摘 要

近些年来，随着通信网络核心网的不断演进，带宽不断拓展，业务内容、业务种类也不断丰富。网络的发展集中呈现出：宽带化、多媒体化、IP 化等趋势。无线通信的网络结构和接入技术均在朝着以上这三个目标不断演进，并完成了从实验室技术理论到实际网络实现的过程。

与此同时，软交换技术的不断成熟使网络的业务功能与底层传送技术在“物理上”分离开来、不同媒体类型的业务交互的实现变得简单、不同网络之间的互联互通变得可能。使用软交换技术组网也成为电信运营商的必然选择。

在移动通信网 3G 标准之一的 WCDMA 网路中，其核心网络（UMTS 系统）R4 版本中出现了新的接口——Mc 接口。软交换技术正是通过该接口在 UMTS 系统的网络结构和系统运行上得到体现。对于 WCDMA 网络而言，核心网电路域的 Mc 接口控制了网络内各媒体类型通信业务的运行，对它的监测是整个 WCDMA 核心网监测的关键。

本文首先介绍了 WCDMA 的特点和网络结构，并对 Mc 接口监测的必要性与监测要求进行了分析；然后对 Mc 接口上的 H.248 协议做了研究，并着重分析该协议的连接模型、消息结构以及它在 Mc 接口上的应用；接着介绍 WCDMA 网络测试仪的体系结构，包括测试仪的软件体系结构、数据处理流程和多线程结构，并重点介绍了 Mc 接口监测模块在测试仪中的位置和作用；随后，文章从 BER 编码规范和 ABNF 语法规范入手，详细阐述对以二进制方式、文本方式两种方式编码的 H.248 协议数据的解码功能的设计和实现；最后，文章对 Mc 接口上的业务流程作了研究和分析，根据对典型业务流程的分类总结了呼叫合成的关键信息（Key 值），提出了 Mc 接口监测模块的呼叫合成方案，并根据该方案从“类”的封装设计这一角度阐述了呼叫合成与统计分析功能的实现；并根据网络、业务未来的发展对下一步的工作做了探讨和展望。

论文用现网数据对该模块的实际运行效果进行了测试，结果表明本监测模块达到了测试期望，已经具有商用水平。

关键词：WCDMA，网络测试，Mc，H.248/Megaco，CDR

Abstract

With the constant evolution of core network, bandwidth continues to expand, the contents and types of service continuously enriched. Network's development has been concentratly known as broadband, multimedia, mobile and so on. In such a trend, both the architecture and access technology of wireless communication network are developing towards those goals. And their technical theories have been applicated into the existing networks from laboratory.

At the same time, soft-switching technology has matured. It physically separate network's service functions from underlying transmission technology. The intracions between different types of services became simple. The intercommunications between different networks became possible. Using soft-switching technology to organize their own network has become Network operators' inevitable choice.

WCDMA which is one of the 3G standards has add a new interface in its core network (UMTS system). By this addition, the soft-switching technology was imported into the network structure of WCDMA and its system's operation. Mc interface controls all types of multimedia services in core network's Circuit Switched Domain. Monitoring Mc interface is the key to monitoring WCDMA's core network.

This article first introduced the characteristics and network structure of WCDMA. It analyzed the necessity and damends of monitoring Mc interface. After that it did the thorough research on the H.248 protocol standard which is using on Mc interface. Then it introduced the system structure of WCDMA network tester including its data processing and multi-threading structure, specially focused on Mc Monitoring Module's function and position in the whole tester. Based on BER and ABNF, this article elaborated the design and implement of H.248 data's decoding function for both binary mode and text mode. In the last chapter, articl did a thorough research of the service processing on Mc interface. According to the result, article reached the critical values for synthesizing Calling Details by classified service processing's types. It proposed CDR's synthesizing scheme, then designed and implemented the CDR synthesizing function on the basis of scheme.The elaborating of the design and implement was from the perspective of class packaging. The end of the article made a summary of the text, explored and prospected the next phase of work by analyzing network's future development.

Articl has run some tests about Mc Monitoring Module's functions by using datas

from existing network. The result shows that this moudle's operating effect has met our expectation, reaches commercial level.

Keywords: WCDMA, Network Testing, Mc, H.248/Megaco, Calling Detail Records

图表目录

略缩词表

英文缩写	中文译文	英文全称
WCDMA	宽带码分多址	Wideband Code Division Multiple Access
CDMA2000	码分多址 2000	Code Division Multiple Access 2000
TD-SCDMA	时分同步码分多址	Time Division-Synchronous Code Division Multiple Access
3G	第三代移动通信	3rd-generation
IP	互联网协议	Internet Protocol
GSM	全球移动通讯系统	Global System for Mobile Communications
MAP	移动应用部分(协议)	Mobile Application Part
UMTS	通用移动通信系统	Universal Mobile Telecommunications System
UTRAN	全球路上无线接入	Universal Terrestrial Radio Access
MSC Server	移动交换中心	Mobile Switching Center Server
MGW	媒体网关	Media GateWay
ISDN	综合业务数字网	Integrated Services Digital Network
ITU	国际电讯联盟	International Telecommunication Union
IETF	互联网工程任务组	Internet Engineering Task Force
TDM	时分复用和复用器	Time Division Multiplex and Multiplexer
ATM	异步传输模式	Asynchronous Transfer Mode
SCTP	流控制传输协议	Stream Control Transmission Protocol
M3UA	MTP 第三级用户的适配层协议	MTP3-User Adaptation layer
MTP-3b	信息传输部分第 3 级	Message Transfer Part level 3
UDP	用户数据包协议	User Datagram Protocol
H.248/Megaco	媒体网关控制协议	Media Gateway Control protocol
MGC	媒体网关控制器	Media Gateway Controller
MG	媒体网关	Media Gateway
PDU	协议数据单元	Protocol Data Unit
SDU	服务数据单元	Service Data Unit
BER	基本编码规则	Basic Encoding Rules

PER	紧缩编码规则	Packet Encoding Rules
CER	正则编码规则	Canonical Encoding Rules
TLV	类型-长度-值	Type-Length-Value
IANA	The Internet Assigned Numbers Authority	互联网数字分配机构
DTMF	Dual Tone Multi Frequency	双音多频信号
ABNF	扩展巴斯克范式	Augmented Backus-Naur Form
RE	正则表达式	Regular Expression
CDR	呼叫详细记录	Calling Detail Records
UE	用户设备	User Equipment
SG	信令网关	Signaling Gateway

第一章 绪 论

1.1 研究背景

随着软交换等技术的深入应用，网络的发展集中呈现出：宽带化、多媒体化、IP 化等趋势。移动通信系统的网络结构和接入技术在这种趋势的影响下也不断演进，并完成了从实验室技术理论到实际网络实现的过程。在国内，以 WCDMA、CDMA2000、TD-SCDMA 为标准的 3G 网络也各自开始进入运营阶段，从而满足移动通信用户日益增长的多媒体化、宽带化的业务需求。随着网络架构的发展和技术的提升，业务量和复杂度也在不断提高。在这种发展背景下，移动通信运营商如何通过合理有效的技术监测网络运营情况，保障网络的正常运行是运营商和网络测试公司在下一阶段的运营监测中必须解决的关键问题，其中对于核心网络的监测又是这一关键问题的核心。对于 WCDMA 网络而言，其核心网络内的 Mc 接口控制了网络内各种媒体类型通信业务的运行，对它的监测成为整个核心网监测的关键。

1.2 WCDMA

1.2.1 WCDMA 特点

WCDMA[1]由欧洲ETSI 和日本ARIB 提出，它的核心网基于GSM-MAP信令结构。WCDMA采用直接序列扩频码分多址（DS-CDMA）、频分双工（FDD）方式，码片速率为3.84Mcps，载波带宽为5Mhz。系统能同时支持电路交换业务（如PSTN、ISDN）和分组交换业务（如IP 网），该系统使用灵活的无线协议可在一个载波内同时支持话音、数据和多媒体业务，并通过透明或非透明传输来支持实时、非实时业务。

①扩频码

在 WCDMA 系统中采用长的扩频码，不同的小区和上行链路中不同的用户由不同的扩频码来区分。在下行链路中用Gold 码，便于小区的快速搜索。上行链路采用短码和扰码，易于实现多用户接收机技术。扰码为Gold序列，对于信道的区分则采用正交码，如树形结构的正交码。

②多码率

同一连接的多种服务可以复用在一个专用物理数据信道（DPDCH）上。在服务的多路复用和信道编码之后，多路复用的数据流映射到一个DPDCH上。如果总的数据率超过了一个码分信道的传输上限，则可以将多路复用映射到不同的DPDCH上，

分配多个DPDCH实现一种多码道的方式，分别进行编码及交织。此时，功率和每个服务的质量均可分别独立地进行控制。

③分组数据

WCDMA 提供了数据包发送机制。短的数据包可以直接附加在随机接入信道中，这种方法称为公共信道分组传输。更大和更经常进行的数据传输则在专用信道上完成，单个大的数据包用单数据包方式传输，当数据传送完成时，专用信道立即释放。在多分组方式传输时，专用信道将通过功率控制及同步信息保持来维持专用信道。另外，WCDMA 还可支持基站间的异步操作。随着技术的不断发展，该系统还可支持自适应天线阵技术和多用户检测等技术。

1.2.2 WCDMA 网络结构

WCDMA 采用 UMTS[2]系统的网络结构，其结构与第二代移动通信系统类似。UMTS 网络结构主要由用户终端（UE）、无线接入网络（UTRAN）、核心网络（CN）和运营支撑子系统（OSS）组成，能够由现有的 GSM 网络平滑演进。

无线接入网（UTRAN）处理所有与无线有关的功能；核心网（CN）处理 UMTS 系统内所有的话音呼叫和数据连接，并实现与外部网络的交换和路由功能。在逻辑上可将 CN 分为电路交换域（CS）和分组交换域（PS）和业务应用域。UMTS 系统的 R99 版本和 GSM 的网络结构相比，在核心网侧的变化不大，只是在无线侧引入了一些新的接口。而从 R4 版本开始，核心网开始了向 NGN 网络的演进，其核心网结构如图 1.1 所示。

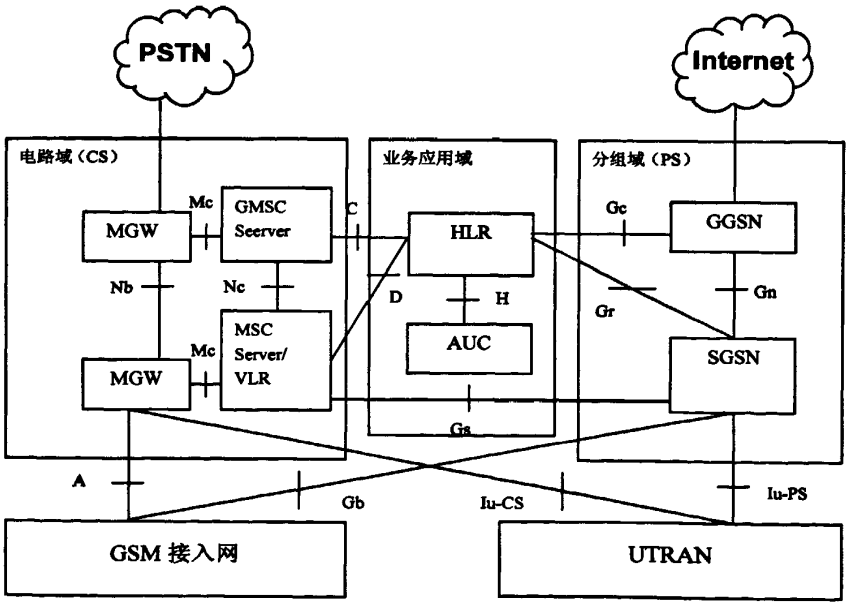

图 1.1 R4 版本的核心网结构

如上图所示，R4 版本的核心网分为电路域、分组域和业务应用域。电路域为用户提供“电路型业务”或相关信令连接的实体，其包含的网元设备有：移动业务交换中心服务器（MSC Server）、MGW（媒体网关）、拜访位置寄存器（VLR）、网关移动业务交换中心（GMSC Server）。分组域为用户提供分组数据业务，其包含的网元设备有：业务 GPRS 支持节点（SGSN）、网关 GPRS 支持节点（GGSN）。业务应用域提供了业务支撑的功能，其包含的网元设备有：鉴权中心（AuC）、归属位置寄存器（HLR）、短消息中心（SC）和设备标识寄存器（EIR）。业务应用域的网元设备供电路域和分组域共同使用。

在图 1.1 中可以看到，GSM 核心网中的移动交换中心（MSC）已经分离为移动业务交换中心服务器（MSC Server）和 MGW（媒体网关）这两个系统网元。这一点的改动对于 UMTS 核心网的发展有重要的意义，它将承载和传输功能从传统的 MSC 中分离出来，交由 MGW 完成各类型媒体的转换，传输，交互等功能；而 MSC Server 只完成对电路域交换业务的控制、用户信令处理、地址更新、位置登记、呼叫连接和安全性管理等功能。

经过这一分离之后，MSC Server 和 MGW 之间引入了新的接口——Mc 接口，MSC Server 正是通过这一接口使用 H.248 协议来完成对 MGW 的控制。UMTS 系统就此引入了软交换技术，并在网络结构上完成了控制与承载的分离，整个核心网也朝着 NGN 网络开始演进。

1.3 Mc 接口测试

1.3.1 Mc 接口的协议栈结构

如上所述，Mc接口[3]是3GPP R4版本中的新增接口，它是MSC Server(或GMSC Server)与媒体网关MGW之间的标准接口，其协议遵从H.248协议。

Mc接口的物理接口可选择ATM、IP或TDM方式；接口采用的H.248协议消息包含二进制或文本方式编码方式；底层传输机制采用MTP-3b(基于ATM 的信令传输)或SCTP(基于IP的信令传输)为上层提供协议承载。

Mc接口的协议栈结构有：

①纯IP连接[4]时，协议栈为H.248/SCTP/IP，也可将M3UA加在SCTP之上。为了更好地与基于H.248的软交换系统进行互通，在纯IP连接时，可选采用UDP承载，即H.248/UDP/IP；

②纯ATM连接[5]时，协议栈为H.248/MTP3b/SSCF/SSCOP/AAL5/ATM；

③混合ATM，IP连接时，协议栈为H.248/M3UA/SCTP/IP。

1.3.2 Mc 接口的监测

Mc接口向MSC Server(或GMSC Server) 提供了在呼叫处理过程中控制MGW中各类静态、动态资源(IP / ATM / TDM)的能力(包括终端属性、终端连接交换关系及其承载的媒体流)；提供了独立于呼叫的MGW状态维护与管理能力。它的接口特点如下：

①存在支持不同呼叫模型的灵活的连接处理以及使H.323用户使用不受限制的不同媒体的处理；

②开放的结构支持该接口的包定义和扩充；

③MGW物理节点资源的动态共享，一个物理的MGW 可以分割成多个相互独立的逻辑MGW；

④根据H.248协议实现在MGW 控制的承载和管理资源之间的动态传输资源共享；

⑤支持移动通信网络特定的功能，如SRNC重定位 / 切换等。

在一次呼叫的处理过程中，MSC Server 与 MGW 之间通过 H.248 消息进行控制、交互与统计，完成链路的建立、维持、修改、释放和拆除，进而实现 WCDMA 核心网电路交换的完成。

因此，对于 Mc 接口的监测是对整个核心网电路域监测的关键。对于它的监测包括以下主要内容：[6]

①对不同传输承载的数据采集。

②对 H.248 协议的解码实现。

③对接口所涉及的各媒体类型业务的呼叫流程重现。

④对话务量，掉话率等各关键数据的统计与分析。

1.4 本文主要研究的内容

本论文主要研究的内容来源于研发项目——3G 网络测试仪（WCDMA 网络测试仪），本人在该产品开发过程中参与并担任了 Mc 接口监测模块的 H.248 协议解码及合成的研发工作。

WCDMA 网络测试仪是根据 ITU 和 IETF 的相关协议标准研制而成的。它通过对采集数据进行解码、呼叫流程分析、呼叫统计、呼损分析、话务统计、语音链路质量分析等操作，为 WCDMA 网络运营商和设备生产商提供准确、有效、便捷、实用的测试手段；适用于 WCDMA 网络的日常运营维护、故障定位排除，网络设备运行状态分析，业务开通的配合测试，以及 WCDMA 网络设备研制开发的协议、性能和一致性测试等。

作为 WCDMA 核心网电路域的主要接口，在该接口上采集到的 H.248 协议数据

对整个核心网电路域的网络运行状况监测有着十分重要的意义。该接口的信令解码、呼叫合成及统计等监测功能是整个 WCDMA 网络测试仪不可或缺的功能。本论文正是基于此，重点介绍了 Mc 接口监测模块的设计与实现。

本文共分为六章，章节内容安排如下：

第一章介绍了本论文研究课题的背景、Mc 接口在 R4 版本的 WCDMA 网络中的位置、功能和作用，并阐述了 Mc 接口监测技术对于 WCDMA 网络测试的研究目的和意义，介绍了本论文主要研究工作。

第二章对 H.248 协议规范及其特点做了研究。

第三章介绍了 WCDMA 网络测试仪系统体系结构以及 Mc 接口监测模块在其中的位置及作用。

第四章详细描述了 Mc 接口监测模块解码功能的设计与实现。

第五章详细描述了 Mc 接口监测模块呼叫合成、统计分析功能的设计与实现。

第六章总结了本论文所做的工作，并探讨了下一步的研究重点与方向。

第二章 H. 248 协议研究

H.248 协议[7]，另称媒体网关控制协议(Media Gateway Control protocol，Megaco)。它是 ITU-T 与 IETF 合作制定的标准，被国际软交换协会 ISC 指定协议为软交换设备与媒体网关之间的标准通信协议。

软交换技术是下一代网络的核心技术，它的核心思想是将呼叫控制与传输承载分离。在网络结构中表现为：传统的 IP 电话网关（或交换中心）在物理上划分为媒体网关控制器(Media Gateway Controller，MGC)和媒体网关(Media Gateway，MG)两个独立的网元实体。H.248 协议则用于这两个网元实体之间的交互和控制。它定义了 MGC 对 MG 的控制方式和标准通信格式。MGC 和 MG 这一对网元再加上他们之间的控制协议，就组成了“软交换系统”。在呼叫过程中，MGC 完成路由寻址、连接建立等功能；MG 则负责传输的承载功能；MGC 通过 H.248 协议来控制 MG，进而完成业务的承载。H.248 协议在软交换网络中的位置如图 2.1 所示：

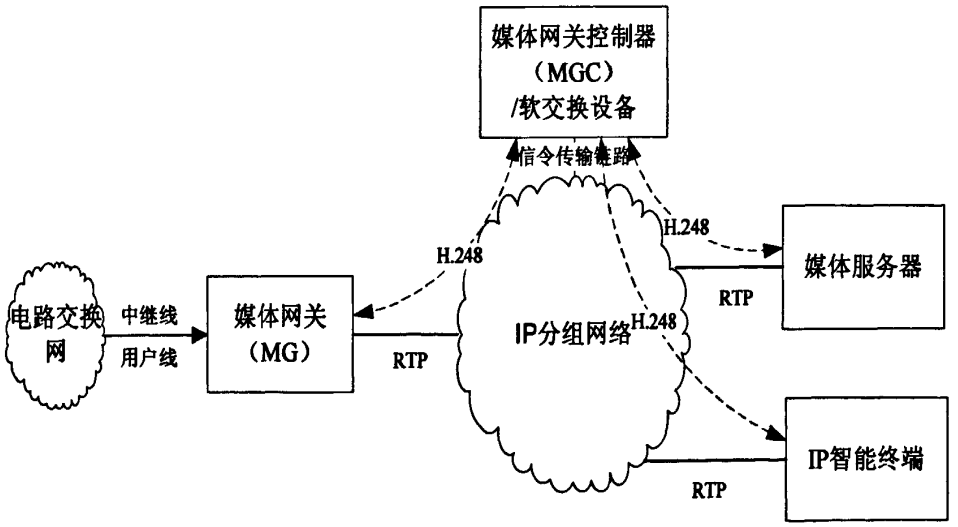

图 2.1 协议在网络中的位置

2.1 H.248 协议连接模型

H.248 协议的主要思想框架[8]表现为 H.248 协议的连接模型，它由几个抽象概念组成，包括终结点（Termination），关联（Context）和包（Package）。这些抽象概念构成了 H.248 协议的连接模型，它描述了 MG 中所有可由 MGC 控制的逻辑实体。MGC 通过对这些逻辑实体的控制来实现对 MG 的控制。

2.1.1 终结点

终结点（Termination）是H.248协议模型中一个逻辑实体，它是媒体流和（或）控制流在逻辑上的发起者。终结点由 MG 创建，并同时由 MG 赋予一个唯一的TerminationID 对其标识；每个终结点都有一系列属性（Property），这些属性可由一系列包含在协议命令中的描述符（Descriptor）来读取或修改。

H.248 协议定义了两种类型的终结点，分别是半永久性和临时性终结点。半永久性终结点代表物理终结点（例如一个 TDM 信道上的终结点），只要这个物理实体始终存在于 MG 内，这个终结点就存在；临时性终结点是临时性的终结点（例如 RTP 流的某一端），只有 RTP 连接存在并在被使用，这个终结点才存在。MG 本身被称为根终节点（Root）。

临时性终结点可以在关联中被创建或删除。半永久性终结点的创建和删除具有“不彻底性”。一个物理终结点并不能被凭空的创造：当它被创建时，MG 只是将它从空关联中移入创建它的关联中；当它被删除时，MG 又将它移回空关联中。也就是说，MG 中所有的物理终结点都被存放在空关联中。

终结点可以加载信号（Signal）；终结点可以对事件（Event）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采取相应的动作；终结点还可以对数据进行统计（Statistics），当MGC发出审计（AuditValue）请求或者当终结点被删除时，终结点就将这些统计数据报告给MGC。

MG可以利用终结点来处理复用媒体流。在处理复用媒体流的连接模型中，用于携带部分复用流的每个数字承载通道都有一个物理或临时的承载终结点与之对应，所有处于这些数字通道的起始和终结位置的承载终结点都与一个名为复用终结点（Multiplexing Termination）的独立终结点相连接。复用终结点代表一个面向帧会话的临时性终结点，它使用Mux描述符来描述所使用的多路复用方式（例如H.320会话中使用的H.221）及其数字通道的帧封装顺序。

一个终结点可以携带多个类型的媒体，并携带多个Stream流描述符分别描述每一媒体类型。因此，终结点可以代表复用的承载能力（例如ATM ALL2）：每创建一个新的复用承载能力，也就相当于在关联中创建了一个临时性终结点；每一个终结点，也就删除了一个复用承载能力。

①终结点的动态性

H.248协议中规定，可以创建新的终结点，也可以修改已存在的终结点的属性值，或向终结点添加、删除事件或信号。MGC可以释放终结点及该终结点代表的资源，但前提是已取得了改接点的控制权（如用Add命令将终结点添加在关联中）。

②终结点标识符TerminationID

终结点由TerminationID进行标识，该标识的分配方式由MG自主决定。其中，物理终结点的TerminationID是MG预先规定好的，这些TerminationID通常具有有某种约定的结构，例如：一个中继组号及其中的一个中继号组成一个ID。

对于TerminationID可以使用一种通配机制，包括两种通配值（Wildcard）："ALL"和"CHOOSE"。通配值"ALL"用来表示多个符合要求或可用的终结点，"CHOOSE"则用来指示符合制定条件的终结点。MGC可以通过这种方式指示MG选择一个中继群中的某一条中继电路。

当命令中的 TerminationID 是通配值"ALL"时，则对每一个匹配的终结点重复该命令（根终结点不包括在内）。当命令要求通配响应时，每一次重复命令将分别产生一个通配响应。当命令不要求通配响应时，多次重复命令只会产生一个通配响应，该通配响应中包含所有单个响应信息的集合。

③终结点的属性

可以用一些属性来描述终结点。每个属性由一个 PropertyID 标识，大部分属性具有缺省值，其缺省值在本协议中或某个扩展包中进行定义。在扩展包中定义的属性由包名（PackageName ）和属性标识符（PropertyID）来标识。对于没有缺省值的属性（除 TerminationState 和 LocalControl 外），在终结点刚被创建或返回到空关联时，均认为该属性为空或无值（no value）；若 MGC 想要完全控制已经由 MG 预订了缺省值的属性，则需要在使用 Add 命令将终结点加入关联时就为这些属性提供明确的取值。对于还处于空关联中的物理终结点，MGC 可以通过 Audit 命令确定它的所有预定值。

终结点具有一些公共属性以及与特定媒体流相关的非公共属性，这些公共属性称为终结点状态(TerminationState)属性。对于每个媒体流，都有各自的本地（Local）属性和远端（Remote）属性。

属性可以是只读的（ReadOnly）或可读写的（Read/Write）。属性能够采用的值及它们的当前值都可以被审计（Audit）。对于可读写的属性，MGC 可以设置它们的值。若某一属性被设定为全局的（Global），它的值就是唯一的，并被所有实现这个包的终结点共享。相互有关的属性可以组合成描述符的形式。

