的应用范围。

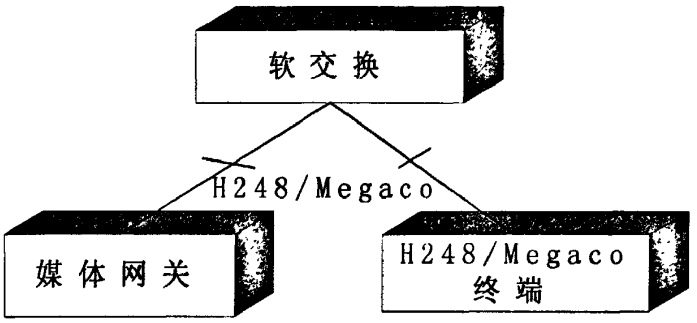

图 2.2 Megaco/H.248 应用范围

图 2.3 为适合简单终端的大规模 VOIP 网络中 H.248 协议的应用位置。

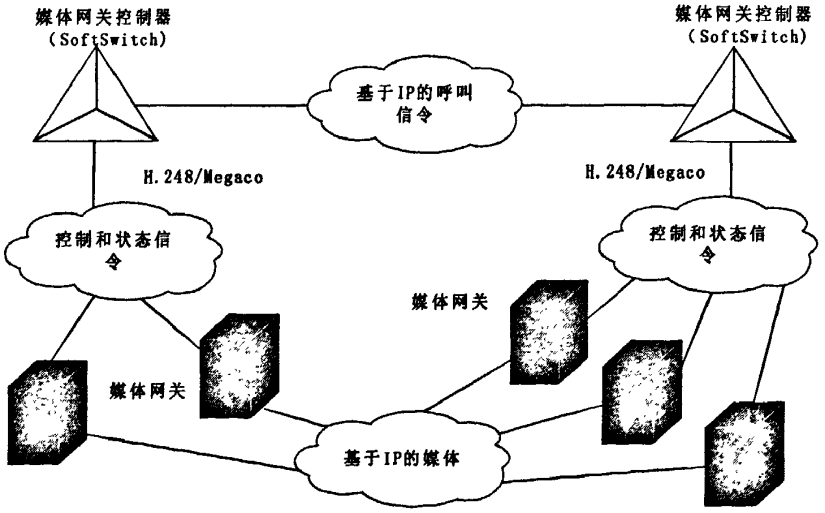

图 2.3 适合简单终端的大规模 VOIP 网络

H.248 协议的连接模型描述 MG 中能由 MGC 控制的逻辑实体，连接模型的主要抽象概念包括终端（Termination）和关联（Context），它们与包（Package）的概念一起构成了 H.248 协议的主要思想框架[6]。

2.2.1 终端

终端（Termination）是MG的一个逻辑实体，可以发送（接收）媒体流和（或）控制流，终端可用特性来进行描述，在终端中，封装了媒体流参数、modem和承载能力参数，这些特性可以组成一系列描述符而包含在命令中。

1）终端类型

终端通常可分为两类，一类是半永久终端，用来表示物理实体。例如TDM 信道，只要这个TDM 信道在媒体网关中被配置，就一直存在，只有当配置信息被删除时，与之对应的终端才会消失。另一类称为临时终端，代表临时性的信息流，例如RTP流，当需要时创建，使用完毕后就删除。临时终端通过ADD 命令创建，通过SUBTRACT 命令清除。与此不同，当一个半永久终端被加入一个特定关联时，它是从NULL 关联中获取，而当从特定关联中删除时，它又被返回到NULL关联。

2）终端功能

终端可支持信号，这些信号可以是 MG 产生的媒体流（如信号音和录音通知），也可以是信路信号（如 Hook Flash）。通过编程可以设置终端对事件进行检测，一旦检测到这些事件发生，MG 就向 MGC 发送 Notify 消息进行报告或由 MG 采取相应的操作。终端可以对数据进行统计，当 MGC 发出 AuditValue 命令进行统计请求时，或者当终端从它所在的关联被删除时，终端就将这些统计数据报告给 MGC。

3）终端 ID

终端可用 Termination ID 进行标识，Termination ID 由 MG 分配。Termination ID 可以使用通配值“ALL”和“CHOOSE”。通配值“ALL”用来规定多个终端，当命令中的 Termination ID 是通配值“ALL”时，则对每一个匹配的终端重复该命令；“CHOOSE”则用来指示 MG 必须选择符合条件的终端，例如 MGC 可以指示 MG 选择一个中继群中的一条中继点电路。

例如，在协议的文本格式编码中，有 R13/3/1, R13/3/2, R13/3/3 三个终端，则 R13/3/*将匹配所有这三个终端。一些特殊场合必须引用所有终端，这时 “ * ”就可满足要求。当需要引用一个 Termination ID，但不能确定该终端是否存在，则可以选用“CHOOSE”，即“ $ ”，则 R13/3/$将匹配三个终端中的其中一个。

4）终端特性和描述符

终端可用特性进行描述，每个特性由一个PropertyID标识，由这些特性可以组成一系列描述符。终端具有一些公共特性以及与特定媒体流相关的非公共特性。公共特性与特定媒体流无关，也称为终端状态（TerminationState）特性。与特定媒体流相关的特性包括本地（Local）特性和接收/发送流特性。终端的非公共特性由包进行定义，这些特性可由包名（PackageName）和特性标识符（PropertyID）来标识。

特性具有只读（ReadOnly）和可读写（Read/Write）两种属性，对于可读写的特性，MGC可以设置它们的值。除了TerminationState和LocalControl之外，在终结点刚被创建或返回到空关联时，都认为该属性为空或“无值（no value）”。

描述符（Descriptor）是协议中的一种语法元素，用来描述一组相互联系的特性。H.248 协议所规定的描述符见表 2.1。

表 2.1 H.248 协议描述符

描述符名称	说明
调制解调（Modem）	标识所使用的调制解调器类型和属性
复用（Mux ）	描述多媒体终结点的复用类型(如 H.221、H.223 和 H.225.0)和组成复用终结点的终结点
媒体（Media）	媒体流的列表 终结点状态（TerminationState）包中所定义的与特定媒体流无关的终结点属性
本地（Local）	包含媒体网关从远端实体接收到的媒体流属性
远端（Remote）	包含媒体网关发送至远端实体的媒体流属性
本地控制（LocalControl）	包中所定义的包含与媒体网关和媒体网关控制器有关的一些属性。
事件（Events）	描述可以被媒体网关检测的事件，以及当事件被检测发生时的处理机制。
事件缓存（EventBuffer ）	描述当事件缓存处于激活状态时，可以被媒体网关所检测的事件
信号（Signals ）	描述适用于终结点的信号
审计（Audit）	适用于 Audit 命令，描述需要审计的信息
包（Packages）	适用于 AuditValue 命令，返回由终结点实现的包的列表
数字映射（DigitMap）	定义一组特定的事件被匹配的模式，使得事件可以按组而不是单个上报
业务改变（ServiceChange）	适用于业务改变（ServiceChange）命令，描述何种业务发生改变以及业务发生改变的原因
被观察事件（ObservedEvents ）	适用于 Notify 或者 AuditValue 命令，上报已检测到的事件
统计值（Statistics）	适用于 Subtract、Audit 命令，上报与终结点有关的统计数据
拓扑（Topology）	描述关联中终结点之间的媒体流流向

关联特征（ContextAttribute）	包中所定义可以影响整个关联的属性
差错（Error）	包含差错代码和可选的差错描述文本，适用于 Notify 请求或所有命令响应

5）根终端

根终端（Root）是特殊的终端，代表整个 MG。当 root 作为命令的输入参数时，命令可以作用于整个网关，而不是一个终端。

2.2.2 关联

关联（Context）为一组终端之间的联系。如果一个关联中超过两个终端，那么关联就对终端之间的拓扑结构和媒体混合和（或）交换参数进行描述。空关联是一种特殊的关联，它包含所有那些与其它终端没有联系的终端，例如，在一个中继网关中，所有的空闲线路被作为终端包括在“空”关联当中。图 2.4 给出了终端和关联的例子，但不包括所有类型。

关联中的最大终端数是媒体网关的一个特性。仅支持点到点连接的媒体网关在每个关联中仅允许两个终端存在。支持会议呼叫的媒体网关可以允许三个或更多的终端同时存在于一个关联中。通过 H.248 协议中定义的命令能够对终结点进行添加、修改、删除等操作。

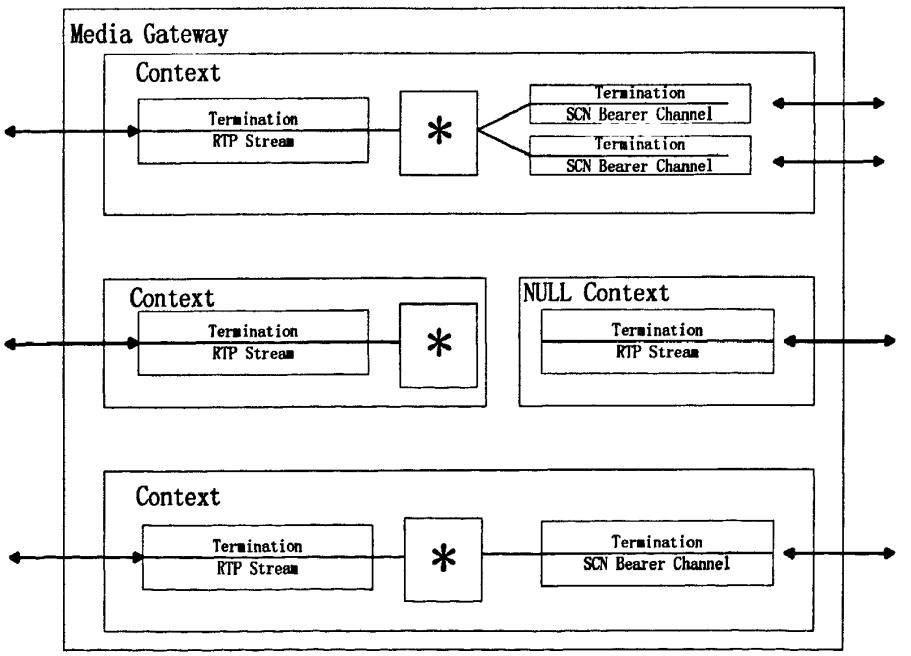

图 2.4 关联与终端

关联具有以下特性：

◆ ContextID：关联标识，一个由媒体网关（MG）选择的 32 位整数，在 MG 范

围内是独一无二的。特殊关联编码对照如表 2.2 所示。

表 2.2 特殊关联编码对照表

关联	二进制编码	文本编码	含义
空关联	0	“_”	表示在网关中所有与其它任何终端都没有关联的终端
CHOOSE 关联	0xFFFFFFFE	“$”	表示请求 MG 创建一个新的关联
ALL 关联	0xFFFFFFFF	“*”	表示 MG 的所有关联

- Topology：拓扑结构，关联的拓扑结构描述关联中终端之间的媒体的流向。终端的 Send/Receive 方式指示媒体在媒体网关的流入或流出方向。有三种连接值：单向，双向，隔离。单向是指两个终端之间的单向媒体流。双向是指的两个终端之间的双向媒体流。隔离是指两个终端之间没有媒体流。拓扑结构只用于描述关联。它可在“Add”或“Modify”命令中使用。
- 优先权：表示 MG 处理关联的先后次序。“0”为最低优先级，“15”为最高优先级。
- 紧急呼叫的标识符：用于关联向 MG 提供紧急呼叫关联的信息。MG 优先处理使用紧急呼叫标识符的呼叫。

2.2.3 包

不同类型的网关可以支持不同类型的终端，本协议通过允许终端具有可选的特性、事件、信号和统计来实现不同类型的终端。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），MGC 可以通过审计命令 Audit 来确定终端实现了哪一种类型的包。

终端具有可选的特性、事件、信号和统计，这些可选项组合成包。这些项以及包含的参数分别由标识符 ID 进行标识。包的定义特性、事件、信号、统计和程序五个部分。表 2.3 列出了几种常用的包。

表 2.3 H.248 常用包

包名	中文名	包 ID	含义
Generic	通用包	g	常见项目里都会用到通用包
Base Root Package	基础根包	root	该包定义了网关范围内的属性

包名	中文名	包 ID	含义
Tone Generator Package	音生成器包	tonegen	该包定义了生成放音的各种信号。基于扩展性的考虑，该包没有指定参数值。放音一般定义成单个的信号，信号包含一个参数 ind、一个放音 ID。参数 ind 表示 interdigit 时延，放音 ID 用于放音。放音 ID 对于任何相同的语音来说都应该与语音生成保持一致。MG 应提供其所在国家支持的各种放音的特性。
Tone Detection Package	音检测包	tonedet	该包定义了用于音检测的各种事件。各种音通过其名称(放音 ID)来选择。MG 应提供其所在国家支持的各种放音的特性。
Basic DTMF Generator Package	基本 DTMF 生成器包	dg	该包将基本的 DTMF 音定义成各种信号，并扩展了 tonegen 中 playtone 的参数 tl 的允许取值。
DTMF detection Package	DTMF 检测包	dd	该包定义了基本的 DTMF 音检测。该包扩展了“start tone detected”、“end tone detected”和“long tone detected”事件中放音 ID 的可能的取值。
Call Progress Tones Generator Package	呼叫进展音生成器包	cg	该包将基本的呼叫进展音定义成各种信号，并扩展了 tonegen 中 playtone 的参数 tl 的允许取值。
Call Progress Tones Detection Package	呼叫进展音检测包	cd	该包定义了基本呼叫进展检测音。该包扩展了“start tone detected”、“end tone detected”和“long tone detected”事件中放音 ID 的可能的取值。

