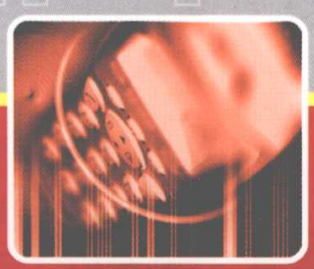
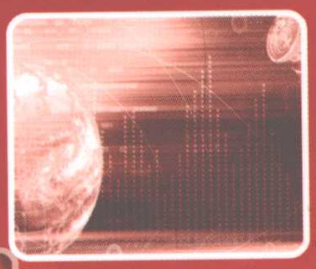
电子通信新技术丛书

软交换与SIP实用技术

徐培文 谢水珍 杨从保 编著



机械工业出版社
CHINA MACHINE PRESS



ISBN 978-7-111-21371-0

上架指导：工业技术／电子技术／通信技术

ISBN 978-7-111-21371-0
9 787111 213710 >

地址：北京市百万庄大街22号　邮政编码：100037
联系电话：(010) 68326294　网址：http://www.cmpbook.com
(010) 68993821　E-mail:online@cmpbook.com

定价：36.00元

电子通信新技术丛书

软交换与 SIP 实用技术

徐培文　谢水珍　杨从保　编著

机 械 工 业 出 版 社

TN91/67

本书从整体出发，全面讲述了软交换的各种技术，包括下一代网络技术概况、软交换的系统框架以及软交换中的媒体网关与信令网关技术；详细介绍了软交换所支持的各种协议，如 H.323、MGCP 和 H.248 协议；并在此基础上详细论述了软交换的各种实用技术，如软交换管理技术、计费技术、服务质量和业务提供技术。同时本书还结合实际应用对软交换的核心协议 SIP 进行了详细的介绍，内容包括协议架构、SIP 应用和安全以及开发设计过程。

本书内容详尽，结合理论和实际开发经验，适合于通信专业的工程技术人员，尤其是可以适合作为从事下一代网络开发和 SIP 开发的专业技术人员、管理人员作为工作和实际开发的参考书。同时，本书也可以作为高等院校信息与通信等专业的高年级本科生或研究生的参考书。

图书在版编目（CIP）数据

软交换与 SIP 实用技术/徐培文，谢水珍，杨从保编著. —北京：机械工业出版社，2007.4

（电子通信新技术丛书）

ISBN 978-7-111-21371-0

Ⅰ. 软… Ⅱ. ①徐…②谢…③杨… Ⅲ. ①通信交换－通信网②计算机网络－通信协议 Ⅳ. TN915.0

中国版本图书馆 CIP 数据核字（2007）第 058535 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：张俊红 责任编辑：朱 林 版式设计：冉晓华

责任校对：张 媛 封面设计：陈 沛 责任印制：杨 曦

北京机工印刷厂印刷

2007 年 6 月第 1 版第 1 次印刷

169mm×239mm · 12.625 印张 · 491 千字

0 001—4 000 册

标准书号：ISBN 978-7-111-21371-0

定价：36.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

销售服务热线电话：（010）68326294

购书热线电话：（010）88379639 88379641 88379643

编辑热线电话：（010）88379768

封面无防伪标均为盗版

丛 书 序

当今，经济全球化和网络化已成为一种潮流，电子信息产业在全球范围内的结构调整步伐加快，发达国家制造能力加速向发展中国家转移；与此同时，全球电信业转型步伐加快，技术、网络、业务融合的趋势更加明显，跨国公司纷纷创新发展模式，向更广的信息服务领域拓展。作为发展中国家，我国必须在社会发展过程中更加积极地推进信息化，应用先进的电子信息技术，提高各行业信息化水平，进一步提高我国在全球竞争中的综合国力。

改革开放几十年来，我国信息产业以年均25%的速度递增，领先于其他产业的发展，成为我国国民经济的第一支柱产业，在促进国民经济增长、促进经济增长方式转变、促进经济社会协调发展、促进先进文化传播、保障网络与信息安全等方面有着重要的地位和作用。以2006年1~9月份为例，全国通信业务总量完成11131.1亿元，比去年同期增长25.1%。其中，电信业务总量为10589.2亿元，增长25.5%；邮政业务总量为541.9亿元，增长17.9%。全国通信业务收入完成5278.5亿元，比去年同期增长11.7%。其中，电信业务收入4799.3亿元，增长11.3%；邮政业务收入479.1亿元，增长16.2%。对于电子信息制造业，2006年1~9月，我国规模以上电子信息产业实现销售收入29988亿元，同比增长25.4%，其中制造业实现销售收入27311亿元，同比增长25.6%；软件产业实现收入2677亿元，同比增长23.8%。制造业实现工业增加值为5740亿元，同比增长26.7%；利税总额为1231亿元，同比增长27.7%；出口交货值为16314亿元，同比增长26.8%。在当前阶段，全面建设小康社会的发展战略的实施，社会主义市场经济体制的不断完善，为信息产业发展创造了良好的条件；而信息产业经过多年的发展，整体水平不断提高，这使其进一步发展成为可能。

信息产业作为我国重点发展的高科技产业，对实现以信息化带动工业化，全面推动我国经济的跨越式发展起着重要作用，其技术创新能力在相当程度上将决定未来我国的综合竞争实力。与此同时，我们应该清醒地认识到，电子信息产业是一个受技术影响较大的产业。对于我国的电子信息产业，目前面临着诸多问题：制造业产业结构不尽合理，软件产业比重偏低，集成电路产业整体水平还有较大差距，基础电子产品发展相对滞后；自主创新能力差，核心技术受制于人，产品和应用技术水平不高；运营业务创新能力还不强，业务结构有待优化，网络资源利用率偏低；城乡通信发展不够协调，农村通信发展相对滞

后，普遍服务压力较大；队伍整体素质还不高，高层次、复合型人才严重缺乏，还不能满足行业发展的需要等问题和矛盾。然而，电子信息技术涵盖的面非常广，包括计算机通信及网络、图像通信、数字程控交换、移动通信与无线通信技术、数字信号处理技术、电磁场与微波、智能仪器及系统设计、仪器仪表等多个领域。层出不穷的各种电子信息新技术，如3G、WiMAX、RFID、无线传感器、FMC、IMS、软交换、NGN、NGI等，一方面给我们带来了新的机遇，另一方面也使得我们面临的问题和矛盾更加激化。若不能够及时跟上技术发展的步伐，则必然会被全球产业发展所淘汰；然而过度地追求新技术，同时会因为资金、技术、人才、政策等问题而陷入困境。

为此，我们策划出版这套电子通信新技术丛书，力求从实际应用出发，以新的观点、新的视角来向大家介绍电子信息技术的理论前沿、应用前景和最新发展动态，以及电子信息产业发展状况，为工程技术人员、科研开发人员和电子信息相关专业在校学生提供参考和帮助，为我国电子信息产业的发展做出自己的一份努力。

丛书编委会

前　言

什么是下一代网络？对下一代网络，目前业界有不同的理解，对通信网从交换的角度来讲，软交换就是下一代网络；对互联网来讲，IPv6 就是下一代网络；对传输网来讲，光传输网就是下一代网络。本书着重从通信网的角度来分析下一代网络，并从各个层面进行详细探讨。

从目前的通信网发展来看，IP 网通信业务量的爆炸式增长已成为世界瞩目的焦点。作为下一代网络（NGN）的核心，软交换技术近年来得到了广泛关注，其目的是以分组为基础建设下一代公众网络，实现从 PSTN 到 IP 网的平滑过渡。利用 IP 网的优势节约投资和运营成本，开发新业务，在继承的基础上实现目前各个业务网络之间的互通。它将在业务融合、用户终端控制、第三方应用集成中起重要作用。下一代网络是业务驱动的网络，通过业务与呼叫控制分离以及呼叫控制与承载分离实现相对独立的业务体系，使业务真正独立于网络，灵活有效地实现业务的提供。用户可以自行配置和定义自己的业务特征，不必关心承载业务的网络形式及终端类型，使得业务和应用的提供有较大的灵活性，从而满足用户不断发展、不断更新的业务需求。软交换通过优化网络结构实现了网络和业务的融合，采用了开放式应用编程接口，允许在交换机制中灵活地引入新业务。

下一代网络将是以 IP 为代表的分组化网络，这已成为业界的共识。然而，分组化网络还存在不少问题，而且从传统的电路交换网络向分组化网络的演进将是一个长期过程，因此在未来很长一段时期内，电信运营商将不得不同时支持两种网络，解决两网之间的互通以及各自业务和应用之间的互操作性，从而最终完成平滑过渡，软交换将是完成这一过渡任务的关键。

近年来，以软交换为核心的下一代网络成为电信产业的焦点，我国信息产业部对软交换的定义是："软交换是网络演进以及下一代分组网络的核心设备之一，它独立于传送网络，主要完成呼叫控制、资源分配、协议处理、路由、认证、计费等主要功能，同时可以向用户提供现有电路交换机所能提供的所有业务，并向第三方提供可编程能力。"目前，开发下一代可持续发展的网络来支持语音业务和变得日益重要的数据以及多媒体业务，已经成为众多电信运营商的战略目标。从发展的角度来看，NGN 是从传统的以电路交换为主逐渐迈向以分组交换为主，它承载了原有 PSTN 的所有业务，把大量的数据传输卸载到 IP/ATM 网络中，以减轻 PSTN 的重荷，形成开放的、融合的、统一的网络平台。

因此，NGN 是基于 TDM 的 PSTN 语音网络和基于 IP/ATM 的分组网络融合的产物，它使得在新一代网络上进行语音、视频、数据等综合业务成为了可能。这样不仅能降低网络成本，而且能派生出许多新型的、集成的业务，为运营商创造了新的利润增长点，使网络向着信息传送更加高效、业务生成更加灵活的方向发展。

软交换就是在这样的需求背景下产生的，它是下一代网络的核心技术。本书紧紧围绕软交换技术，全面介绍了下一代网络和软交换技术概况，着重讲述其结构及特点，同时提出了软交换技术和 IPv6 的融合思想。紧接着对软交换技术背景进行了介绍，阐述了软交换的框架及其核心技术，以及支持的网关设备和协议，并特别详细地介绍了软交换管理系统的关键技术、计费管理、服务质量。

SIP 凭借其简单、易于扩展、便于实现等诸多优点越来越得到业界的青睐，它正逐步成为 NGN 和 3G（3rd Generation）IP 多媒体子系统（IP Mutimedia Subsystem，IMS）域中的重要协议，并且市场上出现越来越多的支持 SIP 的客户端软件和智能多媒体终端，以及用 SIP 实现的服务器和软交换设备。SIP 在软交换中是一个重要的协议，因此本书着重详细介绍了 SIP 标准、SIP 的安全机制，同时介绍了 SIP 在软交换和 IMS 中的应用，以及 SIP 代理服务器的开发应用举例，以供读者参考。

本书共分 13 章。第 1 章为下一代网络软交换的概述，着重讲述下一代网络技术概况、结构及特点，软交换的定义及其核心技术，同时介绍了 IPv6 技术，提出了软交换技术和 IPv6 的融合思想。第 2 章为软交换中的网关技术，包括信令网关和媒体网关技术，主要讲述信令网关的功能以及软交换和 No.7 信令网互通功能。第 3 章为 H.323 协议，第 4 章为 MGCP 和 H.248/Megaco 协议，至此就讲述了软交换支持的主要呼叫协议。第 5 章为软交换的管理技术，主要是将电信管理网运用于软交换系统中，同时详细讲述了在软交换管理体系中可以使用的 CORBA、JMX 实现技术，另外还提出了和 NGOSS 融合的思想。第 6 章为软交换的计费管理，详细介绍了软交换网络的计费管理理论和方法。第 7 章为软交换的服务质量，主要将 IP 网的 QoS 理论运用于软交换技术，以提高软交换的运行质量。第 8 章为软交换的业务提供技术，主要讲述了 PARLAY 和 JAIN 技术。在本书第 9 章软交换实践中，分析了软交换部署中容易出现的各种实际问题，如私网穿越，软交换接入、安全、性能和组网等相关问题。第 10 章主要介绍了 SIP 标准，包括 SIP 框架、SIP 消息结构及消息类型、SIP 用户代理和 SIP 代理服务器的行为等相关内容。第 11 章为 SIP 的安全机制，主要分析了 SIP 存在的安全隐患，以及响应的解决方法和响应的安全机制。第 12 章为 SIP 的应用，主要介绍了 SIP 在软交换和 IMS（IP 多媒体子系统）中的应用。在第 13 章的 SIP 应

用开发中，结合工程实践，介绍如何选择相应的开源协议栈来进行应用开发，另外，本章还举例介绍了 PartySIP 的应用开发分析。

本书在写作过程中得到北京邮电大学马严教授、王志谦高级工程师，中国人民公安大学安全防范系的洪卫军教授、李锦涛教授、卜凡亮教授、杜治国高工、王志军高工等老师的帮助，在 SIP 在 IMS 中的应用部分，得到西门子的朱丽、俞小良高级工程师的帮助，在 SIP 的应用开发部分得到王洋、吴明豪、郭薇等同学的帮助，同时本书还得到了蒋亮、王鲲鹏、姜雪松、张俊红、夏珂、廖林生、魏薇、王树峰、余颖、陈磊的帮助，在此表示感谢。

由于编写时间有限，加上技术本身的快速发展，书中难免存在错误或不足之处，敬请广大读者批评指正，来信请发往 xupw@ bupt. edu. cn。

作者

目　录

丛书序

前言

第 1 章　下一代网络技术与软交换技术的介绍 …… 1

1.1　下一代网络简述 …… 1

1.1.1　下一代网络概述 …… 2

1.1.2　下一代网络的分层结构 …… 3

1.2　软交换技术 …… 5

1.2.1　软交换技术概况以及同 IMS 的关系 …… 5

1.2.2　引入软交换的意义 …… 6

1.2.3　软交换支持的主要协议 …… 6

1.2.4　软交换实现的主要功能 …… 7

第 2 章　软交换中的网关技术 …… 12

2.1　网关控制协议的特征 …… 12

2.2　媒体网关技术 …… 14

2.2.1　媒体网关及其功能 …… 15

2.2.2　RTP …… 17

2.2.3　RTCP 概述 …… 19

2.3　信令网关技术 …… 26

2.3.1　No. 7 信令系统 …… 26

2.3.2　SCCP 简介 …… 30

2.3.3　信令网关概述 …… 31

2.3.4　SIG TRAN 协议介绍 …… 32

2.3.5　SCTP …… 35

2.3.6　MTP3 协议 …… 43

2.3.7　M2PA 协议 …… 47

第 3 章　H. 323 协议 …… 50

3.1　H. 323 协议族体系 …… 50

3.2　H. 323 系统结构介绍 …… 51

3.3　RAS 协议 …… 59

3.4　呼叫信令信道 …… 60

3.4.1　H. 323 通信控制的一般过程 …… 60

3.4.2　呼叫信令信道选路 …… 60

3.4.3　媒体控制信道路由 …… 61

3.5　呼叫信令流程 …… 62

3.5.1　呼叫建立 …… 62

3.5.2　端点均无注册 …… 63

3.5.3　两个端点都注册于同一网守 …… 63

3.5.4　只有主叫端点在网守注册 …… 64

3.5.5　只有被叫端点在网守注册 …… 65

3.5.6　两个端点注册在不同的网守 …… 67

3.5.7　可选的被叫端点信令 …… 71

3.5.8　快速连接流程 …… 72

3.6　通信初始化和能力交换 …… 72

3.7　音频、视频通信的建立 …… 72

3.8　呼叫服务 …… 73

3.9　H. 245 协议 …… 74

3.10　呼叫结束 …… 76

第 4 章　MGCP 和 H. 248 协议 …… 78

4.1　MGCP 的简介和功能 …… 78

4.2　MGCP 的结构 …… 79

4.2.1　MGCP 的端点 …… 81

4.2.2　MGCP 的连接建立过程 …… 84

4.2.3　MGCP 的命令种类 …… 84

4.2.4 返回码和出错码 …… 85
4.2.5 原因代码 …… 86
4.2.6 MGCP 的消息格式 …… 87
4.2.7 UDP 上的传输 …… 88
4.2.8 状态失效修复能力和竞态情况 …… 89
4.3 Megaco/H.248 协议的简介和功能 …… 90
4.3.1 Megaco/H.248 简介 …… 90
4.3.2 Megaco 的产生 …… 90
4.3.3 Megaco 的网络结构 …… 90
4.3.4 Megaco 和其他协议的关系 …… 91
4.4 H.248 协议的主要内容和流程 …… 92
4.4.1 Megaco 的连接模型 …… 92
4.4.2 Megaco 的描述符 …… 93
4.4.3 Megaco 的命令 …… 100
4.4.4 Megaco 中的事务 …… 102
4.4.5 Megaco 的传输 …… 104
4.4.6 Megaco 的安全考虑 …… 104
4.4.7 Megaco 的包定义 …… 105
4.5 软交换中的 Megaco …… 106
4.6 MGCP 和 H.248 协议的比较 … 108
4.6.1 MGCP 与 Megaco 的相同和相似之处 …… 108
4.6.2 MGCP 与 Megaco 的区别 …… 108

第 5 章 软交换的管理技术 …… 112

5.1 TMN 概述 …… 112
5.1.1 TMN 的管理功能 …… 112
5.1.2 网络管理的新要求 …… 114
5.1.3 TMN 的技术基础 …… 114
5.1.4 TMN 的实用技术 …… 114
5.2 软交换的网络管理 …… 117
5.2.1 软交换的基本管理 …… 117
5.2.2 软交换的基本网络管理功能 …… 119
5.3 软交换的网络管理技术实现 … 120
5.3.1 CORBA 技术 …… 121
5.3.2 JMX 技术 …… 126
5.4 软交换的管理技术与下一代运营支撑系统的融合 …… 131
5.4.1 电信业务运营支撑系统介绍及带给软交换技术的挑战 …… 131
5.4.2 业务运营支撑系统的发展与组成 …… 133
5.4.3 NGOSS 概述 …… 134