④描述符

属性可以被组合成描述符，从而作为命令的输入或输出参数。使用 Add 命令将一个终结点添加到一个关联中时，可以以适当的描述符作为命令的输入参数来设置可读写的属性值；使用Move 将一个终结点从一个关联中转移到另一个关联中时，也可以改变终结点的属性值；当然，也可以使用 Modify 命令对关联中的终结点的属性值进行修改。描述符也可以作为命令的输出参数在应答中返回，例如使用 AuditVale 命令对相关属性、事件、信号等信息进行审计，应答的输出参数中就包含

了这些信息的描述符。

通常，在命令中若完全省略了某个描述符，则该命令作用的那个终结点的对应属性值保持不变；另一方面，若一个命令中只省略了一个描述符的某些可读写属性，则对应终结点的对应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方式。

H.248 协议定义的描述符如表 2.1 所示：

表 2.1 H.248 协议描述符

描述符	功能描述
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
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

2.1.2 关联

当若干数量的终结点产生相互的联系并有机地组合在一起，就形成了关联（Context）。关联是一个结合体，当这个结合体包含两个或两个一上的终结点时，就能够表示拓扑结构，及该拓扑结构中的媒体流参数等信息。

几个相关的概念如下：

●关联标识（ContextID）：Context 的标识；

●拓扑结构（Topology）：用于描述一个关联内部终结点间的媒体流方向；

●关联优先级（Priority）：提供关联的优先级处理信息；

●紧急呼叫的标识符（Indicator for Emergency Call）：提供关联紧急处理的信息。

在上一节中提到的空关联是一种特殊的关联，它包含了所有属于对应 MG 所控制的物理终结点和资源。MG 能修改或查看空关联中的终结点的属性值，还可以要求这些终结点对事件进行检测。

H.248 协议中定义了相关的命令对终结点进行操作。使用 Add 命令向关联添加终结点（若没有指定特定的关联，MG 在将创建一个新的关联同时将终结点加入该关联），将半永久性终结点加入关联时，只是将该终结点从空关联中取出再加入目标关联。使用 Subtract 命令将终结点从关联中删除。当删除的终结点为半永久性终结点时，该终结点并不会被删除，而是将它重新移回空关联中。若删除的终结点为该关联中的最后一个终结点，则同时删除该关联。使用 Move 命令可以将终结点从一个关联移至另一个关联。

终结点与关联的关系有从属意味，但同一时刻一个终结点只能存在于一个关联中。一个关联中所能拥有的最大终结点数量是 MG 的既有属性。一个只提供点到点连接的 MG 的关联中最多只允许两个终结点；一个提供多点会议连接模式的 MG 的关联中则可以支持三个或三个以上的终结点。

2.1.3 包

本协议通过允许终结点具有的可选项为：属性（Property）、事件（Events）、信号（Signals）和统计（Statistics）。这些可选项的不同组合实现并区别了不同类型的终结点，这种组合被称为包（Package）。对某一终结点使用审计命令可以查看该终结点实现了哪些包。

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。协议规定了各标识符有效范围。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）都有相互独立

的命名空间。同一个标识符可以用在每个包中，而不同的包中的两个属性标识符也可以相同。包可以被扩展，扩展后的属性、事件、信号和统计等标识符既可以遵循扩展包的定义也可以用基本包的标识符来指定。

2.2 H.248 协议消息

2.2.1 消息类型

①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。H.248 命令及其含义参见表 2.2：

表 2.2 H.248 命令

命令名称	描述
Add	MGC→MG，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	MGC→MG，修改一个终结点的属性、事件和信号。
Subtract	MGC→MG，删除一个终结点与它所在的关联的联系，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	MGC→MG，将一个终结点从一个关联转移到另一个关联。
AuditValue	MGC→MG，获取终结点属性、事件、信号和统计的当前信息。
AuditCapabilities	MGC→MG，获取终结点的属性、事件和信号的所有可能值的信息。
Notify	MG→MGC ，向 MGC 报告 MG 中所发生的事件。
ServiceChange	MGC↔MG 或 MG→MGC， 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。

②响应

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。

2.2.2 消息结构

H.248 协议发送或接收的信息单元称为消息。在 H.248 协议中，一个或多个命令或响应被封装成一个消息进行发送或接收。

H.248消息采用二进制格式和文本格式两种编码方式。二进制格式采用ITU-T X.680 ASN.1[9]（抽象语法规则1，Abstract Syntax Notation One）定义的规范描述，使用X.690定义的BER规则编码[10]；文本格式遵循RFC 2234 ABNF[11]（扩展巴科斯范式，Augmented Backus-Naur Form）定义的规范描述。

采用文本格式编码的H.248协议也称为Megaco协议。由于Megaco和H.248的区别仅在于编码方式的不同，因此本论文除特别指明外，均用H.248协议泛指Megaco和H.248。在实际网络中，MGC必须支持两种编码格式，MG则可以支持其中任何一种或两种方式。这两种编码方式的消息结构[6]均相同，如图2.2所示。

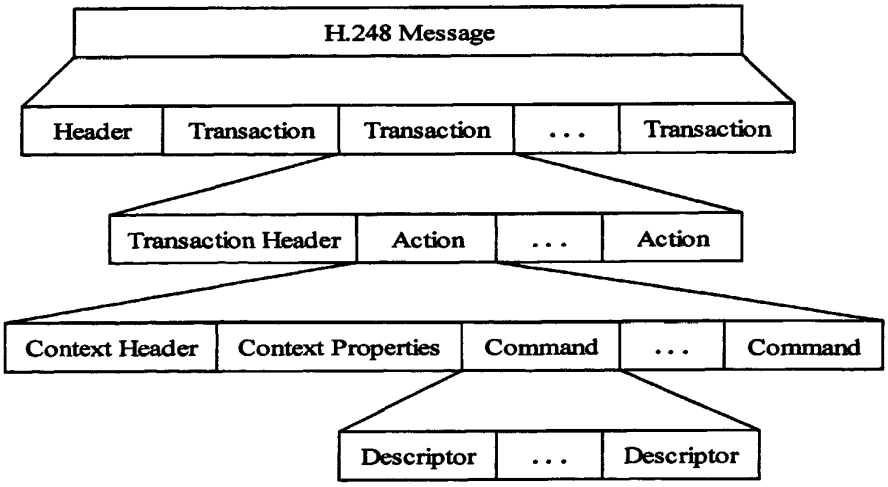

图 2.2 H.248 消息体结构

①消息体

一个消息体（Message Body）由一个消息头（Header）和若干个事务组成。消息头携带了该消息的标识符（MID，Message Identifier）和版本字段：其中，MID 标识了消息的发送者，它可以是域地址、域名或设备名（一般采用域名）；版本字段

标识了消息遵守的协议版本，版本字段有 1 位或 2 位数，目前普遍使用版本 1。

②事务

一个消息（Message）中包含一个或多个事务（Transaction），消息内的事务是相互独立的。协议中没有规定多个事务被独立处理的先后次序。

事务包括请求和响应两种类型：事务请求即 TransactionRequest，而事务响应则分为 TransactionReply 和 TransactionPending 两种。

MG 和 MGC 之间的一组命令组成了事务（Transaction），每一组 Transaction 都由一个 TransactionID 进行标识。每个 Transaction 由一个或者多个动作（Action）组成，每个 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、修改和审计都只是一个关联内的操作，因而每个动作都由一个关联标识进行标识。下面两种情况下动作可以不指定关联标识符：一是对关联之外的终结点进行修改或审计操作；另一种情况是 MGC 要求 MG 创建一个新的关联。事务、动作和命令这三者之间的关系如图 2.3 所示。

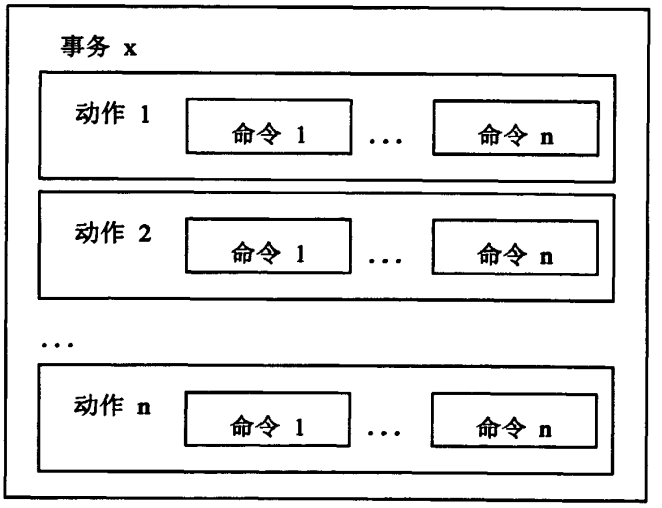

图 2.3 事务、动作、命令的包含关系

③动作

如前所述，动作与关联（Context）具有密切的相关性，ContextID 在标识关联的同时也唯一标示了一个动作。在动作内部，命令被按顺序执行。

一个动作从关联头部（Context Header）开始，其中包含的 ContextID 标识了该动作对应的关联。ContextID 由 MG 指定，在该 MG 的范围内具有唯一性。MGC 在以后的操作中使用相同的 ContextID 标识针对同一关联的各项操作。Context Header 后紧跟着若干个命令，这些命令只针对该 ContextID 标识的关联。

④命令

命令是 H.248 消息的核心内容，它实现了对关联和终端属性的控制。这些控制包括：指定终端报告检测到的事件，通知终端使用何种信号和动作，以及指定关联

的拓扑结构等。命令由命令头部（Command Header）与命令参数构成，这些命令参数以“描述符”（Descriptor）的形式出现。

综上所述，H.248 协议消息构成机制如图 2.4 所示。

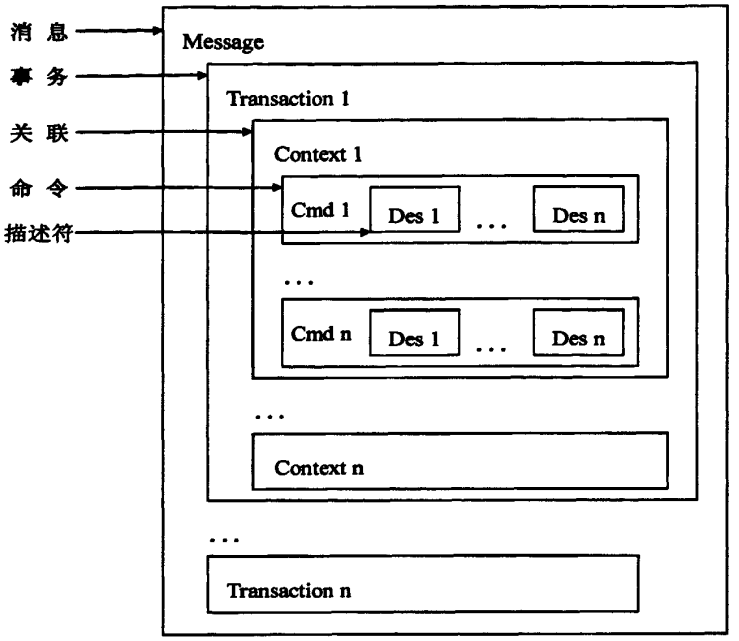

图 2.4 H.248 协议消息机制

2.3 H.248 协议在 Mc 接口上的应用

如第一章所述，Mc 接口是 WCDMA 核心网电路域在向 NGN 网络演进过程中产生的新接口。接口两端的 MSC Server 和 MGW 作为软交换技术在 UMTS 系统内的实现网元，分别起到了 MGC 和 MG 的作用。由此可知，Mc 接口上 H.248 协议主要完成了 MSC-Server 与 MGW 之间的事物处理[4]。具体如下：

与呼叫无关的事物处理主要包括：MGW 的注册、重启、重注册、退出服务、恢复通信，MSC Server 的恢复、退出服务，终结点退出服务、恢复服务，审计，MGW 资源拥塞处理－激活/指示。

与呼叫相关的事物处理主要包括：建立承载，准备承载，改变流方向，隔离承载终结点，加入承载终结点，改变导通，激活互通功能，释放程序（释放承载、释放终结点），承载已释放、已建立，发送信号音，播放录音通知，发送/检测/报告 DTMF 音，录音通知播放完毕，激活语音处理功能，预留电路，隧道信息上传/下传，信号音已完成，录音通知停止，停止发送信号音，停止 DTMF 检测，停止 DTMF 音，修改承载特征，协议协商结果，速率改变，承载修改支持，确认/预留承载特征，紧急呼叫指示。

H.248 协议最初被用于固定 IP 电话网。WCDMA 核心网络的电路域部分虽属于固定电话网络，但由于其从属的 UMTS 系统环境的移动性，H.248 标准包及其扩展 Q.1950 已经不能满足一些移动环境内的特有应用，如 SRNS 重定位 / 切换、实时媒体流（RTP 流）传输控制等。为此，3GPP 等组织在 3GPP 29.232 等标准[12-15]中为 H.248 协议扩展并新增了支持承载与控制分离的 BICC 包（承载特性包、承载网络直通连接包，通用承载连接包），以及为支持 3G 应用的 UMTS 包（3GUP、电路交换数据包、TFO 包）等新的包来解决这一问题。

现有的国际标准没有对 Mc 接口上的 H.248 协议编码方式做出统一规定，我国的通信行业标准 YD/T 1593-2007 中也同时允许采用文本和二进制方式对 Mc 接口上的 H.248 消息进行编码。将两种编码方式对比来看，虽然文本编码方式更符合"IP 化"的网络演进方向，也被更多的新协议所采用，但是由于在移动网中现有的协议大多采用二进制方式以及各厂家的习惯性等原因，Mc 接口上的 H.248 协议大部分还是采用了二进制编码。

2.4 本章小结

本章对 H.248 协议的原理进行了研究。首先简要指出了该协议在采用软交换技术组网的网络中的位置和作用；接着从 H.248 协议的连接模型这一角度介绍了协议的基本内容，包括终结点、关联、包的概念；然后分别对 H,248 协议的消息类型和消息结构进行了介绍和总结；最后对 Mc 接口上的 H.248 协议应用做了概括与归纳。

第三章 Mc 接口监测模块

3.1 WCDMA 网络测试仪的体系结构

WCDMA 网络测试仪的主要功能是实时捕获 WCDMA 网络数据、对现网数据进行信令分析、CDR 合成（Call Detail Records，呼叫详细记录）和统计分析。整个测试仪通过 WCDMA 网络各协议基于消息或 CDR 的 KPI 统计指标实时地反映网络运行状况，为 WCDMA 设备制造商和网络运营商工程的网络开通/验收测试、故障定位、日常维护、网络优化和性能分析等工作提供必备的检测手段与检测依据。

测试仪的硬件结构主要由 PC 机（工控机）、数据采集卡及总线构成。数据采集卡负责原始物理层数据流的采集，并通过 PCI 总线与工控机进行交互，将数据提交给软件进行分析处理，进而实现测试仪为用户提供的与网络分析相关的高级功能。

硬件模块之间的交互与协作、协议数据的解码、呼叫合成与分析统计等高级功能的实现都需要软件系统来控制、协调与实现。如果说测试仪的硬件划分是“水平上的划分”，那么其软件体系结构就是对整个测试仪系统在逻辑上的“立体划分”。它不但在“水平上”完成了对各硬件模块的控制和它们的交互协调，而且“垂直地”实现了对数据从采集，到存储提交，再到统计合成，直至形成分析结果这一流程的控制。

在实现中，我们基于面向对象的模块化设计思想和多线程技术来设计 WCDMA 测试仪的软件体系结构，并将其划分为：数据采集、数据存储、详细解码、合成解码、呼叫合成、统计分析、界面显示等模块。如图 3.1 所示：

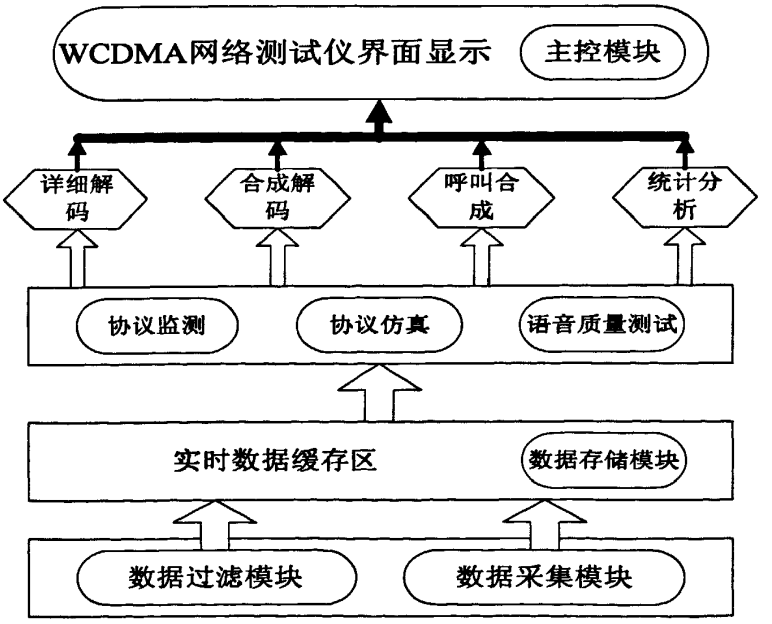

图 3.1 WCDMA 网络测试仪软件体系结构

其中，数据采集模块根据不同的链路选择相应的采集卡，例如 E1 卡、网卡等，通过采集卡驱动所提供的 API 函数实时捕获网络中的数据；数据过滤模块根据预设置的过滤配置信息，使用户可以只采集所关心的原始数据，以缓解硬盘压力便于数据分析；采集到的原始数据经过排序、添加消息 ID 等预处理，被送入实时数据缓冲区，再调用数据存储模块，将它们映射到硬盘文件上，从而缓解内存压力；软件的高层提供了协议监测、协议仿真、语音质量测试等应用功能。这些功能主要通过对原始消息进行深度的协议分析，实现协议的详细解码、合成解码、呼叫合成及业务统计等功能，在此基础上用户可以对网络质量做出评估、及时掌握现网运行状况、分析网络业务瓶颈、优化网络。

在本节接下来的内容中，将从 WCDMA 网络测试仪的数据处理流程与多线程结构两方面来介绍测试仪的软件体系结构。

3.1.1 数据处理流程

测试仪的核心功能是协议分析。它主要包括协议解码、呼叫合成、统计分析等子功能。协议分析的处理对象是网络数据，数据处理流程的设计是网络测试仪的核心设计之一。

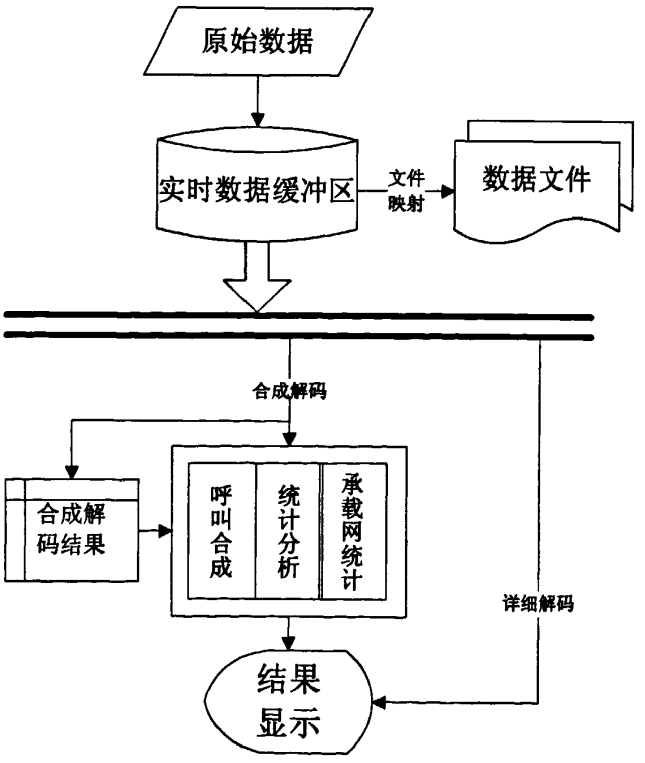

图 3.2 WCDMA 网络测试仪数据处理流程

在上图中可以看到：首先，网络原始数据经由“数据采集”和预处理过程（时

间排序，添加消息 ID 等操作）被注入实时数据缓冲区，并按照一定的数据格式映射到硬盘文件上；之后，数据缓冲区中每条消息的关键信息经由合成解码过程被逐条地提取出来，用于消息列表的消息特征信息显示或为呼叫合成、统计分析等协议监测功能服务；用户若需要查看详细解码结果，系统可根据所添加的消息 ID 从数据缓冲区提取出对应的消息，经由详细解码过程即得到详细解码结果。

3.1.2 多线程架构

由于网络测试仪的多任务性及高实时性要求，我们在测试仪的软件架构设计过程中采用了多线程技术。除主线程外，主要的辅助线程有：数据捕获、数据存储、消息处理、消息显示、消息过滤等线程。

主控线程负责软件的启动、鉴权、终止、用户操作响应、界面显示和各个辅助线程的调度等操作。

数据捕获线程负责硬件驱动消息的响应、接收硬件驱动程序的数据、向数据缓冲区中格式化存储数据、防止高速率下的丢包等操作。此线程被设置为高优先级。

数据存储线程是后台线程，它负责数据缓冲区中原始数据的映射磁盘文件存储映射，以减少内存消耗，便于消息查找和离线分析，防止意外断电造成的数据丢失。

消息处理线程触发消息的合成解码、呼叫合成及统计分析功能。

消息显示线程和消息了过滤线程均利用数据处理线程的执行结果，分别完成完成消息列表的显示和从列表中过滤出用户指定的特定类型消息。

整个多线程体系的架构如下图所示：

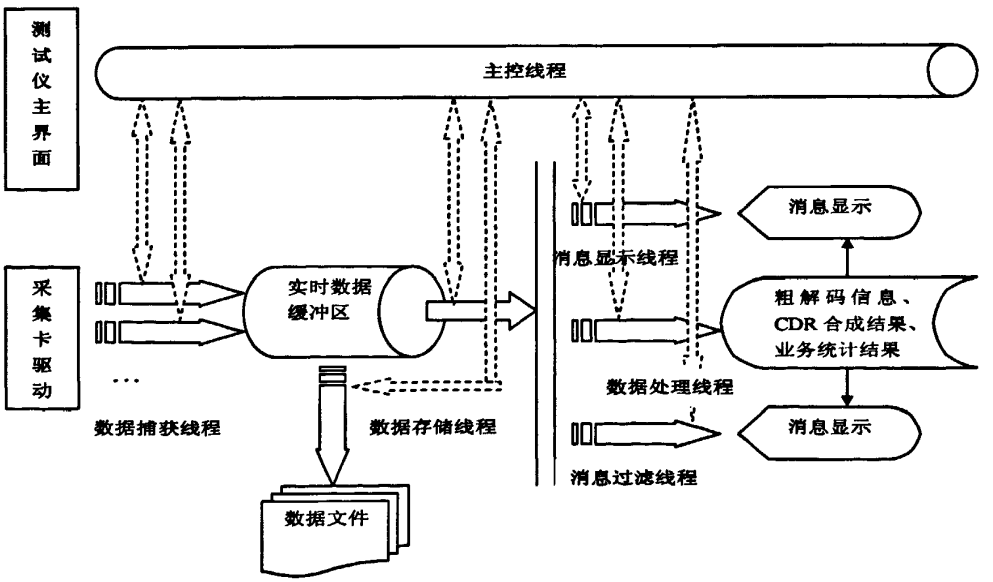

图 3.3 WCDMA 网络测试仪多线程体系架构

3.2 Mc 接口监测模块

3.2.1 模块化设计

模块化设计指：用主程序、子程序、子过程等框架把软件的主要结构和流程描述出来，并定义和调试各个框架之间的输入、输出链接关系，最终形成一系列以功能块为单位的算法描述。这其中，以功能块为单位进行程序设计，实现其求解算法的方法称为模块化。模块化设计降低了程序的复杂度，使程序设计、调试和维护（特别是更新与升级）等操作变得简单。

根据模块化的设计思想，我们将 WCDMA 网络测试仪的各功能模块分别采用 DLL（Dynamic Link Library，动态链接库）的形式单独实现，使各项功能之间相对独立，而软件的主控模块实现各功能模块之间同一的控制和调度。采用模块化思想进行设计，有效地增强了各功能的独立性，消除了各协议、接口模块之间不必要的耦合，有助于软件在整体上的插件化，使得软件体系结构更加清晰，降低了软件开发、测试、维护等阶段的代价，易于软件的维护和升级，便于模块的移植和系统集成。另外，协议的模块化与协议栈的分层思想也可在结构上相统一。

3.2.2 Mc 接口监测模块的技术要求

我们将 Mc 接口上的 H.248 协议解码、呼叫合成及统计等功能封装在一个模块中，称为 Mc 接口监测模块(以下简称 Mc 模块)。该模块以 DLL 形式加载在系统中，并通过接口函数的形式向主控模块提供对采集自 Mc 接口协议栈应用层的 H.248 协议的分析方法和相关功能操作。