包名	中文名	包 ID	含义
Analog Line Supervision Package	模拟线监控包	al	该包定义了模拟线的各种事件和信号。
Basic Continuity Package	基本导通包	ct	该包定义了用于导通测试的各种事件和信号。导通测试包括提供环回或收发器功能。
Network Package	网络包	nt	该包定义了与网络类型无关的网络终端的属性。
RTP Package	RTP 包	rtp	该包用于支持通过实时传输协议 RTP 方式的分组多媒体数据传输。
TDM Circuit Package	TDM 电路包	tdmc	该包用于支持 TDM 电路终结点。

包中常用的特性名、事件名和信号等。其通常为包名/特性名、包名/事件名和包名/信号的格式。在解码过程中通常需要对包中的这些特性名、事件名和信号等进行解析。表 2.4 列出了这些事件名所对应的含义。

表 2.4 H.248 包特性表

事件名	含义
al/fl	模拟线包中的拍叉事件
al/of	模拟线包中的摘机事件
al/on	模拟线包中的挂机事件
al/ri	模拟线包中的振铃音信号
cg/bt	呼叫音包中的忙音信号
cg/ct	呼叫音包中的拥塞音信号
cg/cw	呼叫音包中的呼叫等待音信号
cg/dt	呼叫音包中的拨号音信号
cg/rt	呼叫音包中的回铃音信号
dd/ce	DTMF 检测包中的 DigitMap Completion 事件
nt/jit	Network Package 中的抖动缓存最大值，单位为毫秒
tdmc/ec	TDM 电路包中的回声取消特性
tdmc/gain	TDM 电路包中的增益控制特性

2.3 H.248 协议消息结构

H.248 消息是 H.248 协议发送的一个信息单元。消息可以采用文本格式编码和二进制格式。文本方式编码时，遵循 RFC 2234 ABNF 规范；二进制编码时，使用 ITU-T X.680（ASN.1）定义的规范描述，使用 X.690 定义的 BER 规则编码。MGC 必须支持两种编码格式，MG 可能支持其中任何一种或两种方式。H.248 消息都有相同的结构，一个 H.248 消息的结构如图 2.5 所示。

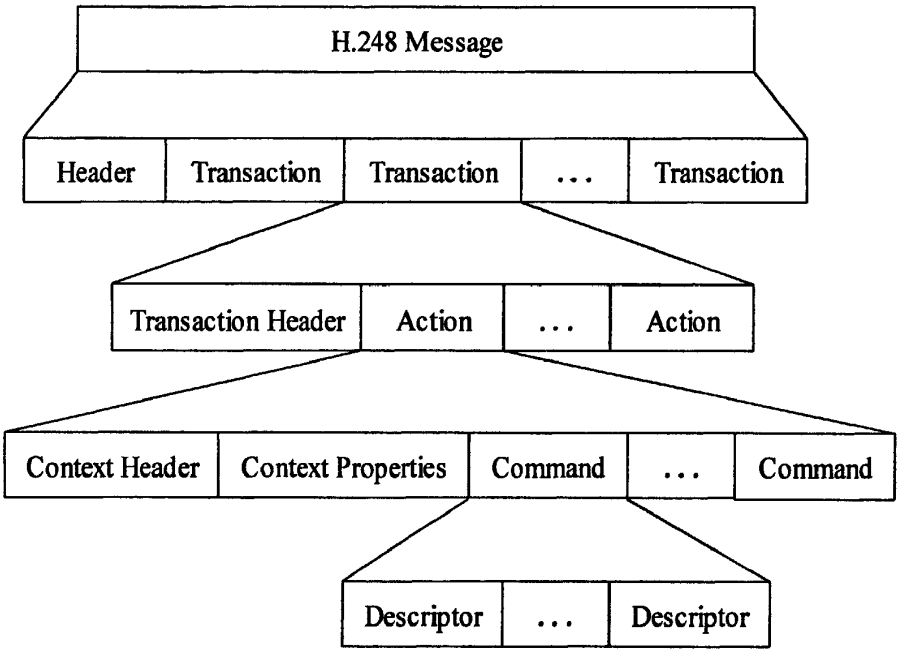

图2.5 H.248消息结构

一个H.248 消息（Message）包含多个事务交互（Transaction），消息中的事务交互之间没有关系，可以单独处理；一个事务交互由多个动作（Action）构成，动作对应关联（Context），动作由一系列局限于一个关联的命令（Command）组成。

由此，H.248 消息构成机制如图2.6所示。

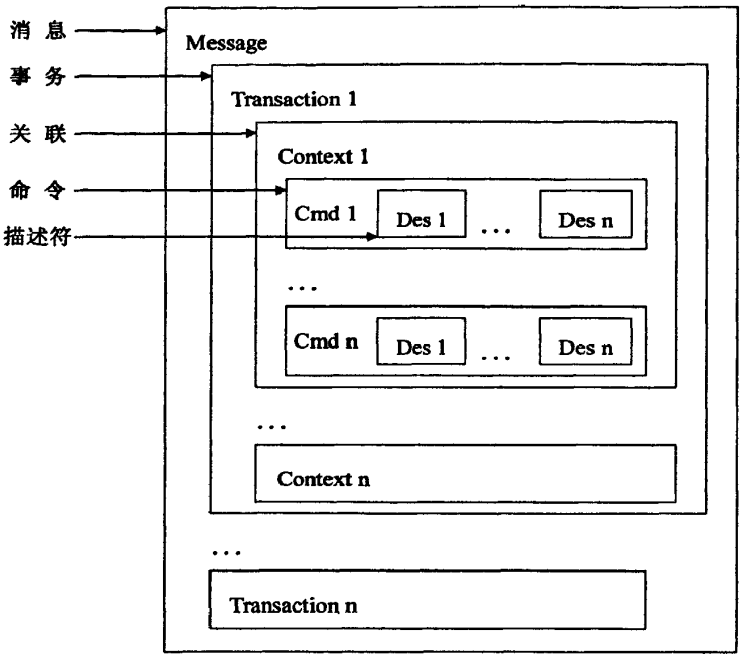

图2.6 H.248消息构成机制

2.3.1 消息

H.248 协议发送或接受的信息单元称为消息，消息从消息头（Header）开始，后面是若干个事务交互。

消息头中包含消息标识符（MID，Message Identifier）和版本字段：

- MID 用于标识消息的发送者，可以是域地址、域名或设备名，一般采用域名。
- 版本字段用于标识消息遵守的协议版本。版本字段有 1 位或 2 位数，目前版为 1。

消息内的事务交互是相互独立的，当多个被独立处理时，消息没有规定处理的先后次序。

2.3.2 事务交互

MGC 和MG 之间的一组命令构成事务交互，事务交互由TransactionID 进行标识。事务交互包含一个或多个动作，一个动作由一系列局限于一个关联的命令组成。一个事务交互从“事务头部”（TransHdr）开始。在TransHdr 中包含TransactionID。TransactionID 由事务交互的发送者指定，在发送者范围内是唯一的。事务交互包括请求和响应两种类型，而响应也有两种：TransactionReply 和TransactionPending。

2.3.3 动作与命令

1）动作

动作与关联（Context）是密切相关的，它由一系列局限于一个关联的命令组成。动作由ContextID 进行标识。在一个动作内，命令需要顺序执行。一个动作从关联头部（CtxHdr）开始，在CtxHdr 包含ContextID，用于标识该动作对应的关联。ContextID 由MG 指定，在MG 范围内是唯一的。MGC 必须在以后的与此关联相关的事务交互中使用ContextID。在CtxHdr 后面是若干命令，这些命令都与ContextID 标识的关联相关。

2）命令

H.248定义了8 个命令，用于对协议连接模型中的逻辑实体（关联和终端）进行操作和管理，命令提供了实现对关联和终端进行完全控制的机制。

H.248规定的命令大部分用于MGC 实现对MG 的控制。通常MGC 作为命令起始者，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。表2.5列出了H.248命令及其含义。

表 2.5 H.248 命令

命令名称	命令代码	描述
Add	ADD	MGC→MG，增加一个终端到一个关联中，当不指明 ContextID 时，将生成一个关联，然后再将终端加入到该关联中。
Modify	MOD	MGC→MG，修改一个终端的属性、事件和信号参数。
Subtract	SUB	MGC→MG，从一个关联中删除一个终端，同时返回终端的统计状态。如关联中再没有其它的终端将删除此关联。
Move	MOV	MGC→MG，将一个终端从一个关联移到另一个关联。
AuditValue	AUD_VAL	MGC→MG，获取有关终端的当前特性，事件、信号和统计信息。
AuditCapabilities	AUD_CAP	MGC→MG，获取 MG 所允许的终端的特性、事件和信号的所有可能值的信息。
Notify	NTFY	MG→MGC ，MG 将检测到的事件通知给

		MGC。
ServiceChange	SVC_CHG	MGC↔MG 或 MG→MGC， MG 使用 ServiceChange 命令向 MGC 报告一个终端或者一组终端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 进行注册，并且向 MGC 报告 MG 将要开始或者已经完成了重新启动工作。同时，MGC 可以使用 ServiceChange 命令通知 MG 将一个终端或者一组终端进入服务或者退出服务。

所有的 H.248 命令都要接收者回送响应。命令和响应的结构基本相同，命令和响应之间由事务 ID 相关联。

响应有两种："Reply" 和 "Pending"。"Reply" 表示已经完成了命令执行，返回执行成功或失败信息；"Pending" 指示命令正在处理，但仍然没有完成。当命令处理时间较长时，可以防止发送者重发事务请求。

2.4 协议栈结构

H.248 消息可基于多种传输协议传输，例如承载在纯 IP 网络上的 TCP、UDP、SCTP 等协议，承载在 ATM 网络上的 MTP3b 协议，承载在混合 ATM、IP 网络上的 M3UA 协议，H.248 协议传输可以基于 IP，也可基于 ATM 具体结构如图所示。

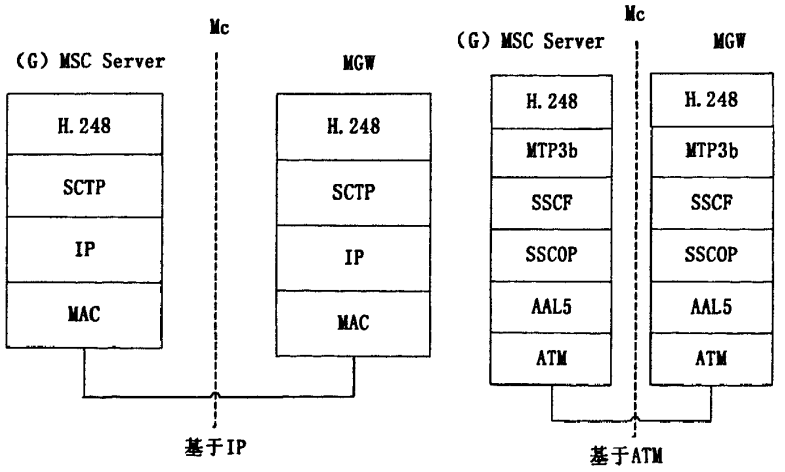

图 2.7 协议栈结构

2.5 本章小结

本章首先说明了传统的网关过于复杂而制约了 IP 电话系统的大规模部署，网关分离思想可以解决这一问题，通过对网关分离模型的介绍说明了媒体网关产生的过程，即传统网关分解为信令网关和媒体网关。为了处理媒体网关控制器（MGC）与媒体网关（MG）这两个分离实体之间的通信 IETF 和 ITU—T 分别推出了对应的媒体网关控制协议即 MGCP 和 H.248 协议。本章分析比较了媒体网关控制器与媒体网关之间的这两种通信协议，列出了 H.248 协议相对于 MGCP 协议的优点以及 H.248 协议广泛的应用范围。

然后，从终端（Termination）、关联（Context）和包（Package）三个方面说明了 H.248 协议的主要思想框架。同时介绍了 H.248 协议的消息结构以及消息的构成机制为后续章节中 H.248 协议消息解码奠定了基础。最后介绍了 H.248 协议消息的协议栈结构。

第三章 网络管理系统体系结构

3.1 网络管理系统功能结构

网络管理系统软件架构的基本设计思想是采用面向对象的模块化设计和多线程技术。从功能上划分，可以分为数据采集、数据存储、协议识别、协议解码、事务关联、呼叫跟踪与追踪、统计分析、界面显示等部分。