第 6 章 软交换的计费管理 …… 139

6.1 计费系统概述 …… 139
6.2 传统计费系统原理 …… 140
6.3 软交换的计费系统 …… 143
6.3.1 软交换的计费系统体系结构 …… 144
6.3.2 软交换的计费协议 …… 145
6.3.3 软交换的计费策略 …… 146
6.3.4 软交换的计费方案模型 … 147
6.3.5 软交换的计费方案 …… 148
6.4 软交换对计费的基本要求 …… 148
6.4.1 计费协议 …… 148
6.4.2 计费方式 …… 148
6.4.3 计费对象 …… 148
6.4.4 计费精度要求 …… 148
6.4.5 分组语音计费内容 …… 149
6.4.6 计费系统需要解决的问题 …… 149
6.4.7 软交换计费基本内容 …… 150
6.5 软交换计费系统的特点及要求 …… 150
6.6 软交换计费系统的数据采集 … 152
6.7 软交换计费系统的计费方案 … 152
6.8 软交换计费的相关协议和规范 …… 153

6.8.1 AAA 服务器的应用 …… 153
6.8.2 RADIUS 协议 …… 155
6.8.3 Diameter 协议 …… 161
6.8.4 AAA 技术的发展 …… 167
6.9 软交换计费系统与外部系统的接口 …… 168
6.10 实际系统中关于计费的例子 …… 169

第 7 章 软交换的服务质量 …… 171

7.1 QoS 的背景和定义 …… 171
7.1.1 QoS 的背景 …… 171
7.1.2 QoS 的定义 …… 171
7.1.3 IP QoS 原理 …… 173
7.1.4 QoS 技术分类 …… 176
7.1.5 综合服务模型 …… 178
7.1.6 区分服务模型 …… 182
7.1.7 两种模型的比较 …… 185
7.2 软交换的 QoS 分析 …… 186

第 8 章 软交换的业务提供技术 …… 190

8.1 软交换的业务综述 …… 190
8.2 Parlay 技术 …… 191
8.2.1 Parlay 组织的组成和现状 …… 192
8.2.2 Parlay 规范定义的商业角色和商业模型 …… 192
8.2.3 Parlay3 技术 …… 193
8.2.4 Parlay X 协议综述 …… 193
8.2.5 Parlay X 提供的第三方呼叫接口 …… 194

第 9 章 软交换的实践 …… 198

9.1 软交换的控制功能 …… 198
9.1.1 呼叫模型的设计要求 …… 198
9.1.2 软交换的呼叫模型 …… 199
9.1.3 BCSM 呼叫控制模型 …… 200
9.2 软交换的私网穿越 …… 204
9.2.1 软交换部署时易出现的问题 …… 204
9.2.2 软交换私网穿越的解决方案 …… 204
9.3 软交换的接入 …… 207
9.3.1 媒体接入层的组网研究 … 207
9.3.2 媒体接入层的接入方法 … 207
9.3.3 现阶段的接入方式 …… 208
9.4 软交换的安全问题 …… 209
9.5 安全解决方案 …… 209
9.5.1 网络设备安全防线 …… 210
9.5.2 网络安全防线 …… 210
9.5.3 接入安全防线 …… 211
9.6 软交换的组网形式 …… 211
9.7 软交换设备的性能要求 …… 213

第 10 章 SIP …… 214

10.1 SIP 介绍及协议框架 …… 214
10.1.1 SIP 概述 …… 214
10.1.2 SIP 中的术语 …… 216
10.1.3 SIP 在网络体系中的位置 …… 216
10.2 一个 SIP 通信的实例 …… 219
10.3 SIP 消息结构及基本消息体 …… 224
10.3.1 请求消息 …… 224
10.3.2 应答消息 …… 227
10.3.3 头字段 …… 235
10.3.4 消息体 …… 251
10.4 用户代理的行为 …… 252
10.4.1 用户代理客户端 …… 253
10.4.2 用户代理服务器端 …… 264
10.5 SIP 代理服务器 …… 269
10.5.1 概述 …… 269
10.5.2 有状态代理服务器 …… 270
10.5.3 确认请求的有效性 …… 271
10.5.4 路由信息预处理 …… 273

10.5.5 确定请求的发送目的地 …… 273
10.5.6 请求转发 …… 275
10.5.7 应答处理 …… 280
10.5.8 异常处理 …… 285
10.5.9 无状态代理服务器 …… 286
10.5.10 代理服务器路由处理小结 …… 288
10.6 SIP 事务 …… 291
10.6.1 客户端事务 …… 292
10.6.2 服务器端事务 …… 298
10.6.3 定时器 …… 303
10.7 SIP 在传输层上的通信处理 …… 303
10.7.1 客户端 …… 304
10.7.2 服务器端 …… 305
10.7.3 数据帧与错误处理 …… 307

第 11 章 SIP 的安全机制 …… 308

11.1 SIP 遭受的攻击和威胁模式 … 308
11.1.1 注册攻击 …… 308
11.1.2 伪装服务器 …… 308
11.1.3 篡改消息 …… 308
11.1.4 恶意修改以结束会话 …… 309
11.1.5 拒绝服务与放大 …… 309
11.2 SIP 的安全机制 …… 310
11.2.1 传输层和网络层安全 …… 310
11.2.2 SIPS URI 方案 …… 311
11.2.3 HTTP 鉴权 …… 312
11.2.4 S/MIME …… 317
11.3 SIP 安全机制的实现 …… 317
11.3.1 用户终端与代理服务器/注册服务器 …… 319
11.3.2 中间服务器与中间服务器 …… 319
11.3.3 终端系统与终端系统 …… 320
11.3.4 DoS 防护 …… 320
11.4 SIP 安全机制带来的限制 …… 321
11.4.1 HTTP Digest …… 321
11.4.2 S/MIME …… 322
11.4.3 TLS …… 322
11.4.4 SIPS URI …… 323

第 12 章 SIP 的应用 …… 325

12.1 SIP 在软交换中的应用 …… 325
12.2 SIP 在 IMS 中的应用 …… 326
12.2.1 IMS 概述 …… 327
12.2.2 IMS 网络单元分析 …… 329
12.2.3 IMS 组网结构分析 …… 333
12.2.4 IMS 业务体系介绍 …… 334
12.2.5 SIP 与 IMS …… 337

第 13 章 SIP 应用开发 …… 338

13.1 SIP 开源协议栈 …… 338
13.1.1 OPAL …… 338
13.1.2 VOCAL …… 339
13.1.3 sipX …… 340
13.1.4 ReSIProcate …… 340
13.1.5 oSIP …… 341
13.2 oSIP 简介 …… 342
13.2.1 oSIP 的结构 …… 342
13.2.2 解析器模块 …… 343
13.2.3 状态机模块 …… 348
13.2.4 工具模块 …… 352
13.2.5 oSIP 使用概要 …… 352
13.3 基于 oSIP 的 PartySIP 开发举例 …… 356
13.3.1 PartySIP 的主要数据结构 …… 356
13.3.2 PartySIP 的状态机图 …… 361
13.3.3 PartySIP 的初始化过程 … 361
13.3.4 osip_timers_thread 的简单处理过程 …… 370
13.3.5 向 PartySIP 中加入模块 …… 373

附录 …… 376

附录 A Megaco/H.248 协议基本包 …… 376

附录 B 缩略语 …… 380

参考文献 …… 387

第1章　下一代网络技术与软交换技术的介绍

1.1　下一代网络简述

人类跨入21世纪，信息技术迅速发展。特别是软件领域，软件技术的不断发展，软件开发方式已经由面向对象、面向组件的方式代替了传统的面向过程的软件开发方式。在通信领域，在传统交换发展了几十年后的今天，软交换吸引了广大电信界人士的眼球。作为下一代网络的核心技术，软交换已经成为电信领域内最受关注的焦点之一，无论是电信运营商、设备制造商，还是致力于该领域的业务开发商，都希望借助软交换技术推广和发展电信服务业，巩固自身的市场地位，获得直接经济效益。下一代网络应该是一个能够屏蔽底层通信基础设施多样性，能够提供一个统一开放的、可伸缩的、安全稳定和高性能的服务平台，以支持快速灵活地开发、集成、定制和部署网络业务应用。

随着网络技术、软件技术和通信技术的发展，有线网络、Internet和无线网络之间的融合成为网络发展的主要特征。下一代网络主要目标就是为公众提供包含语音业务和互联网业务在内的各种视频业务，而当前的电信网是为电话业务所设计的，很难适应NGN多业务特征的需求，因此需要有新的网络结构来支持这些业务特征，这就是NGN产生的一个重要背景。

对于下一代网络的研究主要有电信界以及计算机界两大阵营。两大阵营都试图将各自当前的网络向下一代网络过渡。计算机界标准化组织的代表是IETF（互联网工程任务组，Internet Engineering Task Force）。在IETF看起来，下一代网络就是NGI（下一代互联网，Next Generation Internet）。IETF在传输层的规范集中在IPv6协议，在业务层的规范主要基于智能终端采用端到端控制方式；电信界对下一代网络的研究主要集中在ITU（国际电信联盟，International Telecommunication Union），ETSI（欧洲电信标准化协会，European Telecommunication Standards Institute），3GPP（第三代合作伙伴计划，3rd Generation Partnership Project），3GPP2（第三代合作伙伴计划2，3rd Generation Partnership Project 2）等组织，在电信界看起来，下一代网络（Next Generation Network，NGN），在传输层上主要体现在自动交换光网络（ASON）上，在应用层主要体现在软交换上。在电信界ITU-T（国际电信联盟标准化部门）是权威的标准化组织，ITU-T对NGN的研究主要由第11工作组和第16工作组承担。本书提到的下一代网络的相关技

术均是从电信界角度进行分析的。

在下一代网络的协议规范上，ITU-T 和 IETF 在相关的标准化制定方面已经取得了重要的进展。如 H.248 协议、BICC（与承载无关的呼叫控制）协议、SIP（会话起始协议）和 SIGTRAN（信令传输）系列协议等。NGN/软交换体系相关的协议目前正在发展之中，ITU、IETF 等标准化组织都已制定了相应的标准或者草案。但是部分相关的扩展包，只能由设备的研发部门提出实施草案，并提交相关的标准化组织，等待批准。

1.1.1 下一代网络概述

从技术的角度上讲，NGN 的范围相当宽泛。如果特指业务网层面，下一代网络是指下一代业务网；对于交换网是指软交换体系；对于数据网，则指下一代互联网，例如：IPv6 技术；而对于移动网，则指 3G 和 B3G；如果特指传送网层面，则 NGN 指下一代传送网，特别是智能光网络；如果特指接入网层面，NGN 则指下一代宽带接入网。

广义的 NGN 实际包容了所有新一代网络技术，而狭义的 NGN 往往特指软交换。今天国内谈得比较多的是狭义的 NGN，即软交换。传统的程控交换机都是不同厂商开发自己的硬件平台，而软交换则是建立在业界标准的硬件平台之上，通过软件包的方式来提供所有的呼叫控制业务生成的功能。这种硬件平台是 IT 领域常用到的服务器。

“软交换”是目前 NGN/3G 讨论的热点，从实际意义上讲，软交换设备是在 IP 网上语音传送 VoIP 体系中把呼叫控制功能从媒体网关中分离出来，通过服务器上的软件实现基本呼叫控制功能后才真正实现的。1999 年后，软交换设备这一名词被业界普遍认可，成为最后的通用概念。顺应固定和移动融合以及整个电信网全 IP 化演进的趋势，第三代移动通信网络的发展在很多方面应用了 NGN 技术。2000 年前后，3GPP 制定 WCDMA R4 阶段规范时，首次把 NGN 提出的软交换概念引入移动核心网。从网络结构、接口协议、业务以及业务开发等方面看，3G 与 NGN 的发展是协调一致的。在网络结构方面，NGN 和 3G 都提出了承载和控制分离的网络体系结构；在接口协议方面，3G 与 NGN 所采用的协议很多都是一致的，包括 BICC、SIP/SIP-T、Megaco/H.248 和 SIGTRAN；在业务方面，3G 和 NGN 不仅提供的业务种类相似，例如都提供语音和多媒体业务，而且业务的实现方式也类似，3G 和 NGN 都支持开放的业务接口，因此，两者在业务层面和架构上均是统一的，相似的业务可以同时构建在 NGN 和 3G 之上。从广义上讲，NGN 包括了固定和无线领域，WCDMA R4 和 R5 及其后续阶段都属于 NGN 架构。

1.1.2　下一代网络的分层结构

可以看出，下一代网络采用分层、开放的体系结构，并将传统交换机的功能模块分离成独立的网络实体，各实体间采用开放的协议或应用编程接口（API），从而打破了传统电信网封闭的格局，实现多种异构网络间的融合。下一代网络的体系通过将业务与呼叫控制分离、呼叫控制与承载分离来实现相对独立业务体系，使得上层与下层的异构网络无关，灵活、有效地实现了业务的提供，从而满足了人们对多样的、不断发展的业务需求。可以说，下一代网络完全体现了业务驱动的思想和理念，很好地实现了多网的融合，提供了开放、灵活的业务提供体系。图 1-1 为下一代网络的体系结构介绍。

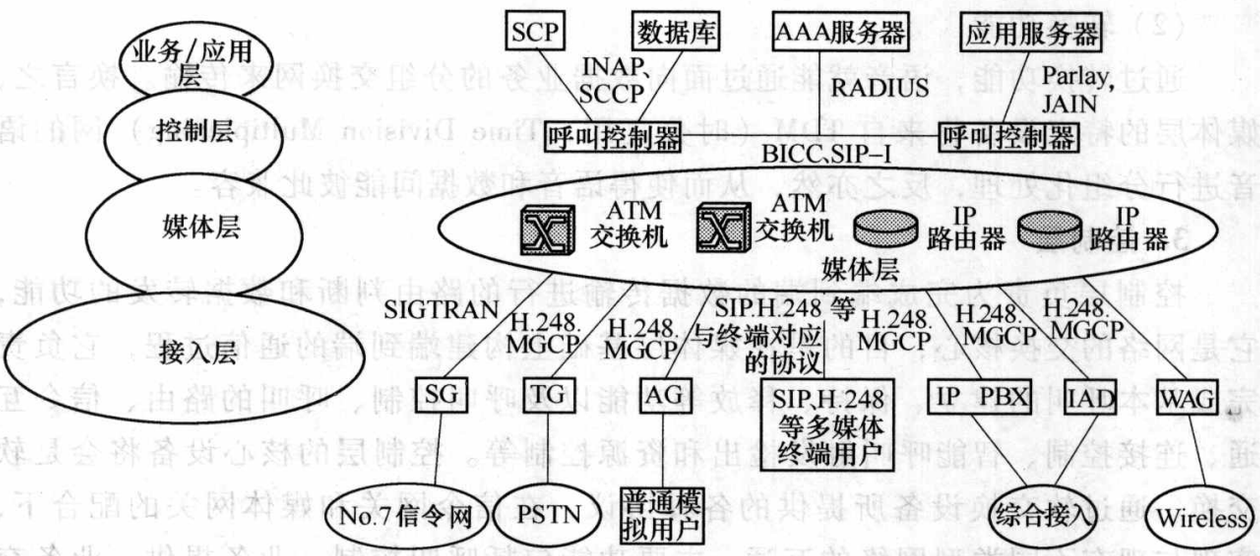


图 1-1　下一代网络的体系结构

SG—信令网关
TG—中继网关
AG—接入网关
IP PBX—基于 IP 技术的程控用户交换机
IAD—综合接入设备
WAG—无线接入网关
SCP—业务控制点
INAP—智能网应用部分
SCCP—信令连接控制部分
RADIUS—远程鉴别拨入用户服务
PSTN—公用电话交换网

下一代网络从功能上可以分为四个层次：接入层、媒体层（即传输层）、控制层以及业务/应用层。图 1-1 具体说明了这种结构，各层的功能如下：

1. 接入层

该层主要提供各种网络和设备接入到核心骨干网的方式和手段，主要包括信令网关、媒体网关和接入网关等多种接入设备。它利用各种接入设备实现不同业务的接入，并实现信令和媒体信息格式的转换。其中信令网关负责 No. 7 信令网消息的转换，消息由多媒体传送协议（MTP）承载转换为由 IP 承载，并将

信令消息经 IP 网发送给软交换。媒体网关负责适配功能，具体有 ISDN（综合业务数字网）的 IP 中继网关、异步转移模式（ATM）媒体网关、无线媒体网关、用户媒体网关。同时还可以直接和 H. 323 终端、SIP 终端进行连接，提供相应业务。媒体接入网关在软交换的控制下，实现相互间的通信。接入设备应能向上连接高速传输线路，向下支持多种业务的接口。

2. 媒体层（或叫做传输层）

媒体层有两个功能：

（1）传送功能

引用一个面向分组交换的网络来传送各种信息流，实现语音/数据/视频/其他多媒体业务的融合。

（2）转换功能

通过转换功能，语音就能通过面向数据业务的分组交换网来传输。换言之，媒体层的特定设备将来自 TDM（时分复用，Time Division Multiplexing）网的语音进行分组化处理，反之亦然，从而使得语音和数据间能彼此兼容。

3. 控制层

控制层负责为完成端到端的数据传输进行的路由判断和数据转发的功能，它是网络的交换核心，目的是在媒体层基础上构建端到端的通信过程，它负责完成基本呼叫的建立、保持、释放等功能以及呼叫控制、呼叫的路由、信令互通、连接控制、智能呼叫触发检出和资源控制等。控制层的核心设备将会是软交换。通过软交换设备所提供的各种协议，在信令网关和媒体网关的配合下，实现与现有不同类型网络的互通，主要功能包括呼叫控制、业务提供、业务交换、资源管理、用户认证、SIP 代理等。

4. 业务/应用层

该层是一个开放、综合的业务接入平台，在电信网络环境中，智能地接入各种设备，提供各种增值服务。该层是软交换体系结构中的最高层，通过设置各种应用服务器，提供各种业务逻辑，满足用户个性化的需求。为了使得业务的提供和呼叫控制相分离，在软交换设备和应用服务器之间定义了相关的协议或者 API，例如 SIP，JAIN 或者 Parlay 应用编程接口。主要在呼叫建立的基础上提供附加的增值业务，包括传统智能网上的和新的 IP 网上的 SCP、数据库、用户身份认证、授权和计费协议（AAA）服务器、应用服务器等。其中，应用服务器提供开放的应用编程接口（API），方便业务的开发和提供。