根据 Mc 模块在整个测试仪系统功能结构中的位置，Mc 模块必须达到以下功能和要求：

①能够对Mc接口的协议数据（H.248协议）的文本编码方式和二进制编码方式进行基础解码和合成解码，并能对该接口上的扩展包进行正确解码。

②能够对Mc接口上的各类业务的信令流程进行CDR合成，进而重现接口上的呼叫流程。

③能够对Mc接口上的H.248消息进行统计，统计结构能清晰地反映与所监测接口相关的网络和设备的运行状况。

④模块的设计需真正模块化，以“松耦合”的方式嵌入测试仪平台，利用平台的界面功能进行用户交互。

在具体的实现过程中，我们利用面向对象思想将整个模块划分为多个子模块分别实现。上文中提到的每一个功能均可以作为一个子模块。通过对各个子功能块的

集成，实现 Mc 模块的整体功能。设计如下：整体模块可分为解码、合成统计模块。解码模块实现对 H.248 协议 PDU 的详细解码和合成解码，数据调用该模块后将解码信息提供给合成统计模块，合成统计模块将每一次呼叫过程中的消息进行组合，形成完成的呼叫流程，同时实现对成功、失败、呼叫中等各种状态的流程进行统计，完成与测试相关各种高级功能的实现。每个子模块在设计编码完成之后，都要进行相应的单元测试。在单元测试正确的情况下，将模块集成入测试仪平台的主控单元，同时利用实验室的现有测试平台和外场测试环境完成 Mc 接口监测功能的集成测试。以确保程序的健壮性和功能的有效性与完备性。

3.2.3 Mc 接口监测模块的总体方案

Mc 接口监测模块主要由解码单元、合成单元以及统计单元这三部分组成。根据模块化设计思想，将整个 Mc 模块划分为协议解码和分析统计两个子功能模块实现，并生成独立的 DLL 供主控模块调用。实现的功能有：文本/二进制格式 H.248 数据信息的解码、呼叫过程的合成、话务统计、呼损统计、流量统计等以及对统计分析结果的输出等。

根据测试仪的数据流程和多线程结构，Mc 接口监测模块的总体架构设计如下图 3.4 所示：

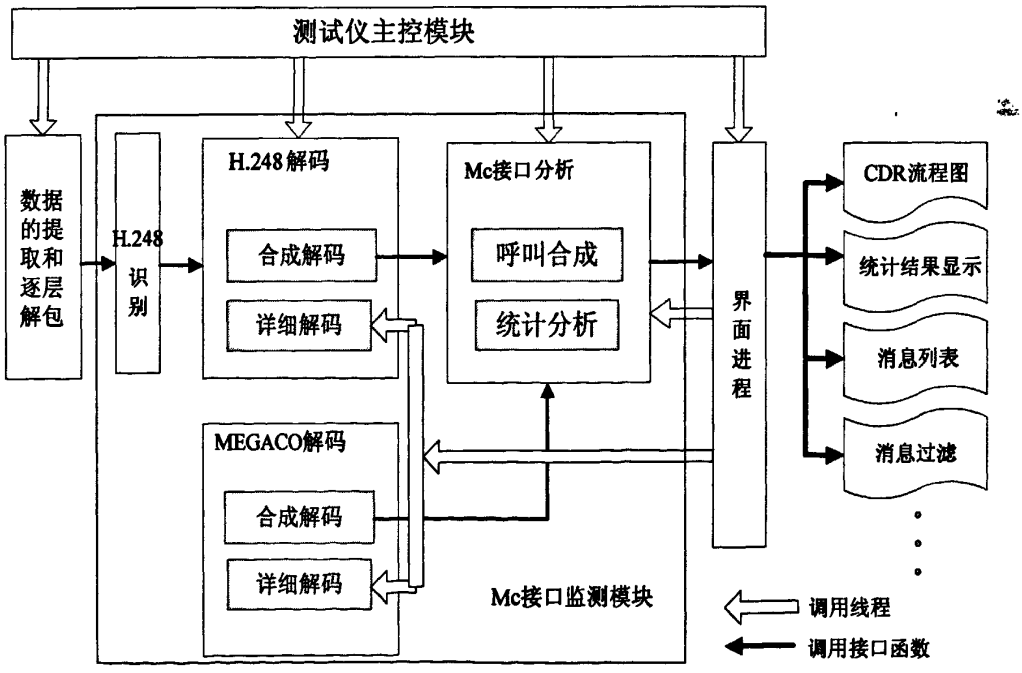

图 3.4 Mc 接口监测模块的总体架构

①数据采集卡受主控模块的进程控制，从测试口(Mc接口)上捕获原始数据帧，经过预处理如：给消息按照时间排序贴上时间戳、加上消息索引等操作。

②当该消息报文依次通过IP层、传输层（UDP/TCP/SCTP）提取出它的上层PDU偏移量、长度等信息后，就调用H248Decode.DLL进行应用层消息解码。解包的同时维护原消息与预处理所添加信息的关联。

③协议识别功能以全局函数的形式供主控模块调用，判断应用层数据是否为H.248协议数据。

④根据合成解码结果，对缓冲区中的H.248消息进行一次便利，并按合成方案及关键值进行归类，进而实现呼叫合成和统计分析功能。把处理结果以文件形式保存到磁盘中。呼叫合成和统计都是在H.248协议分析单元中处理。

⑤最后应用进程通过调用模块接口函数完成特定的显示功能。如调用详细解码结果，实现H.248消息的各个参数信息、原始信息和相关解释的完全呈现；设定过滤条件，实现用户指定条件的消息的查询等等。

3.3 本章小结

本章首先介绍了WCDMA网络测试仪的体系结构，分析了网络测试仪的数据处理流程和多线程结构；然后对Mc接口监测模块的进行了需求分析；最终根据模块化设计思想设计了Mc接口监测模块的总体方案。为下面各章节关于Mc模块的协议解码、呼叫合成和统计分析功能的具体设计实现奠定了基础。

第四章 Mc 接口监测模块解码功能的设计与实现

4.1 解码功能的需求分析

对网络测试仪来说，协议解码功能是其最基本，也是最重要的功能。测试仪的所有高级功能都是对协议解码结果的利用，诸如消息过滤、呼叫合成以及统计分析等。同样，在 Mc 接口监测模块中，对 H.248 协议的解码功能是整个模块的最基本功能。

由于 Mc 模块需要具备呼叫合成、统计分析的功能，以及体现在网络测试仪用户功能中的消息过滤等用户交互功能，我们将 Mc 模块的协议解码功能分为如下一些子功能并分别实现：

①SDU 提取功能：从本层 PDU（协议数据单元，Protocol Data Unit）中提取除消息头域外的净荷数据，用于协议栈解码的层次调用。

②协议识别功能：根据协议自身特点判断数据块是否为 H.248 协议数据，不需要用户配置，配合仪表的协议栈自动识别功能，增强仪表的智能性和使用性。

③合成解码功能：为提高仪表的实时处理速度，根据协议特点，提取数据中的关键信息，用于消息特征信息的显示以及呼叫合成。

④详细解码功能：按照 H.248 协议标准对数据进行比特级的解析，便于用户对消息内容的详细查看。

如 2.3 节中所述，Mc 接口上的 H.248 消息数据有文本和二进制两种编码方式。为了使网络测试仪能够适于不同的网络，Mc 模块的解码功能必须对采用这两种编码方式编码的 H.248 消息数据均能完成解码。文本编码（通常称为 Megcao 协议）和二进制编码（通常称为 H.248 协议）虽遵循同样的连接模型和协议标准，但在数据格式上有很大差别。因此，对于两种编码方式的协议数据的解码方式也有所不同。在本章中将分别针对这两种编码方式介绍解码功能的实现。

4.2 解码功能的类封装设计

根据 WCDMA 网络测试仪软件体系结构的模块化设计思想和面向对象思想[16]，我们通过类的方式来实现各协议的解码功能的封装，即将解码的相关功能以成员函数的形式包含在各个协议类中，再进一步封装成 DLL，从而对外提供能实现各功能的接口函数。在 Mc 模块的设计过程中，我们将 SDU 提取函数，合成解码函数和详细解码函数作为 Mc 模块解码类的功能的接口函数。协议识别函数因其易被频繁调用的特性，未被封装在 Mc 模块解码类中，而是将它作为一个全局函数从模块的 DLL

中导出，供外部调用。这样的设计使得功能的实现与类及其接口相分离，便于模块的分离和维护，便于软件的升级，也符合面向对象的程序设计方法。

由于 WCDMA 网络测试仪涉及到众多的接口和协议，从面向对象的角度出发，以所有接口、协议的解码类为对象设计了基类 CDecoder，此类的对象代表了 PDU 这一抽象概念，其中封装了如获取 bit 头函数、获取数据 bit 长函数等公用函数。对于各模块解码所需的特定功能则可在继承自 CDecoder 类的各协议解码类中进行实现。

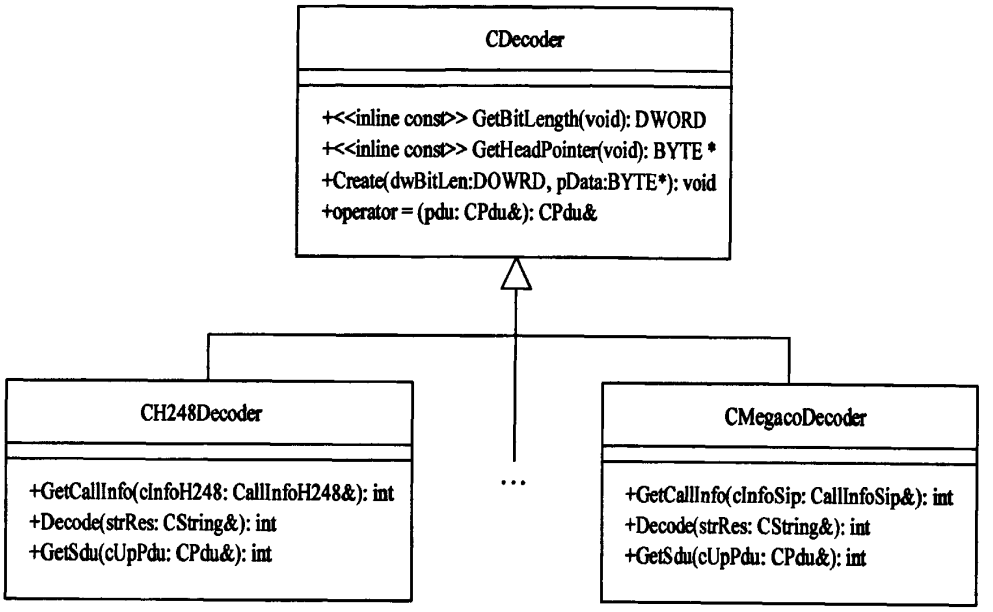

图 4.1 解码类继承类图

上图对解码类的继承关系作了示例，仅列出了一部分公有成员函数，这些函数就是解码类对外的接口。在图中可以看出，CH248Decoder 类和 CMegacoDecoder 类均派生自基类 CDecoder，它们均继承了基类中数据块处理类的公有函数，并根据自身协议特点分别实现了针对本协议的相关功能，同时增加了相应的接口函数。整个类图结构清晰，符合面向对象的编程思想。

基类 CDecoder 提供的四种方法：GetBitLength 方法获取本层 PDU 的比特长度；GetHeadPointer 方法获取本层 PDU 存储在内存中的数据头指针；Create 方法通过传入的比特长度和数据头指针等参数创建一个新的 PDU；等号重载函数便于解码类对象的赋值操作。

解码类 CH248Decoder 除了从基类继承的公有接口外，对外还增加了三个方法：GetCallInfo 方法用于 H.248 协议的合成解码；Decode 方法用于 H.248 协议的详细解码；GetSdu 方法用于提取本层 PDU 的 SDU。

4.3 H.248 协议解码功能的设计与实现

采用 ASN.1 语法中 BER 规范（基本编码规范）描述的二进制格式对媒体网关控制协议进行编码得到的协议被称为 H.248 协议。ASN.1 语法是 ISO/ITU-T 标准，描述了一种对数据进行表示、编码、传输和解码的数据格式，被广泛地应用在通信协议的规范定义中。它是一种数据定义语言，描述了目标信息和结构。ASN.1 语法包括 BER（基本编码规则，Basic Encoding Rules）、PER（紧缩编码规则，Packet Encoding Rules）、CER（正则编码规则，Canonical Encoding Rules）、DER（非典型编码规则，Distinguished Encoding Rules）等编码规则。H.248 协议就采用了其中的 BER 方式[17]。

BER 编码规则采用了一种“TLV 三元组”的形式，它们分别为标识符八位位组(T)、长度八位位组(L)、内容八位位组(V)。

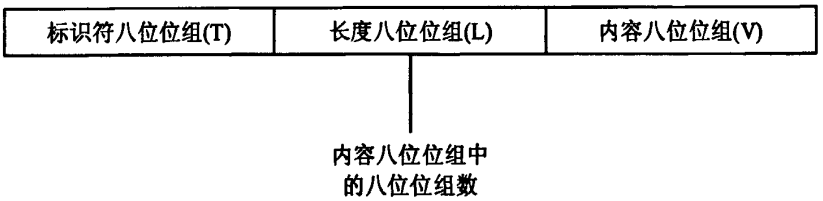

图 4.2 长度八位位组确定形式的 BER 编码结构

另外，还有一种名为内容结束八位位组的位组形式，该位组只有在长度八位位组为不确定形式下才出现，用以指示三元组内容结束。

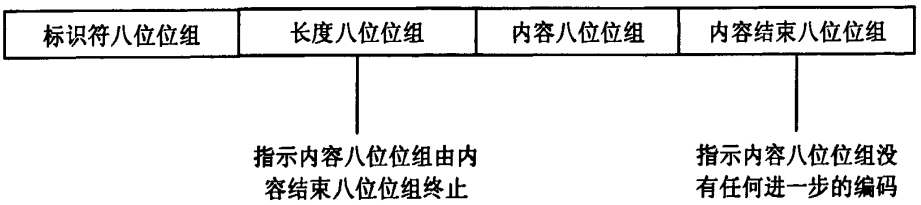

图 4.3 长度八位位组不确定形式的 BER 编码结构

标识符八位位组内的数据是其所属三元组中内容八位位组数据值的类和编号（ASN.1 标签）的编码；长度八位位组在确定形式下指示了内容八位位组中数据的长度，即内容八位位组的八位组组数；内容八位位组是所编码数据值的二进制形式（在这里需注意的是：内容八位位组可能并不是一个简单的数据值，它有可能是向内一个层次的“TLV 三元组”，也就是说通过 V 部分，三元组可以实现层的嵌套）；内容结束八位位组只有在长度八位位组为不确定形式时才出现，它的形式确定——两个值为 0 的八位组，用来指示内容八位位组结束（由于内容结束八位位组的特点，语法规定了其他各部分的八位组不能出现连续的来两个值为 0 的八位组，以此使内

容结束八位位组在规范中唯一)。

在 ITU 标准 X.690 中，对编号为 0~30 的标签代表的数值类型做出了规定，对编号大于等于 31 的数值类型则未作定义，以此保留给标准以外的类型和国际标准使用。图 4.4 给出了标签编号为 0~30 的标识符八位位组结构：

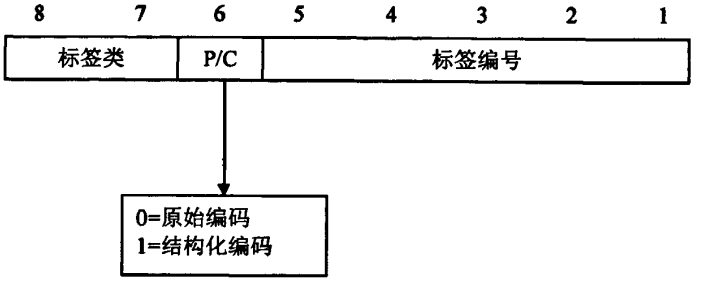

图 4.4 低标签编号的标识符八位位组编码结构

如图所示，位8和位7用来表示标签类的类型的编码，并对四种可能值做了规定：00为通用类型、01为应用类型、10为上下文指定类性、11为专用类型；位6用来表示本层三元组携带的数据是原始编码还是结构化编码，即本层的V组表示的是单纯的数据值还是向内一个层次的TLV三元组；位5至位1表示了标签编号的二进制整数编码，位5为最高有效位。注：对于标签编号从31号开始的标识符八位位组，其长度不局限于1个八位组，而是在第一个八位组内将位5至位1均置为1，利用后续的八位组来表示标签编号。具体方法是将每个后继八位位组的8位置1，剩余的位7到位1组成一个7×N位的二进制数表示标签编号，第一个后继八位位组的位7为最高位。

BER 规范规定了长度八位位组有确定形式和不确定形式两种结构：

确定形式的长度八位位组明确表示了内容八位位组的八位组数，它有短形式和长形式两种类型：

短形式的长度八位位组由单个八位位组组成，其中位 8 为 0，位 7 至位 1 组成的无符号二进制数（可能为 0）表示了本层的内容八位位组的八位组数。由于 7 位无符号二进制数最大只能表示 127，所以短形式的长度八位位组仅在本层内容八位位组的八位组数小于或等于 127 时使用。

长形式的长度八位位组在内容八位位组的八位组数大于 127 时使用，如图 4.5 所示：

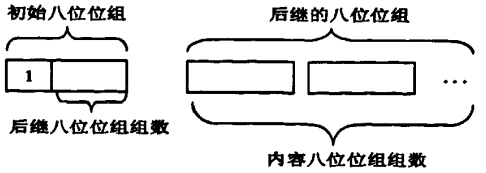

图 4.5 长形式的长度八位位组编码结构

该形式的构成规则是将第一个八位位组的位 8 置为 1，位 7 至位 1 的无符号二进制数表示后续的八位组数；后继的一个或若干个八位组则以第一个后继的八位组位 8 为最高位，所有比特位组成一个无符号二进制数表示内容八位位组的八组数。其中：初试八位位组的位 7 至位 1 不应都为；在长形式中，是否使用比最少必需数更多的后继八位位组是发送器的一个选项。

不确定形式的长度八位位组是一个单一的八位组，其位 8 置 1，位 7 至位 1 均为 0。在使用这种形式的长度位位组时，无法显式地得到内容八位位组的长度值，可以通过计算长度八位位组与内容结束八位位组（紧跟着内容八位位组的最后一个八位组）之间的八位组数得到。

内容八位位组的内容的根据数据值的类型，并遵循 ASN.1 的类型定义对各类型的数据值进行编码的结果，其长度不定（可 0，1 或更多个八位组）。

在 ASN.1 对 BER 规范的描述中，有一点需要特别注意：对于之前提到的原始编码，BER 规则采用确定形式的长度八位位组进行指示（而且原始编码的数值所在的 TLV 三元组层是最内部的层次，它所在的三元组不存在进一步的嵌套）；对于结构化编码且是立即可用的情况，BER 规则也采用确定形式的长度八位位组进行指示（结构化编码指示了该层次 TLV 三元组存在嵌套）；对于结构化编码且不是立即可用的情况，不确定形式的长度八位位组专门负责这种情况的编码。这一编码规则对我们在 H.248 协议的详细解码算法设计尤为重要。

4.3.1 协议识别功能的实现

根据 ITU 的 H.248 标准，在 H.248 协议的二进制编码语法规范[18]中，H.248 协议消息（MegacoMessage）被定义为序列值类型（SEQUENCE），其定义如下所示：

```
MegacoMessage ::=SEQUENCE
{
authHeader AuthenticationHeader OPTIONAL,
mess Message
}
```

ASN.1 定义了简单类型和结构类型两种数值类型，简单类型包括了布尔、整数、枚举、双精度、字串流等类型；而结构类型是简单类型的组合，有序列和集合等类型并以其成员类型的次序及排列方式来进行区分。序列值这一类型表示一种或多种类型的有序集合，其成员类型也可以是结构类型。如本节开头所述，ASN.1 对标识符八位位组的标签编码的 0～30 做了规定，这其中为序列值（SEQUENCE 类型）规定的编码是 16。而 H.248 协议数据的标签类型为通用的，即 H.248 协议数据是标签

类型为通用的序列值数据，由此我们不难得出 H.248 消息数据从最外层的编码封装来看是一个标识符八位位组值为 0x30 的“TLV”三元组。由于 H.248 消息数据与其所在层次的其他协议数据相比，头一个字节为 0x30 和采用 BER 规则编码使其成为区别其他协议数据的标志。由此，我们设计 H.248 协议识别算法如图 4.6 所示：

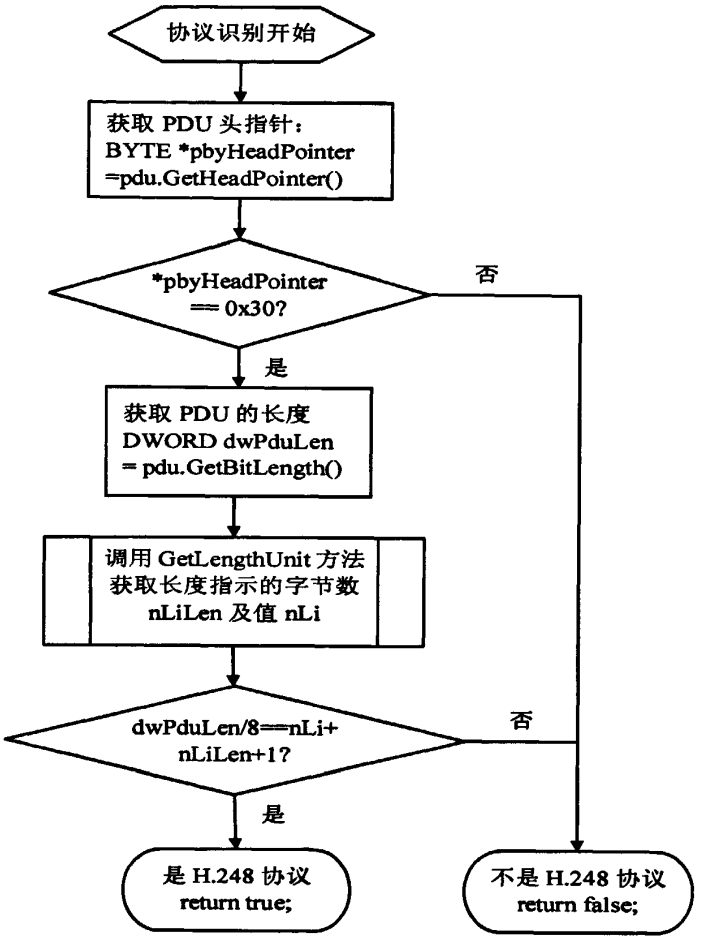

图 4.6 H.248 协议识别流程图

在图 4.6 所示的协议识别流程中，首先调用基类 CDecoder 的 GetHeadPoniter() 函数获取数据块的头指针，使指针指向一条 H.248 消息的第一个字节，即第一个标识符八位位组；判断该字节的值，若不为 0x30，则确定该数据块不是 H.248 协议数据；若该字节值确为 0x30，则继续调用继承自基类的 GetBitlength()函数获取整个数据块的比特长度 dwPduLen；接着，指针下移一个字节，并调用图 4.7 所示的 GetLengthUnit 方法获取长度八位位组的长度指示字节数 nLiLen 及自身携带的内容八位位组字节数 nLi；判断整个 pdu 长度是否等于 1+nLiLen+nLi，即判断该 pdu 内数据是否遵循 BER 编码规则，在且仅在等式成立的情况下，该数据块为 H.248 协议。

图 4.6 中用到的 GetLengthUnit()函数是长度八位位组字段的解码函数，这一解码功能被封装为解码类的私有成员函数，仅供 H.248 解码类调用。该函数实现流程