根据这些功能划分成不同的模块其结构如图 3.1 所示

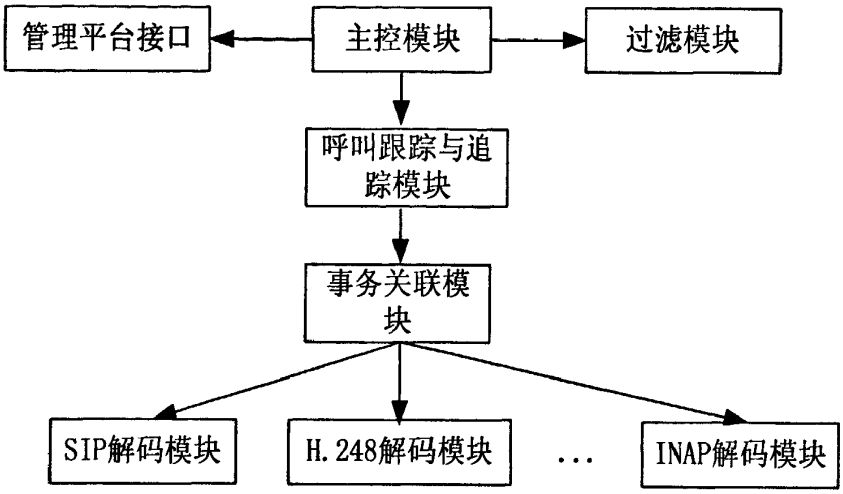

图 3.1 系统模块功能结构

- 主控模块：控制和调用其他模块，网络中的数据通过管理平台接口进入系统，再由主控模块管理数据将数据送到解码模块进行处理。并将解码结果显示出来，利用呼叫跟踪与追踪模块完成呼叫跟踪，呼叫统计等功能。
- 管理平台接口：采集网络中的数据，给数据打上时间戳然后送到数据缓冲区。
- 呼叫跟踪与追踪模块：应用解码模块提供的数据，提供呼叫跟踪功能。它有多个子系统来实现不同的功能，如存储会话细节、Remon 等。网络管理系统提供的很多功能都是建立在呼叫追踪与跟踪之上的。
- 事务关联模块：直接与呼叫跟踪与追踪模块通信，可以将 SUs(信令单元)转发给呼叫跟踪与追踪模块。也可以将同一个会话中的 SU 发送到不同的呼叫跟踪与追踪模块进行处理，这些呼叫跟踪与追踪模块将可以互相通信并将所有的信令单元信息传送到一个呼叫跟踪与追踪模块中以完成对这个会话的关联。
- 解码模块：网络管理系统的所有功能都是依靠解码来实现的。通过对消息中的数据的解析，各个模块都要应用消息解码出来的信息来实现该模块的

功能。由于不同的模块需要的信息不一样，在设计解码模块的时候就需要根据消息的结构来设计。

上述模块间采用松耦合，主要采用动态链接库相互调用，各模块功能相对独立，缩小相互之间的依存以生成的动态链接库（DLL）形式存在，功能模块DLL提供接口，向主控模块提供各种参数。与界面相关的操作由主控模块和应用进程实现，与其它功能模块无关。这样有利于软件中各模块的移植、修改，利于软件升级。

3.2 网络管理中数据处理流程

数据经过管理平台接口采集数据，在数据采集的时候对数据进行简单处理包括时间戳，编号等。再按照一定的格式将数据数据存储在缓存中，在缓存中的数据通过协议识别将其分类，它包含该数据单元在缓存中的位置及其长度、类别、协议栈结构及各层协议 PDU 位置指针及长度等重要信息。根据消息的类别调用不同的解码单元。由于不同的功能对消息解码要求不一样，如：Remon 只需要知道消息的源地址和目的地址即可，而不用知道消息的类型，因此在解码的过程中对消息解码的“深度”要求不一样。后续的处理包括呼叫的跟踪、呼叫统计、以及 States 等都要通过解码来完成，处理结果通过交互界面显示出来。数据处理流程如图 3.2 所示。

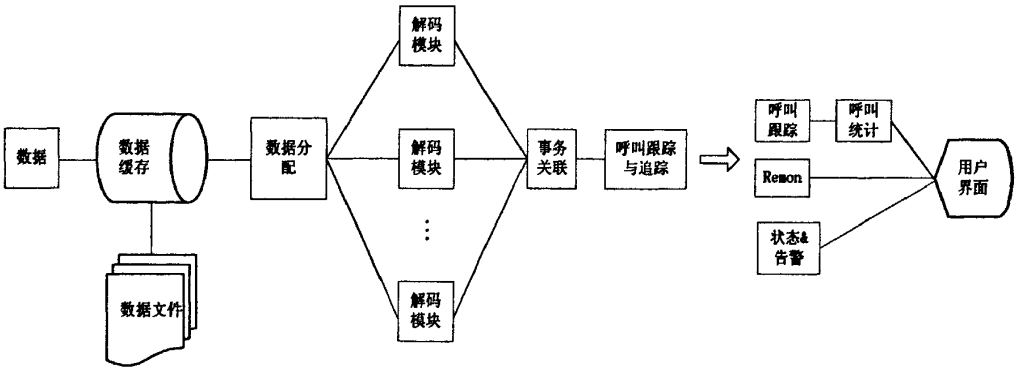

图 3.2 数据处理流程

网络融合过程中网络中的数据越来越大，要求网络管理系统中对数据处理的速度越来越高。面对大量的数据而协议解码又是网络管理系统中各项功能应用的基础，因此只有当解码的效率提高时才能更好地为系统服务。为了提高解码效率我们采用并行处理数据的方式，如图 3.2 中所示每个解码模块都各自独立地处理消息数据。事务关联模块中的相应单元将接受这些数据进行处理。在工作过程中数据分配模块根据数据缓存中的数据，将不同协议数据分配到相应解码模块。

3.3 多线程技术

多线程的设计为我们软件带来了很多优点。一个单独的线程处理每个进入的连接，而另一个线程则一直在运行以管理这些连接。多个客户可以同时进行读操作，而不会彼此影响。但写操作在某种程度上依赖于所用表的类型，若其他客户需要访问正在更新的数据，那么会将其挂起。当某个线程在写一个表时，所有其他需要访问该表的线程都要等待表被释放。客户可以执行任何允许的操作，而不用考虑其他并发的连接。连接管理线程可以防止其他的线程读写正在更新的表。

整个软件的多线程体系结构如图3.3所示。

主线程负责与用户交互提供友好的人机交互界面，包括软件的启动、中止、支持用户设置过滤规则、设置监测点检测位置、画图功能保证用户能够直观地了解观测点和刷新等。主线程具有调度其它辅助线程的作用。

数据读取线程负责从服务器接收数据，为了在高速率时不丢包，这个线程被设置为一个高优先级的线程。该线程的作用是响应硬件驱动的消息，从硬件读取信令消息数据，把接收到的消息数据以一定的格式存放到数据缓冲区中，该缓冲直接映射到硬盘文件上面。

消息解码线程为数据缓冲区中的新消息，进行消息解码和呼叫跟踪。处理完一条消息后，此线程向主线程发送界面显示或刷新的消息，接着处理下一条新消息（如果有的话）。事务关联模块接收消息解码结果并将消息解码结果发送到呼叫跟踪与追踪模块存储。

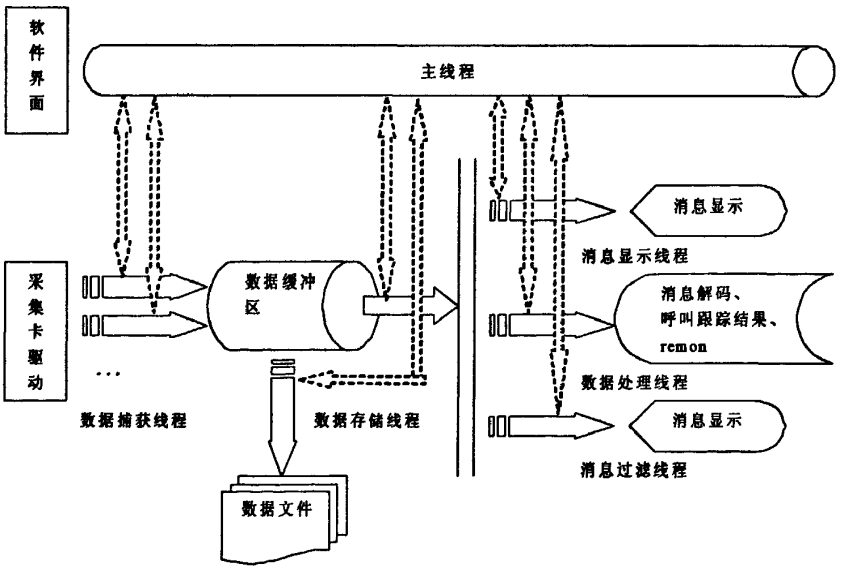

图 3.3 网络管理多线程体系结构

3.4 本章小结

本章概述了网络网络管理系统的体系结构以及各个模块的功能，而后介绍了网络管理系统的数据处理流程，最后还对网络管理系统的多线程体系结构做了简略的介绍，简单描述了多线程技术给软件设计带来的优点。

第四章 H.248 协议解码功能的设计与实现

软件的生命周期由软件定义、软件开发和运行维护三个时期组成[14]。

在进行一个项目开发时，我们一般要遵循以下七个步骤：问题定义，可行性研究，需求分析，总体设计，详细设计，编码和单元测试，综合测试。网络测试仪器或者网络管理设备中的功能，如呼叫跟踪，呼叫统计，呼叫的跟踪与追踪都是基于协议的解码。解码模块为这些功能提供必要的数据。在网络融合的过程中，人们对于宽带及业务的要求在迅速增长，业务需求趋于多样化，网络的压力网络中的数据量会随着融合的进程也越来越大，因此这些网络测试和管理的仪器对解码的效率要求也越来越高。

H.248 协议的解码功能实现采用面向对象的程序设计方法，采用类的方式将解码相关功能的实现封装成 DLL，对外仅提供相应的解码接口函数，这样有助于将功能的实现与接口分离，便于模块的移植和维护。

4.1 解码模块的设计

通过对 H.248 消息的结构和消息构成分析我们可以将消息可以分为消息起始部分、事务部分和动作部分这三个字段这样划分消息可以使得消息分为三个部分。消息起始部分包含了消息头以及发送者标识（IP 地址和端口号），事务部分包括事务类型和事务 ID，动作部分则有消息的关联 ID 命令部分和描述符。在解码模块的上层模块事务关联模块中需要对消息进行重构和消息的“配对”即一个请求消息必然会有一个响应消息，事务关联模块中消息的“配对”是以消息的事务交互为基础的。而消息在重构的时候需要消息头、消息事务和消息关联的信息作为重构的依据，采用这样一种划分满足系统设计的要求。这种划分还可以让消息在结构和功能上保持一种相对的独立。因此我们采用分层的思想来进行解码模块的设计，解码模块分为三层：消息与解码层、事务解码层和动作解码层。这种划分方式基于上面对消息的划分，因此层与层之间可以保持一种相对的独立，主要表现在：

1) 每层在解析过程中只负责本层所需要解析的参数，并不依赖于上层解析的结果。同时也不会对下层的解析产生影响。
2) 为了降低代码实现的复杂度，解码模块采用顺序解码的方式。即任何一条 H.248 消息必须按照消息预解码，事物解码和动作解码的顺序进行解析才能完成消息的详细解码。这样处理可以不必预先取出消息中相应字段从而降低代码实现的复杂度。因此，下层需要上层为其提供原始数据。所以这种独立是相对的。

从解码模块升级和维护的角度来说采用分层解码便于解码模块的升级与维护，当有新的包或者用户需要一些可选包中的参数时只需要在相应层对这些参数的解析进行添加就可以了，而不需要对其它层进行修改或者添加大大方便了解码模块的升级和维护。

解码功能模块总体架构如图 4.1 所示。

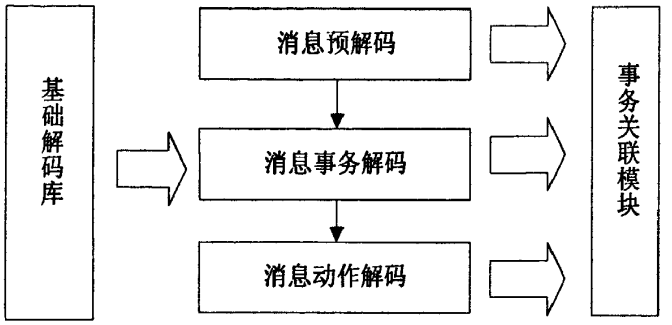

图 4.1 解码功能模块结构

事务关联模块是解码模块的上层，解码模块中的各层在解析过程中将解析出来的参数传递给事务关联模块进行处理。下面我们对各个模块的功能进行说明。

- 消息预解码层：判断消息长度是否合法，如果是，执行下一步判断消息类型对消息头以及发送者标识的解析，否则退出。
- 事务解码层：对事务类型及事务ID解码，并针对两种特殊的事务类型调用相应的事务解码单元进行解析。
- 动作解码层：对动作的上下文ID进行解析，完成对命令类型的解析，完成对描述符字符串的判定并调用相应的描述符解码单元。对描述符解码，生成h.248的内部消息完成详细的消息解码。
- 基础解码库：包含了在H.248消息解码过程中需要用到的解码方法比如：对十进制数的解析、空白线性字符的处理、当前字符或者字符串的判定等等。

由于基础解码库只包含各种解码方法我们并不单独对其进行阐述，但是解码库是非常必要的，将解码的方法放在基础解码库中简化了各个解码层。同时有许多解码的方法在各层都会用到，因此放在基础解码库中当各层需要时直接从解码库中调用即可。基础解码库是解码解码库中的一些解码方法会在接下来的具体实现中有所涉及。下面我们通过对消息预解码层、事务解码层和动作解码层的设计与实现来进一步了解 H.248 消息解码模块的实现过程。

4.2 消息预解码层的设计与实现

消息预解码层是解码模块的第一层需要对外提供接口函数函数：

```
H248Msg * ParseH248Msg (  UInt8 * pData,   //消息内容
UInt32 length,     //消息长度
IBool    parseDigitsChoice,   //是否进行数字填充
IBool    isH248BIN,   //编码格式
IBool    isUDP   //承载方式
```

H.248消息是从消息头开始，后面跟随若干事务。消息头包含消息表示MID（Message Identifier）和版本字段。消息预解码就是要解析出MID以及版本字段，但是在解析的消息头之前我们并不知道消息的编码方式及文本编码还是二进制编码。因此需要对消息编码方式进行解析。不过对于同一消息而言不论采用哪种编码方式其所传递的信息是一样的。H.248消息支持文本和二进制消息，首先我们采用枚举的方式对消息的编码方式进行申明。

```
enum H248MsgEncodeType
    {
    H248_TEXT_MESSAGE = 0,//ABNF 编码的文本消息
    H248_BINARY_MESSAGE,//二进制消息
    H248_UNKNOWN//未知消息
    };
```