下一代网络是业务驱动的网络，通过业务与呼叫控制分离以及呼叫控制与承载分离实现相对独立的业务体系，使业务真正独立于网络，灵活有效地实现业务的提供。用户可以自行配置和定义自己的业务特征，不必关心承载业务的网络形式以及终端类型，使得业务和应用的提供有较大的灵活性，从而满足用

户不断发展更新的业务需求，也使得网络具有可持续发展的能力和竞争力。

1.2　软交换技术

1.2.1　软交换技术概况以及同 IMS 的关系

下一代网络是集语音、数据、传真和视频业务于一体的全新的网络。作为下一代网络控制功能实体的软交换其核心思想是硬件软件化，通过软件来实现原来交换机的控制、接续和业务处理等功能，在下一代网络中更快地实现各类复杂的协议的互通，并更方便地提供业务。

软交换是一种功能实体，它将呼叫控制功能从媒体网关（传输层）中分离出来，通过服务器上的软件实现为下一代网络提供具有实时性要求业务的呼叫控制和连接控制功能，包括呼叫选路、管理控制、连接控制和信令互通。软交换位于网络分层中的控制层，它与媒体层的网关交互作用，接收正在处理的呼叫相关信息，并指示网关完成呼叫。软交换主要处理实时业务，首先是语音业务，也可以包括视频业务和其他多媒体业务。

目前 NGN 的网络演进引起了广泛的关注，即固定网演进是否适于采用 IMS（IP 多媒体子系统，IP Multimedia Subsystem）的架构？IMS 是 3GPP 在 Release 5 版本上提出的可支持 IP 多媒体业务的子系统，它的核心特点是采用了 SIP 和与接入的无关性。目前 IMS 支持 2G 和 3G 的移动接入方式，还不能支持固定接入方式。在网络融合的发展趋势下，3GPP、ETSI 和 ITU-T 都在研究基于 SIP 的网络融合方案，希望能使 IMS 成为基于 SIP 会话的通用平台，同时支持固定和移动的多种接入方式，以实现固网和移动网的融合。

IMS 将来如何提供统一的业务平台、它的相关标准的制定乃至商品的成熟性等问题仍有待进一步的探讨和解决。同时支持固定和移动的多种接入方式，实现固定网和移动网的融合也是需要解决的重大问题。但目前，相当多的运营商面临的问题是如何将传统的基于电路的 PSTN 演进到基于分组技术的 NGN 中。尽管 IMS 可能成为 NGN 的重要组成部分，但是这种模式并不能很迅速地进行商业化的部署，特别是由于移动网和固定网的差异，对 IMS 的推广造成了一些不确定的影响，因为 IMS 是为移动网络设计的。在 2004 年 ITU-T 的 NGN 会议上，大会同意启动基于呼叫服务器（软交换）网络演进的标准化研究。ITU FGNGN 也在 2004 年启动了 PSTN 演进的研究，提出了基于呼叫服务器的 PSTN 演进方案和示例。呼叫服务器将主要完成电路交换网的演进历程，而 IMS 将主要完成多媒体业务的功能，随着 IMS 的成熟性的增加，两者的应用将长期共存。

1.2.2 引入软交换的意义

引入软交换的意义有以下几点：

1）软交换是融合现有 PSTN 和因特网的重要技术手段，是从现有网络向下一代网络平滑演进的重要举措。

2）软交换将承载和控制分离，控制和业务提供相分离，有利于各个层次的分别演进，适应了当前各项技术共同发展的现状。

3）提供开放的 API，使得第三方软件提供商能够很容易地介入增值业务的开发，使得向网络运营商向用户提供灵活、个性化和多样化的信息服务成为可能，在缩短增值业务开发周期的同时，降低了升级系统的费用。

所以，软交换在网络演进进程中的意义是：一方面，它造就了网络的融合；另一方面，也是更重要的一方面，在于它打开了原有封闭网络的一个缺口，为网络成为开放的和可编程的网络迈出了关键的第一步。

1.2.3 软交换支持的主要协议

软交换设备需要支持众多的互通协议，如：TUP，ISUP，H.248、DSS1，SCTP，M3UA，Megaco，SIGTRAN，INAP，SNMP，COPS，SCCP，TCAP，H.323，RADIUS，BICC，MGCP，SIP（SIP-T），Parlay 等。这些协议的层次结构如图 1-2 所示。

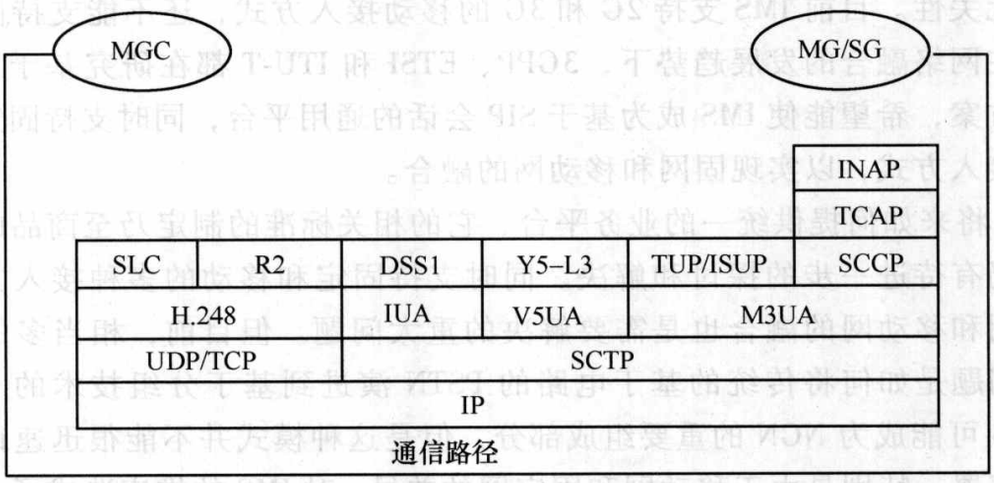


图 1-2 相关协议的层次关系

其中主要的协议有：

1）H.323：软交换与智能终端、VoIP 网关或网守之间的协议；

2）SIP：软交换与 SIP 终端或 SIP 服务器之间的控制协议；

3）MGCP/Megaco/H.248：软交换与媒体网关之间的协议，用于控制媒体网关的动作；也可用于软交换和媒体服务器之间，实现对媒体服务器的访问控制

和资源管理功能；

4）SIGTRAN：软交换与信令网关之间的协议，用于软交换和信令网关间 No. 7 信令的交互；

5）SIP-T、BICC：软交换系统之间互通采用的协议；

6）INAP：软交换与智能网 SCP 之间的协议，用于软交换和智能网之间的互通；

7）COPS：软交换与策略服务器之间的协议，能够使策略服务器通过 COPS 对软交换的 QoS 进行管理；

8）RADIUS：软交换与 AAA 服务器之间的协议，完成用户的鉴权、认证及计费等功能。上述这些协议或者接口可以分为以下几个类别：

1）呼叫控制类的协议：如应用服务器接口 Parlay API、INAP 等相关协议，当软交换设备处于智能网的业务交换点（SSP）时，SCP 通过 INAP 来控制 SSP 的进程；

2）网间互通协议：如 H. 323、TUP、ISUP、R2、BICC、SIP-T，软交换设备通过 SIGTRAN 、TUP、ISUP、R2 完成软交换设备同 PSTN 与 ISDN 的互通；

3）软交换系统之间的互通协议：可以通过 BICC、SIP-T 实现软交换之间的互通；

4）媒体网关控制协议类：如 Megaco/H. 248、MGCP，软交换通过 Megaco/H. 248、MGCP 控制媒体网关或媒体服务器，来达到媒体和信令分离的目的；

5）设备接入协议：如 V5. 2、DSS1，这两协议主要完成 V5. 2 接口和 ISDN 设备的接入。

1.2.4　软交换实现的主要功能

1. 软交换的基本功能

软交换设备是多种逻辑功能实体的集合，为下一代网络提供语音业务、数据业务和视频业务的呼叫控制和连接控制功能，是下一代网络呼叫与控制的核心设备，是电路交换网向分组网演进的重要设备。它独立于底层承载协议，主要完成呼叫控制、资源分配、协议处理、路由、认证、计费等主要功能，并可以向用户提供传统电路交换机所能提供的所有业务以及第三方业务。软交换设备作为一个开放的实体，与外部的接口采用开放的协议。图 1-3 展示了软交换设备对外接口，软交换必须能够维护外围设备的基本运作。

2. 软交换软件系统功能

软交换设备是电路交换网向分组网演进的核心设备，也是下一代电信网络的重要设备。软交换既可以作为独立的下一代网络部件分布在网络的各处，为所有媒体提供基本业务和补充业务，也可以与其他的业务节点结合，形成新的

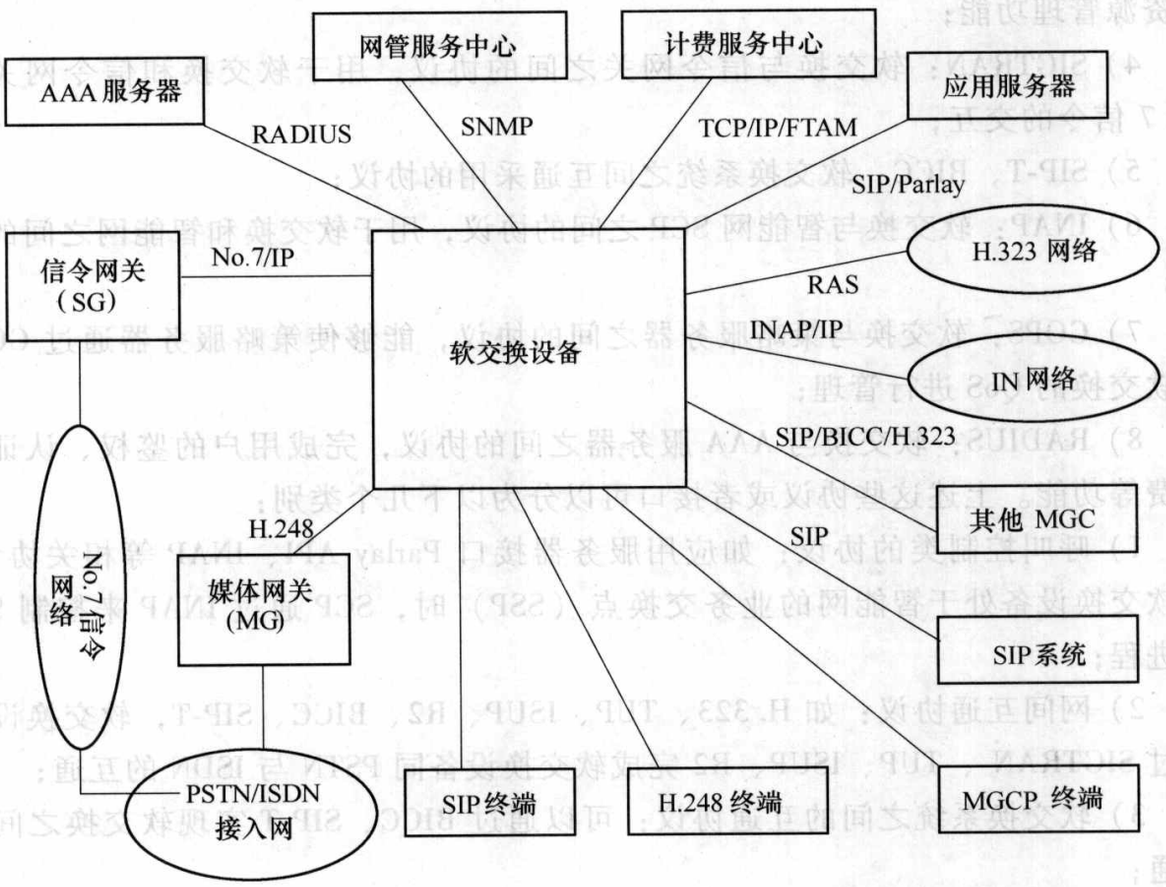


图 1-3 软交换设备对外接口

产品形态，这正是软交换的灵活性，使得它可以应用在以下各个领域：

1）协议互通领域。支持多种信令协议（包括 H. 248. H. 323，SIP. MGCP，ISUP，INAP、RADIUS，SNMP），实现现有网络的互连互通。

2）电路交换网领域。软交换核心控制设备与媒体网关和信令网关相结合，呼叫控制和承载连接，可作为汇接局和长途局的接入，提供现有的 PSTN 中的基本业务和补充业务。

3）电路/分组网领域。软交换可与分组终端进行互通，实现分组网与电路网的互通。如在 H. 323 呼叫中，软交换可视为 H. 323 终端，在 SIP 呼叫中，可视为用户代理。

4）智能网领域。软交换与媒体网关相结合，完成 SSP 功能，通过支持 INAP 和 CAP（信道接入协议）与现有智能网的 SCP 相结合，提供各种智能业务。

5）业务提供领域。目前软交换支持提供基本语音、会议和网上浏览等业务，理论上软交换还可与应用服务器相配合，提供各种新的增值业务和补充业务。

作为下一代网络的核心设备之一，软交换为具有实时性要求的业务提供呼叫控制和连接控制功能，它独立于底层承载协议，主要完成呼叫控制、媒体网关接入控制、资源分配、协议处理、路由、认证以及计费等主要功能。并可以向用户提供现有电路交换机所能提供的所有业务和多样化的第三方业务。概括

地说，软交换主要实现以下功能：

（1）接入功能

各种媒体网关可以接入软交换，如 PSTN/ISDN 中继媒体网关、ATM 媒体网关、用户媒体网关、无线媒体网关和综合接入网关等，支持 MGCP、Megaco/H.248 协议，实现对媒体网关的控制、接入和管理，支持 SIP 终端通信接入，支持 H.323 协议及对 H.323 域的管理。

（2）呼叫控制功能

这是软交换的核心功能，它对基本呼叫的建立、维持和释放提供控制，包括呼叫处理、连接控制、资源控制、智能呼叫触发等，并提供支持多方呼叫控制、二次拨号等功能。

（3）业务提供功能

软交换能够提供传统交换机所能提供的全部业务，还与智能网相配合提供智能业务。此外，软交换提供可编程的、开放的 API，实现与外部应用平台的互通，从而易于新业务的引入和开发。

（4）互连互通功能

在现有网络向 NGN 的发展演进过程中，不可避免地要实现与现有多个网络的互连互通（包括 PSTN、No.7 信令网、VoIP 网、智能网等）。软交换设备支持相应的信令与协议（如 ISUP，INAP，MAP 等），从而完成与上述网的互通，实现网络融合。

（5）网管与计费功能

支持本地的维护管理以及实现与外部网管中心的互通，实现对软交换系统的管理、配置、统计、报警以及计费信息的采集等功能。

详细来讲，软交换软件系统需要为多种形式的业务提供有效的运行、开发和管理的支持，其主要功能应包括以下一些方面：

1）独立的呼叫控制功能独立于底层承载协议，为完成基本呼叫的建立、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，可以说是整个网络的灵魂；

2）接收来自业务交换功能的监视请求，并对其中与呼叫相关的事件进行处理；

3）接收来自业务交换功能的呼叫控制相关信息，支持呼叫的建立和监视；

4）支持基本的双方和多方呼叫控制功能，提供对多方呼叫控制的功能，包括多方呼叫的特殊逻辑关系、呼叫成员的加入、退出、隔离、旁听以及混音过程的控制等；

5）能够识别媒体网关报告的用户摘机、拨号和挂机等事件，控制媒体网关向用户发送各种声音信号，如拨号音、振铃音、回铃音等，提供满足运营商需

求的拨号计划；

6）实现多种异构网络间的无缝融合，这就要求软交换不仅要支持多种不同的网络协议，而且还需提取不同网络中呼叫控制的通用特征，在异构的网络资源层上叠加一个通用的呼叫控制模型，实现对跨网呼叫的统一处理；

7）控制媒体网关发送自动语音应答（IVR），以完成诸如二次拨号等多种业务；

8）软交换可以同时直接与 H.248 终端、MGCP 终端、H.323 终端和 SIP 客户终端进行连接，提供相应业务；

9）当软交换位于 PSTN/ISDN 本地网时，应具有本地电话交换设备的呼叫处理功能；

10）当软交换位于 PSTN/ISDN 长途网时，应具有长途电话交换设备的呼叫处理功能；

11）承载控制功能，在融合网络中，特别是 IP 分组网中，信令链路和承载链路的建立是分离的和分别可控的，因此软交换还必须能够对承载链路的建立进行控制和监视，以保证多媒体业务和特定业务对承载链路服务质量的要求；

12）媒体接入功能，此项功能可以认为是一种适配功能，它可以连接各种媒体网关，如 PSTN/ISDN IP 中继媒体网关、ATM 媒体网关、用户媒体网关、无线媒体网关、数据媒体网关等，完成 H.248 协议功能，同时还可以直接与 H.323 终端和 SIP 客户终端进行连接，提供相应业务，因此软交换应支持 H.248、H.323、SIP、SCTP、M3UA、M2PA、V5UA、IUA 等等；

13）业务提供功能，由于软交换在网络从电路交换向分组网演进的过程中起着十分重要的作用，因此软交换应能够提供 PSTN/ISDN 交换机提供的业务，包括基本业务和补充业务，同时还应该可以与现有的智能网配合提供现有智能网所提供的业务；

14）互连互通功能，软交换应可以通过信令网关实现分组网与现有 No.7 信令网的互通，可以通过信令网关与现有智能网互通，为用户提供多种智能业务，允许业务控制功能（SCF）控制 Vole 呼叫且对呼叫信息进行操作（如：号码显示等），可以通过软交换中的互通模块，采用 H.323 协议实现与现有 H.323 体系的软交换网的互通，可以通过软交换中的互通模块，采用 SIP 实现与未来 SIP 网络体系的互通，可与其他软交换设备互通互连，它们之间的协议可以采用 SIP 或 BICC，提供 IP 网内 H.248 终端、SIP 终端和 MGCP 终端之间的互通；