如图 4.7 所示：

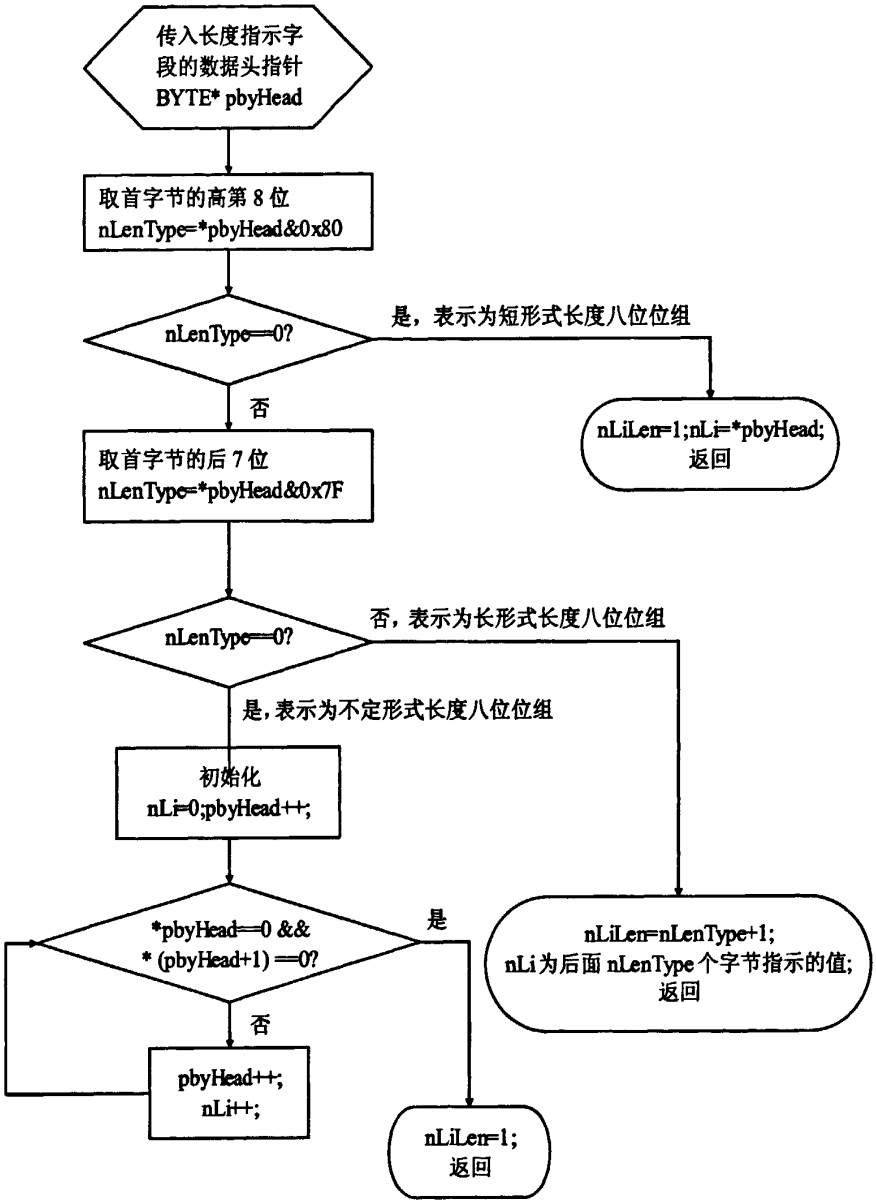

图 4.7 长度八位位组字段解码流程图

在该函数实现中，我们通过传入的长度指示字段头指针，获取长度八位位组的第一个八位组的位 8 的值，判断其是否为 0（即是否为确定形式中的短形式长度八位位组，首字节位 7 至位 1 的值即为 nLi）；若位 8 为 1 则取其位 7 至位 1 的值，为 0 则可知此字段采用是不定形式的长度八位位组，需将指针逐字节向后移位，查找内容结束八位位组以确定三元组的长度；若首字节位 7 至位 1 的值非 0，则确定该字段是确定形式中的长形式长度八位位组，该值即为 nLiLen（长度八位位组的长度指示字节数），根据该值解可解出内容八位位组长度 nLi。

4.3.2 包解析功能的设计

Mc 接口上的 H.248 协议是在 ITU 的 H.248 标准的基础上，扩充了软交换业务和 WCDMA 网络特定应用所需的各类型包。各类型的包的正确解析，是对 Mc 接口上的 H.248 消息解码的一个重要功能。

H.248 标准的扩展是通过包的形式完成的。一个新的包必须向 IANA（The Internet Assigned Numbers Authority，互联网数字分配机构）进行注册。IANA 对所有应用环境下的包有统一的标识。其中，文本编码的协议消息使用字符串命名的方式对各类型包进行标识，如“dd”、“mfd”等；二进制编码的协议消息使用包 ID 的方式对各类型包进行标识，譬如 UMTS 包中的 3GUP 包 ID（Pkgid=0x002f）、BICC 包中的 Bearer Characteristics 包 ID（Pkgid=0x001e）。

包 ID 的长度为 4 个字节。我们对所有类型的包 ID 进行了预定义，并设定所有可能值及其对应包的结构体格式（该结构体中包含了每种包内可能存在的描述符），解码函数截取包 ID 字段之后即可将消息携带的包信息保存入固定的解码结果结构体中。

4.3.3 详细解码功能的设计与实现

网络测试仪中的协议详细解码功能，是为了满足用户对截取到的协议数据中具体某一条消息进行查看而设定的功能。用户通过该功能可以对该条消息中所携带的消息类型、参数结构、参数值等所有内容进行逐比特的精确查看。这就对详细解码功能中消息解码的完整性提出了要求。Mc 模块的 H.248 详细解码功能不但要求能对二进制编码的 H.248 消息进行正确、完整的解码，而且要求对 Mc 接口上特殊应用的包有正确的解析。最终的解码结果必须按照协议标准中的消息结构进行保存。根据这几点，我们以 H.248 协议事务请求消息的详细解码流程为例进行设计的介绍，如图 4.8 所示：

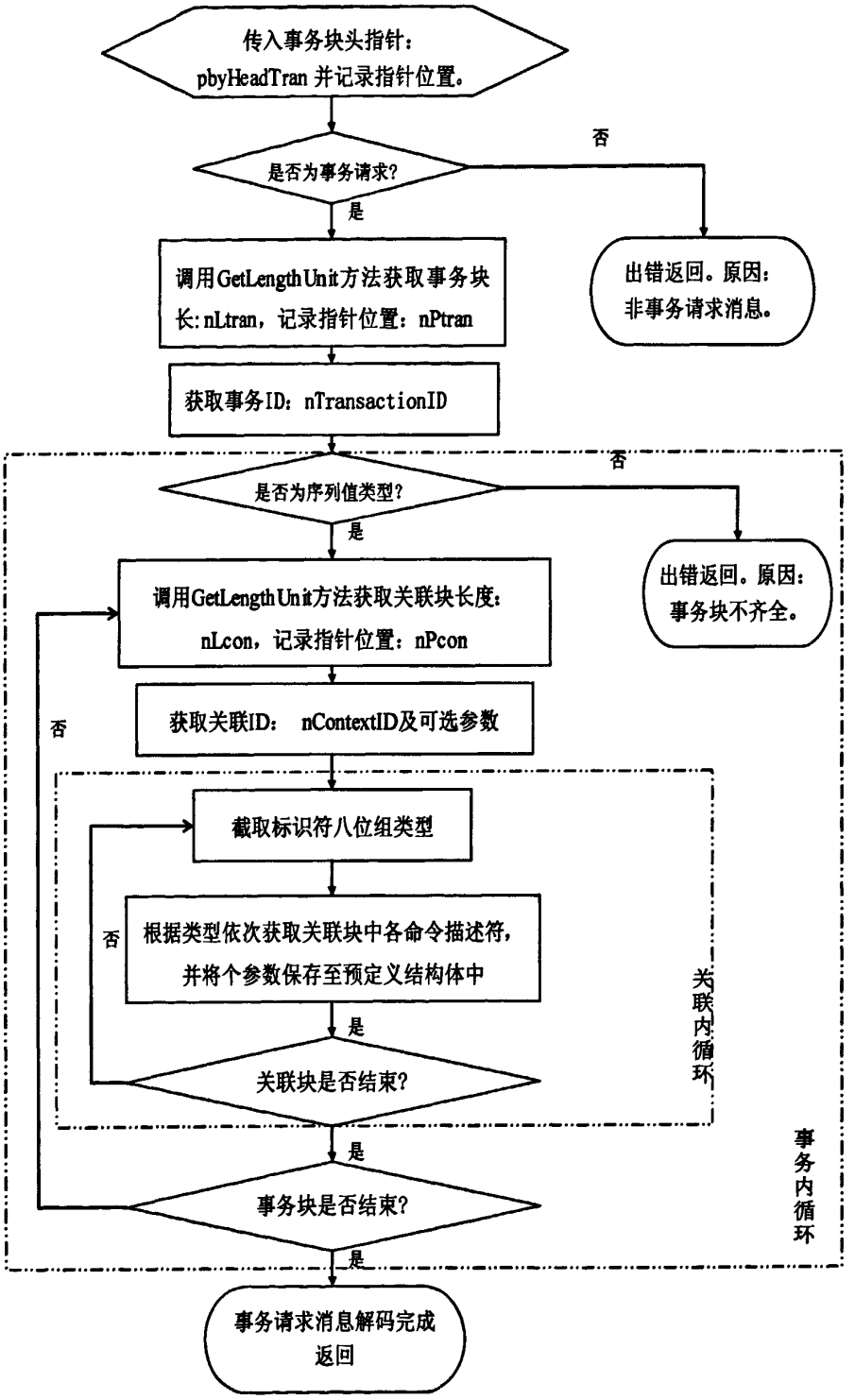

图 4.8 H.248 协议事务请求消息的详细解码流程图

由 H.248 协议的消息结构可知：消息体（也就是 PDU）内存在若干个事务块；事务块内部包含了若干个关联块（动作）；关联块内部包含了若干个命令；每个命令包含了自身的输入、输出参数及命令描述符。根据 H.248 消息的这种消息结构，我

们在详细解码流程采取了事物内循环嵌套关联内循环的算法流程。这样的流程，使得被解析消息的整个解码过程能够严格按照自身的比特流顺序进行一次解码，并能够在每进入内一层循环之前精确地定位到每一层次自身携带的参数如TransactionID、ContextID等层头数据，进而及时地对这些关键数据进行保存。在图中可以看到，最外层循环为 PDU 内循环，中间层为事务内循环，最内层是关联内循环，而整个解码流程完成的是一个逐层解析的工作。其中，事务块的结束的标志为：指针当前位置减去 nPtran 等于 nLtran（事务块字节长）；关联块结束的判断依据为：指针当前位置减去 nPcon 等于 nLcon（关联块字节长）。这里需要注意的是：在关联内循环中，各命令的命令描述符数量结构都大不相同，需要根据 H.248 协议的二进制编码规范进行精确的解析。

4.3.4 H.248 协议详细解码功能的测试及结果分析

上面提到的是 H.248 协议中事务请求消息(TransactionRequest)的解码流程。根据 ITU 标准，H.248 消息还包括事务响应消息(TransactionReply)、事务临时响应消息(TransactionPending)、事务响应证实消息(TransactionResponseAck)等。他们的区分标识是事务块的标识符八位位组的值（均是上下文指定类的结构化编码：事务请求为 0xA0，事务响应为 0xA2）。它们的语法结构各不相同，我们按照图 4.8 的设计思想分别对它们进行了解码流程的设计。由于详细解码功能属于用户发起的即时查看功能，对实时性的要求并高，所以在我们在 H.248 协议解码功能的设计中，采用了上述的解码流程对 H.248 消息进行详细解码，其结果如图 4.9 所示：

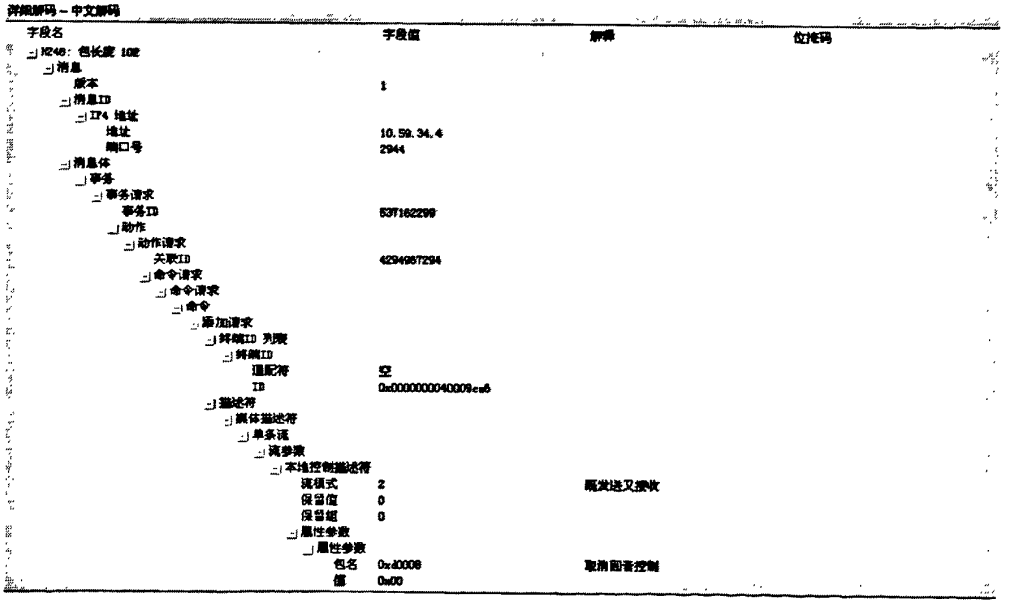

图 4.9 H.248 协议详细解码结果演示

结果中显示：该条 H.248 消息的消息 ID 字段为“10.59.34.4：2944”；第一个事务块为事务请求消息，事务 ID 为“537162299”；事务块内第一个关联 ID 为“4294967294”；第一个命令为一条 Add Request 消息，携带了命令描述符为 Local（RTP 流属性为 SendReveice）；实现的包为呼叫进展信号音检测包，信号值为 0x00（回铃音终止）。整个解码结果准确无误。

4.3.5 合成解码功能的设计

合成解码与详细解码功能的不同之处在于：详细解码为用户提供了查看具体一条协议消息的解码结果，而合成解码则为呼叫合成等高层协议分析功能服务。以呼叫合成功能为代表的高层功能实现的基础都是获取每条消息中的关键信息，这些高层功能操作的对象是海量的消息数据，这就对关键信息的提取速度提出了很高的要求。合成解码并不像详细解码那样对协议消息进行完全的解码，而只关注关键数据的提取。它的功能特点是实时性高，速度快和获取精确。合成解码功能的解码对象是原始数据，解码目标是合成的关键数据，解码要求是快速、准确。根据这些特点，我们在设计 H.248 协议合成解码过程中单独地设计了合成解码流程。

对于 H.248 协议来说，事务包含关联的嵌套结构不仅使消息结构变得清晰，更重要的是这样的嵌套使得协议的连接模型很好地被诠释在消息数据中，表现为关联 ID 是呼叫连接的标识。根据我们的合成方案设计（见第五章），需要对每一个关联都触发一次呼叫合成（关联建立前由终结点 ID 与事务 ID 相配合触发合成）。在实现中，就是基于关联（及每一关联建立之前相关的动作），来保存每个关联内的信息，如事务类型（命令请求还是命令响应）、信令方向、主被叫信息等。

我们设计 H.248 合成解码结果类 H248CallInfo 来保存这些信息：

```
class CH248CallInfo : public CallInfo
{
public:
    uint32 TransactionID;              //事务 ID
    uint32 ContextID;                  //关联 ID
    Ter_Msg TerMsg[MaxTerMsgNum];      //消息终端容器
    int8 nTerMsgNum;                   //消息终端数目
    CSDPInfo LocalSdp;                     //本地 SDP 信息
    CSDPInfo RemoteSdp;                    //远端 SDP 信息
    uint16 ErrorCode;                  //错误码
    Top_Msg TopMsg;                        //拓扑消息
```

```
    Top_Direct TopDirect;              //拓扑方向
    Action AC;                         //动作
    int Transaction;                   //事务类型
    int CMDSeq;                        //命令序号
    char    TelNum[MAX_CALL_NUMBER_LEN];          //被叫电话号码
public:
    CH248CallInfo();
    ~CH248CallInfo(){}
    ...
};
```

根据面向对象的思想，合成解码结果类也采取从基类继承的方式实现。不同的协议中的特定关键信息由各协议合成解码结果子类实现。在 Mc 模块的 H.248 协议合成解码结果类 H248CallInfo 中，定义了包括事务 ID、关联 ID、终结点 ID（存储在消息终端数组中 TerMsg）、本地/远端 SDP 信息、事务类型、拓扑消息/方向、错误码等在内的与呼叫合成密切相关的关键信息。同时类中还实现了关于合成解码结果清空等方法对合成解码结果类进行维护。

合成解码的流程在设计思想上与详细解码相同，但对具体信息命令参数、包等不过于关心，而是直接利用之前所述的协议识别函数、长度八位位组解码函数等方法，控制指针跳动、定位至事务块内，继而有效地获取关联 ID、事务 ID、终结点 ID 等关键信息，并能将完成合成的消息位置进行保存，在查看 CDR 时选择某条消息的详细解码时传递给详细解码函数作为入口。

4.4 Megaco 协议解码功能的设计与实现

Megaco 协议是采用 ABNF 语法规范描述的文本格式编码的媒体网关控制协议。ABNF[11]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，如下：

name = elements crlf

其中，name 即规则名（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 即元素，表现为一个或多个规则名亦或是终结符的形式；crlf 即行结束标志，为回车符后紧跟换行符的形式。等号将规则名与它的定义分隔开；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）值遵循 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），同时 ABNF 还规定了各操作符之间的优先级。

例如下面的一条 Megaco 消息：

```
MEGACO/1[10.35.2.101]:2944
T=452422{C=299{
N=A19{OE=385879552{al/on}}}}
```

根据 Megaco 协议的 ABNF 语法规范[11]解析上面的消息，结果如下所示：

```
MegacopToken: MEGACO    //  指示为 Megaco 协议
Version: 1              //  版本为 1
mId: [10.35.2.101]:2944 //  MG 的 IP 地址和端口号分别为 10.35.2.101:2944
TransToken: T           //  事务请求
TransactionID: 452422   //  事务 ID 为:452422
CtxToken: C             //  关联指示
ContextID: 299              //  关联 ID 为 299
NotifyToken: N              //  通知命令
TerminationID: A19          //  终结点 ID 为 A19
ObservedEventsToken: OE //  观测到的事件描述符
RequestID: 385879552    //  请求 ID 为 385879552
pkgdName: al/on         //  指示为模拟线包中的挂机事件
```

传统的 ABNF 语句的解码方式是对字符串进行查找和匹配的方式进行解析。即将指针进行诸字符的移动，根据换行结束标志截取一段字符串与协议规范内正确字符组进行匹配，进而取出消息中的每个阈值。但是，由于 WCDMA 网路测试仪对实时性高要求，传统的解码算法显得非常繁琐、效率比较低下，速度无法达到要求。为此，在协议解码功能的设计过程中，我们采用了正则表达式法来解析 Megaco 协议，以达到提高解析速度的目的。

正则表达式[19,20]（regular expression），就是用某种模式去匹配一类字符串的公式。即用一个“字符串”来描述一个特征，然后去验证另一个“字符串”是否符合此特征。它可被用于：

①验证字符串是否符合指定特征，比如验证是否是合法的邮件地址、IP 地址等。

②用来查找字符串，从一个长的文本中查找符合指定特征的字符串，比查找固定字符串更加灵活方便。

③用于替换。

WCDMA 网络测试仪的开发在 C++环境下完成，因此我们选择了全部采用模板、纯 C++代码编写，并可支持 VC++、GCC、Turbo C++等编译环境的 DEELX 正则表达式引擎为 Megaco 协议解码功能服务。

DEELX 接口简单且完全开源，所有代码只有一个头文件，包含该头文件即可调用引擎功能。它主要包括三个类：CRegexpT 模板类、MatchResult 类、CContext 类。其中，CRegexpT 模板类是 DEELX 中最重要的类，负责正则表达式的编译和匹配替换；MatchResult 类用来记录匹配结果，其对象中记录了所匹配到的字符串在整个文本中的位置，以及各个捕获组的位置；CContext 类中存放了匹配过程中的数据。

4.4.1 协议识别功能

在ABNF规范描述的文本编码的网络协议报文中都含有一些固定的字符或字符串来标识报文中特定域的存在或标识报文的类型。

例如 Megaco 协议中，有如下定义[21]（部分）：

```
megacoMessage = LWSP [authenticationHeader SEP ] message
Message = MegacopToken SLASH Version SEP mId SEP messagebody
MegacopToken = ("MEGACO" / "!")
SLASH = %x2F          ; "/"
Version = 1*2(DIGIT)     ; 1 至 2 位的数字
SEP = ( WSP / EOL / COMMENT) LWSP      ; LWSP 是线性空白字符
```

根据以上定义可知：Megaco 消息的 Message 域由一个值为“MEGACO”或“!”的 MegacopToken 域作为开头，随后通过一个“/”的分隔符与版本号 Version 域分开；之后是消息 ID（mId）和消息体（messagebody）两个值域，并通过 SEP（这里是空格符）相互分隔。根据定义：若要判断某一个消息是否为 Megaco 消息，只需要判断协议的开头，即 MegacopToken 和 Version 两个域内的值是否符合协议的定义即可。由此，Megaco 协议识别函数设计如下：

```
1   bool AFX_EXT_CLASS IsMegacoPdu(const CPdu& pdu)
    //AFX_EXT_CLASS 关键字指示全局函数从 DLL 中导出
2   {
3       BYTE *pbyMsg = pdu.GetHeadPointer();          // CPdu::GetHeadPointer()
4       static CRegexpT <char> regexp("(?:MEGACO|!)\d{1,2}");
5       MatchResult result = regexp.Match(pbyMsg);    // Find
6       int nRes = result.IsMatched();
7       if (nRes == 0)                                // Not Matched
8       {
9           TRACE(_T("Error Message");
```

```
10          return false;
11      }
12      return true;
13  }
```

在该函数中，首先通过调用 Cpdu 类的 GetHeadPoniter()函数获取数据块的头指针，然后以字符形式具体化了一个模板类对象 regexp，通过传入正则表达式“(?:MEGACO|!)\d{1,2}”对其进行初始化，该正则表达式的含义为该字符串中含有“MEGACO/”或“!/”并紧跟一个 1 至 2 位的数字；随后的第五行中，利用获取的数据块头指针调用 regexp 的 Match()方法，根据设定的正则表达式对消息进行查找，将查找结果保存在一个 MatchResult 对象中；最后在第六行中，调用 MatchResult 类的 IsMatched()方法判断匹配结果，即该消息是否是 Megaco 消息。

4.4.2 合成解码功能

在上一节中提到，合成解码不追求对协议数据进行逐比特地解码，而是提取消息中的关键数据和关键值，从而为消息显示，CDR（呼叫详细报告）合成等高级功能提供依据。

因此，在设计 Megaco 协议的合成解码功能时，所关注的关键数据有 mId、ContextID 等。本小节以 mId 值域为例，展示如何通过正则表达式法查找、提取消息中该域的值。

在 ABNF 语法规范中对 mId 域的定义如下:

```
mId = (( domainAddress / domainName )[":" portNumber]) / mtpAddress / deviceName
domainName = "<" (ALPHA / DIGIT) *63(ALPHA / DIGIT / "-" / ".") ">"
deviceName = pathNAME
domainAddress = "[" (IPv4address / IPv6address) "]"
IPv6address = hexpart [ ":" IPv4address ]
IPv4address = V4hex DOT V4hex DOT V4hex DOT V4hex
V4hex = 1*3(DIGIT) ; "0".."255"
hexpart = hexseq "::" [ hexseq ] / "::" [ hexseq ] / hexseq
hexseq = hex4 *( ":" hex4)
hex4 = 1*4HEXDIG
portNumber = UINT16
mtpAddress = MTPToken LBRKT 4*8 (HEXDIG) RBRKT
MTPToken = ("MTP")
```

```
ALPHA = %x41-5A / %x61-7A    ; A-Z / a-z
DIGIT = %x30-39              ; 0-9
```

从上面的定义可以看到，mId 域实际上由三部分组成，分别为(domainAddress / domainName)[":" portNumber]、mtpAddress、deviceName。ABNF 语法规范中详细定义了每个部分的语法构成以及是否存在或包含的合法字符。根据正则表达式的语法和 mId 的定义，可以分别编写(domainAddress / domainName)[":" portNumber]、mtpAddress、deviceName 这三部分的正则表达式，再完成 mId 域的完整正则表达式，如下所示：

```
(?:(?:(?<domainAddress>\[(?:((?#IPv4address)?:\d{1,3}(?:\.\d{1,3}){3})|((?#IPv6address)?:(?:[0-9A-F
]{1,4}(?::[0-9A-F])*(?:::(?:[0-9A-F]{1,4}(?::[0-9A-F])*)?)?)|(?:::(?:[0-9A-F]{1,4}(?::[0-9A-F])*)?))))\]
)|(?<domainName><[0-9a-zA-Z][0-9a-zA-Z-.]{0,63}>))(?<portNumber>:\d{1,5})?)
//(domainAddress / domainName)[":" portNumber]部分正则表达式
(?:MTP(?:[\x20\0x09]|;[0-9a-zA-Z\x09\x20-\x2F\x3A-\x40\x5B-\x60\x7B-\x7E]*(?:\x0D\x0A?|\x0A)|
(?:\x0D\x0A?|\x0A))*\{(?:[\x20\0x09]|;[0-9a-zA-Z\x09\x20-\x2F\x3A-\x40\x5B-\x60\x7B-\x7E]*(?:\x
0D\x0A?|\x0A)|(?:\x0D\x0A?|\x0A))*(?<mtpAddress>[0-9A-F]{4,8})(?:[\x20\0x09]|;[0-9a-zA-Z\x09\
x20-\x2F\x3A-\x40\x5B-\x60\x7B-\x7E]*(?:\x0D\x0A?|\x0A)|(?:\x0D\x0A?|\x0A))*\}(?:[\x20\0x09]|;[
0-9a-zA-Z\x09\x20-\x2F\x3A-\x40\x5B-\x60\x7B-\x7E]*(?:\x0D\x0A?|\x0A)|(?:\x0D\x0A?|\x0A))*)
// mtpAddress 部分正则表达式
(?<deviceName>\*?[a-zA-Z]\w{0,63}[\w/*$]*(?:@[0-9a-zA-Z*][0-9a-zA-Z\-*.]{0,63})?)
// deviceName 部分正则表达式
```