在消息解码过程中有可能会碰到编码错误的消息因此未知消息类型的申明是必要的。消息解码是整个解码的开始阶段，根据 H.248 消息结构消息解码主要解析 H.248 消息的消息头部以 H.248 文本消息为例在消息头部最主要的参数是 Message_Id。

```
class H248Msg: public FmbManager
    {
protected:
    UInt8 *          pData;          //消息在内存中的位置
    UInt32           dataLen;        //消息的长度
    IBool            suSyntaxErrFlag;//语法错误标识
    UInt32           megacoVersion;//协议版本
    UInt8            encodeType;     //消息编码方式
..
          //获取消息的基本信息
public:
    inline UInt32          GetDataLength    ( void );  //获取消息的长度
    inline UInt8           GetEncodeType  ( void );//消息编码方式
```

在基类中包括了消息的基本信息如消息在内存中的位置、消息长度（用以保存原始数据）以及协议版本等。消息编码出错时我们就没必要继续对该消息进行解码，因此我们定义了布尔型变量suSyntaxErrFlag来标识消息编码是否存在语法错误。当该值为真时就不需要继续对消息进行进一步的解码，返回下一条消息解码。在消息解码的基类中，由于需要确定消息的编码类型因此在基类中定义了变量encodeType期通过函数GetEncodeType来调用并将结果返回encodeType。基类中的GetDataLength、GetMsgId等函数需要频繁的用到，为了提高模块运行的效率我们申明为内联函数。

消息解码流程如图4.2所示。

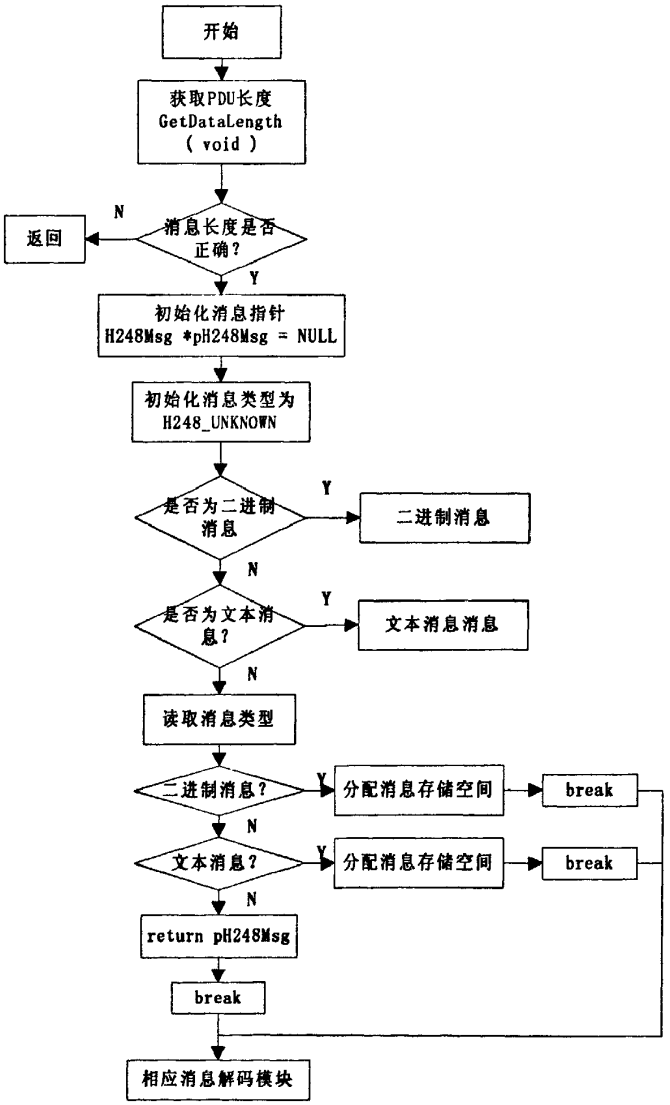

图 4.2 消息类型解码流程

下面对消息类型判定算法进行说明：

（1）通过 GetDateLength 函数获取了原始消息的长度并传递给 dataLen 存储起

来。再对这个值进行判断若是大于了消息长度的最大值（呼叫跟踪与追踪模块允许消息长度最大为 256 字节的消息）则返回，否则就进行下一步。

（2）初始化消息指针，H248Msg *pH248Msg = NULL 避免“野指针”出现。

（3）初始化消息类型为 H248MsgEncodeType encodeType = H248_UNKNOWN，这样做是为了当消息本身存在错误时返回 H248_UNKNOWN，表示不是 H.248 消息。

（4）在判断消息类型时我们采用了一种基于 ASCII 码表的快速判别消息类型的机制。我们知道文本消息在内存中以 ASCII 码的形式存储，二进制消息则是以二进制的形式存储。在判别消息类型时我们可以通过区分消息是否是 ASCII 码就能够区分开该消息是文本消息还是二进制消息。首先我们定义一个类型为 char 型长度为 256 的表 SEPASCIITable[256]使之与 ASCII 表相对应，在表中表示为“1”的位置都是文本消息中可能会用到的字符，其中包括控制字符如 0x09 Horizontal Tab(水平制表符)，0x0A Line feed(换行键)以及可打印字符阿拉伯数字 0~9，小写字母 a~z 以及对应的大写字母和！,*,$等符号。下面我们将说明如何应用 SEPASCIITable[256]来划分消息类型，表如下所示。

```
char SEPASCIITable[256] =
    {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 1,    //0-9
    1, 0, 0, 1, 0, 0, 0, 0, 0, 0,    //10-19
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    //20-29
    0, 0, 1, 1, 1, 1, 1, 1, 1, 1,    //30-39
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //40-49
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //50-59
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //60-69
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //70-79
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //80-89
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //90-99
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //100-109
    1, 1, 1, 1, 1, 1, 1, 1 ,1, 1,    //110-119
    1, 1, 1, 1, 1, 1, 1, 0 ,0, 0,    //120-129
    ...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    //230-239
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    //240-249
    0, 0, 0, 0, 0, 0                 //250-255
    };
```

我们在消息解码的基础类中定义函数IBool IsH248TextEncodeType来判定消息是否是ASCII消息（文本消息）,在解析的时候没有必要对整个消息都进行是否是ASCII判定。在函数中我们设定一个长度i，当消息长度小于这个值时，则对整个消息体进行是否是ASCII的判定。否则只需要判定消息前i个字符即可，在检测时逐个对其进行比较若是ASCII码i--，只要数组中有一个不是ASCII码则该消息一定不是文本编码的H.248消息函数返回FALSE，否则返回TRUE该消息是文本编码消息。下面我们通过对IBool IsH248TextEncodeType(UInt8 * pData, UInt32 length)函数的说明来描述如何应用SEPASCIITable[]来解析消息编码类型。*pData指针指向消息内容首地址，然后在循环判定if (!SEPASCIITable[pData[i]])数组的各个元素（消息里面的单个字符），由于pData[i]所对应的单个字符的ASCII码与其在SEPASCIITable[]中对应的位置是一致的当该位置是‘1’时改字符就是ASCII码表中对应的字符，因此在解析时就不需要每次都去逐一比较从而提高了效率。如果所有的字符都是文本类型的IsH248TextEncodeType函数返回真，否则返回假。

（5)消息类型判定之后为保护原始数据分别为不同类型编码消息分配消息的存储空间。就进入到相应的消息头进行解析。

论文仅以文本消息为例描述消息解码的过程。

4.2.1 Megaco 协议

Megaco协议是采用ABNF（扩展巴科斯范式）语法规范描述的文本格式编码的H.248协议。ABNF是在BNF（标准巴科斯范式，Backus-Naur Form）的基础上扩充的语法规范。ABNF语法规范定义是由一组ABNF规则构成，例如ABNF中规则如下定义：

name = elements crlf

其中，name指规则名，即一个符号序列，该符号序列以字母打头，后面跟一个字母或数字或连字符（下划线）的组合。elements指一个或多个规则名或是终结符。crlf是行结束标志，回车符后紧跟换行符。等号将规则名和规则的定义分隔开。元素构成一个或多个规则名（和/或）值的定义的序列，这些规则名（和/或）值依照ABNF规范中定义的各种操作符，包括"连接"、“选择”、“增式选择”、“值域选择”、“序列组”、“不定循环”、“指定循环”及“可选序列”9种操作符，并且规定了各操作符之间的优先级。ABNF的核心规则是：

ALPHA = %x41-5A / %x61-7A；大写字母 A-Z / 小写字母 a-z

BIT = "0" / "1"；二进制数

CHAR = %x01-7F ； 除 NUL 以外的任何 7 位 US-ASCII 字符

CR = %x0D ； 回车

CRLF = CR LF ； 互联网标准格式的换行

CTL = %x00-1F / %x7F ； 控制字符

DIGIT = %x30-39 ； 数字 0-9

DQUOTE = %x22 ； "（双引号）

HEXDIG = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"； 十六进制数

HTAB = %x09 ； 水平制表符

LF = %x0A ； 换行

LWSP = *(WSP / CRLF WSP) ； 线性空白字符（过去的换行）

OCTET = %x00-FF ； 8 位数据

SP = %x20 ； 空格符

VCHAR = %x21-7E ； 可见（可打印）字符

WSP = SP / HTAB ； 空白字符

消息的解析过程就是对消息的编码的解码过程，在解码过程中都是严格按照编码的方式进行解码。所有的 H.248 文本消息的结构如下：

```
MEGACO/1 <MessageId>
<TransactionOp> = <TransactionId1> {
      Context = <ContextId A> {
    <Command> = <TerminationId> { <DescriptorList> },
    ...
    <Command> = <TerminationId> { <DescriptorList> }
  }
```

```
        Context = <ContextId B> {
      <Command> = <TerminationId> { <DescriptorList> },
      ...
      <Command> = <TerminationId> { <DescriptorList> }
    }
    ...
  }
  <TransactionOp> = <TransactionId2> {
        Context = <ContextId C> {
      <Command> = <TerminationId> { <DescriptorList> },
      ...
      <Command> = <TerminationId> { <DescriptorList> }
    }
        Context = <ContextId D> {
      <Command> = <TerminationId> { <DescriptorList> },
      ...
      <Command> = <TerminationId> { <DescriptorList> }
    }
    ...
  }
  ...
```

下面是H.248命令的文本描述示例：

```
MEGACO/1 [191.169.150.170]:2944
T=372794021{
C= - {
MF=A0{
E=369099784{
dd/ce{DigitMap=dmap1}，al/*},
SG{cg/dt},
DM=dmap1{
([2-9]xxxxxx|13xxxxxxxxx|0xxxxxxxxx|9xxxx|1[0124-9]x|E|x.F|[0-9EF].L)}}}}
```

按照协议规定和编码方式对其进行解释

第一行：MEGACO协议的版本为1。消息发送者标识（MID），此时为MGC的IP地址和端口号：[191.169.150.170]:2944。

第二行：事务ID为“372794021”，该事务ID用于将该请求事务和其触发的响应事务相关联。

第三行：此时，该事务封装的关联为空。

第四行：Modify 命令，用来修改终端A0的特性、事件和信号。

第五行：事务描述符，其RequestID为“369099784”。通过RequestID 可以将事件请求命令和事件发生通知Notify 命令关联起来。

第六行：MGC请求MG监视终端A0发生的以下事件：事件一，根据Digit Map规定的拨号计划（dmap1）收号。事件二，请求网关检测模拟线包（al）中的所有事件。

第七行：信号描述符。表示MGC请求MG给终端A0送拨号音。

第八行：Digit Map描述符。MGC给终端A0下发拨号计划dmap1。

第九行：拨号计划dmap1。其中，“[2-9]xxxxxx”表示用户可以拨2～9中任意一位数字开头的任意7位号码；“13xxxxxxxxx”表示13开头的任意11号码；“0xxxxxxxxx”表示0开头的任意10位号码；“9xxxx”表示9开头的任意5位号码；“1[0124-9]x”表示1开头，3以外的十进制数为第二位的任意3位号码；“E”表示字母“E”；“x.F”；“[0-9EF].L”表示拨以数字0～9、字母“E”、“F”开头的任意位等长定时器超时之后就会上报。

4.2.2 消息解析实现

消息头的解析是整个H.248消息解码的开始下面我们通过消息头解析的流程图来说明消息头的解析过程。消息头解析流程如图4.3所示。

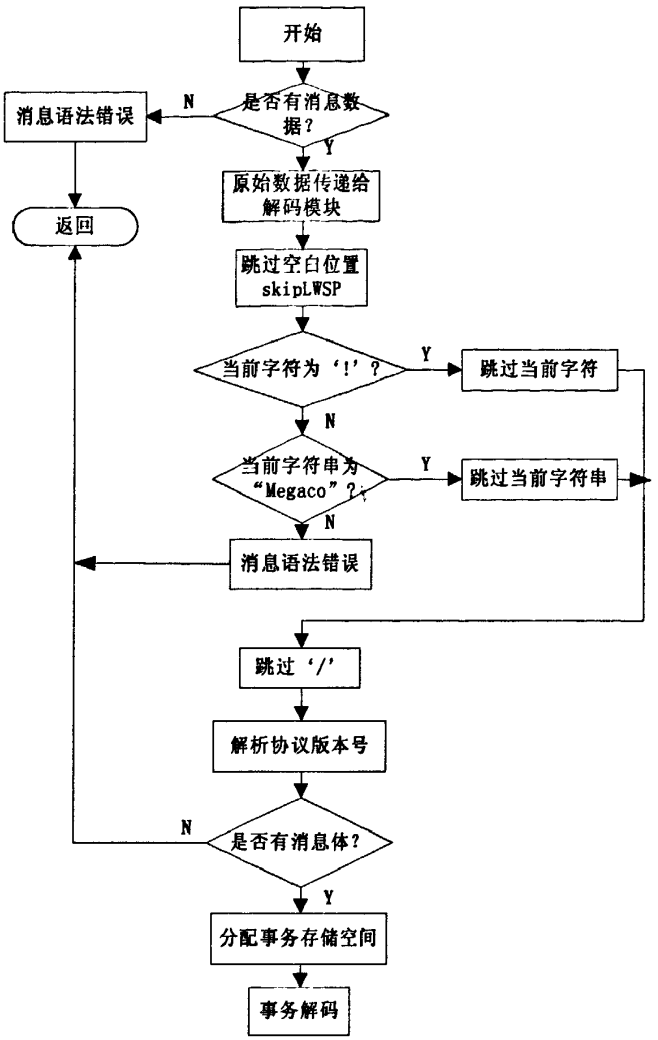

图4.3 消息头解析流程图

下面我们对消息头的解析流程进行说明。

（1）通过if ((!pData) || !(*pData))判断原始数据是否存在消息体，若不存在消息体则设置suSyntaxErrFlag = true，消息存在语法错误不能被解析。否则标记解析的原始位置。进入步骤2。