15）网守功能，即接入认证与受权、地址解析和带宽管理，软交换应能够与认证中心连接，并可以将所管辖区域内的用户、媒体网关信息送往认证中心进行认证与授权，以防止非法用户/设备的接入，软交换应可以完成 E.164 地址至 IP 地址、别名地址至 IP 地址的转换功能，同时也可完成重定向的功能；

16）操作维护功能，主要包括业务统计和报警等；

17）计费功能，软交换应具有采集详细话单及复式计次功能，并能够按照运营商的需求将话单传送到相应的计费中心，当使用记账卡等业务时，软交换应具备实时中断呼叫的功能；

18）语音处理功能，软交换应可以控制媒体网关是否采用语音压缩，并提供可以选择的语音压缩算法，算法应至少包括 G.729 和 G.723 等，软交换应可以控制媒体网关是否采用回声抵消技术，软交换应可以向媒体网关提供语音包缓存区的大小，以减少抖动对语音质量带来的影响。

第2章　软交换中的网关技术

2.1　网关控制协议的特征

软交换系统中，网关分解的结构如图 2-1 所示。

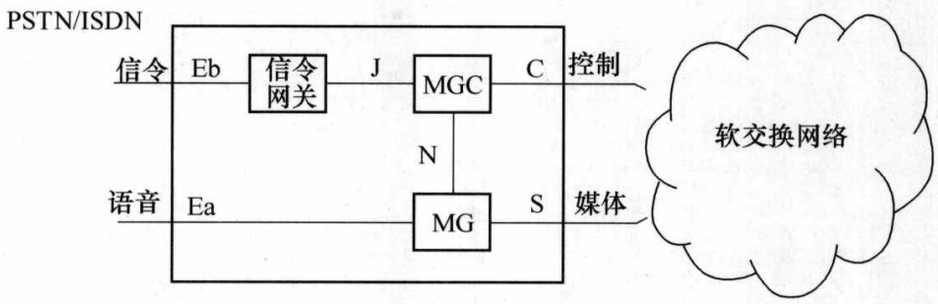


图 2-1　网关分解模型图

其中，媒体网关负责媒体变换以及 PSTN 和软交换两侧通路的连接；信令网关负责信令转换，但仅进行信令的底层转换，并不改变应用层消息；媒体网关控制器（MGC）负责根据收到的信令控制媒体网关的呼叫连接的建立和释放，MGC 才真正对信令消息进行分析和处理，并进行应用层的互通变换。但值得注意的是，图 2-1 中的 MGC 一般由软交换设备来实现其功能。其结构如图 2-2 所示，媒体网关控制器与媒体网关之间采用 MGCP 协议进行通信。

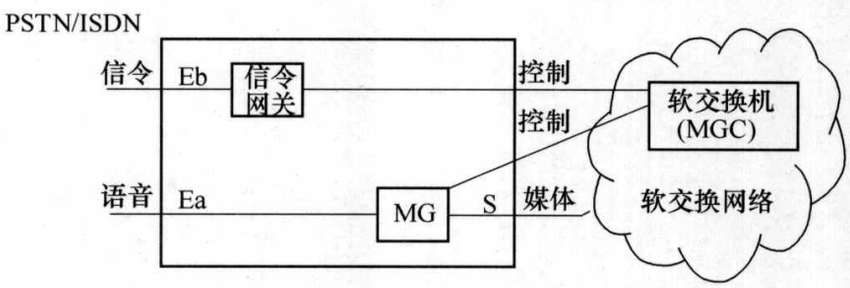


图 2-2　软交换的结构分解

从图 2-2 中可以看出，分离的网关结构将呼叫控制功能和媒体处理功能分别承载在 MGC 和 MG 上。因此，在这两个功能实体间必须存在一种控制协议来保证 MGC 可以根据呼叫需求建立、修改和释放媒体连接。软交换常用网关的种类如图 2-3 所示。

分离网关的实施如图 2-4 所示。

下面将从多个侧面来讨论 MG-MGC 的控制关系：

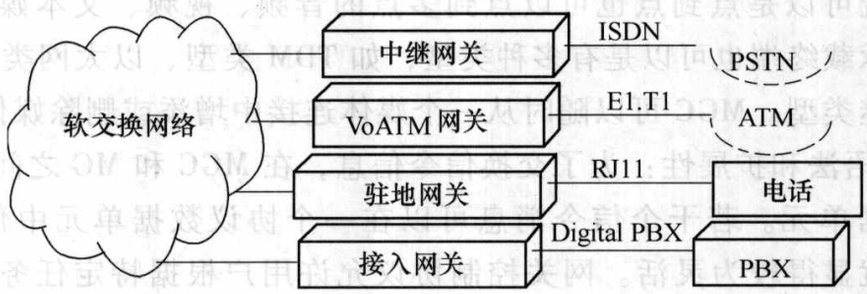


图 2-3　4 种常用的网关类型

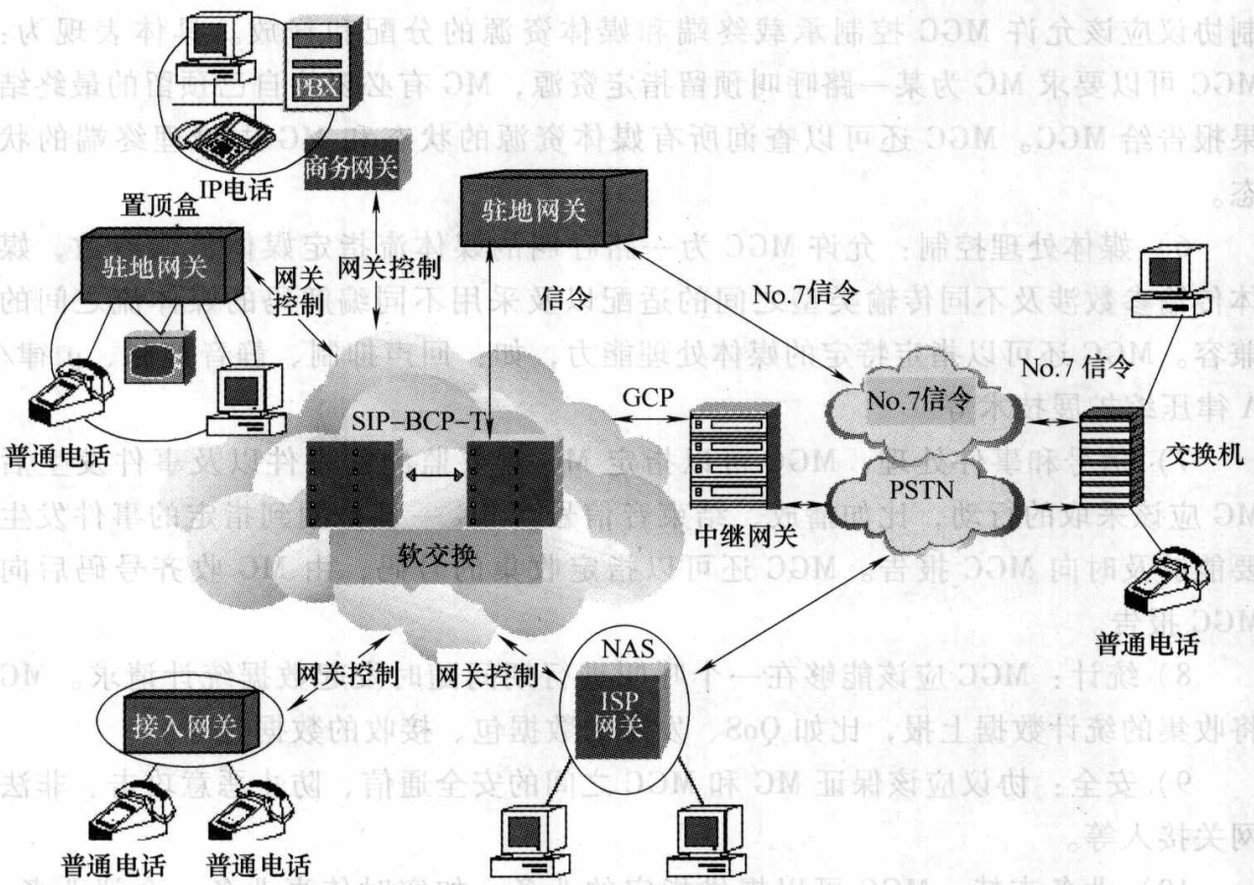


图 2-4　软交换中的各种网关

1）网关控制协议是 MGC 和 MG 之间的协议，所以该协议首先要支持在 MGC 和 MG 之间建立控制关系，控制原则是：一个 MGC 可以控制多个 MG；多个 MGC 可以控制同一个 MG。

2）协议信令的传输：基于分组网络的传输都不得不面临一个数据丢失的问题。为了保证 MGC 和 MG 之间消息的可靠传输，网关控制协议必须提供一定的传输机制，这种机制能够保证 MG 和 MGC 之间正确、灵活的消息交换，保证消息的实时传递，还应能检测出消息丢失并能对丢失的消息进行重传，同时拒绝收到的重复命令或响应。

3）连接管理和流管理：网关控制协议应当允许 MGC 建立多个媒体连接，这

些媒体连接既可以是点到点也可以点到多点的音频、视频、文本媒体流。一个连接涉及的承载终端也可以是有多种类型，如 TDM 类型、以太网类型、ATM 类型以及帧中继类型。MGC 可以随时从一个媒体连接中增添或删除媒体流。

4）协议语法和扩展性：为了交换信令信息，在 MGC 和 MG 之间必须定义传送的协议数据单元。若干个信令消息可以在一个协议数据单元中传输。这样，命令的执行就显得较为灵活。网关控制协议允许用户根据特定任务的需求对协议的大小进行裁减。网关控制协议允许 MGC 和 MG 的协议版本独立升级。

5）资源预留和控制：资源预留对于媒体处理必不可少。一个完善的网关控制协议应该允许 MGC 控制承载终端和媒体资源的分配和释放。具体表现为：MGC 可以要求 MG 为某一路呼叫预留指定资源，MG 有必要将自己预留的最终结果报告给 MGC。MGC 还可以查询所有媒体资源的状态和 MG 中物理终端的状态。

6）媒体处理控制：允许 MGC 为一路呼叫的媒体流指定媒体传输参数，媒体传输参数涉及不同传输类型之间的适配以及采用不同编解码的媒体流之间的兼容。MGC 还可以指定特定的媒体处理能力，如：回声抑制、静音抑制、μ 律/A 律压缩扩展技术等。

7）信号和事件处理：MGC 可以指定 MG 需要监测的事件以及事件发生后 MG 应该采取的行动，比如播放、结束音信号。MG 一旦检测到指定的事件发生要能够及时向 MGC 报告。MGC 还可以指定收集的号码，由 MG 收齐号码后向 MGC 报告。

8）统计：MGC 应该能够在一个呼叫进行期间随时发起数据统计请求。MG 将收集的统计数据上报，比如 QoS、发送的数据包、接收的数据包等。

9）安全：协议应该保证 MG 和 MGC 之间的安全通信、防止恶意攻击、非法网关接入等。

10）业务支持：MGC 可以提供指定的业务，如实时传真业务、会议业务、IVR 业务等。

2.2 媒体网关技术

媒体网关是下一代网络技术软交换系统的核心设备，如果说软交换设备是软交换技术的实现，那么同样媒体网关控制器（Media Gateway Controller，MGC）则能称得上是软交换设备的一种实现。可以说媒体网关控制器所控制下的媒体网关是软交换的“四肢”，几乎大多数的业务都需要媒体网关在媒体网关控制器的控制下进行。因此，媒体网关技术一直都是 IT 设备制造商们追逐的热点。

2.2.1　媒体网关及其功能

人们知道“网关”的作用就是实现两个网络之间消息以及协议的相互转换和传输，从而最终实现两者之间的互通。而媒体网关的作用就是将一种网络中的媒体转换成网络所要求的另一种媒体格式。例如，媒体网关能实现电路交换网中的媒体信息和 IP 分组网中的媒体信息的转换和传输。

分离网关技术提出之前，运营商在采用 VoIP 技术时，就存在以下的问题：

1）现有 IP 电话网关不但要执行媒体格式变换，还要进行信令转换，而且还要控制网关内部资源，为每个呼叫建立网关内部的语音通路，这对电话网关具有特别高的要求，因此降低了 IP 电话系统的运行效率和灵活性；

2）网关功能过于复杂，不利于扩展；

3）与 PSTN 无缝综合方面较差；

4）可用性较差，传统功能合一的电话网关缺乏故障保护机制，因此网关引起的系统可靠性差、可用性低，极大地影响了电话系统的质量；

5）No. 7 信令并非在所有网关都需要支持，容易造成资源浪费；

6）随着 IP 电话使用日益广泛，如果接入要通过本地的 PSTN，这使电信运营商在运营的主导权和灵活性上受到了制约。

后来，以 H. 323 技术为背景的 IP 电话运营商开始尝试直接将 H. 323 的网关设置在用户端，但由于存在绕开运营商发起呼叫的可能性，这种办法并没有获得成功。为此，国际标准化组织先后提出 MGCP、HGCP、H. 248 等一系列协议标准。其核心思想就是将原来的电话网关进行功能上的分解，使之成为负责媒体流处理的媒体网关（MG）和负责呼叫建立控制的媒体网关控制器（MGC）两个部分。功能分解后，不但解决了用户绕开运营商直接通信的问题，还独立出电话网关中以前负责呼叫控制的功能，实现系统的高效性。

在软交换技术中，软交换继承了网关分离技术，将电话网关分解为信令网关和媒体网关以及媒体网关控制器，同时将媒体网关控制器的功能集成到软交换设备的功能中，能够实现 H. 323、MGCP、H. 248 等协议的互通。因此，在软交换系统中，通常意义上的媒体网关控制器就是指软交换设备。

媒体网关主要涉及的功能如下：

1. 用户或网络接入功能

各种用户或各种接入网络的综合接入，如普通电话用户、ISDN 用户和以太网用户接入等，都需要由媒体网关负责。媒体网关设备是用户或用户网络接入核心媒体层的“接口网关”。另外，它也具有软交换系统接入核心媒体网络的功能。目前接入核心媒体网络主要是通过 IP 网络接入技术来实现，同时 ATM 接入技术也非常成熟。

2. 语音压缩编码和处理功能

媒体网关要满足的一个主要要求就是提供优质的语音质量，因为这对用户来说有着直接的影响。但有时候带宽比质量更重要，这时压缩技术便成为一个重要因素。因此，媒体网关应该支持一系列的语音压缩编码方式（如：G.723、G.711、G.726 等）适应不同要求，既满足语音质量要求，又满足带宽要求。

另外，从灵活性的角度出发，对于媒体网关对特定媒体流的语音压缩编码方式是可以有多种方式的：

1）静态指定型，一般由手工方式在媒体网关和软交换设备之间进行选定；

2）动态选定型，一般是媒体网关或者软交换设备根据网络带宽等特殊条件进行动态的选定。

在上面两种方法中，动态选定型会具有更高的灵活性，特别是当网络发生拥塞或带宽变化时，媒体网关对语音压缩编码方式的动态转变，能自动实现从高速率编码方式向低速率编码方式的切换，从而能保证通话的继续进行，这对保证通话质量具有重要作用。

另外，要提供相关的回声控制功能，支持静音压缩和舒适噪声插入功能，同时为了保证通话质量和降低 VoIP 的延时抖动，媒体网关还都具备输入缓冲功能。

3. 媒体流的转换功能

在软交换中，语音、视频信息或综合的数据信息都被抽象成媒体流，传入软交换系统中。但是，实际情况中用户接入和核心媒体之间的信息格式、协议机制可能是不一致的，因而需要一种功能，它能在这种情况下实现将一种媒体流映射成另一种网络要求的媒体流格式，这就是媒体网关的媒体流转换功能。但是由于业务和网络的复杂性，媒体流映射并不是简单的映射，它涉及到媒体编码格式、资源预约和分配、特殊资源的检测处理等多项与媒体流属性相关的内容。此外，对不同的业务特性还有许多特殊的要求。

4. 受控操作和管理网络资源的功能

网络资源管理功能包括对资源状态的管理和资源分配两部分：

1）资源状态管理是指向软交换报告物理实体的状态改变情况；

2）资源分配则是根据软交换的命令对相关连接释放资源或预留资源。

当然，媒体网关受软交换的控制，它绝大部分的动作，特别是与业务相关的动作都是在软交换的控制下完成的，如编码、压缩算法的选择，呼叫的建立、释放、中断，资源的分配和释放，特殊信号的检测和处理等。

媒体网关和软交换之间的特殊关系决定了它们之间控制协议的重要性，就是软交换和媒体网关之间的控制协议。

媒体网关控制协议中，使用广泛的有两个：一个是 MGCP（Media Gateway

Control Protocol)，它属于应用层控制协议，主要完成 MGC 对 MG 的控制，以实现网络中 MG 之间的连接，并且处理 MG 与 MGC 的交互；另一个是 H.248 协议，它是由 IETF Megaco 工作组和 ITU-T 在 MGCP 的基础上分别提出了 Megaco 和 H.248 协议，对 MGCP 进行了扩展。H.248 和 Megaco 在协议文本上相同，只是在协议消息传输语法上有所区别。

5. 管理和统计功能

作为软交换网络系统中的一员，媒体网关同样受到网管系统的统一管理，媒体网关也要向软交换或网管系统报告相关的统计信息。

2.2.2　RTP

随着 Internet 的迅速增长和普及，人们对网络多媒体信息的需求越来越多。为了适应网络多媒体化的发展，流媒体（音频、视频流）技术应运而生。流媒体是指运用流式传输的方式在 Internet 和 Intranet 播放的媒体格式。传统的 TCP/IP 是为非实时传输业务设计的，它的传送机制和拥塞控制都不适合网络多媒体的实时传输，在网络多媒体的实时传输中面临诸多问题，因此，需要新型高速网络协议支持。另一方面，随着流媒体的技术发展，作为网络多媒体应用的核心协议 RTP（Real-time Transport Protocol，实时传输协议）则受到越来越多的重视，被广泛用于各种多媒体传输系统中。