根据上一小节所述的方法，我们可以利用当调用相同的方法查找并匹配我们关心的 mId 域内容。如下所示：

```
1   int CH248Decoder::GetCallInfo(CallInfoMegaco& cInfoMegaco)
2   {
3       BYTE *pbyMsg = GetHeadPointer();
4       ...
5       static CRegexpT <char> regexp("<上述的 mId 域的正则表达式，此处省略>");
6       MatchResult result = regexp.Match(pbyMsg);
7
8       if (result.IsMatched())
9       {
10          int nGroupNumber =
            regexp.GetNamedGroupNumber(_T("domainAddress"));
11          if (nGroupNumber > 0)    // 存在 domainAddress 分组
```

```
12          {
13              int nGroupStart = result.GetGroupStart(nGroupNumber);// 起始位置
14              int nGroupEnd = result.GetGroupEnd(nGroupNumber);// 结束位置
15
16              _tcsncpy(cInfoH248.szDomainAddr, pbyMsg +nGroupStart,
17                  nGroupEnd-nGroupStart<MAX_ADDR_LEN ?
18                  nGroupEnd-nGroupStart : MAX_ADDR_LEN);
19          }
20      }
21      ...
22  }
```

由于我们在正则表达式中采用了命名分组[20](?<name>xxx)的方式来标识 mId 域中的重要信息（如 domainAddress，portNumber 等），经 Match 方法的匹配后，即可以调用 CRegexpT 类的 GetNamedGroupNumber()方法来获取预定义在正则表达式中的“domainAddress”命名分组的编号，如第 10 行代码所示。若该编号小于 0，则表示解析字段中未包含该命名分组。代码第 13－14 行分别调用 MatchResult 类的 GetGroupStart 和 GetGroupEnd 方法分别获取分组的起始位置和结束位置；第 16－18 行通过作为形参传入的消息头指针和该命名分组的起始位、结束位置，取出消息中“domainAddress”的字段信息，保存到合成解码结果中。

Megaco 协议的合成解码还需要对其他关键数据进行提取，在后续的代码中我们采用同样的方法逐一实现各数据的提取和保存；最终的合成解码结果通过带引用的形参进行返回，供呼叫合成函数使用。这里不再赘述。注：由于 Megaco 协议与 H.248 协议的区别仅在于编码方式的不同，我们将其解码实现亦封装在 CH248Decoder 类中实现，采用不同类型参数的函数重载区别两种协议的相关解码函数。

4.4.3 详细解码功能

对于 Megaco 协议的详细解码功能，我们利用与合成解码相同的方法实现。即首先对协议的输出格式进行预定义；接着以正则表达式分组命名方式逐字段地提取消息中每一域的值，在整体上达到比特级的精度；最后按照预先定义的格式保存解码内容到字符串变量，供主控程序的详细解码结果显示模块调用，格式化输出详细解码结果。

4.4.4 Megaco 协议详细解码功能的测试及结果分析

Megaco 协议的详细解码结果如图 4.10 所示：

结果中显示：该条消息是版本 1 的 Megaco 消息，其 MID 为“10.6.160.1：2944”；第一个事务块为事务请求，其事务 ID 为 2943994532；该事务块内第一个关联 ID 为“NULL”；第一条命令为 Modif 消息，作用于 ID 为“USER00200300025”的终结点；该终结点实现的包有：dd（DTMF 检测包），mfd（脉冲信号检测包）。解码结果正确无误。

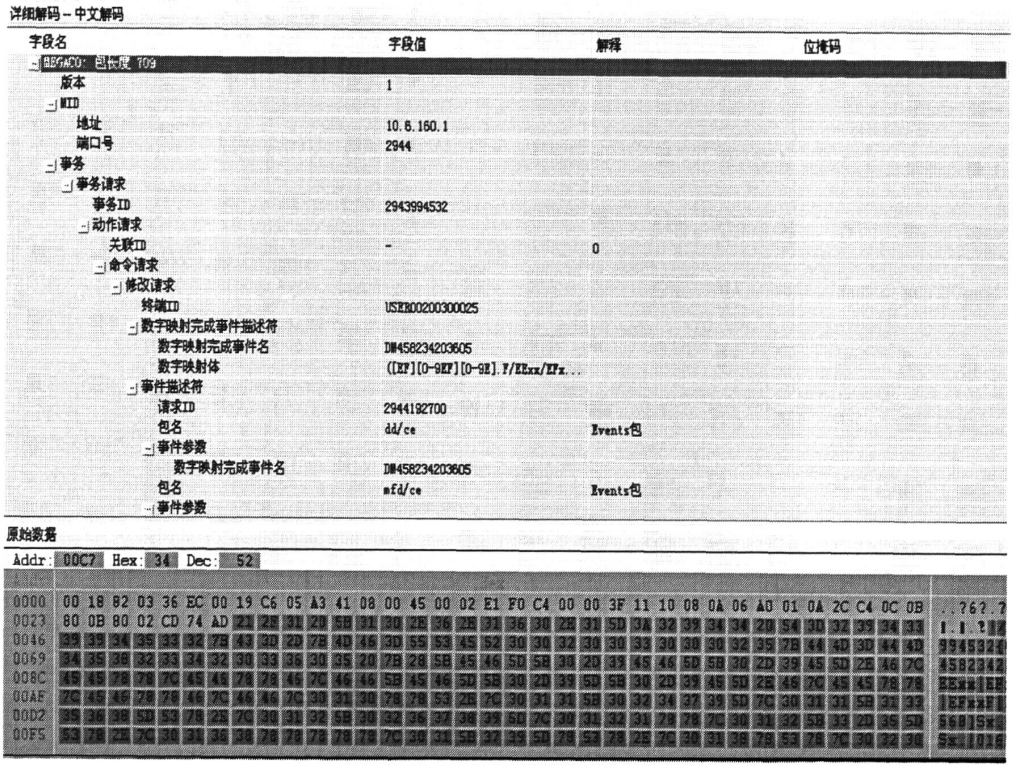

图 4.10 Megaco 详细解码结果演示

4.5 本章小结

本章首先对 Mc 接口监测模块解码功能进行了需求分析；接着根据面向对象的编程思想，设计了协议解码功能的类封装结构；然后通过对 BER 规则的编码原理的研究和“TLV 三元组”结构的理解，对二进制格式编码的 H.248 协议消息的协议识别、详细解码和合成解码功能进行了设计和实现；最后利用正则表达式法对文本格式编码的 Megaco 协议消息的协议识别、详细解码和合成解码功能进行了设计和实现。

第五章 Mc接口监测模块呼叫合成及统计分析功能的设计与实现

5.1 呼叫合成功能的需求分析

CDR(Calling Detail Records)即呼叫详细记录，它描述了呼叫接续的全过程。这里所说的呼叫接续过程是指所有类型的业务流程，呼叫合成功能就是在网络测试仪中重现网络中曾经发生的业务流程，通过对这些记录中的一些重要参数进一步的分析和处理，可以为通信网业务提供分析的基础。呼叫合成功能是网络测试仪应具备的高级功能，它能为用户提供能网络中某一业务过程的重现分析、故障分析及定位等功能。通过对大量呼叫流程的分析，可以对网络或设备的运行质量和运行状况进行更深层次的剖析

呼叫合成的过程是指：网络测试仪从所采集的海量原始信令数据中将属于每一业务流程的信令消息分配到他们自己的业务流程组中，组织成这一业务流程的CDR，并最终通过上层界面的图形功能向用户进行显示。

从呼叫合成的定义中我们可以看到呼叫合成的最大难点首先在于如何从海量的原始信令数据中提取属于每一业务流程的信令，这种提取过程应该是由一次遍历完成的。在前一章我们提到了合成解码功能，它为呼叫合成功能提供了每条信令消息的合成解码功能。通过合成解码，每一条原始信令消息被标记上了它们的自身的标签，即消息ID和呼叫合成关键信息。在这样的基础上，我们需要能制定出合理的呼叫合成方案，以对所有原始数据进行一次完整遍历的过程中，为采集到的信令所涉及的所有业务流程建立它们自己的CDR，并将每条信令消息分配到所属的CDR中（这其中包括不完整的CDR及不完整的消息）。其次，呼叫合成的难点由网络测试仪高实时性要求所决定：采用何种合理的算法设计使得整个合成过程正确、快速、高效地完成。

Mc接口监测模块是WCDMA网络测试仪中的重要模块，它的呼叫合成功能的实现同样面临着上述的难点。在本章下述内容中，本文将首先通过对Mc接口上的H.248协议业务流程进行分析，进而设计出合理的呼叫合成方案，最后在此基础上设计相应的合成类与合成算法。

5.2 Mc 接口业务流程分析

为设计合理的呼叫合成方案，我们必须基于对 Mc 接口上的 H.248 协议标准及业务流程的正确理解，总结出各类型业务流程的关键合成信息。

根据 H.248 协议及国家行业标准 YD/T 1594-2007 中的规定，我们将 Mc 接口上的 H.248 消息信令流程分为两类[22-24]，WCDMA 网络内的呼叫流程和 WCDMA 网络与外部网络间的呼叫流程。以下简称为网内终端呼叫流程与网间中继呼叫流程。它们的本质区别是业务流程是否涉及到 GMSC Server（关口局移动交换服务器），而每一类型又被分为主叫侧和被叫侧两种。

5.2.1 网内呼叫流程分析

以两个 WCDMA 网络内的两个终端之间建立呼叫连接为例说明 Mc 接口上网内呼叫连接模型和呼叫流程（注：网内两个终端之间的呼叫建立和释放流程与该流程基本相同），如图 5.1 所示。其中，UE1 为主叫，连接 MGW1；UE2 为被叫，连接 MGW2；UE1 先挂机（图中实线为 H.248 消息，虚线为 UTRAN 侧信令）。

消息流程分解如下：

事件 1：

MGW1 监测到 UE1 的业务请求消息，将此摘机事件（al/of）通过 Notify Request 消息上报给 MSC Server。

MSC Server 向 MGW1 返回 Notify Reply 消息确认收到用户摘机事件。

事件2：

MSC Server向MGW1发送Modify Request消息，要求MGW1检测用户收号完成（dd/ce）、挂机（al/on）、拍叉簧（al/fl）事件，并向MGW1发送号码表（Digitmap）。

MGW1 向 MSC Server 返回 Modify Reply 消息。与此同时，主叫用户在 UTRAN 内的无线链路建立、鉴权、指配等步骤完成。

事件 3：

UE1 拨号，MGW1 根据 MSC Server 下发的号码表进行收号，并将所拨号码与匹配结果通过 Notify Request 消息上报 MSC Server。

MSC Server 向 MGW1 返回 Notify Reply 应答。

事件 4：

MSC Server 向 MGW1 发送 Add Request 消息，在 MGW1 创建一个新关联（Context），并在新关联中加入 UE1 的终结点（TDM termination）和 RTP 终结

点（RTP termination）。其中 RTP 的连接模式设置为 ReceiveOnly，并设置语音压缩算法。

MGW1 向 MSC Server 返回 Add Reply 响应，分配新的连接描述符、RTP 终端 RTP1，并设置 RTP1 的 IP 地址、端口号、媒体类型及编码类型等信息。

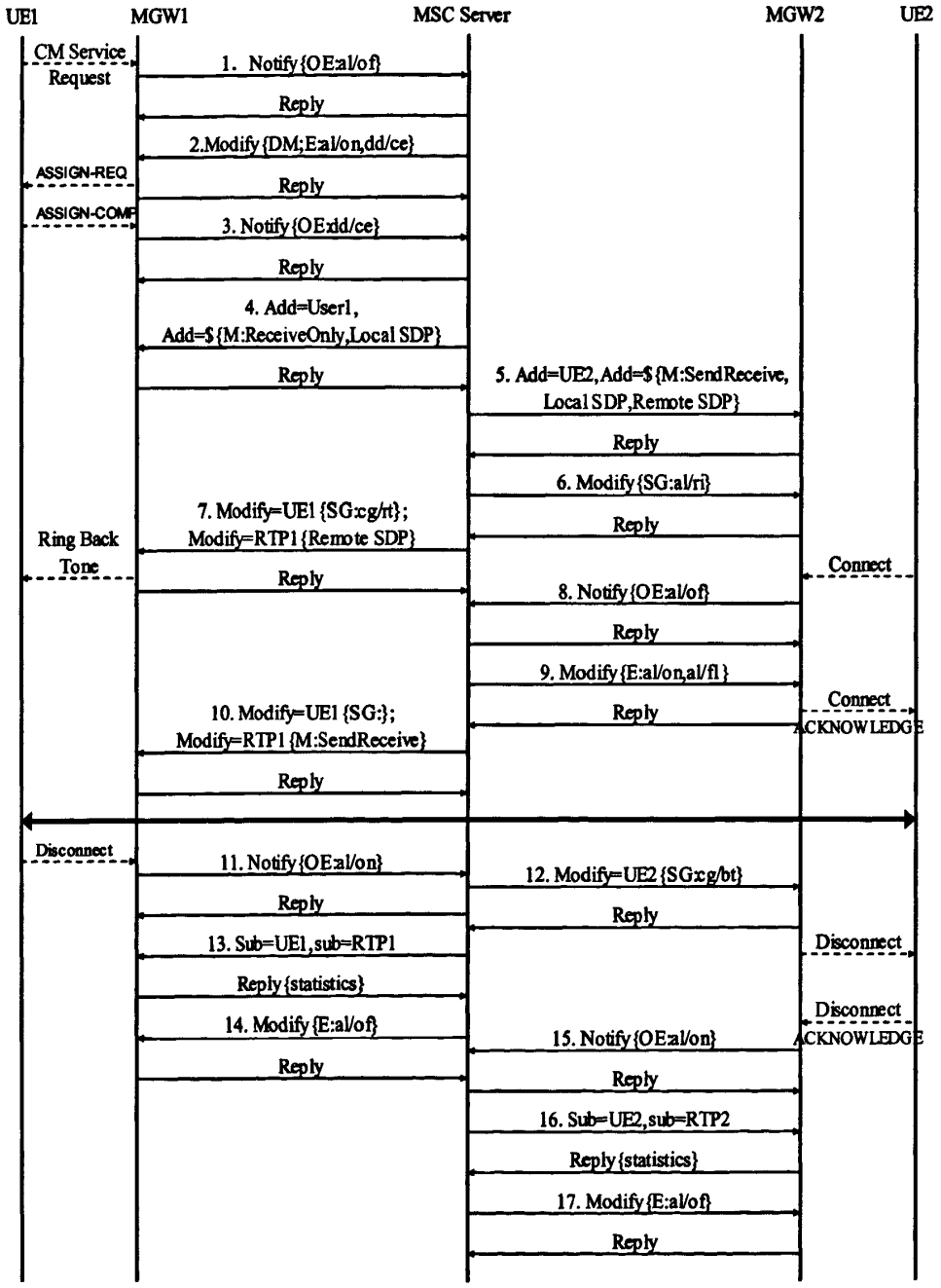

图 5.1 网内终端呼叫流程

事件 5：

MSC Server根据主叫用户所拨号码通过HLR及VLR搜索被叫用户地址信息（小区号、临时漫游号等）等寻址MGW2，向其发送Add Request消息，在MGW2创建一个新的关联，在新关联中加入UE2的终结点和RTP终结点，其中RTP的连接模式设置为SendReceive，设置远端RTP地址及端口号、媒体类型及编码类型等信息。

MGW2向MSC Server返回Add Reply响应，分配新的连接描述符、RTP终端RTP2，并设置RTP2的本地端IP地址、端口号、媒体类型及编码类型等信息。

事件6：

MSC Server向MGW2发送Modify Request消息，请求MGW2向被叫UE2发送振铃音（al/ri）。

MGW2向MSC Server返回Modify Reply应答。

事件7：

MSC Server向MGW1发送Modify Request消息，请求MGW1向主叫UE1放回铃音（cg/rt），并设置RTP1的远端RTP地址及端口号、媒体类型、编码类型等信息。

MGW1向MSC Server返回Modify Reply应答。

事件8：

MGW2监测到UE2摘机（被叫向MGW2发送CONNECT消息），将此摘机事件（al/of）通过Notify Request消息上报给MSC Server。

MSC Server向MGW2返回Notify Reply应答。

事件9：

MSC Server向MGW2发送Modify Request消息，通知MGW2监测UE2的挂机（al/on）、拍叉簧（al/fl）事件。同时MGW2向UE2发送Connect acknowledge消息,被叫端无线链路建立结束。

MGW2向MSC Server返回Modify Reply应答。

事件10：

MSC Server向MGW1发送Modify Request消息，停止向UE1播放回铃音signal{}，并设置RTP1的连接模式为SendReceive。

MGW1向MSC Server返回Modify Reply消息；UE1与UE2正常通话。

事件11：

MGW1监测到UE1的挂机，将此挂机事件（al/on）通过Notify Request命令上报给MSC Server。

MSC Server向MGW1返回Notify Reply应答。

事件12：

MSC Server向MGW2发送Modify Request消息，通知MGW2对UE2放忙音（cg/bt）。

MGW2向MSC Server返回Modify Reply应答。MGW2同时向UE2发送Disconnect消息，被叫端无线链路拆线开始。

事件13：

MSC Server向MGW1发送Subtract Request消息，释放UE1和RTP1。

MGW1向MSC Server返回Subtract Reply应答，释放资源，并向MSC Server上报呼叫的媒体流统计信息。同时MGW1向UE1发送REL释放消息，主叫端无线链路拆线开始。

事件14：

MSC Server向MGW1发送Modify Request消息，通知MGW1监测UE1的摘机（al/of）事件。

MGW1向MSC Server返回Modify Reply应答。

事件15：

Uesr2发送Disconnect Acknowledge消息，被叫端无线链路拆线完成。MGW2认为监测到UE2挂机，将此挂机事件（al/on）通过Notify Request命令上报给MSC Server。

MSC Server向MGW2返回Notify Reply应答。

事件16：

MSC Server向MGW2发送Subtract Request消息，释放UE2和RTP2。

MGW2向MSC Server返回Subtract Reply应答，向MSC Server上报呼叫的媒体流统计信息。

事件17：

MSC Server向MGW2发送Modify Request消息，通知MGW2监测UE2的摘机事件（al/of）。

MGW2向MSC Server返回Modify Reply应答。

5.2.2 网间中继呼叫流程分析

我们以外部网络用户为主叫呼叫WCDMA网络用户的建立呼叫连接为例说明Mc接口上网间呼叫连接模型和呼叫流程，如图5.2所示，其中SG为主叫用户（UE1）侧的信令网关，连接MGW1；UE2为被叫，连接MGW2；主叫先挂机（注：网间中继呼叫的出网呼叫，即WCDMA网络用户做主叫的流程起始步

骤与网内终端呼叫流程类似)。

消息流程分解如下：

事件 1：

UE1 摘机拨号后，GMSC Server 收到主叫交换机经 SG 转发的 ISUP 初始地址消息（IAM)，然后向 MGW1 发送 Add Request 消息，在 MGW1 中创建一个新的关联，并在新关联中加入入局中继的终结点和 RTP 终结点，其中 RTP 的连接模式为 ReceiveOnly，并设置编码类型。

MGW1 向 GMSC Server 返回 Add Reply 响应，分配新的连接描述符、RTP 终端 RTP1，并设置 RTP 的本地端 IP 地址、端口号、媒体及编码类型等信息。

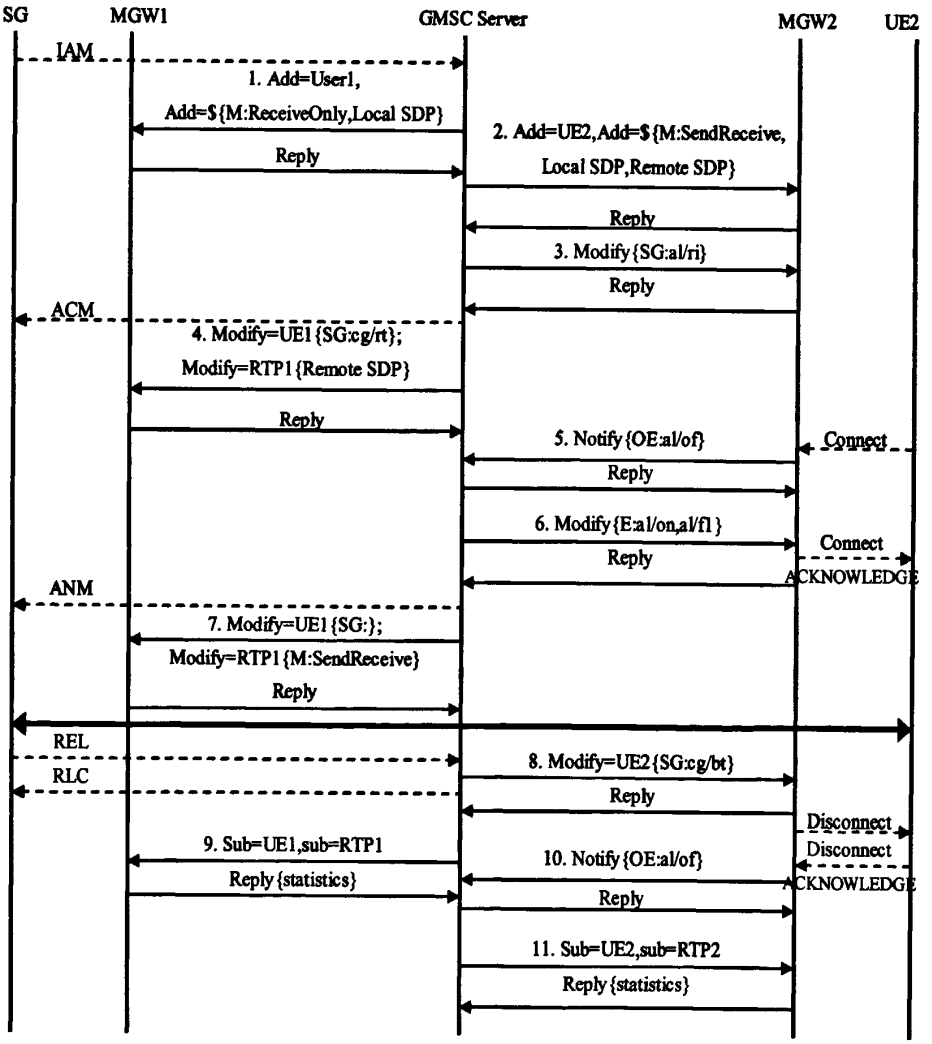

图 5.2 网间中继呼叫流程

事件 2：

MSC Server根据主叫用户所拨号码通过HLR及VLR搜索被叫用户地址信息（小区号、临时漫游号等）等寻址 MGW2，并向其发送 Add Request 消息，在MGW2 创建一个新的关联，并在新关联中加入 UE2 的终结点，其中 RTP 的连接模式设置为 SendReceive，并设置远端 RTP 地址及端口号、媒体类型及编码类型等信息。

MGW2 向 GMSC Server 返回 Add Reply 响应，分配新的连接描述符、RTP 终端 RTP2，并设置 RTP 的本地端 IP 地址、端口号、媒体类型及编码类型等信息。

事件 3：

GMSC Server 向 MGW2 发送 Modify Request 消息，请求 MGW2 向被叫 UE2 发送振铃音（al/ri）。

MGW2 向 GMSC Server 返回 Modify Reply 应答。

事件 4：

GMSC Server 收到MGW2 正确的 Modify Reply 应答后向 MGW1 发送 Modify Request 消息，设置远端 RTP 地址及端口号、媒体类型、编码类型等信息。同时向主叫交换机发送地址全消息（ACM）。

MGW1 向 GMSC Server 返回 Modify Reply 应答。

事件 5：

MGW2 监测到 UE2 摘机（被叫向 MGW2 发送 CONNECT 消息），将此摘机事件（al/of）通过 Notify Request 消息上报给 GMSC Server。

GMSC Server 向 MGW2 返回 Notify Reply 应答。

事件 6：

GMSC Server 向 MGW2 发送 Modify Request 消息，通知 MGW2 监测 UE2 的挂机(al/on)、拍叉簧(al/fl)事件。同时 MGW2 向 UE2 发送 Connect acknowledge 消息,被叫端无线链路建立结束。

MGW2 向 GMSC Server 返回 Modify Reply 应答。

事件 7：

收到 MGW2 的 Modify Reply 消息后，GMSC Server 向主叫交换机发送链路建立消息（ANM）；同时向 MGW1 发送 Modify Request 消息，将其 RTP 连接模式设置为 SendReceive。

MGW1 向 GMSC Server 返回 Modify Reply 应答。链路建立完毕，通话开始。

事件 8：

UE1 挂机，主叫交换机经 SG 向 GMSC Server 发送 ISUP 释放消息（REL），收到该消息后 GMSC Server 向 SG 发送释放完成消息（RLC），并向 MGW2 发送