（2）通过Attach(UInt8 *pDataIn, UInt32 dataLenIn)函数将原始数据传递给解析函数。这样可以保护原始数据不会因意外修改而破坏原始数据，从而保证原始数据安全，pos记录当前解析的位置。

（3）在H.248消息中经常会有大量的线性空白字符（空格符和水平制表符）、换行、回车等。作为用户来说我们只关心消息的内容，而不会去关心那些地方该换行那些地方该回车。SkipLWSP()跳过一些不必要的控制字符如空格，水平制表等在解析过程中不需要解析的字符从而提高模块效率，具体算法为：当解码未完成时即解码位置标记pos小于数据长度时，如果当前位置的的ASCII码是0x20，0x0a，0x0d，

0x09其分别对应分别为Space(空格), Line feed(换行), Carriage return(回车), Horizontal Tab(水平制表符)时pos++进入下一个字符的解析。

（4）ABNF规范中规定：

megacoMessage = LWSP [authenticationHeader SEP] message

Message = MegacopToken SLASH Version SEP mId SEP

MegacopToken = ("MEGACO" / "!")

因此任何一个正确的H.248文本消息都是以“MEGACO”（ABNF规定不区分大小写）或者‘/’作为消息的开始，在对该字段进行解析时我们采用“字符比较”的方式去判定是否是H.248文本消息头。判定方法为：由于消息头的格式是固定不变的，非此（“MEGACO”）即彼（‘!’），我们能够通过将消息需要解析的参数明确的提出来const char* megacoTokenStr = "megaco"，再与读取的数据进行比较。这样处理就不需要和ASCII表中的字符逐一比较来进行当前字符的判定而直接进行比较，提高了检测效率。同时也将字符串长度做了申明const int megacoTokenStrLen= 6，位置指针在移动的时候直接通过Move函数移动跳过已经检测的字符串。通过IsMegacoToken()函数将数据与定义好的megacoTokenStr进行比较，若都不匹配则说明该消息存在消息语法错误，返回。匹配则进行下一步。

（5）消息头中SLASH（‘/’）字符只是做为“megaco”与协议版本号之间的隔离符，不需要对其解析因此直接跳过‘/’。

（6）解析协议版本号。消息头解析完成

（7）在网络管理系统中各个模块只提取消息解码结果后的参数，在消息头解码过程中，事务关联模块就会提取消息头解码数据。在消息头解码完成后就会将消息解码模块的数据删除，因此在解析完版本号之后还需要对后面未经过解析的数据进行保存并对保存的数据进行初步解析，判断保存的数据是否正确。

消息解码模块的核心代码如下：

```
void H248TextMsg::Parse
    (
    IBool    fillInDigitsFlag
    )
    {
  if ((!pData) || !(*pData))
        {
        suSyntaxErrFlag = true;
        OBJ_ERROR_PRINT_1("Empty Message can not be parsed.");
        return;
```

```
        }
    ...
    //解析消息头
    if (h248TextDecode.IsChar('!'))
        {
        h248TextDecode.Move(1);
        }
    ...
    else
        {
        suSyntaxErrFlag = true;
        return;    //Error, Megaco token not found
        }
    h248TextDecode.Move(1);    //跳过"/"
    h248TextDecode.DecodeDecimal(megacoVersion);
    ...
//分配事务存储空间
while( (h248TextDecode.stillMore()) &&
            (h248TextDecode.CurrentChar() != '}') )
        { H248Transaction *pTrans = new H248TextTransaction(
                                &h248TextDecode,
                                fillInDigitsFlag);
        if(pTrans) //存在消息体
            {
            if(pTrans->IsSyntaxError()) //是否有消息语法错误
                {
                suSyntaxErrFlag = true;
                }
        ...
        }}}
```

下面对一些函数做进一步的说明，这些函数在后面事务解码和动作解码都会频繁的用到。

IsChar判定当前字符是否是给定的字符“ch”。在对H.248消息解码的时候经常会对当前位置的字符进行判定，特别是在消息中各个字段分界的位置比如事务解码

模块中事务交互与关联之间，当事务交互类型不一样时根据类型的消息结构紧跟事务交互类型参数的字符就有所区别。

```
inline IBool H248TextDecode::IsChar ( char ch )
    {
    if(pos < dataLen) return ( *(pData+pos) == ch );
    return(False);
    }
```

函数Move在检测到字符匹配后将位置标识pos直接跳过已经检测的字符串，在定义消息字符串的时候就已经将字符串的长度定义了，当字符串匹配时而不需要一个一个字符“走”下去减少不必要的操作。

```
inline void Move(UInt32 positionIn){
        pos = (((pos + positionIn) < dataLen) ? (pos + positionIn) : dataLen);};
```

4.3 事务解码层的设计与实现

事务交互，它是由MGC 和MG 之间的一组命令构成，事务交互由TransactionID进行标识。一个事务交互从“事务头部”（TransHdr）开始。在TransHdr 中包含TransactionID。TransactionID肯定是由发起请求消息的一方进行创建的。MGC和MG都是可以创建的。TracsactionID是一个0～999999999的数字并且有一个“三分钟”原则，就是使用过的ID值在三分钟内不能再使用。由于TracsactionID是由9位数字组成，因此重复的情况只在理论上存在。由此在代码设计时我们就认为TracsactionID是“唯一”的。

由前文介绍我们知道事务交互包括请求和响应两种类型，而响应也有两种：TransactionReply 和TransactionPending。下面根据H.248协议规定具体对这几种事务交互类型进行说明。

TransactionRequest

事务交互由TransactionRequest 激发一个事务包含一个到多个动作，每个动作包含一系列与同一个Context相关的一个到多个命令。TransactionRequest结构如下：

TransactionRequest(TransactionId {
ContextID {Command ... Command},
...
ContextID {Command ... Command } })

TransactionReply

TransactionReply 是事务交互接收者对TransactionRequest 的一种响应，表明接

收者完成该TransactionRequest 命令执行，对每个事务交互都应有一个Reply 响应。有两种情况表明一个TransactionRequest 执行完成：

(1) TransactionRequest 中的所有命令成功执行完毕；

(2) TransactionRequest 中的一个非可选命令执行失败。

TransactionReply 结构如下：

TransactionReply(TransactionID {
ContextID { Response ...Response },
...
ContextID { Response ...Response } })

TransactionPending

TransactionPending 由接收者发送，指示事务交互正在处理，但仍然没有完成。当命令处理时间较长时，可以防止发送者重发事务交互请求。TransactionPending 结构如下：

TransactionPending (TransactionID { })

由此可见，事务交互表现为TransactionRequest，对TransactionRequest 接收者响应一个TransactionReply，而这个Reply是必须的。在此之前可能由许多TransactionPending 响应。事务交互的流程如图4.4所示。

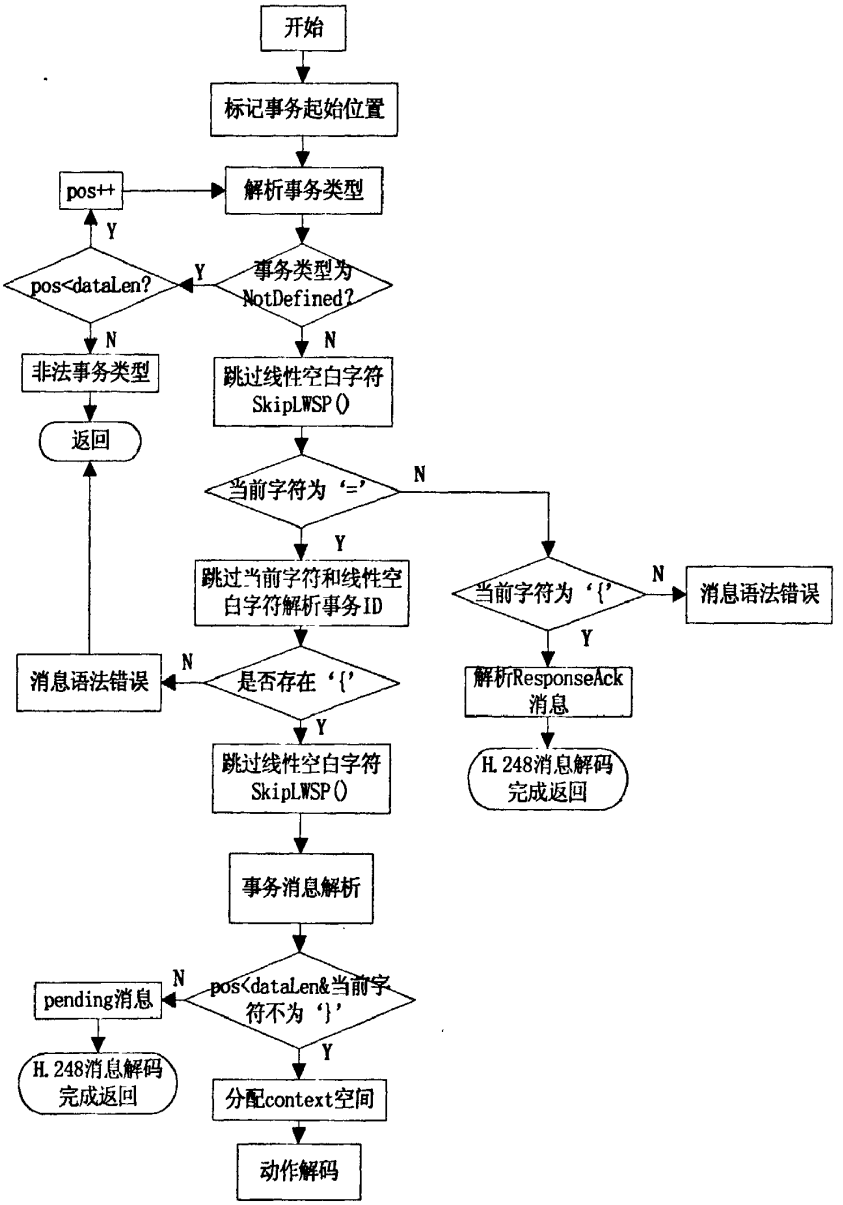

图4.4 事务交互解析流程

在解析事务交互前，我们需要对事务交互的类型做出申明。同样H248_Transaction_NotDefined类型的定义是必要的：

```
enum H248TransactionType
    {
    H248_Transaction_Request = 0,
    H248_Transaction_Reply,
    H248_Transaction_Pending,
    H248_Transaction_ResponseAck,
```

```
    H248_Transaction_NotDefined
    };
class H248Transaction : public FmbManager
    {
protected:
    UInt8 *         pData;          //数据在内存中的位置
    UInt32          dataLen;        //数据长度
    UInt32          transactionStartPos; //事务交互解析开始位置标示
    IBool           suSyntaxErrFlag;  //su 语法错误
    UInt32          transactionId;    //事务交互 ID
    UInt8           transactionType;  //事务类型
    UInt16          transactionErrCode;
    IBool           immAckRequired; //是否需要立即确认
```

这里需要说明的是immAckRequired（需要立即确认），H.248协议规定在事务执行过程中，某些事务执行可能需要较长的时间。从而可能会导致事务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事务被重传多次，或者导致重传定时器数值过大而降低传输的效率。如果协议实体已经预见到某一事务需要较长的执行时间，则它可以先发送一个 TransactionPending临时响应消息来指示该事务正在被处理。当协议实体接收到重复的 TransactionRequest消息，如果该事务正在处理，也应该发送TransactionPending消息。接收到TransactionPending消息的实体必须为TransactionPending消息对应的TransactionRequest消息的重传定时器设定一个不同数值。如果发起TransactionRequest的实体在接收到Transaction Pending消息之后，接收到最终的 TransactionReply消息，则必须立即向对端实体发送一个确认消息，此后，必须重新使用通常的重传定时器。对于发送TransactionPending临时响应消息的实体，必须在发送的最终响应中包含一个“immAckRequired（需要立即确认）”字段，通过这一字段来表明该实体希望能够获得一个立即的确认消息。对于同一个TransactionRequest，协议实体在接收到最终TransactionReply之后，必须忽略接收的TransactionPending消息。所以H.248消息事务解码中定义了IBool immAckRequired; H248_Transaction_ResponseAck则是对响应证实参数（Response Acknowledgement Parameter）的定义。在事务交互类型申明的时候也将ResponseAck单独作为一种事务交互类型来处理。因为含有ResponseAck字段消息结构的特殊性则在对这类H.248消息解码时采用不同的处理方式。