1. RTP 特性

RTP 是用于 Internet 上针对多媒体数据流（例如交互式音频、视频、仿真数据等实时媒体流）而设计的轻型传输协议，RTP 本身用于传输实时数据，其功能是提供净荷类型指示（即数据类型和编码方式）、数据分组序号、数据发送时间戳和数据源指示，接收端则能根据这些信息正确地重组原始信号。它是由 IETF 的音频视频传输工作组（AVT）负责开发，并在 1996 年由 IETF 在 RFC 1889 中正式给出的，RTP 被定义为在一对一的单播或一对多的组播传输情况下工作，其目的是提供时间信息和实现流同步。RTP 在功能上独立于下面的传输层和网络层，但不单独作为一个层次存在，它通常使用用户数据报协议（UDP）来传送数据，但 RTP 也能在传输控制协议（TCP）或 ATM 等协议下工作。

相对于传统的传输层协议，RTP 具有以下特性：

（1）实时性

RTP 提供了具有实时特性的端到端的数据（如视频或音频数据）传输服务。

（2）简单性

RTP 本身并不能提供可靠的传送机制，也不提供流量控制或拥塞控制，甚至不包括长度和报文边界描述。可靠的传送机制、流量控制以及拥塞控制一般是依靠 RTCP（实时传输控制协议，Real-time Transport Control Protocol）来提供。

另外，RTP 将部分传输层协议功能上移到应用层完成，简化了协议本身的复杂程度，提高了效率。

（3）数据流和控制流的分离

RTP 的控制报文和数据报文使用不同端口，这样就实现了数据流和控制流的分离。协议规定，RTP 流使用的端口号应为偶数，相应的 RTCP 使用的端口号应为奇数。

（4）支持多播

RTP 一般运行在 UDP 上，因此可以利用 UDP 的多路复用技术支持组播。

（5）可扩展性

通常 RTP 算法是作为应用程序代码的一部分。开发者可以根据具体要求来对协议进行充分的扩展。

RTP 遵循“应用级分帧，集成层处理”的思想，是一个面向应用的协议，它有紧密相关的部分与其配合进行媒体流的实时传输，即 RTCP，特别是在 RTP 会话期间，各参与者周期性地传送 RTCP 包，RTCP 包中含有已发送的数据包的数量、丢失的数据包的数量等统计资料，因此服务器可以利用这些信息动态地改变传输速率，甚至改变有效载荷类型，这种模式特别适合传送网上的实时数据。应用程序在 UDP 上执行 RTP，以便使用其多路技术和检验的服务，但是 RTP 也可以与其他适合的底层网络或传输协议共同使用。如果底层网络提供组播分配，那么 RTP 可以使用该组播分配支持多路目标文件的数据传输。

在服务质量方面，RTP 本身并没有为及时传送提供任何机制或 QoS 保证，但它依赖于低层服务去实现这一过程。RTP 并不能保证传送过程或防止无序传送，也不能确定底层网络的可靠性。RTP 实行有序传送，RTP 中的序列号允许接收方重建发送方的包序列，同时序列号也能用于决定适当的包位置。

2. RTP 的包结构

RTP 数据包由包头（载荷类型、序列号、时间戳和同步源标志符等）和数据（音频或视频）组成，包头的前 12B 是固定的，其后的 CSRC 为可选部分，数据的长度由数据类型和编码决定。RTP 数据包包头格式如图 2-5 所示。

图 2-5 RTP 数据包包头格式

其中：

1）V 为版本号，字长为 2bit，该字段表明了 RTP 的版本，通常定义的版本

号为 2。

2）P 为填充指示位，字长为 1bit，如果填充位被置位，表明该包的末尾含有一个或多个不属于载荷部分的额外填充字节。

3）X 为扩展指示位，字长为 1bit，如果扩展位被置位，表明该 RTP 包的包头后紧跟着一个头扩展。该头部包含了一些与媒体数据无关的公共信息。

4）CC 为 CSRC 计数，字长为 4bit，CSRC 计数字段包含了跟在固定 RTP 头后的作用源标志符（CSRC）的数目。

5）M 为标记位，字长为 1bit，目的在于允许重要事件在包流中标记出来，例如帧边界，它的解释由框架（Profile）文件定义。

6）PT 为载荷类型，字长为 7bit，载荷类型指出包中数据的格式，这样，RTP 就可以支持不同的数据类型和不同的压缩编码方式，发送者可以根据网络的拥塞情况动态地改变压缩编码方式。

7）序列号的字长为 16bit，每发出一个 RTP 数据包，序列号值就会加一，接收者也可以利用序列号来探测数据包的丢失情况并恢复包顺序，序列号的初始值是随机的（不可预知的）。

8）时间戳的字长为 32bit，时间戳反映 RTP 数据包中第一个 8 位位组的采样时刻，采样时刻必须从单调、线性增加的时钟导出，以允许同步与抖动计算。时钟必须有足够的分辨率，满足所需同步和测定包到达的抖动精度。时钟频率依赖携带数据的格式，并在框架文件中静态指定。

9）SSRC 为同步源（Synchronous Source）标识，字长为 32bit，SSRC 字段指定了同步源，同一个 RTP 连接内的任意两个同步源都不会有相同的 SSRC 标志符。

10）CSRC（Contributing Source）列表，0 ~ 15 项，每项 32bit，CSRC 只是固定包头后 CSRC 标识符的数目。这个数值是用来支持“混合”能力的，它从多个作用源收取多个媒体包，在把它合并成一个包。CSRC 列表指明了包含在包中的有效载荷的作用源。标志符的数目由 CSRC 计数字段（CC）给出。如超过 15 个作用源，也仅标识 15 个。

2.2.3　RTCP 概述

1. RTCP 的基本概念和功能

RTCP 是支持 RTP 工作的一个控制协议，它单独运行在底层协议上。RTCP 是由接收方向发送方发送的报文，它是基于周期性传输的基础上，采用与数据包相同的分配机制，向会话参与者发送控制包。它负责监视网络服务质量、通信带宽以及网上传送的信息，并将其通知给发送端。RTCP 包周期性地向同一个组播组内的所有成员发送。

RTCP 完成 4 个功能服务：

1）主要是提供数据发布的质量反馈，提供 QoS 检测。RTCP 是作为 RTP 的一部分，与其他传输协议的流和阻塞控制有关。这些信息包括所收到数据包的最大序列号、丢失的包数、乱序包的数目以及用于估计传输时延的时标信息，它们反映了当前的网络状况，接收方在接收到这些信息后，可以依此调整它们的发送速率。另外，反馈对自适应编码控制直接起作用，但 IP 组播经验表明，从发送者收到反馈对诊断发送错误是至关重要的。给所有参加者发送接收反馈报告允许问题观察者估计那些问题是局部的，还是全局的。如果是以 IP 组播等发布机制进行 RTP 通信，那么组播可以使网络服务提供商之类的团体可以接收反馈信息，充当第三方监控者来诊断网络问题。反馈功能由 RTCP 发送者和接收者报告执行。

2）提供媒体间的同步机制。在网络会议系统中，RTP 源可能会有几种媒体（音频/视频）要传输，这些不同媒体间的同步需要依靠 RTCP 中包含的时钟信息和相关的 RTP 时标信息来进行同步。RTCP 带有称作规范名（CNAME）的持久传输层标识。一个成员有一个规范名与之对应，它可以用来同步来自于同一个成员的音频和视频流。即使出现冲突或重启程序导致 SSRC 标识符改变的情况，接收者也能根据 CNAME 跟踪参加者。接收者也需要 CNAME 与相关 RTP 连接中给定的几个数据流联系。

3）前两种功能要求所有参加者发送 RTCP 包，因此，为了使 RTP 扩展到大规模数量，速率必须受到控制。让每个参加者给其他参加者发送控制包，这样每个人就能知道会议规模，根据这个数量就能计算包发送的速率。

4）第四个可选功能是传送最小连接控制信息，如辨识参加者。最可能用在“松散控制”连接，此时参加者能自由进入或离开，没有成员控制或参数协调，RTCP 充当通往所有参加者的方便通道，但不必支持应用的所有控制通信要求。

在会话的用户界面上显示会话参加者的标识 RTP 报文中提供了 SSRC 字段来进行源标识，然而，进一步的会话参加者的描述是需要的。RTCP 报文中的源描述提供了会话参加者的详尽描述，包括姓名、住址、Email 等。这是一个附加可选的功能。

2. RTCP 的包类型及格式

根据功能的不同，RTCP 报文可以分为 RR（接收者报告）与 SR（发送者报告）、SDES（源描述）、BYE（离开申明）和 APP（应用）四类，具体描述如下：

RR（Receiver Report，接收者报告），用于传输和接收听众的统计信息。SR（Sender Report，发送者报告），用于传输和接收发送者的一些统计信息。

发送包的类型会根据接收者是否是发送者而采用两种格式中的一种。除包

类型代码外，发送者报告与接收者报告间惟一的差别是发送者报告包含一个 20B 的发送者信息段。如某地址在发出最后和前一个报告间隔期间发送数据包，就发布 SR；否则，就发出 RR。SR 和 RR 都可包括或不包含多个接收报告块。最大可有 31 个接收报告块嵌入在 SR 或 RR 包中，附加 RR 包可堆叠在初始 SR 或 RR 包之后。

RTP 接收者通过 RTCP 中的 SR 和 RR 来提供反馈，SR 和 RR 都包括一个或多个报告块（Report Block），每一个报告块对应着一个同步源标识，这些同步源是在上一个报告周期内收到数据包的用户。SR 是由刚刚发送过报文的用户发出，RR 是由接收者发出。类似于 RTP 数据包，每个 RTCP 包以固定部分开始，紧接着的是可变长结构元素，但以一个 32 位边界结束，图 2-6 是一个 SR 包的包格式，由三部分组成，也可能还有第四个特定设置扩展部分。

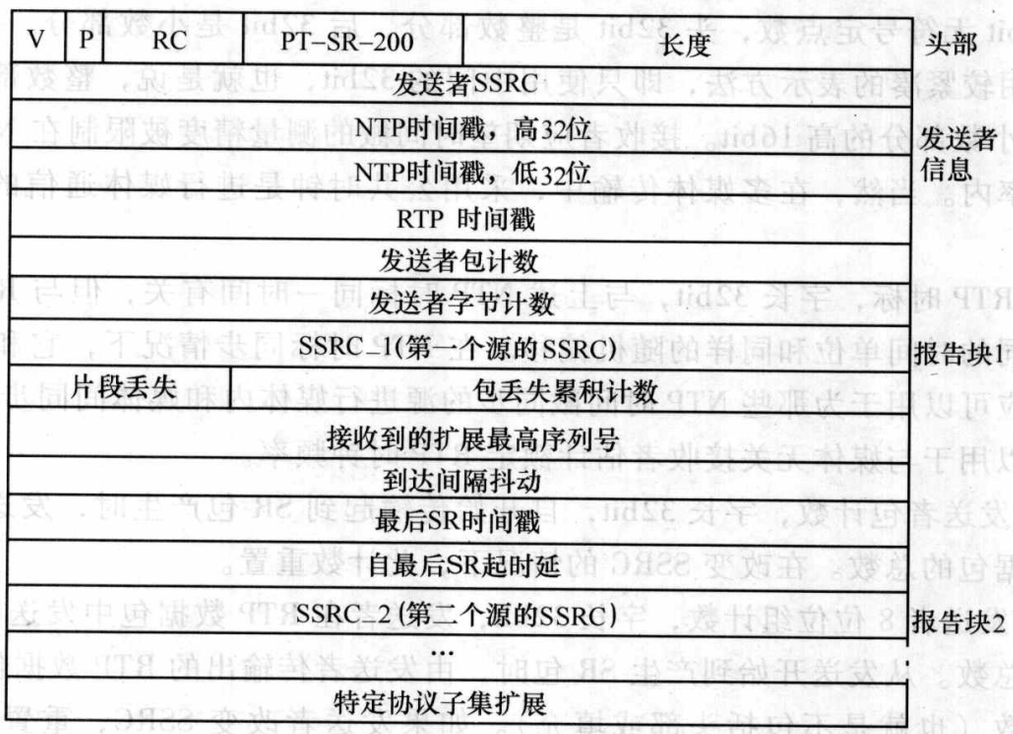


图 2-6　SR 包的包格式

（1）第一部分为头，8 个 8 位位组长，含义如下：

1）V 为版本号，字长为 2bit，该字段表明了 RTP 的版本，通常定义的版本号为 2。

2）P 为填充指示位，字长为 1bit，如果填充位被置位，表明该包的末尾含有一个或多个不属于载荷部分的额外填充字节。有些加密算法需要填充，块大小固定。在组合 RTCP 包内，填充仅在最后一个包中需要，因为组合包加密成一个整体。

3）RC 为接收报告计数，字长为 5bit，指明包含在分组内的接收报告块数目，0 值为有效。

4）PT 为包类型，字长为 8bit，如果值为常数 200，则标识此包为 RTCP 的 SR 包。

5）长度，字长为 16bit，其值为该 RTCP 包以 32 bit 为单位所得的长度减 1，包括头部和任何填充。

6）SSRC，字长 32bit，与 RTP 包的 SSRC 含义类似。

（2）第二部分为发送者信息，20 个 8 位位组，出现在每个发送者报告包中，含义如下：

1）NTP 时标，字长 64bit，表示报告发送时的时钟时间，它可以与来自其他接收者的接收报告中的时间戳一起用来测量到那些接收者的往返传播。这个时间值是一种用网络时间协议（Network Time Protocol，NTP）的时间戳格式表示，是相对于通用协调时间 1990 年 1 月 1 日 0 时的秒数。NTP 时间戳的完整格式是一个 64bit 无符号定点数，头 32bit 是整数部分，后 32bit 是小数部分。有时候，可以使用较紧凑的表示方法，即只使用中间的 32bit，也就是说，整数部分的低 16bit 和小数部分的高 16bit。接收者应期望时间戳的测量精度被限制在 NTP 时间戳分辨率内。当然，在多媒体传输中，采用公共时钟是进行媒体通信的重要基础。

2）RTP 时标，字长 32bit，与上述 NTP 时标同一时间有关，但与 RTP 时标有着相同的时间单位和同样的随机偏移。在 NTP 时标同步情况下，它和 NTP 时标的对应可以用于为那些 NTP 时间戳同步的源进行媒体内和媒体间同步。当然，它也可以用于与媒体无关接收者估计额定 RTP 时钟频率。

3）发送者包计数，字长 32bit，自开始传输起到 SR 包产生时，发送者发送 RTP 数据包的总数。在改变 SSRC 的情况下，此计数重置。

4）发送者 8 位位组计数，字长 32bit，发送者在 RTP 数据包中发送的载荷 8 位位组总数。从发送开始到产生 SR 包时，由发送者传输出的 RTP 数据包中载荷字节总数（也就是不包括头部或填充）。如果发送者改变 SSRC，重置此计数。这部分可用于估计载荷数据平均速率。

（3）第三部分包含接收报告块，大小不固定。每个接收报告块传送单个同步源接收 RTP 包的统计。发生冲突，当源改变 SSRC 时，接收者并不继续统计。这些统计包括：

1）SSRC_n（源标识），字长为 32bit，接收报告块中信息所属源的 SSRC。

2）片断丢失字长为 8bit，前一个 SR 或 RR 包发送以来所丢失的源 SSRC_n 的 RTP 数据包中一部分，定义成所丢失包的数目。

3）包丢失累积计数，字长 24bit，自接收以来所丢失的源 SSRC_n 的 RTP 数据包总数，该数字被定义为期望包数量减去实际接收到的包数量的差值，其中实际接收到的包数量包括迟到包或重复包。因此，迟到的包不计为丢失，如有

复制，此数量可能为负数。期望包数量等于接收到的扩展最高序列号减去接收到的初始序列号。

4）接收到的扩展最高系列号，字长 32bit，低 16bit 包含从 SSRC_n 源 RTP 数据包中收到的最高系列号，高 16bit 以系列号循环计数。如开始时间明显不同，同一连接内不同接收者将对系列号产生不同扩展。

5）间隔抖动，字长 32bit，RTP 数据包间隔时间的统计估计值，以时标为单位，是一个无符号整数。

6）最后 SR 时标（LSR），字长 32bit，接收到的来自源 SSRC_n 最近 SR 中 64bit NTP 时间标志的中间 32bit，若还没有接收到 SR，该域值为零。

7）自最后一个 SR 来的延迟（DLSR）：32bit，延迟以 1/65536s 为单位，表示源 SSRC_n 收到的最后一个 SR 包与发送此接收块之间的时间，如还没有收到 SR，此段设为零。假设 SSRC_r 为发出此接收报告块的接收机，源 SSRC_n 可以通过记录收到此接收报告块的时刻 A 来计算到 SSRC_r 的环路传输时延，可以利用最新的 SR 时间标志域计算整个环路时间 A-LSR，然后减去此 DLSR 域得到环路传播时延。

RR 包格式与 SR 包类似，如图 2-7 所示，但包类型包含常数 201，并删除发送者信息的五个字。当没有数据传输或接收报告时，则将一个空 RR 包（RC = 0）放在组合 RTCP 包的前头。