Modify Request 消息，通知 MGW2 对 UE2 放忙音（cg/bt）。

MGW2 向 GMSC Server 返回 Modify Reply 应答。MGW2 同时向 UE2 发送 Disconnect 消息，被叫端无线链路拆线开始。

事件 9：

GMSC Server 在收到 MGW2 的 Modify Reply 后向 MGW1 发送 Subtract Request 消息，释放主叫中继和 RTP1。

MGW1 向 GMSC Server 返回 Subtract Reply 应答，并上报呼叫的媒体流统计信息。

事件 10：

Uesr2 发送 Disconnect Acknowledge 消息，被叫端无线链路拆线完成。MGW2 认为监测到 UE2 挂机，将此挂机事件（al/on）通过 Notify Request 命令上报给 GMSC Server。

GMSC Server 向 MGW2 返回 Notify Reply 应答。

事件 11：

GMSC Server 向 MGW2 发送 Subtract Request 消息，释放 UE2 和 RTP2。MGW2 向 GMSC Server 返回 Subtract Reply 应答，并上报呼叫的媒体流统计信息。

5.3 Mc 接口呼叫合成功能的设计与实现

5.3.1 呼叫合成方案的设计

经过在上一节中我们对 Mc 接口上的 H.248 协议呼叫流程进行的分析，我们可以把 H.248 协议呼叫流程分为网内终端呼叫的主叫流程、被叫流程以及网间中继呼叫的主叫流程、被叫流程。它们中的每一条具体消息的规则都遵循 H.248 协议标准及其扩充的规定，但它们的起始消息和业务流程有所不同。正是根据这种不同，本文对它们进行了划分。

总结以上四类呼叫流程的特点如下：（a）所有的主叫流程以携带了监测到的 al/of 摘机事件的 Notify Request 消息作为呼叫起始消息。被叫呼叫流程则以 Add Request 消息作为呼叫起始消息；（b）在所有的流程中，发起 Add Request 之前的所有消息的关联 ID（ContextID）均为“NULL”（文本编码中为“-”，二进制编码中为 0），且消息中只有一个终结点 ID（TerminationID）；（c）在呼叫流程的第一条 Add Request 消息中，其关联 ID 为“CHOOSE”（文本编码中为“$”，二进制编码中为 4294967294），而其后的 Add Reply 消息中会携带为其分配的关联 ID

及 RTP 终结点 ID。在后续消息中，均携带了该分配的关联 ID，直至呼叫结束，关联被删除；

由此我们可以得知，在 Mc 接口上呼叫流程进行的不同阶段，都有不同的关键信息能够唯一标识该阶段：在检测到摘机事件到主叫终结点收到第一条 Add 消息之前，由于关联未建立，因此以终结点 ID 作为该阶段的唯一标识最为合理；将终结点加入新建关联的 Add 请求与响应这一对交互消息需要区别对待，若该对消息的终结点 ID 属于一对正在合成中的 CDR，则这对消息属于主叫侧流程。若它们不属于任何以后的 CDR，则此对消息为被叫侧呼叫合成的起始消息，此种情况下事务 ID 能够唯一标识每一段相互独立流程；在主叫侧和被叫侧的呼叫流程中，Add Reply 消息后的阶段都因以该次呼叫所归属的关联 ID 作为唯一。如下图所示：

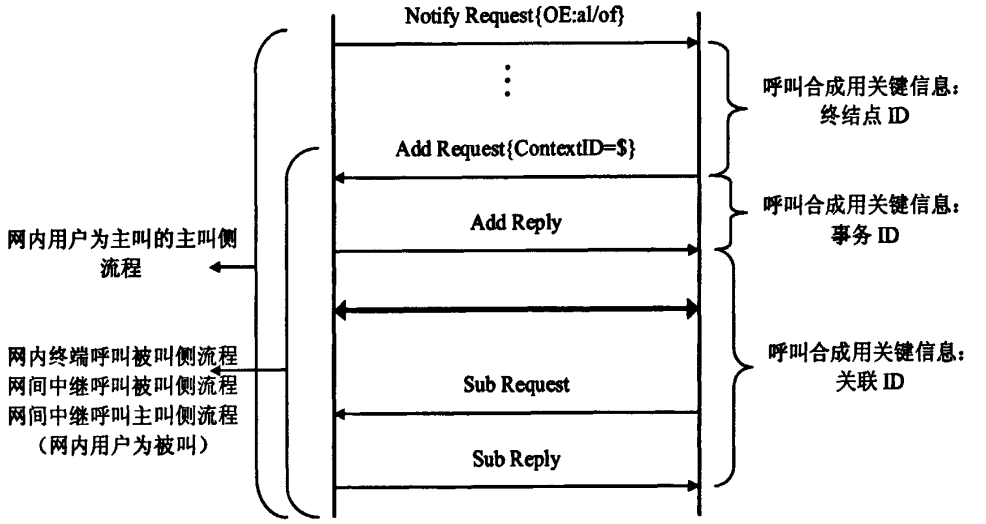

图 5.3 业务流程不同阶段的关键信息

根据本章中对 Mc 接口上的呼叫合成功能的需求分析，呼叫合成就是对海量数据进行一次遍历，经过一次遍历以后达到将每条消息分配到它们自己的 CDR 中。而我们对呼叫合成方案的设计，是针对每当呼叫合成程序获取一条消息时，如何通过一种正确的流程或方法，根据该消息所携带的关键信息判断它属于哪一个业务流程（即哪一个 CDR）。其具体的思想如下：

每处理一条 H.248 消息时，首先查看它的合成解码结果：若该条消息的关联 ID 为空，则意味着该条消息属于主叫侧的起始阶段。因此，提取该条消息的终结点 ID 信息，并据此在合成中的 CDR 列表中进行搜索。若搜索成功，即该消息所属的 CDR 已经被建立，则将此条消息加入对应 CDR 中。若未搜索到，则新建一个 CDR，并将该消息加入其中。

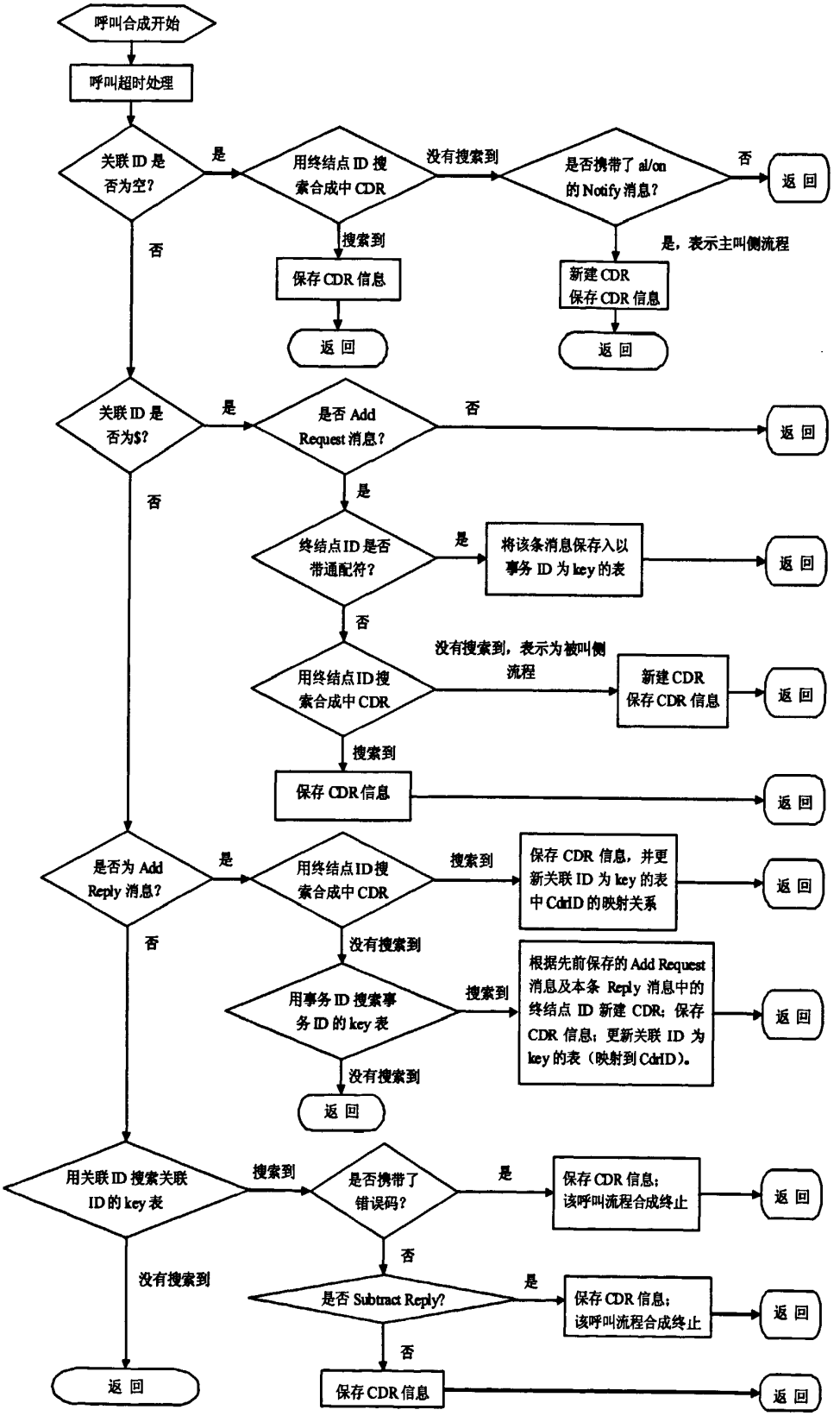

图 5.4 H.248 协议呼叫合成方案流程图

若该消息的关联内容不为空，可以确定它并不属于业务起始阶段。接下来需要先判断关联内容是否包含通配符“$”，若是则该消息应为Add Request消息，否则该消息分析结束，结果为错位返回。在确定为Add Request消息的情况下，继续判断该消息的事务ID是否为通配符，在否定的情况下利用终结点ID搜索CDR列表，保存已有的CDR信息或新建一个被叫侧流程的CDR并保存。若终结点ID为通配符，表明该消息是一个被叫侧流程的起始消息，由于终结点ID信息不完整，应先更新事务ID的key表，并在表中对该消息进行映射供后续合成过程使用。图5.4是Mc接口上H.248协议呼叫合成详细方案的流程图。

若在之前的判断中，该消息的关联内容不为通配符“$”，则先判断该消息是否为Add Reply消息。若是则先利用终结点ID信息搜索合成中的CDR列表，搜索成功则保存CDR信息并更新关联ID为key的表中关联ID与CdrID的映射信息，以供后续阶段的呼叫合成使用。若终结点ID的搜索失败，则利用事务ID搜索事务ID为key的表，在成功的情况下配合之前保存的Add Request消息新建并保存一个被叫侧CDR，同样维护关联ID与CdrID的映射信息。

若消息既不是Add Request又不是Add Reply消息，关联ID也不为空，则该消息是位于业务流程后期阶段的消息。因此应该首先提取该消息的关联ID，并据此搜索关联ID为key的表，若映射信息表中无相关的CDR信息则说明该消息不属于任何已有的CDR，又不是一个CDR流程的开始，这种情况下不对此CDR做任何动作，返回的同时对该条消息做不属于现存CDR的标记。若在现存key表中搜索到该关联ID的CDR则先查看该消息是否携带错误码[25]，若是则保存CDR并终止该CDR的合成，返回主流程对下一条消息进行分析。若不携带任何错误码则先查看是否为Subtract Reply消息，若是则同样保存并终止该CDR。若消息既不携带错误码又不为Subtract Reply消息，则只保存CDR并进入下一条消息的分析过程。

5.3.2 呼叫合成功能的算法设计

上一节中的呼叫合成方案设计基本解决了在一次遍历中完成消息的分，但仅仅这样是不够的。如前面提到，网络测试仪对实时性的要求非常高，所以还需要解决的关键问题就是效率问题。某一时刻，系统的缓存区内可能有海量的数据等待被分类，在我们制定的存储区内有大量未完成的呼叫流程，合成程序每提取到一条消息都需要在未完成的CDR中进行搜索、判断、修改等操作。在这种情况下，如何组织这些未完成的呼叫，以测试仪可以迅速找到每条消息的所属呼叫流程，是实现呼叫合成功能需要解决的首要问题。

为此，我们引入了 HASH（哈希）算法[26]，即通过消息的合成解码结果中的关键信息建立合成中 CDR 的 HASH 散列表。以查找 hash 表的方式搜索合成中的 CDR 能够最大程度的减小算法复杂度，加快程序处理速度。

Hash，即哈希散列，就是把任意长度的输入（又叫做预映射，pre-image），通过哈希算法，变换成固定长度的输出，该输出就是哈希值。这种转换是一种压缩映射，也就是，哈希值的空间通常远小于输入的空间。

哈希散列的数学表述为：$h = H(M)$，其中 $H()$ 为单向哈希函数，M 为任意长度的明文，h 为固定长度的哈希值。Hash 算法需要满足以下特点：

1. 单向性：从预映射，能够简单迅速的得到哈希值，而在计算上不可能构造一个预映射，使其哈希结果等于某个特定的哈希值，这样，哈希值就能在统计上唯一的表征输入值。
2. 抗冲突性：即在统计上无法产生 2 个哈希值相同的预映射。
3. 映射分布均匀性和差分分布均匀性：散列结果中，为 0 的 bit 和为 1 的 bit，其总数应该大致相等，输入中一个 bit 的变化，散列结果中将有一半以上的 bit 改变，这又叫做“雪崩效应（avalanche effect）”。

这里我们应该注意到，Hash 算法设计有一个矛盾，即越复杂的算法能够使出现两个预映射产生相同哈希值的概率越小，但是同时会导致算法复杂度的提高。而简单的 hash 算法必然要带来不同的输入哈希成相同的输出这种情况，即冲突。因此，我们在设计 hash 算法的同时需要设计出相应的冲突解决方法。

在Mc接口上的H.248协议合成功能设计过程中，我们采用采用除法散列法来构造哈希函数，除法散列法的哈希函数h(M)为：h(M)= M mod m（即哈希地址h(M)为关键码K除m所得的余数）。

哈希冲突解决方法采用线性探查法。d为发生冲突的地址，依次探查d的下一个地址（当到达下标为m-1的哈希表尾时，下一个探查的地址是表首地址0），直到找到空闲单位为止。线性探查法的数学递推公式为：

d0 = h(K)

di = (di-1 + 1) 对m取模(1≤i≤m-1)

根据我们的呼叫合成方案，在 Mc 接口上的 H.248 呼叫流程的不同阶段需要由不同的关键信息（Key）来标识该次呼叫流程：终结点 ID、事务 ID、关联 ID。这其中终结点 ID 是呼叫流程中最先出现的关键信息，我们选择终结点 ID 作为 CDR 的唯一标识。与此同时，程序对事务 ID 与终结点 ID 的映射关系以及关联 ID 与终结点 ID 的映射关系进行实时的保存，以便在呼叫流程的不同阶段将消息准确地加入所属 CDR 中。同时考虑到 CDR 的庞大数量，为了避免内存的过度消

耗，我们采用文件的方式组织所有合成中和合成完的CDR，把所有的CDR都顺序存储在文件当中，并建立索引文件，以便能够通过索引迅速搜索到相应的CDR。图5.5是CDR在系统中的存储示意图。

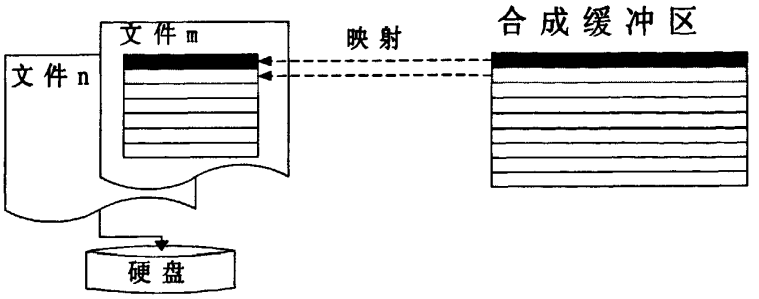

图5.5 CDR在系统中的存储

在合成中，我们需要建立并维护三个hash表，他们分别是以终结点ID、事务ID、关联ID作为预映射的hash表。我们以终结点ID作为预映射的hash表为例进行介绍。

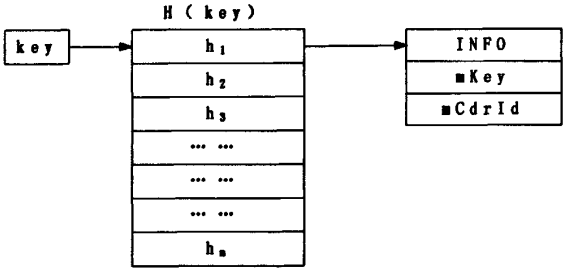

图5.6 终结点ID为key值的hash表结构

如图5.6所示的hash表中，预映射经过hash散列后被定为到hash数组中的某一项，该项为输入的key值和呼叫合成临时信息INFO的结构体。INFO中包含了CdrId、CDR状态、各项key值（可能是类）等关键信息。以事务ID、关联ID和终结点ID为预映射的hash表结构均如上图所示。

在以上的描述中不难看出，CdrId是一个关键的索引值，我们将某一类型的所有CDR都存储在一个该类型CDR的数组中，CdrId就是程序对此数组进行存取的索引值。

在呼叫合成功能的设计中，我们采用了这样的算法思想：按照上一节所述的呼叫合成方案，对各条消息的合成解码结果进行分析；按照合成方案以终结点ID为关键值创建新的CDR（这其中包括利用事务ID的hash表将一对Add命令与响应进行映射，继而创建新CDR），创建新CDR的同时将新的key值（该终结点ID）插入终结点ID的hash表以便在后续操作中查找合成中的CDR；终结点ID的hash表反映出终结点ID与所属的H248Record的映射关系（该H248Record

中包含 CdrId、INFO 的信息等)；实时地更新事务 ID 和关联 ID 为 key 值的 hash 表，并实时地维护他们与所属 CdrId 之间的映射；将每个 CDR 的具体信息（如呼叫其实时间等）存入 CDR 数组所对应数组成员中（即一个 H248CDR 类的对象)；最后将每个 H248CDR 类对象以及对应的 H248Record 类对象中的信息提交给合成函数，调用主控程序界面显示功能完整的显示一个呼叫流程。

5.3.3 呼叫合成功能的类封装设计

与解码模块一样，Mc 接口合成模块也遵守了整个网络测试仪平台的统一 CDR 合成基类，CDR 的文件组织方式，CDR 记录的新建、搜索等方法来实现 Mc 接口上的 H.248 协议呼叫合成功能。

（1）合成缓存类

我们从基类 ICDRBuf 派生出 CH248CDRBuf 这一类型，作为模块向外部的合成缓存封装。其定义如下所示：

```
class CH248CDRBuf : public ICDRBuf
{
    CCDRBuff<CH248Cdr> * m_H248Record;
public:
    CH248CDRBuf();
    virtual ~CH248CDRBuf();

    void SetH248Record(CCDRBuff<CH248Cdr> * pH248Record);

    virtual void SetCdr(uint32 uCdrID, IITEM * pCdr);
    // 修改 uCdrID 的 cdr 为 pCdr 指向的内容

    virtual IITEM * GetCdr(uint32 uCdrID);
    // 分配一个对象，存储 uCdrID 对应的内容，并返回其 IITEM 接口指针

    virtual void ReleaseCdr(uint32 uCdrID, IITEM * pItem);
    // 释放 pItem 对应的内存块

    virtual bool GetCdrMsg(uint32 uCdrID, CSzlList<MsgItem> & msglist);
    // 得到 cdr 对应的消息链表,成功返回 true，否则返回 false
```

```
    virtual bool GetEnumList(int32 nIndex, CSzlList<EnumItem> & list, int32
    nLng);
    // 得到本 CDRBuf 对应的 ITEM 中 nIndex 项对应的所有枚举值链表,
    nLng（0 中文 1 英文）表示仪何种文字呈现。

    virtual int32 GetCdrCount();
    // 得到本 CDRBuf 中 CDR 的条数
};
```

CH248CDRBuf 类中包含了类型为 CCDRBuff 模板类的私有成员，并以 CH248Cdr 类型进行了实例化。该模板类中提供了对 hash 表、映射关系进行操作的方法，并内建了包括 CDR 文件与 CDR 记录、各 CDR 及包括的消息 ID、各 CDR 和消息的对应关系在内的三张索引表以及消息链表，还提供了 SetCdrMsg 方法对消息链表进行操作和标记。该方法如下：

```
SetCdrMsg(uint32 uCdrID, uint32 dwMsgID, int32 nBitOffset, int32 nBitLength)};
//设定 dwMsgID 所标识的消息属于 uCdrID。成功返回 true, 失败返回 false
```

CCDRBuf 模板类是 WCDMA 网络测试仪基础库中的重要类，模块利用这个类直接对“物理上”的合成缓存区进行操作，从而起到组织 CDR 合成，维护 CdrId 与其所属 CDR 及呼叫合成缓存之间的映射的关系等作用。CDH248CDRBuf 类提供的 SetCdr 等成员函数就是调用了 CDRBuf 提供的方法，该类完成了对合成完成后的 CDR 相关操作，并作为 Mc 模块向主控模块提交的封装。

（2）呼叫记录集类

我们从基类 IH248Record 派生出 CH248Record 这一类型，作为 H.248 协议的 CDR 记录集。其定义如下所示：

```
class CH248Record : public IH248Record
{
    CCDRBuff<CH248Cdr> m_H248CallBuf;

    CHash<CH248Info, CH248KeyContext>   m_HashContext;
    CHash<CH248Info, CH248KeyTransaction>   m_HashTransaction;
    CHash<CH248Info, CH248KeyTerminal>   m_HashTerminal;//内建的三张 hash 表
public:
    CH248Record();
```

```
virtual ~CH248Record();
virtual int32 GetCdrCount();                    // 得到当前 Cdr 的条数

virtual bool GetCdr(uint32 unCdrID, CH248Cdr & cdr);
// 得到 ID 号为 unCdrID 的 CDR，成功返回 true，失败返回 false

virtual bool GetCdrMsg(uint32 unCdrID, MsgItem msg[], int32 & nMsgCount);
// 得到 ID 号为 unCdrID 的 cdr 里的所有消息,成功返回 true，失败返回 false

virtual void GetHandleTime(int32 & nStartTimeSec, int32 & nEndTimeSec);
// 得到分析的起始、结束时刻

CH248Info * Search(CH248KeyContext & Key);
void Insert(CH248KeyContext & Key, CH248Info & Info);
bool Remove(CH248KeyContext & Key);
// Context 的 Hash 查找、插入和删除

CH248Info * Search(CH248KeyTransaction & Key);
void Insert(CH248KeyTransaction & Key, CH248Info & Info);
bool Remove(CH248KeyTransaction & Key);
// Transaction 的 Hash 查找、插入和删除

CH248Info * Search(CH248KeyTerminal & Key);
void Insert(CH248KeyTerminal & Key, CH248Info & Info);
bool Remove(CH248KeyTerminal & Key);
// Terminal 的 Hash 查找、插入和删除

inline int32 GetContextComposingCount();
//返回当前有多少个 Context 合成中呼叫

inline int32 GetTransactionComposingCount();
//返回当前有多少个 Transaction 合成中呼叫
```

```
    inline int32 GetTerminalComposingCount();
    //返回当前有多少个 Terminal 合成中呼叫

    bool IsInProgress(const CH248KeyContext & Key);
    bool IsInProgress(const CH248KeyTransaction & Key);
    bool IsInProgress(const CH248KeyTerminal & Key);
    //判断某一个节点（不同的类型的 key 值）是否在合成中，
    线程安全的。返回值:true 合成中，false 不在合成中
};
```

CH248Record 同样封装了一个以 CH248CDR 来实例化的 CDRBuf 模板类为它的公有成员，并以 CDRBuf 提供的方法来管理合成缓存，组织 CDR 文件。同时，CH248Record 记录集在类中内建了三张不同类型 key 值的 hash 表，通过他们维护 key 值与 CdrId 的映射，进而对 CDR 数组进行存取。CH248Record 类封装了在 Mc 模块内部创建、更新 CDR 和统计结果的功能和方法；又避免了对合成缓存区进行直接的操作，起到了封装和保护的作用。（注：记录集类中的 CH248Info 类型是我们为呼叫合成临时信息设计的 H.248 类型，这些临时信息仅存在于呼叫合成过程中，当 CDR 结束后，该参数中的信息会被删除。）