在事务解码模块获取数据消息之后需要先对事务的类型进行解析，事务类型解

析流程如图4.5所示 。

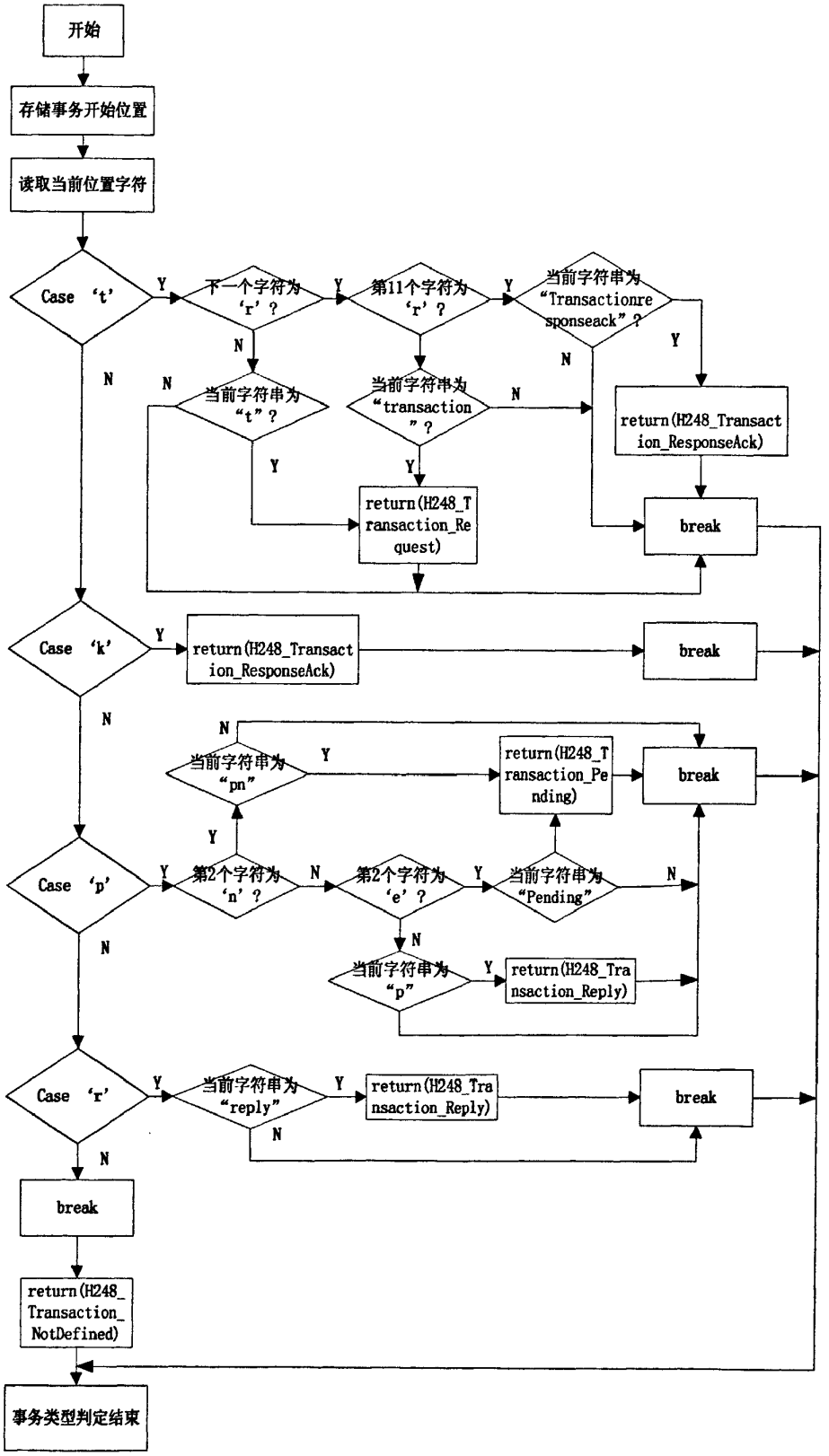

图4.5 事务类型解析流程

遵循ABNF规范中的命名规定H.248消息中事务交互的类型在消息编码中表示如下：

```
...
TransToken = ("Transaction" / "T")
ReplyToken = ("Reply" / "R")
ResponsAckToken = ("TransactionResponsAck " / "K")
PendingToken = ("Pending" / "PN")
...
```

在解析事务交互类型时，依据规范中的规定分别对这些字段进行判定通过switch(h248TextDecode.CurrentCharLc())读取消息中当前指针所指示的字符。以case‘t’为例简单说明解析过程：当前字符为‘t’由于消息编码时采用的简写还是全称在没有对消息解码之前我们并不知道。因此还需要对当前字符之后的字符进行检测，事务类型中当前字符为‘t’的情况只有TransactionRequest和TransactionResponsAck。Transaction在消息中表示TransactionRequest，如果消息是以全称编码，逐一判定的话就需要对当前字符后的10个字符进行判定，这样做带来的的结果是明显的，所以在解析到第二个字符也一样时我们通过Move函数直接跳到第11个字符进行检测就可以避免上述情况，当下一个字符不是‘r’时利用函数将字符。若当前字符不是‘t’则进入相应字符检测，当都不匹配时我们就将这个事务类型归为未定义类型，完成事务类型解析。

在事务交互类型中TransactionResponseAck和Pending消息是特殊的两类消息。对于TransactionResponseAck，其消息结构如下：

```
MEGACO/1 [125.125.125.111]:55555
TransactionResponseAck {
9-13, //TransactionID
...,
50-60
}
```

因此当判定消息类型之后如果下一个字符是“{“， TransactionResponseAck是比较特殊的消息这种消息没有上下文（关联）。通过对类型字段以及后面字符的判断就可以完成对消息的解析。Pending也是没有关联的消息，当监测出事务交互类型是Pending时就不需要进入消息动作解码模块。下面是一个pending消息：

```
MEGACO /1 [124.124.124.222]:2944
PN = 50004 { }
```

对于ParseTransactionReply的解析，由于Reply消息中有关联属性因此在解析了

消息字段之后还需对关联做出判断。当有事务包含关联时，保存关联的原始数据进入动作解码。

4.4 动作解码层的设计与实现

动作解码也可以叫做详细解码是对消息中的关联，命令，描述符中的特性（Property）,事件（Event）,统计（Statistic）等具体参数的解析在对具体参数的解码中我们还是采用字符比较的方法来解码。根据消息的结构先要对 Context Header 进行解码然后是 Context Properties 最后再对 Command 进行解码最终完成 H.248 消息的解析。解析过程涉及到的主要解析参数如图 4.6 所示。

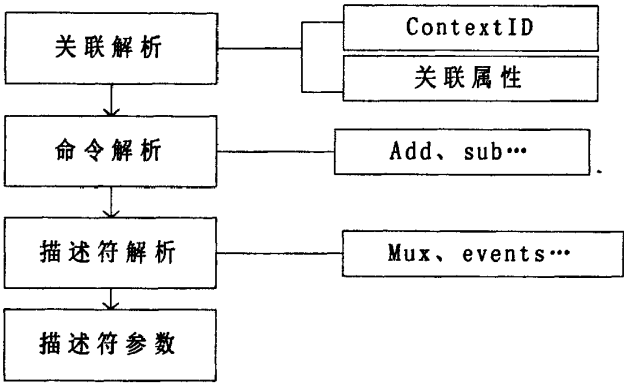

图 4.6 动作解析步骤

我们以文本消息为例说明解析过程。首先对关联进行解析，它以 Context Header 作为关联字段的开始解析流程如图 4.7 所示。

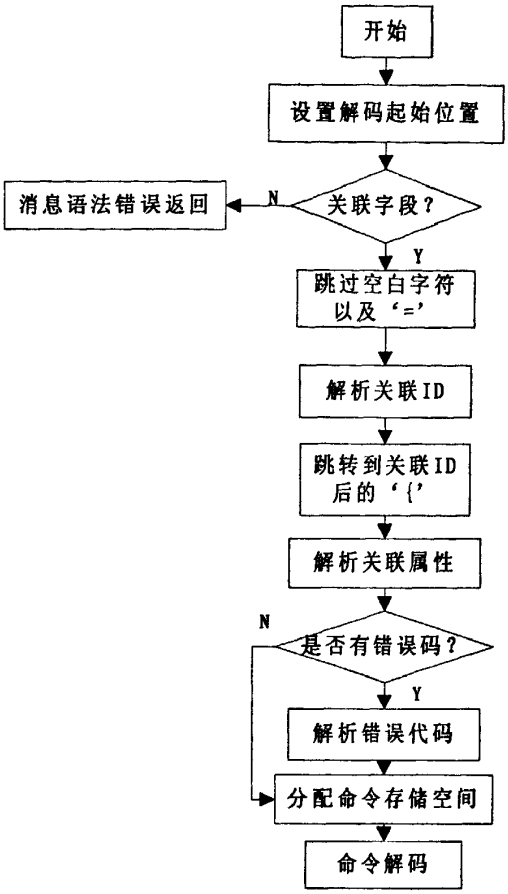

图 4.7 关联解析流程

关联属性以及Context Audit。关联属性时其对应的格式为Topology{ .... } Priority = xxx 或者 Emergency。在解析是时候我们就需要跳过对关联属性函数Move可以快速的进入下一步的解析。同时对于Context Audit它含有且只能含有一对配对的大括号（{}），可以直接跳转到Context Audit的最后。

字符比较时根据ABNF编码中对H.248消息的描述符，命令等的缩写与消息的结构，将存于表中的消息与之对比进而可以判断出消息的描述符，命令等内容从而达到解码的目的。

在协议对关联的定义中我们知道一个由媒体网关（MG）选择的32位整数，在MG范围内是独一无二的三关联标识是由种特殊的关联标识：空关联，choose关联，all关联在对关联解析时就需要对这三种特殊的情况进行解析。

```
case '-'://2D
        contextId = 0;
        break;
case '$': //24
```

```
        contextId = 0xfffffffe;
        break;
case '*': //2A
        contextId = 0xffffffff;
        break;
        default:
        contextId = atoi(tempData);
        break;
```

接下来进入消息命令的解析，H248消息一共8个命令在消息中命令的结构如下所示：

命令名后面的括号中描述的是命令的输入参数，[...]中的参数是可选项。

ADD

ADD(终端ID，[，媒体描述符][，Modem描述符][，MUX描述符][，事件描述符] [，EventBuffer描述符][，信号描述符][，Digit Map描述符][，审计描述符])

Modify

MOD(终端ID，[，媒体描述符][，Modem描述符][，MUX描述符][，事件描述符][，EventBuffer描述符][，信号描述符][，Digit Map描述符][，审计描述符])

Subtract

SUB（终端ID，[，审计描述符])

Move

MOV(终端ID，[，媒体描述符][，Modem描述符][，MUX描述符][，事件描述符][，EventBuffer描述符][，信号描述符][，Digit Map描述符][，审计描述符])

AuditValue

AuditValue(终端ID，[，审计描述符])

AuditCapabilities

AuditCapabilities(终端ID，[，审计描述符])

Notify

Notify(终端ID，观测到的事务描述符，[，Error描述符])

ServiceChange

ServiceChange(终端ID，ServiceChange描述符)

在命令解码的基础类中分别对这8个命令做了申明

```
enum H248CommandType
    {
    H248_Command_Add = 0,
```

```
    H248_Command_Move,
    H248_Command_Subtract,
    ...
    H248_Command_RootAuditCap,
    H248_Command_NotDefined = H248_Command_RootAuditCap + 1
    };
```

通过函数ReadCommandType()获得命令类型。该函数的定义如下：

```
ReadCommandType()
    {
    if((h248TextDecode->CharAfter(1) == '-') &&
            ((h248TextDecode->CurrentCharLc() == 'w')
||(h248TextDecode->CurrentCharLc() == 'o')))
        {
        h248TextDecode->Move(2);
        h248TextDecode->SkipLWSP();
        }
    switch(h248TextDecode->CurrentCharLc())
        {
        case 'a':
          switch(h248TextDecode->CharAfterLc(1))
              {
              case 'd':
                  if (h248TextDecode->IsToken("add",3))
                   {
                   h248TextDecode->Move(3);
                   return(H248_Command_Add);
                   }
                 break; //ADD
        case 'u':
```

在解码的时候根据ABNF中对命令缩写的规定，首先检测第一个字符，根据缩写将其分类检测第二个字符以此来确定命令的类型。如代码中对Add命令解析ABNF规定的缩写是A（不区分大小写），同时AuditValue与AUDITCAPABILITY分别为AC和AV。在解析的时候如果第一个字母是A，继续检测第二个字幕如果是D则一定

是ADD命令，如果第二个字母是U则有可能是AuditValue或者AUDITCAPABILITY命令，由于这两个命令的前5个字母都是一样的因此就跳到第六个字母进行解析。如果第六个字母是V则为AuditValue命令，否则是auditcapability命令。其余命令的解析也按照这种方式归类进行解析。

完成 command 的解析之后就进入描述符的解析 H.248 协议定义了 19 种描述符描述符的形式如下：

DescriptorName=(someID)

{parm=value, parm=value,……}

因此对这些描述符进行了申明

```
enum H248ParserAddressDigitsType
    {
    H248ParserAddressDigitsType_Local,   //0
    H248ParserAddressDigitsType_Remote,
    H248ParserAddressDigitsType_Statistics,
    ...
    H248ParserAddressDigitsType_LocalControl,
    H248ParserAddressDigitsType_Comment,
    H248ParserAddressDigitsType_NotDefined
};
```