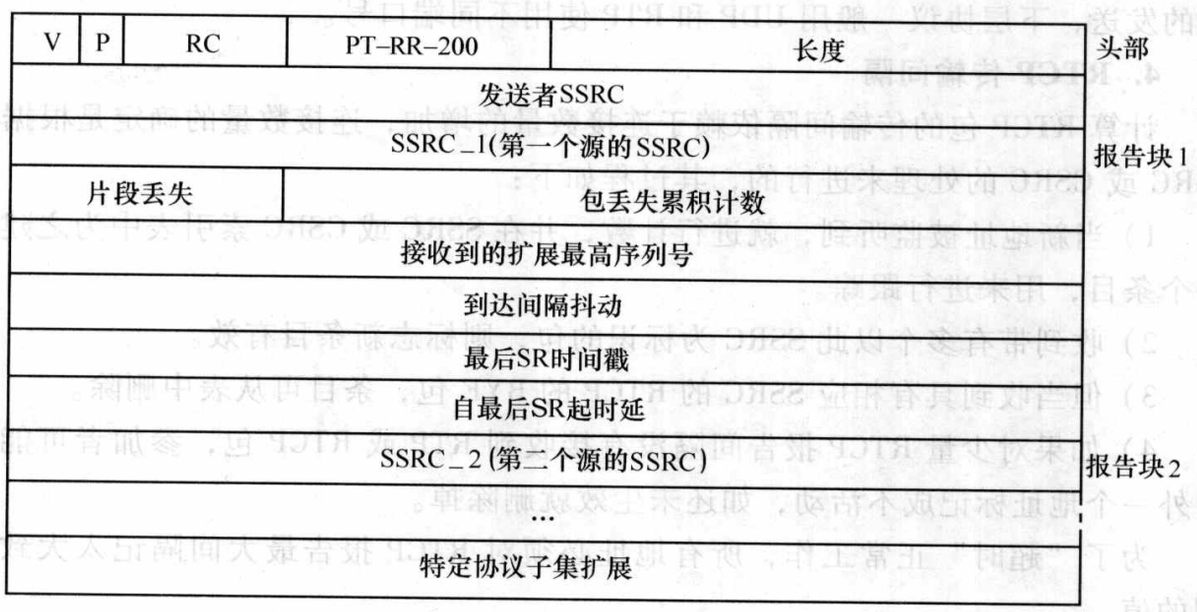


图 2-7　RR 包的包格式

从图 2-6 和图 2-7 的比较可以看出，SR 和 RR 在包结构上的主要区别就是是否有发送者信息这个字长为 20 个 8 位位组的部分。其他都是相同的。

如果接收者或发送者有附加信息通信，那么就应当对接收报告与接收者定义设置或应用扩展。如需要附加发送者信息，应首先包含在发送者报告的扩展

中，但不出现在接收者报告内。

根据发送者信息，第三方监控器可计算出载荷平均数据速率与没收到数据间隔的平均包速率，两者比值给出平均载荷大小。如假设包丢失与包大小无关，那么特殊接收者收到的包数量给出此接收者收到的表观流量。

接收到 RTCP RR 包后，发送者处理步骤如下：

1）RTCP 分析：分析统计包丢失率、包延迟抖动、包的往返时间。

2）网络状态评估：可根据 RTCP 分析结果，对发送方的发送速度进行调整。

3. RTCP 的工作流程

RTCP 为了检测数据流的性能和质量，它周期地把服务质量、传输数据的端口和统计信息等传输情况报告发送给所有参与者。会话中的所有参与者都发送 RTCP 数据包，发送者发送发送者记录 SR，其中包含数据包的数量和字节数，以及用于同步不同会话媒体流的信息。会话参与者周期性地发送接受者记录 RR，RR 包含丢失数据包数目，接收的最大顺序号和用于估计发送和接受者之间的延时的信息。所有复合的 RTCP 控制包都包含源描述（SDES），SDES 包含了源的规范名（CNAME），源描述中还可以附加一些信息，如源名、电子邮件地址、电话号码、应用程序名和一些关于源信息的描述。如果源不再活跃，它将发送 RTCP BYE，BYE 数据包中包括了源离开会话的原因。RTCP APP 数据包为应用程序提供了机制通过 RTP 控制端口定义和发送自定义的信息。RTCP 在会话中定期的发送，下层协议一般用 UDP 和 RTP 使用不同端口号。

4. RTCP 传输间隔

计算 RTCP 包的传输间隔依赖于连接数量的增加，连接数量的确定是根据对 SSRC 或 CSRC 的处理来进行的，其过程如下：

1）当新地址被监听到，就进行计数，并在 SSRC 或 CSRC 索引表中为之建立一个条目，用来进行跟踪。

2）收到带有多个以此 SSRC 为标识的包，则标志新条目有效。

3）但当收到具有相应 SSRC 的 RTCP 的 BYE 包，条目可从表中删除。

4）如果对少量 RTCP 报告间隔没有接收到 RTP 或 RTCP 包，参加者可能将另外一个地址标记成不活动，如还未生效就删除掉。

为了“超时”正常工作，所有地址必须对 RTCP 报告最大间隔记入大致相同的值。

一旦确认地址有效，即使后来标记成不活动，地址的状态也仍保留，也还继续计入共享 RTCP 带宽地址的总数中，最大间隔时间要保证能扫描典型网络分区，建议为 30min。注意，这仍大于 RTCP 报告间隔最大值的五倍。

对于 SDES 包来讲，这个包结构定义了除必需的 CNAME 外的几个源描述项，如 NAME（人名）和 E-mail。它也是定义新特定应用 RTCP 包类型的途径。

给附加信息分配控制带宽应引起注意，因为它将降低接收报告和 CNAME 发送的速率而损害协议的性能。建议分配给单个参加者用于携带附加信息的 RTCP 带宽不要超过 20%。而且并没有有意让所有 SDES 项包含在每个应用中。

5. RTCP 的带宽

RTCP 的速率是软交换实现中要考虑的一个问题。在 RTP 的设计里，允许连接数达到从几个到上千个，并便于能进行自动扩展。如在一般音频会话中，数据流量是受内在限制的，因为同一时刻只有一两个人说话；对音频会议中的组播，给定连接数据率仍是常数，但控制流量不是受限制的。如果每个参加者以固定速率发送接收报告，控制流量将随参加者数量线性增加，因此，速率必须按比例下降。

在会议通信中，假定数据量的总和受会议带宽的限制，该带宽由各成员共同分配。连接带宽是由费用或网络带宽进行限制的，与媒体编码无关，但编码选择会受到连接带宽的限制。当进行一个具体的媒体应用时，连接带宽参数由应用自己提供，但它也可根据单个发送者的数据带宽设置默认值，可以被预留也可强制限制带宽。

控制和数据流量的带宽计算包括底层传输和网络协议（如 UDP 和 IP），因为这是资源预留系统所要知道的，应用程序还需要知道正在使用哪种协议，数据链路层数据头不包括在此计算中，这是因为在传输过程中会使用不同的数据链路层协议。

在传输途中，因为包将包含不同连接层的包头，所以连接层的包头不计算在内。控制流量应限制为连接带宽中已知的一小部分，这样数据才不致受损。建议分配给 RTCP 的连接带宽固定为连接带宽的 5%，而这个数值与间隔计算中其他常量并不重要，连接中所有参加者必须使用相同数值，因此使用相同间隔计算。在协议的实现中所用的算法可以满足上述的目标。计算发送 RTCP 复合包之间的间隔，以在各成员间分配所允许的控制带宽。这允许在小型会议（其中对成员的识别是相对重要的）里快速响应，也可以自动适应大型会议。此算法包含以下特性：

1）所有的发送源可以共享至少 1/4 的控制用带宽，这样即使在有多个接收方但是发送方很少的会议里，新加入的会议成员也可以快速收到发送源的 CNAME。

2）为避免在小型会议里 RTCP 突发包超出所允许的带宽，影响业务量的平滑性，RTCP 包之间的发送间隔必须大于 5s。

3）RTCP 包发送间隔以计算出的间隔乘以在 0.5 ~ 1.5 范围内变化的随机值，以避免所有成员不期望的同步。为防止多点同时启动，加入会议后发送的第一个 RTCP 包也要延迟大约 RTCP 最小间隔一半的随机值。

4）动态估计 RTCP 复合包平均大小，包括所有发送的和接收的，以自动适应被传输的控制信息的数量改变。

此算法用于允许所有成员发送的会议里，此时会议带宽是指单个发送方带宽和成员数目的乘积，而推荐 RTCP 带宽是该值的 5%。

2.3 信令网关技术

信令技术属于电信网的三大支撑技术之一，No.7 信令技术作为目前电信网中的主流技术一直以来更是具有举足轻重的地位。随着 IP 技术的飞速发展及信息化程度的不断提高，人们的生活习惯及通信行为在不断改变，利用 IP 网实现各种业务已经成为通信业的大势所趋。目前通信行业中，软交换技术是业界公认的下一代通信网络的演进方向，该体系通过 IP 分组技术使得原来采用独立技术组网的 PSTN、数据网及移动网得以真正的融合，通过各种终端接入的用户可以享有统一业务及特征，因此它的出现得到业界广泛的关注，随着软交换技术的完善和发展，越来越多的运营商对该技术进行了试验及商用。信令网关是 No.7 信令网与 IP 网的边缘接收和发送信令消息的信令代理，对信令消息进行中继、翻译或终结处理。其实质就是为了实现 PSTN 端与软交换设备的 No.7 信令互通，由其实现信令承载层电路交换形式与 IP 形式的转换功能。

2.3.1 No.7 信令系统

在中国国内 SS7（Signaling System No.7）被叫作 No.7（读作七号）信令系统。No.7 信令系统协议一开始就是按分层结构的思想设计的，但 No.7 信令协议在开始发展时，主要是考虑在数字电话网和采用电路交换方式的数据通信网中传送各种与电路有关的信息，所以 ITU 在 20 世纪 80 年代提出的 No.7 技术规范黄皮书中对 No.7 信令协议的分层方法没有和开放系统互连（OSI）的 7 层参考模型取得一致，对 No.7 协议只提出了 4 个功能层的要求。这 4 个功能层如下：

1）物理层：就是底层，具体是 DS0 或 V.35。

2）数据链路层：在两个节点间提供可靠的通信。

3）网络层：提供消息发送的路由选择。

4）用户部分/应用部分：就是数据库事务处理，呼叫建立和释放。

但随着综合业务数字网（ISDN）和智能网的发展，不仅需要传送与电路有关的消息，而且需要传送与电路无关的端到端的消息，例如用于维护管理，面向 ISDN 用户之间的端对端的信息等，因此原来的 4 层结构已不能满足要求。于是在 1984 年的红皮书中增加 SCCP（信令连接控制部分）满足面向连接和无连接端到端信息传递的要求。但随着 ISDN、智能网以及其他一些业务的发展，仅

增加 SCCP 仍显不足，于是在 1988 年的蓝皮书协议中又增加了事务处理能力（TC）及其应用部分（TCAP）等一些内容，目的是增强信息传送的能力。蓝皮书对 No. 7 信令协议提出了双重要求：一方面是对原来 4 个功能级的要求，另一方面是使 No. 7 协议的分层结构尽量向 OSI 的 7 层参考模型靠近。No. 7 信令协议栈结构如图 2-8 所示。

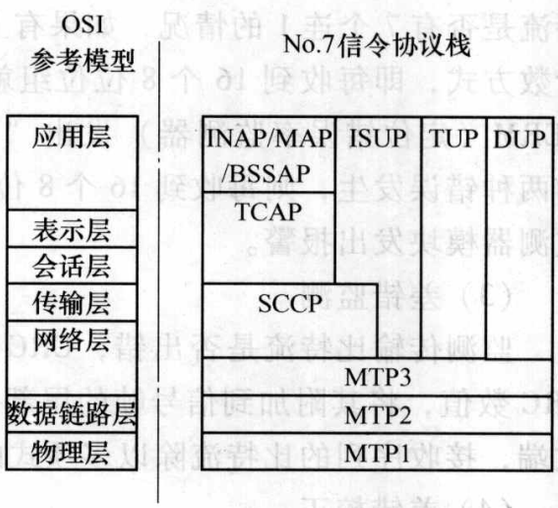


图 2-8　No. 7 信令协议栈结构

在整个 No. 7 信令协议栈中：

1. MTP

MTP 为消息传递部分，它的主要功能是提供一个可靠的传递系统，保证两个节点传递的信令消息无差错，不丢失，不错序，不重复。

1）MTP Layer1（消息传递部分第一层，简称 MTP1）：为物理层。是 No. 7 信令协议栈中的最底层，对应于 OSI 参考模型中的物理层，这一层定义了网络物理上电气上及功能上的特性。物理接口的定义包括：E1，T1，DS－1，V. 35 等。

2）MTP Layer2（消息传递部分第二层，简称 MTP2）：为数据链路层。确保信令消息在链路上实现精确的端到端传送。MTP2 提供流控制、消息编号、差错检查等功能。当传送出错时，出错的消息会被重发。MTP2 对应 OSI 参考模型中的数据链路层。具体功能如下：

（1）信号单元定界

首先介绍一个概念“信号单元分界”，它是指用标志码作为信号单元的开始和结束。结尾的标志码通常又是下一个信号单元的开始标志码。标志码的码型固定为 01111110（7E）。

从数据流到信号单元，实际上就是通过识别“7E”码型来提取帧数据；在实施这种操作时，接收部分需要在识别“7E”之后对信号数据去 0，即每逢 5 个 1 就抽去后面的 0；发送部分需要在加入“7E”之前对信号数据进行插 0，即每逢 5 个 1 插入一个 0。

（2）信号单元定位

判别开通业务的信令链路是否失去定位，如果失去，将转入信号单元差错率监视过程；在开通业务的信令链路上与定界密切相关的定位：8 的整数倍问题，在去标志和去 0 之后，每帧的比特数应该是 8 的整数倍；8 位计数和 7 个连 1 的检测：比特计数器计算比特流数是否 8 的整数倍，8 位计数器计算 8 位位组数目是否超过 $m+6$，其中 m 是信号单元最大 8 位数（272），并且监视接收的比

特流是否有 7 个连 1 的情况，如果有上述的两种情况之一发生，则进入 8 位位组计数方式，即每收到 16 个 8 位位组就向 SUERM（信号单元错误率监视器）和 AERM（定位错误率监测器）发出“受到错误信号单元”消息；如果没有上述的两种错误发生，则每收到 16 个 8 位位组就输出一次，并不再向上述两个错误监测器模块发出报警。

（3）差错监测

监测传输比特流是否出错，CRC 的计算是将比特流除以多项式，余数即为 CRC 数值，将其附加到信号帧的尾部；No. 7 信令加入了一些特别的操作，在接收端，接收序列的比特流除以多项式的余数不是 0，而是 0001 1101 0000 1111。

（4）差错校正

出现差错之后用来重新获得正确的信号单元方法。有基本差错校正（Basic Error Correction，BEC）方法和预防循环重发校正（Preventive Cyclic Retransmission，PCR）方法。

- 对于基本差错校正方法：

非互控；

肯定证实；

否定证实；

重发纠错。

- 预防循环重发校正方法：

与 BEC 不同的是，如果没有新的消息信号到达，则不断重发在重发缓存中尚未证实的消息信号单元；强制重发过程：监视未经证实的消息信号单元数 N1 和未经证实的消息信号单元的 8 位位组数 N2，如果它们超过极限，则强制停止发出新的消息信号单元，开始强制重发，如果重发一遍之后，N1 和 N2 数量恢复正常，则继续正常的 PCR 过程，否则，所有信号单元需要继续重新发出，直到 N1 和 N2 恢复正常为止。

3）MTP Layer3（消息传递部分第三层，简称 MTP3）：即网络层，总体上讲，它的功能分为两大类：信令消息处理功能和信令网络管理功能。在 No. 7 信令网中提供两个 SP（Signalling Point，信令点）间消息的路由选择功能，信令消息在依次通过 MTP1、MTP2、MTP3 之后，可能会被发送到别的信令点，也可能会传递给某个应用层，如 SCCP 或 ISUP 层。MTP3 还提供一些网管功能的支持，包括：流量控制，路由选择和链路管理。MTP3 对应 OSI 参考模型中的网络层。结构如图 2-9 所示。

2. SCCP

SCCP（Signaling Connection Control Part）为信令连接控制部分，位于 MTP 之上，为 MTP 提供附加功能，以便通过 No. 7 信令网在 SP 之间传递电路相关和

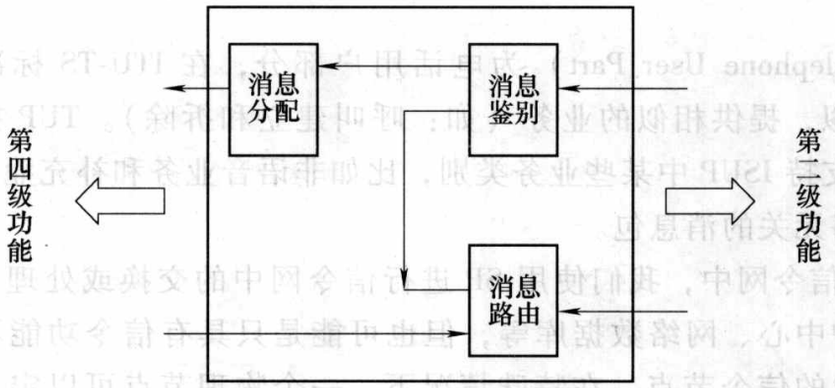


图 2-9　MTP3 的功能

非电路相关的消息，提供两类业务：

1）无连接业务是指在两个应用实体间，不需要建立逻辑连接就可以传递信令数据；

2）面向连接的业务在数据传递之前应用实体之间必须先建立连接，可以是一般性的连接，也可以是逻辑连接。

SCCP 以全局码（GT）的形式扩展 No. 7 信令协议的寻址能力和路由能力，这些扩展基于被叫号码的寻址信息。它与 MTP 一起形成网络服务部分（NSP）。

3. TCAP

TCAP（Transaction Capabilities Application Part）为事务处理应用部分，允许应用调用远端 SP 的一个或多个操作，并返回操作的结果。比如：数据库访问或远端调用处理命令等。使用 SCCP 无连接业务（基本的或有序的），TCAP 在两个用户应用之间提供事务处理对话。

4. INAP

INAP（Intellgent Network Application Part）为智能网应用部分，INAP 负责业务交换点（SSP）和业务控制点（SCP）之间的信息传递，用于智能网业务。

5. BSSAP

BSSAP（Base Station Subsystem Application Part）为基站子系统应用部分，BSSAP 用于移动通信网中的 A 接口，负责移动交换局（MSC）和基站（BS）之间的信息传递。

6. MAP

MAP（Mobile Application Part）为移动应用部分，MAP 主要用于移动通信网中的 C 接口，负责移动交换局（MSC）和归属者位置登记器（HLR）之间的信息传递。

7. ISUP

ISUP（ISDN User Part）为 ISDN 用户部分，在交换局提供基于电路的连接，它直接和 MTP3 层通信。ISUP 提供基础电信业务，包括连接建立、监示和释放。