（3）呼叫合成相关类型参数类

在实例化模板类 CDRBuf 时，我们用到了一个 CDR 类型参数，该类被用来存储每一个 CDR 需要保存的呼叫相关信息。这些信息包括用户关心的 CDR 属性以及统计分析需要的信息，例如：终结点 ID、Mc 接口两端 IP、本地/远端 SDP 描述[27]、事务 ID、关联 ID、CDR 状态等。CDR 参数中的内容在 CDR 结束后保存在文件中，再经由 CH248CDRBuf 统一封装供 CDR 显示或统计分析使用。该 CDR 类型 CH248Cdr 类设计如下：

```
class CH248Cdr : public CCallCdr
{
public:
    CH248Cdr()
    {
        uCDRSubType = P_H248; // 设置本 CDR 的协议类别
        memset(TerminID, 0, MaxTerIDLen);
        MGW_IP = 0;
        MSCServer _IP = 0;
```

```
        ErrorCode = reason_null;
        MTS = MGW;    //网关类型初始化为 MGW
        ZBX = OnUnknownSide;    //呼叫侧初始为未知
        State = CA_INVALID;
    }
    virtual ~CH248Cdr()
    {
    }
public:
    char TerminID[MaxTerIDLen];          //终端 ID
    uint32 MGW_IP;                       //MGW 地址
    uint32 MSCServer_IP;                 //(G)MSC Server 地址
    uint8 ZBX;                           //呼叫侧
    MGType MTS;                          //网关类型
    reason_C ErrorCode;                  //错误码
    CSDPInfo LocalSdp;                   //本地端 SDP
    CSDPInfo RemoteSdp;                  //远端 SDP
    int32 State;                         //CDR 当前状态
    uint32 uTransactionID;               //事务 ID
    uint32 uContextID;                   //关联 ID
};
```

其中 ZBX 为 CDR 的呼叫侧类型，对 H.248 协议来说分为主叫侧和被叫侧两种；State 用于指示 CDR 的当前状态，主要包括未接通、已接通、接通未应答、已应答、通话中、通话结束、呼叫结束等七种，可由消息中携带的参数信息来判断得来；本地及远端 SDP 包含了两端的 RTP 媒体类型、地址、端口及编码类型等信息，由 SDP（会话协议，Session Description Protocol）进行描述[27]。

在上面的描述中，可以看出：在呼叫合成过程中，key 值起到了至关重要的作用。根据合成方案，我们设计三种类型的 key 类型。它们分别是 CH248KeyContext、CH248KeyTransaction、CH248KeyTerminal，设计如下：

```
class CH248KeyContext
{
public:
    uint32   m_unContextID;          // 关联 ID
```

```
    uint32    m_unSrcIP;                    // 源端 IP
    uint32    m_unDstIP;                    // 远端 IP

public:
    CH248KeyContext();
    virtual ~CH248KeyContext();
    bool operator == (const CH248KeyContext & Key);
};

class CH248KeyTransaction
{
public:
    uint32    m_unTransactionID;            // 事务 ID
    uint32    m_unSrcIP;                    // 源端 IP
    uint32    m_unDstIP;                    // 远端 IP

public:
    CH248KeyTransaction();
    virtual ~CH248KeyTransaction();
    bool operator == (const CH248KeyTransaction & Key);
};

class CH248KeyTerminal
{
public:
    char   m_chTerminID[MaxTerIDLen];// 终结点 ID
    uint32    m_unSrcIP;                    // 源端 IP
    uint32    m_unDstIP;                    // 远端 IP

public:
    CH248KeyTerminal();
    virtual ~CH248KeyTerminal();
    bool operator == (const CH248KeyTerminal & Key);
```

```
};
```

呼叫合成过程中的临时信息，由 CH248Info 类封装：

```
class CH248Info
{
public:
    CH248Info();

    virtual ~CH248Info();

public:
    uint32 m_unCdrID;          // 本呼叫 CdrID
    H248Flag m_Flag;           // 标志（关于 H.248 协议连接模型状态的标识）
    int m_nTerNum;             // 终端数目
    int64 m_nTransactionID;    // 事务 ID
    int m_Transaction;         // 事务类型

    int32 m_nOldState;         // CDR 之前状态
    int32 m_nNewState;         // CDR 当前状态

    Ter_Msg TerMsg[MaxTerMsgNum];    //消息终端容器
    int8 m_nTerMsgNum;
    Top_Direct TopDirect;            //拓扑方向

    Mediatx_Type m_eMediatx;
    uint8 ZBX;
};
```

5.3.4 呼叫合成功能的实现

根据 WCDMA 网络测试仪软件架构的设计思想，Mc 接口的合成功能的实现采用了面向对象的程序设计方法。表现为采用类的方式将呼叫合成功能的实现封装成 DLL，对外仅提供相应的呼叫合成接口函数。H.248 协议呼叫合成类 CH248Analyzer 的设计如下：

```
class CH248Analyzer : public CAnalyzer
```

```
{
public:
    CH248Analyzer();
    virtual ~CH248Analyzer();
    virtual void Handle(CallInfo * pCallInfo);
    virtual void Reset();
private:
    CH248Record * m_pH248Record; // 用来存储 cdr 和统计记录集
    CH248Statistic * m_pH248Statistic; //统计类成员对象指针
    CTimeCheck<uint32> m_H248TimeCheck; //定义超时

    CH248KeyContext KeyContext;          // Context 的 key
    CH248KeyTransaction KeyTransaction;  // Transaction 的 key
    CH248KeyTerminal KeyTerminal;        // Terminal 的 key
    CH248StatisticKey KeyStatistics;     //统计的 key

    static IProRecognizer * m_pH248Recognizer;//协议识别变量

    CDRTYPE m_cdrType;//CDR 状态

    uint32 Handle248(CH248CallInfo * pH248CallInfo);//H248 合成函数

    virtual void InitCdr(CH248CallInfo *pH248CallInfo,CH248Info &info,
    CH248Cdr &cdr); // NTFY 初始化 CDR

    virtual void InitAddCdr(CH248CallInfo *pH248CallInfo,CH248Info &info,
    CH248Cdr &cdr); // ADD 初始化 CDR

    void UpdataCdr(CH248CallInfo *pH248CallInfo, CH248Info *pinfo, CH248Cdr
    &cdr); // 更新 CDR

    void ServiceChange(CH248CallInfo * pH248CallInfo, CH248Cdr &cdr);
    //ServiceChange（MGW 注册、注销）消息处理函数
```

```
void ErrorHandle(uint16 nErrorCode, reason_C &ErrorCode);//出错处理

bool IsStr(char* pch, char ch);//字段匹配判断函数

void CallState(CH248CallInfo *pH248CallInfo, CH248Info *pInfo, CH248Cdr
&cdr, int nms);
//呼叫状态统计函数
};
```

CH248Analyzer 中封装了呼叫合成（及统计分析）功能的所有操作方法，并确使这些方法以函数接口的形式提供给主控函数。其中：Handle()及 Reset()方法是一对针对 H.248 协议合成解码结果的操作函数，并能够完成记录分析起始、结束时间、处理 CDR 超时设置和超时节点、多协议关联触发等功能，是整个 Mc 接口呼叫合成功能的功能处理函数，Reset()是重置合成相关变量（呼叫记录集、超时信息等）；HandleH248()方法则是呼叫合成功能的功能实现函数，该方法按照我们所设计的呼叫合成方案，完成了 Mc 接口上 H.248 协议的合成功能；根据呼叫合成方案中的两种触发新建 CDR 的情况，InitCdr()和 InitAddCdr()方法实现了对不同情况的处理；UpdataCdr()方法实现了对 CDR 的更新和保存；ServiceChange()方法是对 ServiceChange 消息的处理函数，它能够根据 ServiceChange 消息中携带的信息修改 CH248Cdr 类对象中的相关参数，如 MSC Server 和 MGW 的 IP 地址等；ErrorHandle()是出错描述符处理方法；CallState()方法通过合成解码和合成临时信息设置呼叫合成状态的统计参数。（注：超时处理是指了防止由于网络丢包导致消息未收全使得大量合成中 CDR 占用内存而采取的一系列措施，如删除 hash 节点、搜索项、超时 CDR 等。）

5.3.5 呼叫合成功能的测试及结果分析

图 5.7-5.10 是 H.248 协议的呼叫合成模块在 WCDMA 网络测试仪上的实际运行结果图：

在 CDR 流程图中，左侧的时间表示发送消息报文的时间，上部的两个设备表示 MGW 和 MSC Server 地址，箭头方向代表消息发送的方向，箭头上均带有消息说明，如消息序号、命令名称、关键的参数信息等；用户如需查看流程中某条消息的详细解码，只需双击该消息即可（如图右半部份所示）。

图 5.7-5.10 分别为网内终端呼叫的主/被叫流程，网间中继呼叫的主/被叫流

程。所有流程均得到正确的重现和显示，拓扑方向清晰，各条消息正确无误。达到了 Mc 接口监测模块呼叫合成功能的设计要求。

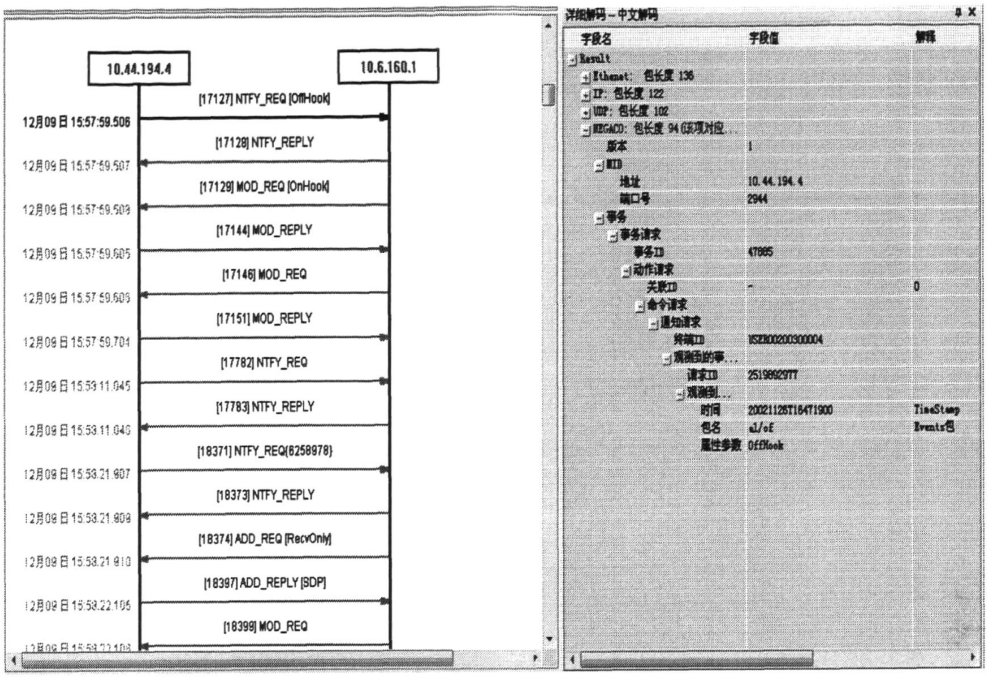

图 5.7 呼叫合成功能运行结果（网内终端呼叫的主叫侧流程）

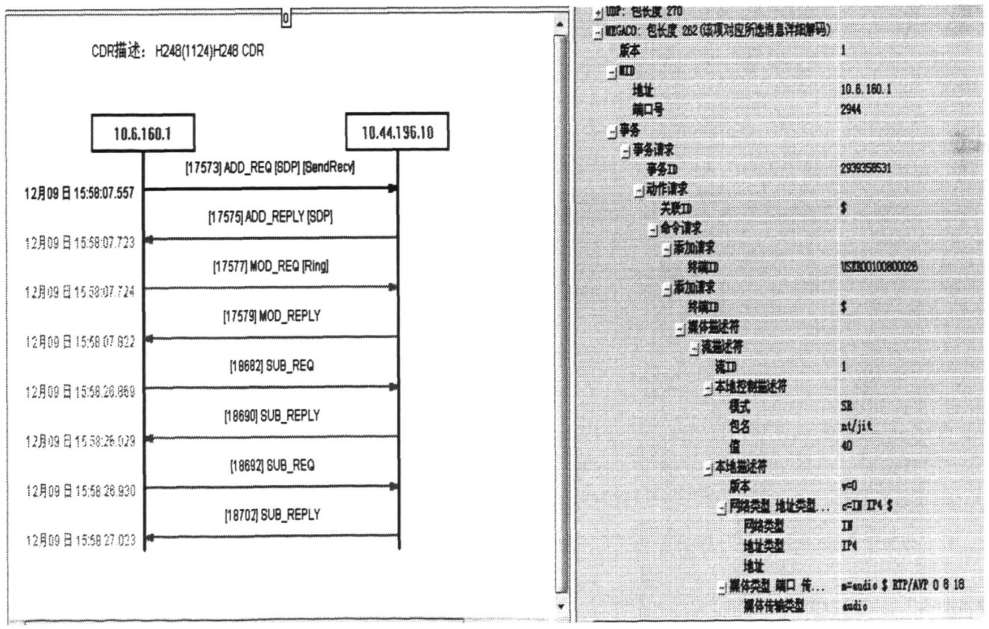

图 5.8 呼叫合成功能运行结果（网内终端呼叫的被叫侧流程）

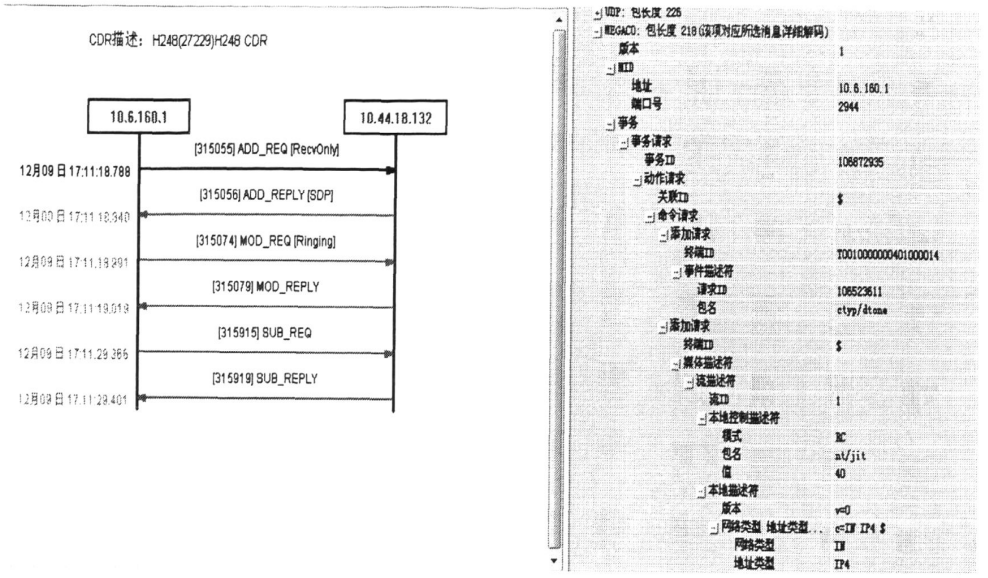

图 5.9 呼叫合成功能运行结果（网间中继呼叫的主叫侧流程）

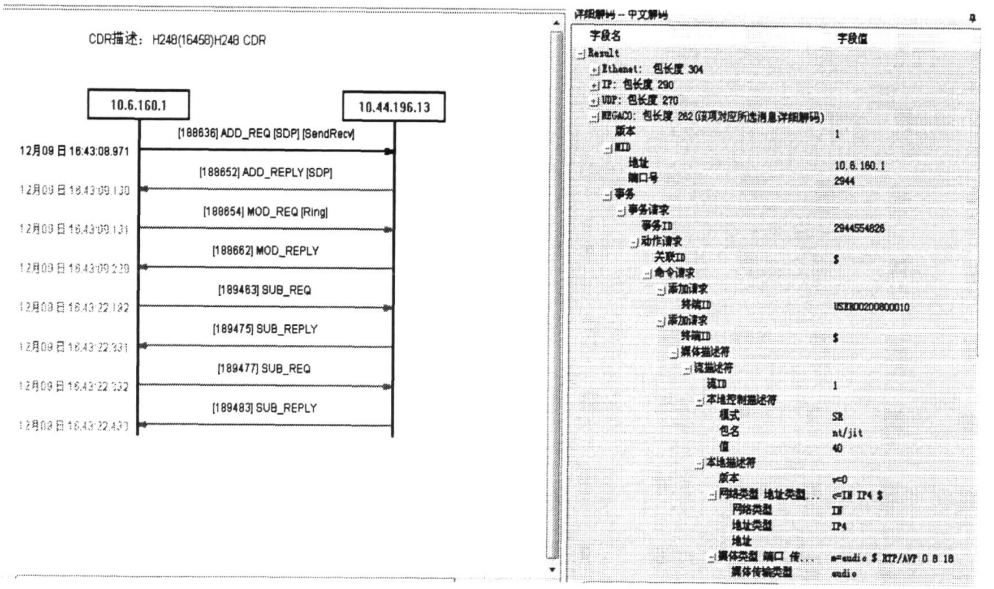

图 5.10 呼叫合成功能运行结果（中继呼叫的被叫侧流程）

5.4 Mc 接口统计分析功能的设计与实现

统计分析功能通过对大量 CDR 的分析，得到设备和网络的运行状况。从该功能的需求分析来看，大量 CDR 数据是分析的对象；而从实现角度来说，对海量消息进行一次遍历的这一 CDR 合成过程是统计分析实现的关键阶段。因此，我们在统计分析功能的设计中，将统计分析部分的代码内嵌入 CDR 合成代码中，与呼叫合成功能同步实现。

5.4.1 流量分析与业务相关指标统计

Mc接口的统计分析功能包括以下内容：

①流量统计：对指定时间内Mc接口上的协议数据进行吞吐量、包流量及其峰值等指标进行统计统计；

②呼叫统计：对MGW与MSC Server之间的呼叫总数、接通次数、应答次数、接通率、应答率等指标进行统计；

③话务统计：对单个MGW的通话总时长、应答总时长、接通总时长、以及该MGW内通话时间在各区间内的次数统计；

④性能统计：对MGW与MSC Server间建立的呼叫中，最长/最短接通时长、最长/最短应答时长、最长/最短通话时长的统计；

⑤呼损统计：对MGW与MSC Server之间按照终止原因和错误码统计呼损情况。

呼叫、话务、性能、呼损统计与CDR关联密切，对这些指标的统计是以CDR为单位的统计，是对每条CDR以及每条CDR中保存的呼叫信息的综合，关注的是Mc接口上各业务的运行状况，是对“高层”的统计。流量统计区别于其他指标的统计，它是对特定时长（5Min）内Mc接口上所有消息进行数据量的统计和处理。在Mc模块中表现为每5分钟形成一个流量文件对该5分钟内的流量统计结果进行保存。流量统计关注的是网络的带宽、传输能力、负荷情况等，是对“底层”的统计。

在流量统计中，比较重要的指标计算公式如下：

$$\text{包峰值(包/秒)} = \frac{{}_{N}\text{SliceMaxPacketnum}}{5*60} \quad N=1，2，3... \qquad (式5.1)$$

式5.1表示包峰值是选取包含最多消息条数的流量文件，计算每秒传输的消息速度。

$$\text{吞吐量峰值(KBPS)} = \frac{{}_{N}\text{SliceMaxBytenum}}{5*60*10^{3}} \quad N=1，2，3... \qquad (式5.2)$$

类似的，式5.2表示从所有5分钟时间段的流量结果文件中选取出包含最大字节数的文件，计算每秒传递的K字节数，以衡量数据传输媒介的带宽。

5.4.2 统计分析功能的设计

在统计分析功能的设计中，我们采用了hash算法对除流量统计外的其他指标统计进行实现：即通过统计节点的Key值创建hash表，再通过Key值的查找对应的统计结果。在CDR合成完成后，再根据每条CDR的状态修改、补充对应

统计项的属性和结果。对于流量统计，则采取了时间分片的方法，以一定的时间（5 分钟）为间隔，分段地统计并详细记录各时间段内的协议数据包个数和字节数。

①统计项结构体的设计：

统计分析功能的结果最终经由主控模块的控制在测试仪界面上进行显示。因此 Mc 模块必须将统计的结果以统一的封装结构向主控模块提交结果。我们设计了结构体 SH248StatisticItem 作为这一封装结构：

```
struct SH248StatisticItem
{
    ////////统计 Key//////////
    uint8 ZBX;                          //呼叫侧
    uint32 MGW_IP;                      //网关地址
    uint32 MSCServer _IP;               //软交换地址

    ////////呼叫统计//////////
    int32 nCallCount;           // 试呼次数
    int32 nInvalidCount;        // 无效呼叫次数
    int32 nEarlyRelCount;       // 呼叫早逝次数
    int32 nConnectCount;        // 用户接通次数
    int32 nAnswerCount;         // 应答次数
    int32 nCallOverCount;       // 呼叫结束次数

    //////////话务统计//////////
    uint64 nTotalConTimeUSec;           // 总接通时长
    STimeStatistic sConTimeStatistic;   // 接通时长分段统计
    uint64 nTotalAnsTimeUSec;           // 总应答时长
    STimeStatistic sAnsTimeStatistic;   // 应答时长分段统计
    uint64 nTotalTalkTimeMSec;          // 总通话时长(需考虑时间特别大的情况)
    STalkTimeStatistic sTalkTimeStatistic;  // 通话时长分段统计

    //////////性能统计//////////
    uint32 uUnUsed;//兼容结构体偏移使用
    int32 nConMaxTimeUSec;      // 最长接通时长
```

```
    int32 nConMinTimeUSec;          // 最短接通时长
    int32 nAnsMaxTimeUSec;          // 最长应答时长
    int32 nAnsMinTimeUSec;          // 最短应答时长
    int32 nTalkMaxTimeMSec;         // 最长通话时长
    int32 nTalkMinTimeMSec;         // 最短通话时长

    ////////////呼损统计//////////
    uint32 nErrorCode[reason_over];          //错误类型统计(预定义的错误码标识)
    ////////////流量统计//////////
    ......
    ////////////媒体统计//////////
    //统计各种媒体类型如' data autio video'等
    // 2 对应 audio，3 对应 video,4 对应 data,5 对应 application, 0.1 对应其他；
    //编码类型统计
    uint32 nEncodeType[FORMAT_COUNT];
    uint32 nMediaType[MDT_COUNT];
};
```

结构体 SH248StatisticItem 是用来记录 Mc 接口的一组 IP 地址间产生的业务数据。按照用户需求，定义了统计显示所需要的所有参数，用于主控模块调用相关统计项输出显示。

②业务相关指标统计 Key 类的设计：

业务相关指标的统计分析功能同样采取了选定Key值的Hash算法进行实现，Mc 接口上的 H.248 协议统计 Key 类定义如下：

```
class CH248StatisticKey
{
public:
    CH248StatisticKey();
    virtual ~CH248StatisticKey();
    bool operator == (const CH248StatisticKey & Key);

    uint32 MGW_IP;                          //MGW 的 IP 地址
    uint32 MSCServer_IP;                    //(G)MSC Server 地址
    uint8 ZBX;                              //呼叫侧
```

```
    uint16 uTimeSlice;                    //该统计项所属的时间片
};
```

通过这个 Key 值，我们在上一节描述的呼叫合成记录集类 CH248Record 中内建第四张 hash 表，即统计值的 hash 表，并提供对该表插入、查找、删除的方法。这样，H.248 呼叫记录集类所保存的内容又添加了统计方面的相关属性。

③流量分析的结构体封装及流程设计

流量统计不依附于 CDR 合成状态，与 CSIPCDR 类没有关联。它对 Mc 接口上的 H.248 消息不以 CDR 为单位进行统计，而是对特定时长（5 分钟）内所有的 H.248 消息进行数据量的统计和处理。可形象理解为每五分钟对接收的 H.248 消息形成一个 m_FlowSlice 文件。

```
struct SH248StatFlowSlice
{
    ////////流量统计//////////
    int64 nByteNum;                       //该时间片内的字节数
    int64 nPacketNum;                     //该时间片内的数据包数量
    int32 CMDNum[KAIWEN_GLAHANM];    //记录请求命令
};
```

一次流量分析需要对该过程中所有生成的 FlowSlice 进行汇总，为此我们定义了 SH248StatFlowTotal 来对整个流量分析过程的结果进行保存：

```
struct SH248StatFlowTotal
{
    ////////流量统计//////////
    int64 nByteNum;
    int64 nSliceMaxByteNum;
    int64 nPacketNum;
    int64 nSliceMaxPacketNum;
    int32 CMDNum[KAIWEN_GLAHANM];
};
```