由于描述符种类繁多本文就不一一列出，仅以 ServiceChange 中的 ServiceChangeMethod 参数为例讲述描述符参数的解析过程。ServiceChangeMethod（MT）参数指示将要发生或已经发生的 ServiceChange 的类型，该参数规定 MG 发生业务改变的 6 种方式：

Graceful ：指示终端将在延迟 ServiceChangeDelay 之后离开服务；已经建立的连接暂不影响，但 MGC 将避免新建连接并试图文明关闭已存在连接。

Forced ：指示终端突然中断服务，已建立的连接丢失。

Restart：指示指定终端在延迟 ServiceChangeDelay 之后重起。

Disconnected：拆线方式适用于根终端。用来指示 MG 曾中断与 MGC 的通信连接但是随后连接又重新恢复。因为 MG 的状态发生改变，所以 MGC 可以审计命令来使 MG 与 MGC 重新同步。

Handoff：当该参数由 MGC 发送给 MG ，用于指示 MGC 将退出服务，MG 必须与一个新的 MGC 建立新的连接；当该参数从 MG 发送给 MGC 时，指示 MG 试图与新的 MGC 建立新的连接 。

Failover：该参数从MG 发送给MGC，指示主控MG 将退出服务，备用的MG 将

开启服务。对servicechangemethod的参数做了申明。

```
static const char *H248SChgMethod[] = {  "failover", "forced", "graceful",
                                         "restart", "disconnect", "handoff"};
static const int H248SChgMethodLen[] = {8, 6, 8, 7, 10, 7};
```

通过 ParseMethod 函数对消息中携带的参数进行解析。

```
H248TextCommand::ParseMethod (void)
    {
        if (commandType != H248_Command_ServiceChange)
        return Aok;
    h248TextDecode->SkipLWSP();
    if(!h248TextDecode->IsChar('=')) return Error;
    h248TextDecode->Move(1);
        h248TextDecode->SkipLWSP();
        int indexOfMethod = -1;
    switch(h248TextDecode->CurrentCharLc())
        {
          case 'f':
            if(h248TextDecode->CharAfterLc(1) == 'a') indexOfMethod = 0;
            else indexOfMethod = 1;
          break;
                ...
            default:
        break;
        };
        if(indexOfMethod != -1)
        {
              if(h248TextDecode->IsToken(H248SChgMethod[indexOfMethod],
                                  H248SChgMethodLen[indexOfMethod]))
              {
248TextDecode->Move(H248SChgMethodLen[indexOfMethod]);
                    serviceChgMethod = indexOfMethod;
              }
              else h248TextDecode->SkipToken();
        }
```

```
        else h248TextDecode->SkipToken();
    return(Aok);
}
```

解析开始先判定是否是 ServiceChange 命令，不是则返回否则继续解析。跳过不必要解析的控制字符，还是通过字符比较的方式解析参数。直到解析完消息中携带的所有描述符及与其对应的参数完成 H.248 消息的解码。

4.6 本章小结

首先介绍了软件模块的设计思想和实现方法，通过对 H.248 消息的结构和消息构成机制的分析将消息分为三个部分消息头、消息事务和消息动作。通过这种划分可以保持消息这三个部分的独立同时也满足解码模块上层模块在消息重构和消息“配对”功能上对解码模块的要求。在此基础之上采用分层的思想将网络管理系统中 H.248 协议解码模块分为三层即：消息预解码层、事务解码层和动作解码层。解码模块采用分层解码的方式实现 H.248 协议的解码。在此思想的指导下重点介绍了消息预解码的实现、事务解码的实现以及动作解码的实现，最终实现 H.248 消息的详细解码。同时为了降低代码实现的复杂度消息详细解码的过程采用了顺序解码的方式。并对消息预解码模块中采用了一种基于 ASCII 码表快速划分 H.248 消息类型的机制做了阐述。

第五章 H.248 解码模块功能测试与分析

5.1 软件测试的方法

软件测试对于保证软件产品的可靠性和安全性是十分重要的，在软件设计过程中不可避免的会出现问题，测试目的是通过测试及时发现软件设计过程中存在的问题。并及时解决这些问题。同时在测试的时候也要根据产品的功能和用途在产品测试过程中尽最大可能的模拟产品在实际应用中的环境。以免在产品使用过程中出现问题造成损失。

因此测试用例的编写就显得很重要了。在测试用例中要首先要明确需要测试的功能，当输入测试数据时能够预见测试的结果。尽可能多的使用不同的测试数据，做到全面的测试。同时也要注意测试的效率和成本问题，在测试时应该选用高效的测试数据。测试的环境也是非常重要的，测试环境的正确与否这节影响到测试的结果。比如在本文对解码模块进行测试，由于 H.248 协议传输可以基于 IP 或者 ATM。在搭载测试环境时一定要明确测试时的协议栈否则在查看测试结果的时候，显示界面显示不出解码的结果，在分析问题的时候会带来很大的不便。

软件测试方法分为：白盒测试、黑盒测试和灰盒测试。白盒测试也称结构性测试，测试人员必须了解程序内部的逻辑结构，分别对程序内不同位置的运行状态进行检查。这包括中间状态以及最终结果，对于本论文的解码模块中分别要注意的是对解码过程中消息头，事务解码以及最终的动作解码这三个部分能否正常运作。特别是消息数据的传递过程，否则一旦数据传送失败就无法对后面的解码模块进行测试。例如：在消息头解码完成时能否正常的将事务交互的数据保存下来并传递给事务解码模块。黑盒测试又称功能测试，测试人员不需要知道软件的内部结构只需要知道软件所要实现的功能，仅在软件接口进行测试。通过输入测试用的数据查看输出结果。灰盒测试和黑盒测试一样通过用户界面测试，但是测试人员已经对被测试软件有所了解。测试的时候能够通过对软件结构的了解有针对的进行某一方面的功能测试。

5.2 测试环境的搭建

测试环境是很重要的任何软件都会在一定环境下工作，工作环境。解码模块应用于网络管理系统中。网络管理系统，这是泰克为管理下一代融合网络提供的业内领先的一体化保证应用套件的一部分。平台为执行实时、多协议、跨架构性能监测提供了广泛的功能，同时使得网络运营商能够监测 IP 融合传送网络中的新接口，包括 Gb/IP、IuFlex 和 SIGTRAN 承载的 A 接口。

针对产品的应用测试环境如图所示。

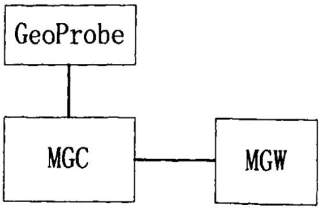

图 5.1 测试结构

监测媒体网关控制器，采集里面的消息数据进行处理。

首先要设置网络管理系统中的配置文件使其可以监测 H.248 消息，如图所示同时在服务器上可以直观的将监控的设备显示出来。

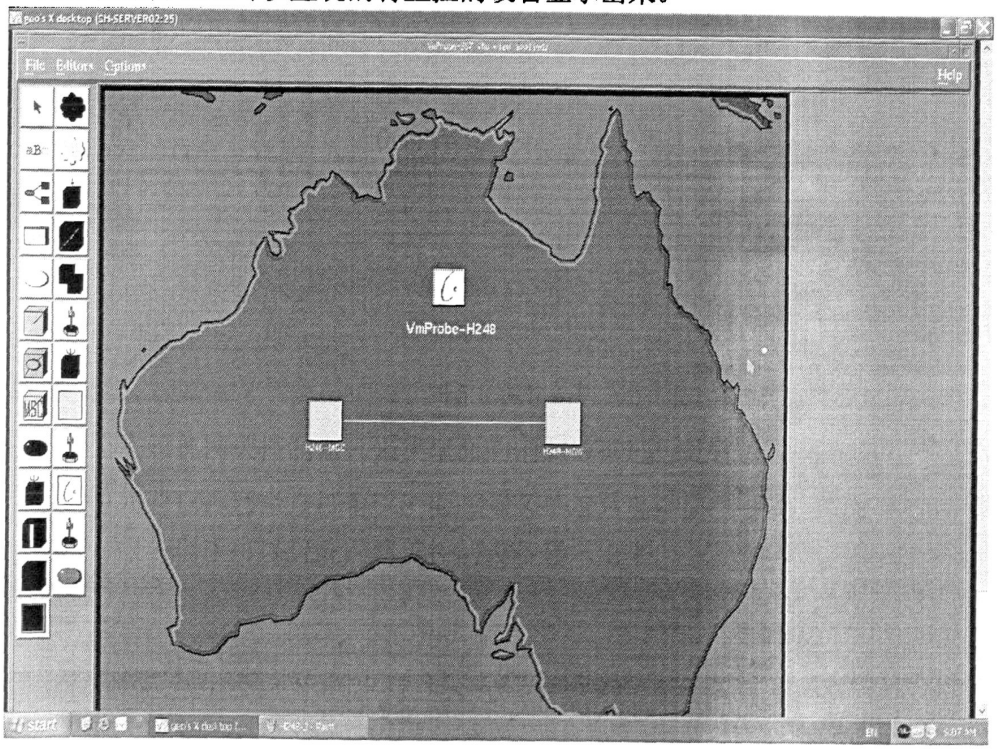

图 5.2 管理系统用户界面

左边的是 MGC 右边是 MG，网络管理系统用于检测 MGC 与 MG 之间的通信。基于 IP 传输时底层传输机制将采用 SCTP 为其提供协议承载。在搭建测试环境时就需要注意。