8. TUP

TUP（Telephone User Part）为电话用户部分，在 ITU-TS 标准里，TUP 和 ISUP 功能相似，提供相似的业务（如：呼叫建立和拆除）。TUP 提供的业务比 ISUP 少，不支持 ISUP 中某些业务类别，比如非语音业务和补充业务，还有 TUP 不传递与电路无关的消息包。

在 No. 7 信令网中，我们使用 SP 进行信令网中的交换或处理工作，如交换局、操作维护中心、网络数据库等，但也可能是只具有信令功能不具备通信业务功能的独立的信令节点。在特殊情况下，一个物理节点可以定义为逻辑上分离的两个信令点。例如，国际出入口局既是国内信令网中的一个信令点，又是国际信令网的一个信令点，常称为网关（Gateway）点。

另外，在 No. 7 信令网中某信令点既非消息源点，又非消息目的地点，其作用仅是将消息从一条信令链路转发到另外一条信令链路上去，而并不处理这些消息，则该信令点为信令转接点（Signalling Transfer Ponit，STP）表示。STP 可以是交换机的一部分，也可以单独存在，当然，有一些 SP 也具备 STP 的功能 。

No. 7 信令通过“信令链路”传递两个信令点间的信令消息，直接连接两个信令点的一束信令链路构成一个信令链路组（ Link Set）。由于 SLC（Signalling Link Code）只有 4bit，每两个信令点之间最多建立 16 条信令链路（Link），即每个 Link Set 最多有 16 条链路。通常一个信令链路组包括两个信令点之间全部平行的信令链路，但不是绝对的，两个信令点之间可以有多个信令链路。同一个链路组里，具有某些相同特征的链路（例如：速率相同）称为链路群（Link Group）。

2.3.2 SCCP 简介

SCCP 能传递与呼叫控制无关的消息，如：网管中心之间的信息、ISDN 补充业务的端到端信令消息和智能网里业务交换点和业务控制点之间的信令信息等。同时它具有较强的寻址选路功能，可以在全球互连的不同的 No. 7 信令网之间实现信令的直接传输，并增加全局码（Globle Title）、子系统号（Subsystem Number）寻址方式。因此，SCCP 为 MTP 提供了附加功能，实现了在任意两个信令点之间透明传递信令信息和其他类型的信息，建立无连接和面向连接的网络业务。SCCP 具体有如下 4 类协议：

1）基本的无连接型业务。这种业务以数据报文的方式传递消息，每个消息都带有地址和路由信息，在网络中可以独立传送。该业务在传递的过程中，SLS 码随机插入，因此不保证消息以正确的顺序到达目的地。

2）有序的无连接型业务。在这种业务中，消息仍以数据报文的方式传送，但 SCCP 负责进行顺序控制。

3）基本的面向连接型业务。这种业务可以在两个信令点之间建立虚电路连

接，虚电路建立后，和这个连接有关的信令消息全部在同一路由上传送，从而保证了消息的顺序不变。

4）带有流量控制的面向连接型业务。这类业务在第二类业务的基础上增加了流量控制的功能。同时，它还具有一定的差错检测功能。

2.3.3　信令网关概述

一般意义上的信令网关（SG）主要功能是能够接收和发送 No. 7 信令消息，具有 IP 网边缘的信令代理功能，能对信令消息进行中继、翻译或终结处理，能实现信令承载层 TDM 形式与 IP 形式的转换功能。信令网关的一般网络结构如图 2-10 所示。

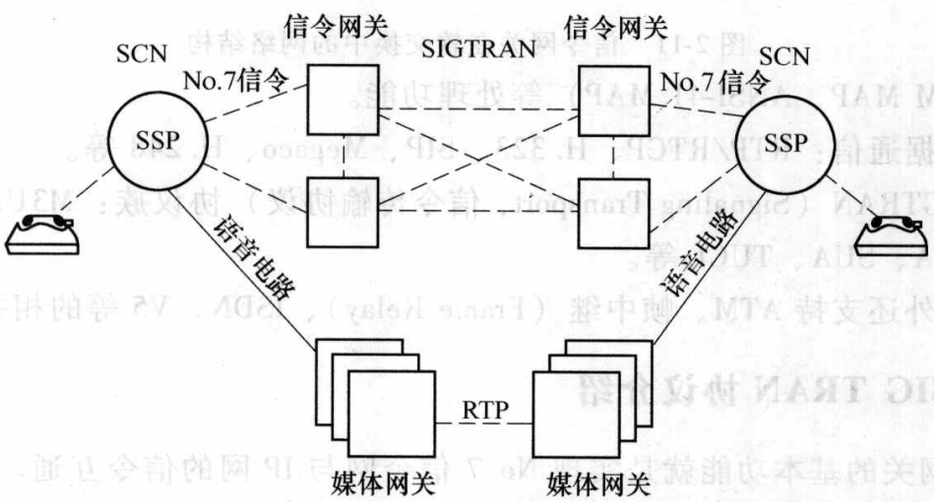


图 2-10　信令网关的一般网络结构

在软交换中，它的位置是接入层，它是软交换体系结构中的重要功能实体，主要实现完成 No. 7 信令到 SIGTRAN 信令的转换，实现软交换系统与现有 PSTN/ISDN 的互通。其实质就是为了实现信令网端局与软交换设备的 No. 7 信令互通。信令网关在软交换中的网络结构如图 2-11 所示。

目前信令网关的功能日趋丰富，信令网关可以为智能网、语音信箱、短消息中心、VoIP、全球漫游、基于位置的服务、预付费、移动互联网等电信新业务和增值业务的开发和部署提供信令网络的访问能力和信令的互通功能。例如信令网关可以部署于短消息中心，提供与移动通信网中其他的 No. 7 信令点（如 MSC、HLR 信令）互通的能力。

同时，信令网关的协议转换范围的扩大，也是信令网关发展的又一个趋势，传统信令网关对 SCN No. 7 信令和 IP 网 SIGTRAN 信令协议互通的支持如今已发展到对 SIP、H. 323 等多信令协议互通的支持，广泛的协议栈支持能力包括：

1）No. 7 信令：支持 ANSI、ITU、中国、日本等 No. 7 信令变体，提供基本信令（MTP、SCCP）、呼叫控制（ISUP、IUP、TUP）、面向业务的消息（TCAP、

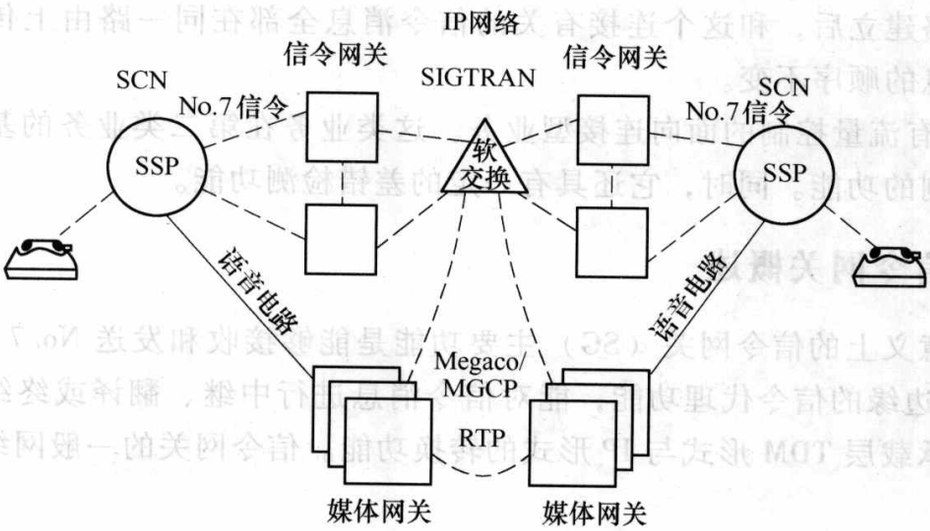


图 2-11 信令网关在软交换中的网络结构

INAP、GSM MAP、ANSI-41 MAP）等处理功能。

2）数据通信：RTP/RTCP、H.323、SIP、Megaco、H.248 等。

3）SIGTRAN（Signaling Transport，信令传输协议）协议族：M3UA、M2UA、M2PA、IUA、SUA、TUCL 等。

4）另外还支持 ATM、帧中继（Frame Relay）、ISDN、V5 等的相关协议族。

2.3.4 SIG TRAN 协议介绍

信令网关的基本功能就是实现 No.7 信令网与 IP 网的信令互通。No.7 信令网与 IP 网的信令互通主要采用 SIGTRAN 协议族中的协议进行实现。

IETF SIGTRAN 工作组于 1999 年 12 月就已经制定了第 2 版的 M2PA 草案，并于 2000 年 10 月制定了 SCTP 的标准 RFC2960，比较充分地满足了在 IP 网络上点到点传送 No.7 信令的应用需求。在之后的两年中，IETF SIGTRAN 工作组将重点放在完善 M3UA 和 M2UA 上，使它们能适应信令网关（即 TDM 到 IP 的转换）的应用需求，而这期间 M2PA 则相对稳定。

对于 No.7 信令应用层而言，最关心的是 No.7 信令向它们提供的接口，而对 No.7 信令的运作并不关心，也就是说 No.7 信令网对上层用户保持透明性。因此在 IP 网络基于 IP 传送 No.7 信令最关键是实现 No.7 信令协议栈中低层协议和 IP 的结合，同时保持到应用层接口与传统 No.7 信令提供的接口的一致性。在 SIGTRAN 中采用的体系结构如图 2-12 所示。

图 2-12 SIGTRAN 体系结构

在图 2-13 的中，SIGTRAN 引入了 SCTP（Stream Control Transmission Protocol，流控制传输协议）实现基于 IP 网上的信令传送；同时又引入多种适配模块以保持到用户层接口的

一致性。但在特殊情况下可能采用 TCP 作为低层通用传送协议。另外，针对 No. 7 信令协议的各个高层协议，如：ISUP、TUP、SCCP 等，在 SIGTRAN 协议栈中相应的有 M2UA（MTP2 User Adaptation）、M3UA（MTP3 User Adaptation）、SUA（SCCP User Adaptation），另外为了保持对传统 UNI 信令 Q. 931 的支持，在 SIGTRAN 中还引入了 IUA 作为 SCTP 与传统的 Q. 931 传送协议接口之间的适配层如图 2-13 所示。

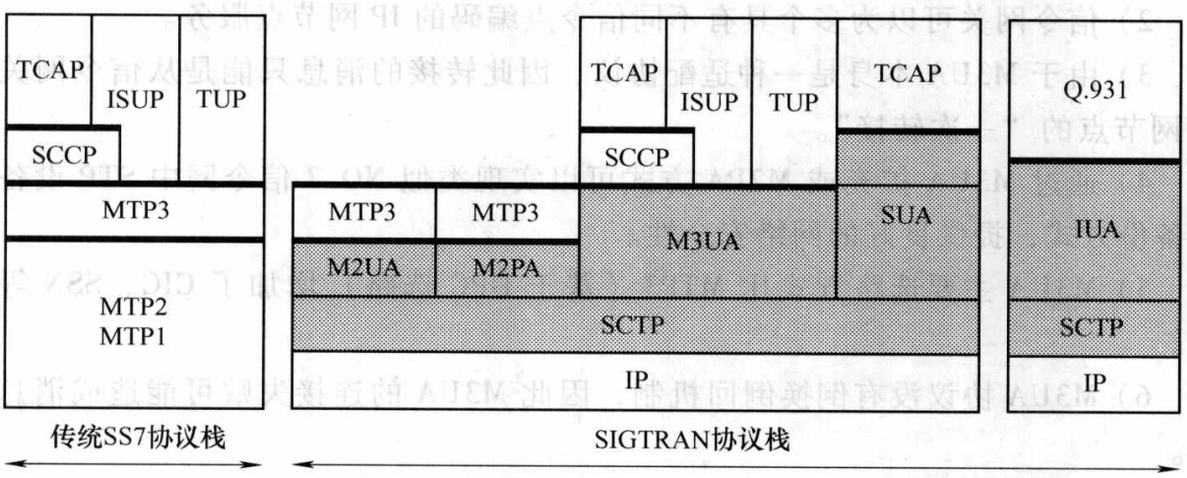


图 2-13　SIGTRAN 协议栈的体系结构

在上图中相同层面上，No. 7 信令的接口与 SIGTRAN 的接口一致。

1）M2UA 与 M2PA 向上提供标准的 MTP2 接口；

2）M3UA 向上提供标准的 MTP3 接口；

3）SUA 向上提供标准的 SCCP 接口；

4）IUA 向上提供标准的 Q. 921 接口。

对于 M2UA、M2PA、M3UA、SUA 以及 IUA 之上的应用协议，SIGTRAN 的传输是透明的。信令网关设备的组网应用主要分为信令点代理的组网应用和信令转接点组网应用两种。

（1）使用代理方式时的信令网关的组网能力

在此种工作方式中，信令网关采用 M3UA 协议，信令网关与所连的 IP 网中的节点使用相同的信令点码，即 IP 网中的节点是信令网关的一个高层用户。在这种情况下，IP 网节点只需具有 MTP3 以上的协议。但这种方式存在不足，由于只有一个信令点码，所以网络不提供经另一信令网关到 IP 网节点的功能，另外，从信令网关到 IP 目的地点通路的状态无法通过合适的 No. 7 信令消息反馈回 No. 7 信令网。由于代理的特性，IP 网的一个节点只能通过固定的信令网关和 No. 7 信令网交互。如果信令网关需要为多个 IP 网节点服务，则需要网关支持多信令点功能，针对不同的 IP 网节点分配不同的信令点码。

（2）使用 STP 方式的信令网关的组网能力

信令网关可具有 STP 功能，这是目前业界信令网关设备采用的普遍方式。信令网关采用 STP 方式工作时，要单独占用一个信令点码，而 IP 网中的节点能有不同的信令点码，从 No.7 信令网收到的信令消息可以向 IP 网内不同的信令点转发，使用协议可以是 M3UA 或 M2PA。

如果信令网关使用 M3UA 协议的转接方式，则整个 IP 网具有如下特征：

1）IP 网节点（或称信令点）不需要具备 MTP3 和其下层的窄带协议；

2）信令网关可以为多个具有不同信令点编码的 IP 网节点服务。

3）由于 M3UA 本身是一种适配协议，因此转接的消息只能是从信令网关到 IP 网节点的“一次转接”。

4）通过 M3UA 扩充或 M2PA 方式可以实现类似 NO.7 信令网中 STP 设备配对备份方式，提供良好的网络安全性。

5）M3UA 主要选路方式比 MTP3（基于 DPC 选路）增加了 CIC、SSN 等方式。

6）M3UA 协议没有倒换倒回机制，因此 M3UA 的连接失败可能造成消息丢失。

如果信令网关使用 M2PA 协议，则整个 IP 网具有如下特征：

1）和 M3UA 协议不同，M2PA 协议是一种对等适配协议，即 M2PA 和 SCTP 一起实现 MTP2 的功能。使用 M2PA，相当于更换了 NO.7 信令的链路层，因此使用 M2PA 的信令网关组网方式，类似于 No.7 信令网中 STP 设备的方式。

2）IP 网节点需要具备 MTP3 协议。

3）这种方式的信令网关可以为多个具有不同信令点编码的 IP 网节点服务。

4）由于 M2PA 是对等的适配协议，信令网关通过 MTP3 层进行信令转接，因此消息可以在 IP 网进行多次转接。

5）信令网关可以实现类似 NO.7 信令网中 STP 设备的冗余备份方式，提供良好的网络安全性。

6）网络的可管理性强。由于使用 M2PA，相当于更换了 NO.7 信令的链路层，因此对信令路由的管理与 NO.7 信令网中的管理一样。

7）由于 MTP3 协议中规定有信令网管理功能，并且 M2PA 协议有比较完备的倒换倒回过程，因此 M2PA 协议相对于 M3UA 协议，在信令网关做 IP 网中的 STP 设备时具有相当的优势，既保证了承载层带宽增加，具有 IP 特性的同时，又可以继承窄带信令网安全可靠的特征。

由于业界主要考虑将 SG 用于 TDM（Ttime Division Multiplexing，时分多路复用）与 IP 的信令格式转换，而非主要发展其 STP 转接功能，因此目前主要应用实例都为 M3UA，而 M2PA 发展相对缓慢。若 SG 的转接功能要求增强，则应加快 M2PA 协议的完善过程。

2.3.5　SCTP

1. SCTP 介绍

在过去的 20 年内，传统 Internet 网中 TCP 在网络传输层上提供了许多应用服务，但随着 IP 网的多业务化，尤其是 VoIP 的发展，TCP 也暴露出很多局限性。因此，IEFT 的信令传输工作组（SIGTRAN）制定了新的面向多媒体通信的流控制传输协议（SCTP），用于在无连接的 IP 网络上传输 PSTN 信令消息，即通常所说的 No. 7 over IP。目前，IEFT 将 SCTP 传输层协议作为主要研究目的，与 TCP 和 UDP 共筑于 IP 层之上。同 TCP 一样，SCTP 提供面向连接的、点到点的可靠传输，它继承了 TCP 强大的拥塞控制、数据包丢失发现等功能，并提供对 TCP 的兼容性。它能在确认方式下，无差错、无重复传送用户数据；根据通路的 MTU 的限制，进行用户数据的分段；并在多个流上保证用户消息的顺序递交；把多个用户的消息复用到 SCTP 的数据块中；利用 SCTP 内部机制来提供网络级的故障保证，同时 SCTP 还具有避免拥塞的特点和避免遭受泛播和匿名的攻击。

不同于 TCP 的是，SCTP 提供了许多利于信令传输的功能，同时，对于其他一些对性能和可靠性有额外需要的应用，也能提供传输业务来满足这些需要。SCTP 和 TCP 最大的区别在于 SCTP 对多流性（Multi-Streaming）和多宿性（Multi-Homing）的支持。