Mc 接口的流量统计与呼叫合成同步实现，如图 5.11 所示：

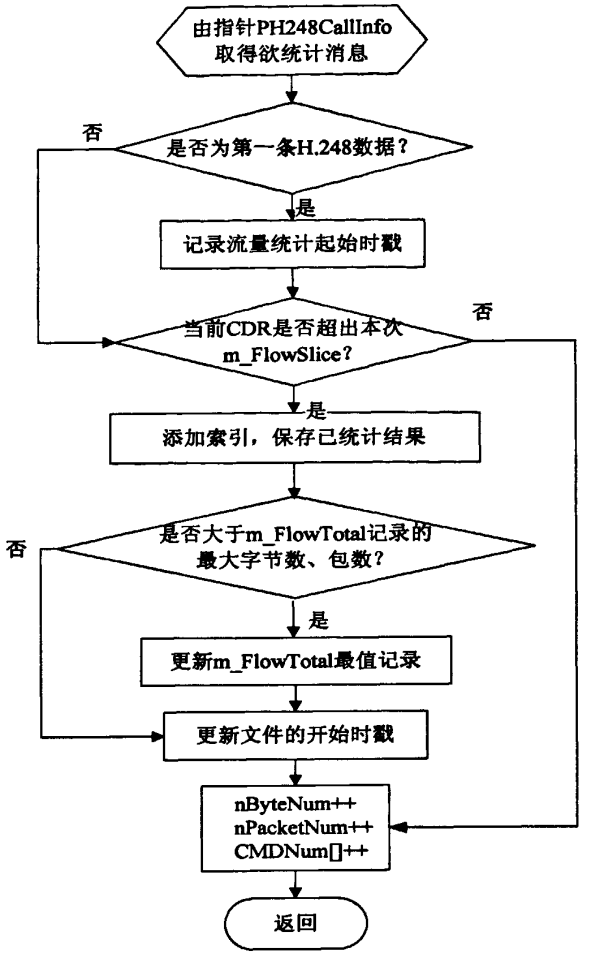

图 5.11 流量统计流程图

整个流程的思想是：以 5 分钟为单位，存储该段时间内所有 H.248 消息的字节数、IP 包数以及请求消息条数；当监测到最新消息的时间戳超过文件的结束时间，说明目前的统计文件 m_FlowSlice 已过期，需要将文件中的数据累加至总的统计结果文件 m_FlowTotal 中，并在统计索引中添加目前 m_FlowSlice 的信息；随后与 m_FlowTotal 记录的以前最值 nSliceMaxBytenum、nSliceMaxPacketnum 相比较，选出最大的重新存入 m_FlowTotal，以便形成峰值数据。

④业务相关指标统计的算法设计

与流量统计不同，其他指标统计（话务统计、呼损统计等）的统计与 CDR 的合成紧密相关，例如当 MGW 发出 ADD 请求时，表明一次呼叫（至少是试呼）开始发生。这说明业务相关指标的统计与 CDR 中呼叫状态变化有紧密关系，因此这类统计的实现被设计为在 CDR 合成分析过程中进行。在实现过程中，我们从呼叫状态的变化这一角度出发完成这一类型的统计。

在 Mc 接口的呼叫合成部分中，我们在 CH248Cdr 类中定义了一个整型的变

量来表示当前 CDR 的状态。它的可能值有 0（未发生）、1（发生，未接通）、2（接通，未应答）、3（应答中）、4（通话结束，正常释放）、5（未接通释放）、6（未应答释放）、8（出现错误码）。

以状态从 1 变化为 5 的情况为例：此时 CDR 状态从“发生，未接通”变为“未接通释放”，即用户的呼叫尝试不成功。Mc 模块统计早释次数、释放原因、无效呼叫次数等统计指标。若在监测的统计结果中，这种情况的出现的次数过多，就意味着网络可能出现了线路故障或拥塞，亦或是 MGW 承载能力达到上限等问题。

在实现中，我们在呼叫合成临时信息 CH248Info 类中维护了关于 CDR 前后状态的两个变量：m_nOldState 和 m_nOldState。根据这两个变量 Mc 模块可以由 CDR 前后状态的不同变化触发对不同指标的统计，完成相应的统计指标的保存和更新。

表 5.1 业务相关的统计指标表

状态变化	触发条件	统计项
未发生-->未接通	含拨号信息的 Notify 消息	统计试呼次数
未接通--> 未应答	含回铃音的 Mofiy 消息	统计接通次数、接通时长
未应答--> 通话中	修改 RTP 属性为“SendReceive”	统计应答次数、应答时长
通话中-->正常释放	Substract 消息	统计通话时长、释放原因
未接通-->未接通释放	未收到回铃音时收到 Sub 消息	统计早释次数、释放原因、无效呼叫次数
未接通-->未应答释放	未收到 RTP 属性修改时收到 Sub 消息	统计释放原因

表 5.1 列出了状态变化时需要进行更新的统计指标，其中仅以主叫端为例列出了这些状态变化的触发条件。

5.4.3 统计分析功能的实现

为了能够使统计分析与呼叫合成同步实现，我们将统计模块的建立、存储、操作等相关方法封装在统计类 CH248Staistic 中：

```
class CH248Statistic : public IStatistic
```

```
{
public:
    CH248Statistic();
    virtual ~CH248Statistic();

    virtual void Handle(CCDR &cdr, int nOldState, int nNewState, int nSecond);
    //H.248 的统计存储

private:
    CH248Record * m_pH248Record;

    void Save();                                    // 保存统计结果和统计索引至文件；

    void Statistic(const CH248Cdr H248Cdr, const int32 nOldState, const int32 nNewState);
    //统计函数

    void State0To1Statistic(const CH248Cdr H248Cdr, SH248StatisticItem *
    pH248StatisticItem);
    // CDR 状态从“未发生”变为“发生，未接通”时的统计

    void State1To2Statistic(const CH248Cdr H248Cdr,SH248StatisticItem *
    pH248StatisticItem);
    // CDR 状态从“发生，未接通”或“同抢，未接通”变为“接通，未应答”时的统计，

    void State2To3Statistic(const CH248Cdr H248Cdr,SH248StatisticItem *
    pH248StatisticItem);
    // CDR 状态从“接通，未应答”变为“应答通话中”时的统计

    void State3To4Statistic(const CH248Cdr H248Cdr,SH248StatisticItem *
    pH248StatisticItem);
    // CDR 状态从“应答通话中”变为“通话结束，正常释放”时的统计

    void State1To5Statistic(const CH248Cdr H248Cdr,SH248StatisticItem *
```

```
pH248StatisticItem);
// CDR 状态从“发生，未接通”或“同抢，未接通”变为“未接通释放”时的统计

void State2To6Statistic(const CH248Cdr H248Cdr,SH248StatisticItem *
pH248StatisticItem);
// CDR 状态从“接通，未应答”变为“未应答释放”时的统计

void State8To8Statistic(const CH248Cdr H248Cdr,SH248StatisticItem *
pH248StatisticItem);
//统计各种错误码产生的次数

void TimeStatistic(int32 TimeUSec, STimeStatistic & sTimeStatistic);
void SubTimeStatistic(int32 TimeUSec, STimeStatistic & sTimeStatistic);
// 接通、应答时长分段统计

void TalkTimeStatistic(const CH248Cdr H248Cdr, STalkTimeStatistic & sTalkTimeStatistic);
// 通话时长分段统计

virtual void HandleFlow(CH248CallInfo & H248Info);
void StatisticFlow(CH248CallInfo &H248Info);
//流量统计

void SaveFlow(const int32 nSkipCount);
//存储流量统计
};
```

CH248Statistic 类中封装的私有方法 Statistic()封装了统计功能的核心操作：先分别出传入 CDR 参数中的呼叫阶段属性，再根据不同的呼叫阶段对不同的统计指标进行统计和保存。该方法被封装为类 CH248Statistic 的对外接口 Handle()方法中提供给H.248合成类，在呼叫合成功能实现的同时实现各统计指标的实现。最终的统计结果和统计索引以文件的形式存储在测试仪硬盘中。

同时，为了能够是统计分析功能与呼叫合成功能同步完成，我们在上一节所述的呼叫合成功能类 CH248Analyzer 维护了包括统计接口指针、统计记录集等成员变量和统计模块的相关操作方法。通过这样的封装形式，使得统计功能的实现

与呼叫合成功能的实现能够实现了同步完成，这里不再赘述。

5.4.4 统计分析功能的测试及结果分析

Mc 接口的统计分析功能实际运行结果，如图 5-11 至图 5-14 所示：

网关地址	MSC地址	CDR类别	试呼次数	呼叫早释次数	用户接通次数	用户应答次数	呼叫结束次数	接通率(%)	应答率(%)
10.44.196.12	10.6.160.1	1	3448	3373	60	30	30	1.74	0.87
10.44.20.132	10.6.160.1	2	709	18	648	648	639	91.40	91.40
10.44.19.132	10.6.160.1	2	707	15	645	645	634	91.23	91.23
10.44.18.132	10.6.160.1	2	656	3	649	649	644	98.93	98.93
10.44.196.11	10.6.160.1	1	594	173	419	289	285	70.54	48.65
10.44.17.132	10.6.160.1	1	573	533	37	24	24	6.46	4.19
10.44.18.132	10.6.160.1	0	495	0	495	0	0	100.00	0.00
10.44.196.4	10.6.160.1	1	299	118	179	136	134	59.87	45.48
10.44.19.132	10.6.160.1	1	286	84	202	154	152	70.63	53.85
10.44.20.132	10.6.160.1	1	248	73	175	120	119	70.56	48.39
10.44.196.10	10.6.160.1	1	215	82	132	92	92	61.40	42.79
10.44.196.11	10.6.160.1	2	174	0	171	171	169	98.28	98.28
10.44.196.28	10.6.160.1	1	156	48	107	71	68	68.59	45.51
10.44.196.6	10.6.160.1	1	138	65	72	51	51	52.17	36.96
10.44.10.132	10.6.160.1	2	126	0	120	120	120	95.24	95.24
10.44.194.132	10.6.160.1	1	120	72	47	32	31	39.17	26.67
10.44.196.4	10.6.160.1	2	113	0	107	107	107	94.69	94.69
10.44.196.13	10.6.160.1	1	103	32	71	49	45	68.93	47.57
10.44.194.4	10.6.160.1	1	102	37	65	43	42	63.73	42.16
10.44.136.18	10.6.160.1	1	100	40	60	39	39	60.00	39.00
10.44.19.132	10.6.160.1	0	99	0	34	0	0	34.34	0.00
10.44.20.132	10.6.160.1	0	92	0	31	0	0	33.70	0.00
10.44.196.15	10.6.160.1	1	91	29	62	45	43	68.13	49.45
10.44.196.30	10.6.160.1	1	72	46	26	20	19	36.11	27.78

图 5.12 统计分析功能运行结果（呼叫统计）

网关地址	MSC地址	CDR类别	总接通时长(毫...	总应答时长(毫...	总通话时长(毫...	10分钟<=通话...	5分钟<=通话...	3分钟<=通话...	1分钟<=通话...	通话时长<1分钟
10.44.196.11	10.6.160.1	1	1474405	3685644	16404946	0	6	8	74	197
10.44.196.4	10.6.160.1	1	1010075	2137452	17471904	3	12	12	43	64
10.44.196.10	10.6.160.1	1	596151	1436129	9365834	1	1	7	32	51
10.44.196.28	10.6.160.1	1	570452	1132903	6976195	1	5	3	22	37
10.44.196.15	10.6.160.1	1	291919	681318	3757021	1	0	3	15	24
10.44.196.6	10.6.160.1	1	285458	601356	3609445	1	1	2	15	32
10.44.196.13	10.6.160.1	1	278502	666342	3994051	1	2	1	9	32
10.44.136.18	10.6.160.1	1	267082	545869	2967016	1	0	2	9	27
10.44.194.4	10.6.160.1	1	258724	629340	7503441	1	8	6	11	16
10.44.196.16	10.6.160.1	1	237820	467662	5705992	2	4	1	10	14
10.44.196.12	10.6.160.1	1	166741	348542	2974749	1	0	1	10	18
10.44.194.132	10.6.160.1	1	166112	402189	3721402	1	1	3	11	15
10.44.196.30	10.6.160.1	1	129045	244705	474722	0	0	0	2	17
10.44.18.132	10.6.160.1	2	127214	127214	19900922	4	10	8	31	591
10.44.196.20	10.6.160.1	1	113327	165731	1089138	0	1	1	2	7
10.44.19.132	10.6.160.1	2	106557	106557	42757868	4	17	28	149	436
10.44.20.132	10.6.160.1	2	106037	106037	48107351	9	15	21	162	432
10.44.136.17	10.6.160.1	1	87924	127074	993946	0	0	2	4	5
10.44.196.7	10.6.160.1	1	63343	159677	2043667	1	0	2	3	5
10.44.19.132	10.6.160.1	1	56996	1330392	11723861	1	6	6	45	94
10.44.136.132	10.6.160.1	1	47807	73713	311602	0	0	0	2	5
10.44.196.8	10.6.160.1	1	47695	79338	672219	0	0	0	6	0
10.44.20.132	10.6.160.1	1	46768	1075841	8182637	0	3	5	34	77
10.44.194.6	10.6.160.1	1	44911	65637	2190225	1	2	0	1	2

图 5.13 统计分析功能运行结果（话务统计）

网关地址	MSC地址	CDR类别	最长接通时长(...	最短接通时长(...	最长应答时长(...	最短应答时长(...	最长通话时长(...	最短通话时长(...
10.44.196.13	10.6.160.1	1	19055	615	57758	3697	1136735	505
10.44.196.11	10.6.160.1	1	17546	247	38620	1694	501714	300
10.44.196.28	10.6.160.1	1	17066	268	47799	5031	742421	3501
10.44.196.15	10.6.160.1	1	16998	399	36397	399	671421	4801
10.44.196.20	10.6.160.1	1	16876	1104	24465	7697	418712	2501
10.44.196.10	10.6.160.1	1	16647	382	48806	382	2403972	4018
10.44.196.12	10.6.160.1	1	16635	198	30425	375	1524543	200
10.44.196.4	10.6.160.1	1	16508	696	51426	1185	1719247	200
10.44.196.16	10.6.160.1	1	15909	769	53750	6835	1561448	11802
10.44.196.17	10.6.160.1	1	14288	780	16158	8199	16875	11556
10.44.136.17	10.6.160.1	1	13593	836	18896	1195	286908	11408
10.44.194.4	10.6.160.1	1	13536	302	33196	5942	1172210	3500
10.44.136.132	10.6.160.1	1	13505	398	18859	398	121504	3899
10.44.196.30	10.6.160.1	1	13097	397	49093	397	105006	5600
10.44.196.6	10.6.160.1	1	10869	285	31452	399	617723	698
10.44.136.18	10.6.160.1	1	10321	284	32396	5328	735223	10001
10.44.196.8	10.6.160.1	1	9395	696	27197	2796	166305	72701
10.44.194.132	10.6.160.1	1	9392	296	27953	406	1227937	1596
10.44.8.1	10.6.160.1	1	8327	1388	44090	6407	90503	16788
10.44.196.7	10.6.160.1	1	7876	399	28917	399	1142936	4800
10.44.18.132	10.6.160.1	1	7703	171	34502	455	721519	630
10.44.194.6	10.6.160.1	1	7604	420	14697	6117	1326117	10999
10.44.194.132	10.6.160.1	2	7592	95	7592	95	623100	2663
10.44.196.22	10.6.160.1	1	7122	6497	19497	17122	80403	24501

图 5.14 统计分析功能运行结果（性能统计）

网关地址	MSC地址	CDR类别	语法错误	协议错误	未授权	事务请求语法...	协议版本不支持	标识符错误	事务指向未知...	无可用的关联...
10.44.196.12	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.196.12	10.6.160.1	2	0	0	0	0	0	0	0	0
10.44.194.8	10.6.160.1	0	0	0	0	0	0	0	0	0
10.44.194.8	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.194.8	10.6.160.1	2	0	0	0	0	0	0	0	0
10.44.17.132	10.6.160.1	0	0	0	0	0	0	0	0	0
10.44.17.132	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.196.29	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.196.4	10.6.160.1	0	0	0	0	0	0	0	0	0
10.44.196.4	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.196.4	10.6.160.1	2	0	0	0	0	0	0	0	0
10.44.196.11	10.6.160.1	0	0	0	0	0	0	0	0	0
10.44.196.11	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.196.11	10.6.160.1	2	0	0	0	0	0	0	0	0
10.44.8.1	10.6.160.1	0	0	0	0	0	0	0	0	0
10.44.8.1	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.8.1	10.6.160.1	2	0	0	0	0	0	0	0	0
10.44.196.6	10.6.160.1	0	0	0	0	0	0	0	0	0
10.44.196.6	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.196.6	10.6.160.1	2	0	0	0	0	0	0	0	0
10.44.196.13	10.6.160.1	0	0	0	0	0	0	0	0	0
10.44.196.13	10.6.160.1	1	0	0	0	0	0	0	0	0
10.44.196.13	10.6.160.1	2	0	0	0	0	0	0	0	0
[illegible]	[illegible]	[illegible]	0	0	0	0	0	0	0	0

图 5.15 统计分析功能运行结果（呼损统计）

在结果中可以看到：演示数据采集自 IP 地址为 10.44.196.30（MGW）和 10.6.160.1（MSC Server）之间的 Mc 接口；统计结果包括了呼叫统计、话务统计、性能统计、呼损统计分项的统计，统计结果清晰准确，达到了期望。

Mc 接口监测模块的流量统计功能实际运行结果如下：

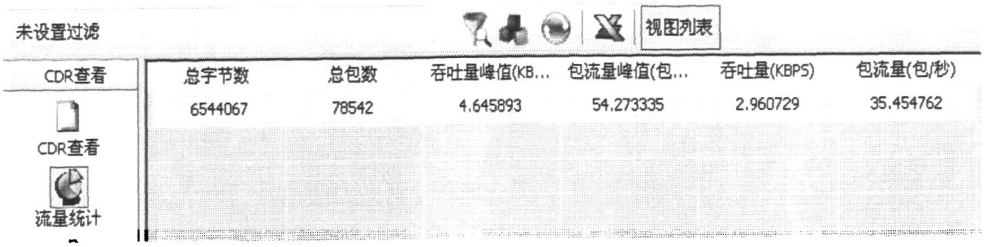

图 5.16 流量统计运行结果

在结果中可以看到：该Mc接口上总字节数、总包数、吞吐量/包流量峰值、吞吐量、包流量等指标均得到了有效的统计，符合设计要求。

5.5 本章小结

本章首先对Mc接口监测模块的呼叫合成功能进行了需求分析；接着在参考大量现网数据的基础上，对Mc接口上的常见呼叫流程进行了分类，以此为依据提出了Mc接口呼叫合成的关键值并设计了Mc接口的呼叫合成方案；然后根据在本章开头所作的需求分析提出了以hash算法和文件形式来组织CDR合成的实现方法，解决了Mc模块呼叫合成功能实现的难点；最后设计了呼叫合成功能的类封装结构，实现了Mc模块的呼叫合成功能。

本章的第二部分是对Mc接口统计分析的设计与实现。在该部分中，文章提出了统计分析功能与呼叫合成功能同步实现的设计思想，并据此设计了统计分析功能的关键值类型和类封装结构，指定了Mc接口监测的统计指标；最终将统计分析功能的实现内嵌在呼叫合成功能的代码中，解决了高实时性和大数据量对测试仪统计分析功能提出的要求。

第六章 结论与展望

6.1 全文总结

本论文的研究课题是基于WCDMA网络测试仪研发项目。本论文从理论和实践两方面入手，对WCDMA网络结构、核心网电路域的Mc接口、Mc接口上的H.248协议标准、协议编码方式、呼叫建立流程等进行了较为深入的研究和探索。接着从Mc接口监测模块在整个测试仪系统中的从属和功能出发，对模块的协议解码、呼叫合成与统计分析三项重要功能的设计思想进行了详细的阐述，最终提出并实现了三项功能的设计方案。

本论文所研究和开发的Mc接口监测模块已应用于WCDMA网络测试仪中。通过实验网与现网测试，达到了商用水平。

6.2 下一阶段的工作

随着软交换技术与WCDMA网络及技术的发展，会对Mc接口的监测技术提出更高的要求，从应用角度出发本课题仍然有需要进一步研究和实践的地方。

1) 由于Mc接口上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) 随着协议标准的不断升级，Mc监测模块也需要进行相应的升级和扩展；
3) 随着网络业务的不断扩充可能会引起现有业务流程的改变，同时新业务的业务流程可能给现有的呼叫合成方案提出新的要求。因此对模块的呼叫合成和统计分析功能可能会有相应的添加和修改；
4) 随着现网的不断完善和成形，与Mc接口上的业务相关的接口及所使用的协议标准也在不断成形，本设计的下一步工作将主要放在Mc接口业务的多段关联功能实现上。

致 谢

值此论文结束之际，我特向所有给予我帮助和关心的人致以真诚的感谢！

首先，我要衷心感谢我的导师程方副教授。在三年的研究生期间，程老师一丝不苟的工作作风和宽以待人的为人之道以及她精湛渊博的学术修养与严谨细致的治学态度，对我产生了深刻的影响，并永远是我日后工作和学习的榜样。在课题阶段中，程老师以她在该领域的丰富经验和严谨的学术态度及时地对我进行指导，使得课题最终能够顺利并圆满完成。本论文的工作就是在程老师悉心指导和严格要求下完成的。从选题到论文的撰写和修改，都倾注了老师的心血。在此向她致以衷心的感谢！

其次，我要感谢实验室和项目组其他老师、及同学的帮助和支持。在研究生学习的三年中，我与他们结下了真诚的友谊，在良好的科研氛围中共同完成了研究生阶段的学习。感谢苟由柯师兄，他的言传身教使我在软件编程能力上的下了扎实的基础；感谢汪世龙师兄对我在完成课题研究与项目实现过程中的无私帮助。衷心地感谢以上提及的所有师长与朋友，课题的顺利完成离不开他们的关心和帮助。

此外，我感谢父母和家人对我的关心和帮助，是他们一如既往的支持和关怀使我最终顺利完成了研究生的学习，谨以此文表达我对他们的谢意。

最后，我还要感谢所有对本论文提出宝贵意见的老师和同学，以及将评审该论文的老师、教授。

时间飞逝，充实的三年研究生学习时光即将离我远去，我很怀念它。

参考文献

[1] 张平等.WCDMA移动通信系统(第2版)[M].北京.人民邮电出版社.2004:pp11-14

[2] T1.722-2002. UMTS References-3G Specifications (Release 99, Release 4, & GTT)[S]. 2002:pp18-25

[3] YD/T 1593-2007. 2GHz TD-SCDMA/WCDMA 数字蜂窝移动通信网移动软交换服务器与媒体网关间的Mc接口技术要求(第二阶段) [S].中华人民共和国通信行业标准.2007:pp6-8

[4] ITU-T Recommendation H.248.4 (2000) Corr.1 (2004), Gateway control protocol: Transport over Stream Control Transmission Protocol (SCTP)[S]:pp5-7

[5] ITU-T Recommendation H.248.5 (2000), Gateway control protocol: Transport over ATM[S]:pp7-14

[6] YD/T 1594-2007. 2GHz TD-SCDMA/WCDMA 数字蜂窝移动通信网移动软交换服务器与媒体网关间的 Mc 接口测试方法[S].中华人民共和国通信行业标准.2007

[7] ITU-T Recommendation H.248.1. Gateway control protocol[S]. 2005

[8] YD/T 1292-2003. 基于 H.248 的媒体网关控制协议技术要求[S]. 中华人民共和国通信行业标准. 2007:pp4-7

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

[10] ITU-T Recommendation X.690 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]. 2002

[11] IETF RFC 2234. Augmented BNF for Syntax Specifications: ABNF[S]. 1997

[12] 3GPP TS 29.232. Media Gateway Controller (MGC) – Media Gateway (MGW) interface：Stage 3[S]

[13] ITU-T H.248.30 Gateway control protocol: RTCP extended performance metrics packages[S]

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

[15] 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]. 2004

[16] 邵维忠, 杨芙清. 面向对象的系统设计(第2版)[M].北京.清华大学出版社.2007

[17] GB16263.1-2006-T.信息技术 ASN.1 编码规则第 1 部分基本编码规则(BER)[S].2006:pp3-16

[18] IETF RFC 2885, Megaco Protocol version 0.8 [S]. 2000

[19] Jeffrey E.F.Friedl, 余晟译.Mastering Regular Expressions（第 3 版）[M]. 北京：电子工业出版社. 2007:pp24-28

[20] 廖凯，雒江涛，张治中. 利用正则表达式实现 Megaco 协议解析[J]. 通信技术. 2008.11vol41:pp81-87

[21] IETF RFC 3525. Gateway Control Protocol Version 1 [S]. 2003

[22] 徐红军, 糜正琨. H.248 协议的实现及其应用[J]. 数据通信. 2005.4: pp42-45

[23] 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:pp36-38

[24] 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:pp15-19

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

[26] Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest, Clifford Stein, Introduction to Algorithms(Second Edition), 北京. 机械工业出版社. 2006.9:pp132-146

[27] IETF RFC 2327. SDP: Session Description Protocol.[S]. 1998

附 录

1. 攻读硕士期间参与的主要科研项目

[1] TD-SCDMA、WCDMA 移动通信网络信令测试技术的研究与开发（重庆市科技攻关 2005AC2040，12 万)子课题:WCDMA 网络测试仪的研究与开发。本人负责：Mc 接口监测模块的研发工作，主研。（2008.7-）

[2] 重庆市科委重点攻关项目(IPTV 测试仪的开发及产业化).“CYDD-600 IPTV 测试仪”. 2008.9-。本人负责：参与项目的前期调研和需求分析。（2008.4-2008.7）

2. 发表的论文

[1] 第一作者，NGN 网络测试仪 H.248 解码模块的设计与实现,《电子测试》，2010 年 03 期，pp：62-66。

[2] 第一作者，IPTV 业务测试综述,《黑龙江科技信息》，2009 年 26 期，pp：89。