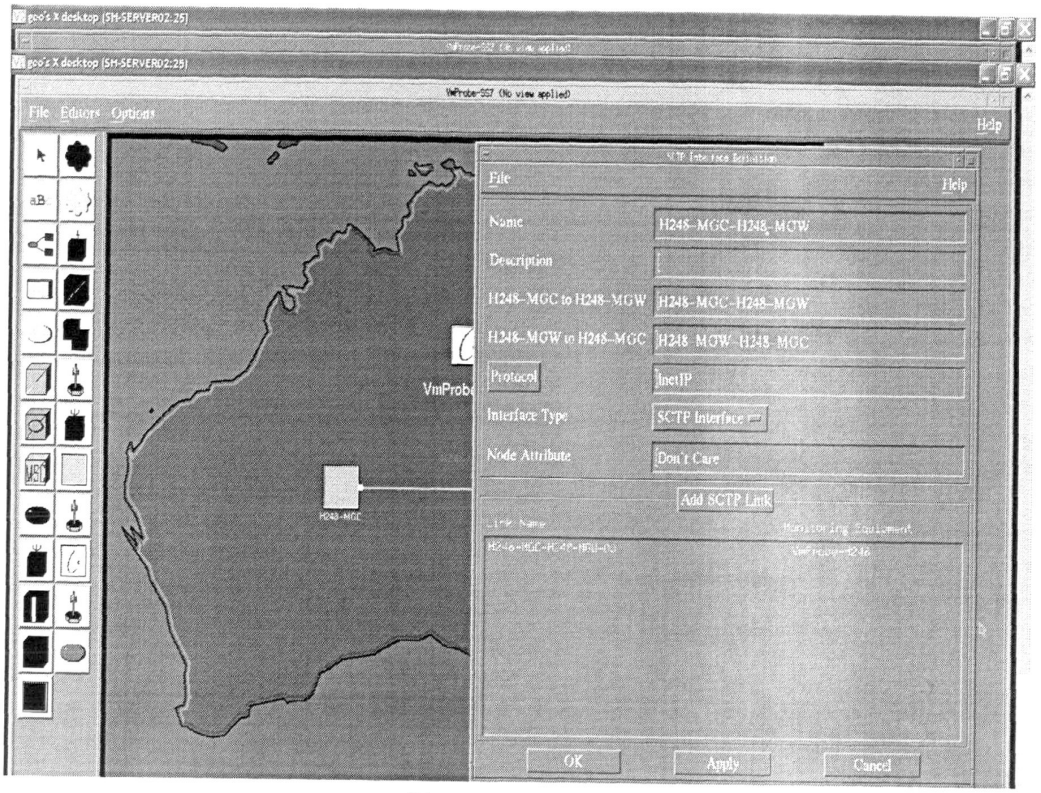

图 5.3 基于 IP 传输

5.3 H.248 消息解码测试结果展示与分析

MGC 与 MG 之间有通信时网络管理系统捕获消息信号并对其处理分析。通过用户界面显示给用户，如图所示。

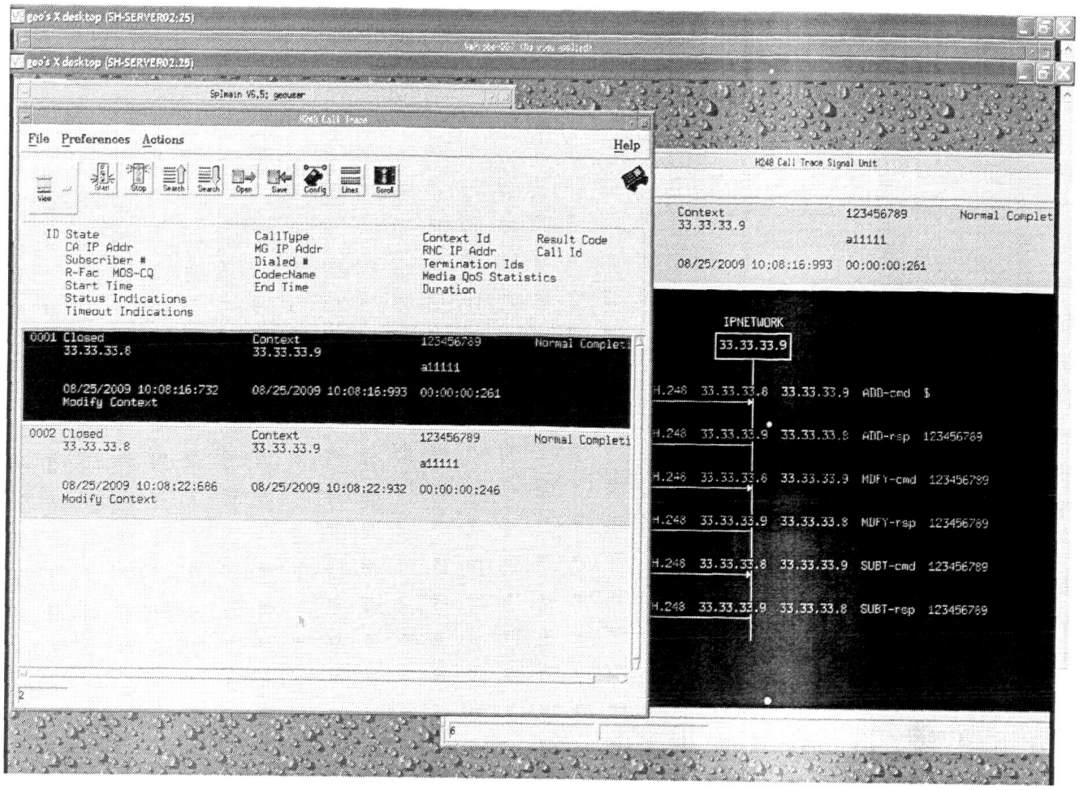

图 5.4 消息接收

通过动作解码事务关联模块提取出 contextID，在获得的消息中消息状态为 Closed，表明这是一条完整的消息，即每一个请求都有相对应的响应消息。

双击消息将弹出呼叫跟踪功能界面，通过呼叫跟踪功能的演示我们可以读出该消息是基于 IP 传输的 H.248 消息。解码模块能够准确的解析出呼叫跟踪功能所需要的参数：发送者和接收者的 IP 地址(通过消息预解码层解析获得)，在呼叫跟踪功能中 ADD-cmd 表示事务交互类型为请求命令类型为 ADD 命令，ADD-rsp 则是对 ADD 请求消息的响应。这两个参数通过事务解码层和动作解码层解析。

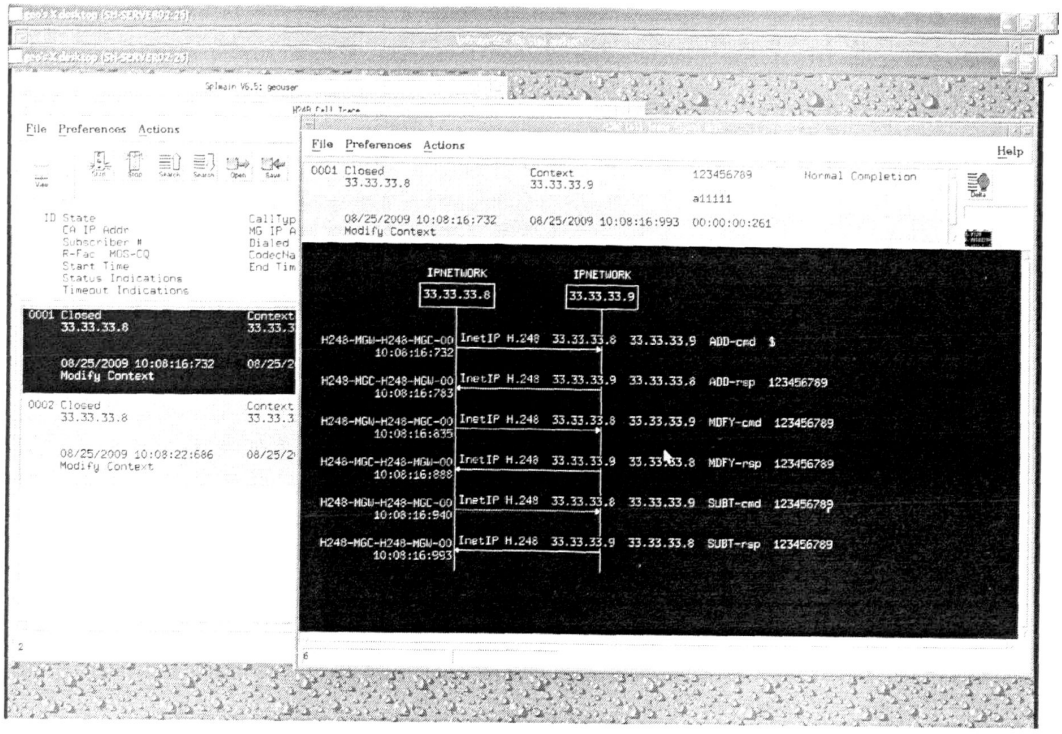

图 5.5 呼叫跟踪

双击对应的消息就能够看到详细解码的消息。

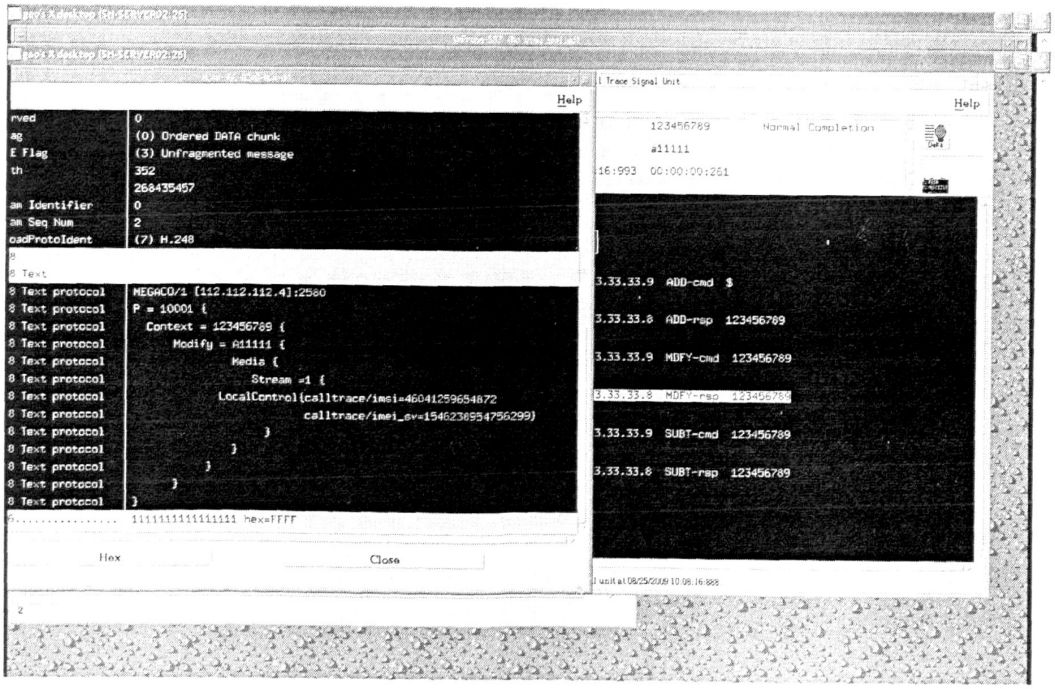

图 5.6 消息详细解码

图 5.6 展示的是对单条 H.248 消息的详细解码。该消息为 calltrace 包中的 imsi(国际移动用户识别码)和 imei(国际移动设备身份码)信息。

5.4 本章小结

本章对解码模块在网络管理系统中的实际运行过程进行了图例分析。通过对网络管理系统呼叫跟踪功能的演示我们可以看到解码模块各层都能为其提供实现功能的参数。通过对详细解码结构分析解码模块能够准确地解析H.248协议消息。

第六章 结 论

6.1 全文总结

论文的研究课题是基于网络管理系统。本论文从理论和实践两方面入手，对H.248 协议标准、协议编码方式、协议的消息结构和构成机制等进行了较为深入的研究和探索，基于网络管理系统体系架构的设计思想，提出了 H.248 协议解码模块在网络管理系统中的设计思路，并且实现了 H.248 协议解码并成功用于呼叫跟踪及统计分析等监测功能。

本论文所研究和开发的H.248协议监测模块已应用于泰克公司的网络管理系统中。目前，该产品已正式投入商用，并且通过了全国多家运营商和研究院的测试，取得了良好的效果。

6.2 下一步的工作

随着网络的发展，网络融合的应用会越来越广泛，因此本课题仍然有需要进一步研究和实践的地方。

1) 由于 H.248 协议的良好的扩展性，导致各设备厂商均有自己的私有协议标准，因此对于私有协议标准的解码需要进一步跟踪和完善；
2) 随着 H.248 协议标准的不断升级，解码模块也需要进行相应的升级和扩展；
3) 根据用户的需求在解码模块中加入可选属性的解析。

致 谢

本论文是在导师何丰教授亲切关怀和悉心指导下完成的。何老师渊博的知识、严谨的治学态度和勤奋的工作作风给我留下了深刻的印象，并深深地感染和激励了我。三年来，何老师不仅在学业上指导了我同时还在生活中和思想上给予了我无微不至的关怀，特别是在思想指导方面将对我今后的发展产生深远的影响。在此谨向何老师表示衷心的感谢和崇高的敬意！

同时感谢重庆邮电大学实验室和通信学院的各位老师、以及实验室的全体师兄师弟们给我多方面的帮助！

感谢父母多年来含辛茹苦的养育之恩和亲朋好友的鼓励与支持，使我能顺利的完成学业！

参考文献

[1] 赵慧玲, 叶华. 以软交换为核心的下一代网络技术[M]. 北京：人民邮电出版社. 2002，8. 4-6

[2] 科林斯. VoIP 技术与应用[M].北京人民邮电出版社.2003，05. 106-114

[3] 赵学军. 软交换技术问答[M]. 北京：人民邮电出版社. 2005, 11. 3-5

[4] 叶华. 软交换技术的研究[J]. 宽带世界. 2003, 1-2：18-22

[5] 高艳玲, 马玉霞. 软交换技术及应用浅析[J]. 计算机与网络. 2006, 6.：32-35

[6] ITU-T Recommendation H.248.1(2005) Gateway control protocol[S]

[7] IETF RFC3525(2003), Gateway Control Protocol[S]

[8] 李俊杰, 梁满贵, 简锐锋. H.248 / Megaco 协议在下一代呼叫中心中的应用[J]. 中国数据通信. 2005, 6.：22-25

[9] YD/T 1292-2003,中华人民共和国通信行业标准：基于 H.248 的媒体网关控制协议技术要求 [S].

[10] 3GPP TS 29.232: " Media Gateway Controller (MGC) – Media Gateway (MG) interface[S]; Stage 3"

[11] ITU-T Recommendation X.680 (2002) ISO/IEC 8824-1:2002, Information technology – Abstract SyntaxNotation One (ASN.1): Specification of basic notation.

[12] ITU-T Recommendation X.690 (2002) ISO/IEC 8825-1:2002 ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)[S].

[13] IETF RFC 2234 (1997), Augmented BNF for Syntax Specifications: ABNF. [S]

[14] 软件工程导论，张海潘，清华大学出版社，第四版

[15] 张慧媛, 杨放春, 詹舒波. 媒体网关控制标准 Megaco/H.248 协议的分析与研究 [J] 北京邮电大学学报. 2002, 9:15-19

[16] 田战毅, 王荧蓉, 莫益军. 软交换呼叫模型的研究与设计[J]. 电信快报 网络通信. 2005, 1.:33-36

[17] ITU-T Recommendation H.248.4 (2000) Corr.1 (2004), Gateway control protocol: Transport over Stream Control Transmission Protocol (SCTP)[S].

[18] ITU-T Recommendation H.248.5 (2000), Gateway control protocol: Transport over ATM.[S]

[19] IETF RFC 2327 (1998), SDP: Session Description Protocol.[S]

[20] ITU-T Recommendation H.248.8 (2005), Gateway control protocol: Error code and service change reason description.[S]

[21] ITU-T Recommendation H.248.15 (2002), Gateway control protocol: SDP H.248 package attribute[S]

[22] YD/T 1309-2004 中华人民共和国通信行业标准, 与承载无关的呼叫承载控制规范[S].

[23] 徐红军, 糜正琨. H.248 协议的实现及其应用[J]. 数据通信. 2005, 4:42-45

[24] 李健芳. H.248 协议在软交换网络中的应用[J]. 电信网技术. 2002, 12:19-22

[25] 王明昌, 黄瑞光. Megaco/H.248 协议在下一代网络中的应用[J]. 无线电工程. 2002, 2:36-38

[26] 姚辉军, 薛海英. H.248 协议在 VoIP 中的可靠机制和安全机制[J]. 现代电信科技. 2006, 5:42-45

[27] 徐书华, 黄本雄, 徐丽娜. MEGACO 协议栈高可用性实现方案研究[J]. 网络技术. 2005, 9:60-63

[28] 吴乃星, 朱晓民, 廖建新. 支持分布式应用的 Megaco/H.248 协议栈系统的研究和实现.[J] 现代电信科技. 2004, 6:46-49

[29] 孙鑫,余安萍等. VC++深入详解[M]. 北京：电子工业出版社. 2007,3. 354-356

[30] Stephen Prata. C++ Primer Plus(第五版)中文版[M]. 北京：人民邮电出版社. 2005.2. 263-265

[31] Robert C.Martin.敏捷软件开发原则、模式与实践[M].北京：清华大学出版社, 2004.2

附 录

1. 发表的论文

[1] 何西良，高益. ASN.1 编解码运行库系统的研究和设计，电子测试，2009.11.

[2] 何西良，王庆敏. 软交换系统中 H.248 协议解码研究与实现，中国新通信，2010.2.