（1）多流性

SCTP 的名称源于其“多流”功能。这种功能允许数据被划分多个流，这些流具有独立排序的特性，因此任何流中的任何消息的丢失将只会影响那个流的传送，而不会影响其他的流；而 TCP 只使用一个数据流，并保证了此数据流的传送，并保证其传送的顺序。对于电话信令而言，就不需要严格保证同一呼叫的消息的顺序。但同时，SCTP 的传送又属于同一个 SCTP 偶联（association），因此所有的流（stream）又服从于同一数据流（flow）和拥塞控制机制，在传送层降低了开销。

SCTP 通过创建“data transmission”与“data delivery”间的独立性实现了“多流”特性。每一个消息由 TSN 和 SSN 来确定其“多流”特性：

1）TSN（Transmission Sequence Number，传输顺序号）：在 SCTP 一个偶联的一端为本端发送的每个数据块顺序分配一个基于初始 TSN 的 32 位顺序号，以便对端收到时进行确认。TSN 是基于偶联维护的。

2）SSN（Stream Sequence Number，流顺序号）：在 SCTP 一个偶联的每个流内，为本端在这个流中发送的每个数据块顺序分配一个 16 位顺序号，以便保证流内的顺序传递。SSN 是基于流维护的。

TSN 和 SSN 的分配是相互独立的。

当每个净荷数据使用 TSN 和 SSN 两组数字后，TSN 就能用于控制消息传送和检测消息丢失，而 SSN 则可用于决定接收数据的传送顺序。

（2）多宿性

多宿性是指一个 SCTP 端点支持多个 IP 地址。但在一个端点对应于整个地址列表将使用惟一的 SCTP 端口（port），从而有利于进行特定会话。SCTP 端点地址结构如图 2-14 所示。

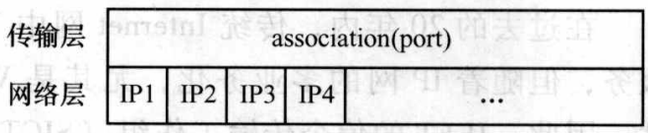


图 2-14　SCTP 端点地址结构

SCTP 的多流和多宿特性如图 2-15 所示。

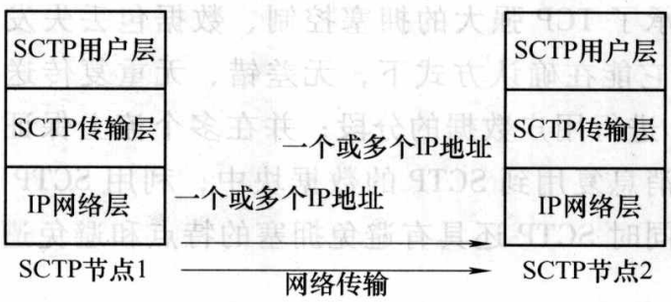


图 2-15　SCTP 的多流和多宿特性

2. SCTP 的特点及功能

SCTP 用“偶联”来定义交换信息的两个对等 SCTP 用户间的协议连接。SCTP 也是面向连接的，但在概念上，SCTP 的“连接”比 TCP 连接更为广泛：TCP 的连接只有一个源地址和一个目的地址，而 SCTP 则提供一种对“多宿性”方式的支持，使得每个 SCTP 端点能为另一个对等端点提供一组传输地址（即传输地址 = 一组 IP 地址 + 一个 SCTP 端口）。其特点如下：

1）一种新的 IP 传送协议，与 TCP，UDP 处于同一层；

2）SCTP 是单播协议，支持两个端点之间的数据交换；

3）采用面向连接的机制，意味着 SCTP 两个端点之间的关系先于数据传送建立，且此关系将保持一致至所有数据传送完成；

4）无差错，非复制，按顺序的用户数据传送；

5）SCTP 提供了可靠的传送机制；

6）SCTP 是全双工的；

7）数据拆分，以满足发现的通道 MTU 大小，SCTP 同 TCP 一样，会根据路径的 MTU 对用户消息进行分片，在接收端，分片被重组后传给上层 SCTP 用户；

8）SCTP 是面向消息的，支持对单独的消息定界，而 TCP 是面向比特的，在传送的比特流中不保存任何不明确的结构以作为增强；

9）SCTP 与 TCP 一样是速率可适应的，会根据当前网络负荷状态调整数据传送；

10）支持“多流性”；

11）支持“多宿性”，实现网络级的容错，当 SCTP 传送数据包给目的 IP 地址时，如果此 IP 地址是不可到达的，SCTP 可以将消息重路由给一个交替的 IP 地址。这样，在关联的一端甚至两端，可容忍网络级错误；

12）支持将多个用户消息捆绑在同一 SCTP 块（chunk）中，即多个用户消息可选择地绑定到一个 SCTP 包上，通过将消息放到一个或多个 SCTP 数据结构——“块”中，SCTP 保留了应用程序的消息框架边界，不同类型的块可绑定到一个 SCTP 包中，但是控制块必须放在任何一个数据块之前。

SCTP 的功能框架如图 2-16 所示。

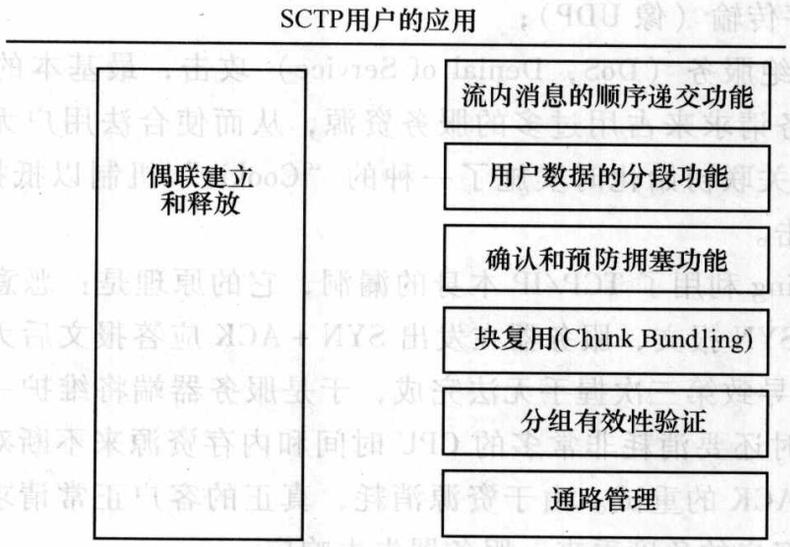


图 2-16　SCTP 的功能框架

偶联是由 SCTP 用户发起请求来启动的。出于安全的目的，为了避免遭受攻击，在偶联的启动过程中采用了 Cookie 机制。同时提供了一个对激活偶联的完善关闭程序，它必须根据 SCTP 用户的请求来执行，当然 SCTP 也提供一种非完善的关闭程序，但 SCTP 不支持半打开状态（类似 TCP），即一端可以在另一端结束后继续发送数据当一个端点执行了关闭程序后。对于完善的关闭程序，当一端点执行了关闭后，在偶联的两端应停止接收从用户来的新数据，并只递交队列中的数据的。

以下我们将进行 SCTP 同 TCP 的比较，SCTP 所具备的额外功能为：

1）多流性特点；

2）支持多宿；

3）根据路径 MTU（最大传输单元）大小对消息进行数据分片；

4）选择性确认（SACK）和拥塞控制，选择性确认用于数据包丢失时，SCTP 反馈给发送端的是丢失的并且要求重传的消息序号，这比 TCP 具有更高的灵活性，另外，SCTP 还运用了 TCP 中的拥塞控制技术，包括慢启动，拥塞避免

和快速重传；

5）块（chunk）绑定；

6）路径管理，SCTP 路径管理功能主要负责从远端提供的一组传输地址中选择目的传输地址，它根据两个方面来选择目的地址：SCTP 用户指示和当前可达的合格目的地，当其他流控制不能提供可达性信息时，路径管理功能定时地扫描链路的可达性，并向 SCTP 报告远端传输地址所发生的变化，SCTP 路径管理功能模块同时还负责在建立链路时，向远端报告可用的本地地址，并把远端返回的传输地址告诉 SCTP 用户；

7）支持多种传输模式，严格有序传输（像 TCP），部分有序传输（像 Per-Stream）和无序传输（像 UDP）；

8）防范拒绝服务（DoS，Denial of Service）攻击，最基本的 DoS 攻击就是利用合理的服务请求来占用过多的服务资源，从而使合法用户无法得到服务的响应，SCTP 在关联初始化时实施了一种的“Cookie”机制以抵抗 SYN Flooding 对目标主机攻击。

SYN Flooding 利用了 TCP/IP 本身的漏洞，它的原理是：恶意的攻击者大量向服务器发送 SYN 报文，服务器在发出 SYN + ACK 应答报文后无法收到客户端的 ACK 报文，导致第三次握手无法完成，于是服务器端将维护一个非常大的半连接列表，同时还要消耗非常多的 CPU 时间和内存资源来不断对这个列表中的 IP 进行 SYN + ACK 的重试。由于资源消耗，真正的客户正常请求无法满足，所以此时从正常客户的角度看来，服务器失去响应。

但在一次 SCTP 四路握手中，INIT 消息的接收端不必保存任何状态信息或者分配任何资源，这样就可防范 SYN Flooding 等 DoS 攻击。它在发送 INIT-ACK 消息时，采用了一种机制——状态 Cookie，该 Cookie 具有发送端要建立自己状态所需的全部信息。

SCTP 产生一个状态 Cookie 的过程如下：

1）使用收到的 INIT 和发出的 INIT- ACK 块中的信息创建一个关联的 TCB（传输控制块）。

2）在 TCB 中，将当前日期设为创建日期，将协议参数“有效 Cookie 时间”设为生存期间。

3）根据 TCB，收集重建 TCB 所需的最小信息子集，将该子集和密钥产生一个 MAC（信息认证编码）。

4）结合上述最小信息子集和 MAC 产生状态 Cookie。

5）在发送完 INIT ACK（包含状态 Cookie 参数）后，发送方必须删除 TCB 以及任何与新关联有关的本地资源。

INIT 和 INIT-ACK 都必须包含建立初始状态所需的参数：

1）一组 IP 地址；

2）保证可靠传输的初始 TSN；

3）每个被接收的 SCTP 包中必须含有的初始标签；

4）每一端请求发出的流数目和每一端能支持接收的流数目。

交换完这些消息之后，INIT 的发送端以 COOKIE-ECHO 消息的方式发送回状态 Cookie。接收端根据所接收到的 COOKIE-ECHO 中的状态 Cookie，完整地重建自己的状态，并回送 COOKIE- ACK 来确认关联已建立。COOKIE-ECHO 和 COOKIE-ACK 都可将用户数据消息绑定到各自的包中。

由此可见，采用以上方式，即使接收再多的 INIT 消息，接收端也没有任何资源的消耗：它既不分配任何系统资源，也不保存此次新关联的状态，它只是把相应重建状态所用的状态 Cookie 作为参数，包含在每一个回送的 INIT-ACK 消息中，最后该状态 Cookie 会被 COOKIE-ECHO 消息发送回来。

3. SCTP 数据交换

在两个 SCTP 主机间的正常数据交换。SCTP 主机发送 SACK 块，用来确认每一个收到的 SCTP 包，因为 SACK 能完整地描述接收端的状态，因此，依据 SACK，发送端能做出重传判决。

4. SCTP 关闭关联

作为面向连接的传输协议，SCTP 也运用三路握手来关闭一个关联，但与 TCP 有一点不同：一个 TCP 终端在“关联关闭”的过程中能够保持连接开启，并从对端接收新的数据，而 SCTP 不支持 TCP 的这种“半关闭”状态。

1）主机 A 发出“关闭”（SHUTDOWN）块来终止与主机 B 的关联，主机 A 进入“SHUTDOWN- PENDING”状态，对应的动作是：不再接受上层接口发送的数据，只发送队列中剩余的数据，进入“SHUTDOWN-SENT”状态。

2）主机 B 一旦接收到“关闭”块，就进入“SHUTDOWN-RECEIVED”状态，同主机 A 一样，不再接受上层应用的数据，只发送队列中剩余的数据。

3）主机 A 再次发送“关闭”块，通知主机 B 所发送的剩余数据已到达，并且重申了关联正在关闭。

4）当第二次收到“关闭”块时，主机 B 发送“确认关闭”块。

5）主机 A 随后发送“关闭结束”块，完成本次关联的关闭。

5. SCTP 高层接口

SCTP 通过接口接收高层协议（SCTP 用户）发送的原语请求为 SCTP 的用户提供的业务，并且根据不同事件从 SCTP 向 SCTP 用户发送通知。

6. 由高层协议向 SCTP 发送的原语

（1）INITIALIZE（[local port]，[local eligible address list]）

返回结果：local SCTP instance name

这个原语允许 SCTP 启动其内部的数据结构，并为建立操作环境分配所需的资源，一旦 SCTP 启动，则高层协议就可以与其他 SCTP 端点直接通信，而不需要再调用该原语。SCTP 将向高层协议返回一个本地 SCTP 实例名。

（2）ASSOCIATE（local SCTP instance name，destination transPort addr，outbound stream count）

返回结果：association id［，destination transport addr lst］［，outbound stream count］

该原语用来由高层启动一个到特定端点的偶联。对等端点将按照在该端点定义的传送地址的方式进行通信，如果偶联成功建立则返回结果中还应包含到对端的完整的传送地址和已经到达本端点的输出流的数量；如果本地 SCTP 实例未启动或建立偶联失败，则认为该原语是一个错误。

（3）SHUTDOWN（association id）

返回结果：result

该原语用来完善地关闭一个偶联，任何用户层数据 SCTP 停止接收，同时将本地排队的用户数据发送到对端。该偶联将在收到所有发送的 SCTP 分组的证实后被中止。返回结果用来指示是否成功地关闭了该偶联。

（4）ABORT（association id［，cause codd］）

返回结果：result

该原语用来非完善地关闭一个偶联，本地排队的用户数据将被丢弃，并发送一个 ABORT 数据块到对端。返回结果用来指示是否成功地非完善地关闭了该偶联。

（5）SEND（association id；buffer address，byte count［，context］［，stream id］［，lifetime］［，destination transport address］L，unorder flag［，no－bundle flag］［，payload protocol－id］）

返回结果：result

SCTP 用户使用该原语通知 SCTP 发送数据，返回结果用来指示是否成功的发送了数据。

（6）SETPRIMARY（association id，destination transport address，［source transport address］）

返回结果：result

高层协议用该原语指示本地 SCTP 把给定的目的地传送地址作为发送 SCTP 分组的首选通路。该操作的返回结果用来指示这个操作是否成功执行。

（7）RECEIVE（association id，buffer address，buffer size［，stream id］）

返回结果：byte count［，transport addressj［，stream id］［，stream seouence number］［，partial flag］［，delivery number］［，payload protocol

- id]

该原语用来把在 SCTP 队列中的可用的第一个用户消息读到由高层协议规定的缓冲区中。所读消息的字节数将作为结果返回，如果有可能也可以返回其他信息。

如果 SCTP 队列中没有消息，则可以向上返回这种情况，或者是先不执行该操作，直到队列中有新消息为止。

(8) STATUS (association id)

返回结果：status data

该原语用来要求 SCTP 返回一个包含以下信息的数据块：

1) 偶联连接状态；

2) 目的地传送地址表；

3) 目的传送地址的可达性状态；

4) 当前的接收方窗口大小；

5) 当前的拥塞窗口大小；

6) 未确认的数据块的数量；

7) 收到的数据块的数量；

8) 首选通路、首选通路上最近收到的 SRTT；

9) 首选通路的 RTO 和其他目的地地址的 SRTT 和 RTO 等。

(9) CHANGEHEARTBEAT (association id, destination transport address, new state [, interval])

返回结果：result

高层协议用该原语指示本地端点允许或禁止向指定的目的地传送地址发送心跳消息。返回结果用来指示该操作的执行情况。如果处于允许情况下，如果目的传送地址未空闲，心跳程序也不执行。

(10) REQUESTHEARTBEAT (association id, destination transport address)

返回结果：result

高层协议用该原语指示本地端点执行并向给定偶联的特定目的地端传送心跳消息，返回结果用来指示传送给目的地地址的 HEARTBEAT 数据块是否成功。

(11) GETSRTTREPORT (association id, destination transport address)

返回结果：srtt result

高层协议用该原语指示本地 SCTP 报告对给定偶联上规定的目的地传送地址的当前 SRTT 测量值，返回结果应当是一个包含最近 SRTT 的毫秒值。

(12) SETFAILURETHRESHOLD (association id, destination transPort address, failure threshold) 返回返回结果：result

该原语允许本地制定可达性失败的最大重传检出门限，返回结果用来指示

该操作是否成功。

(13) SETPROTOCOLPARAMETERS (association id, [, destination transnort address,] protocol parameter list)

返回结果：result

该原语允许制定协议参数，返回结果用来指示该操作是否成功。

(14) RECEIVE _UNSENT (data retrievel id, buffer address; buffer size [, stream id] [, stream sequence number] [, partial flag] [, payloadprotocol -id])

返回结果：result

该原语允许用户接收未发消息。

(15) RECEIVE_UNACKED (data retrieval id, buffer address, bufter size, [, stream id]

[, stream sequence number] [, partial flag] [, payload protocol -id])

返回结果：result

该原语允许用户接收没有得到目的端应答的消息。

(16) DESTROY (local SCTP Instance name)

返回结果：result

该原语允许 SCTP 删除一个 SCTP 实例。

7. 由 SCTP 向高层协议发送的通知原语

(1) DATA ARRIVE notification

当一个用户消息被成功的接收，并且准备向高层传输时，SCTP 将用该原语通知高层协议。

可能传递的参数：关联标志号，流编号；

(2) SEND FAILURE notification

当一个消息不能被 SCTP 递交时，SCTP 将发送该原语通知高层协议。

可能传递的参数：关联标志号，待取回的数据的标志，原因代码，上下文；

(3) NETWORK STATUS CHANGE notification

当目的地传送地址被标记为未激活时（例如，当 SCTP 检测出故障），或标记为激活时（当 SCTP 检出故障恢复），SCTP 将用该原语通知高层协议。

必须传递的参数：关联标志号，目的端传送地址，新状态信息；

(4) COMMUNICATION UP notification

SCTP 用该原语通知高层协议本地 SCTP 已经准备好发送或接收用户数据，或者是一个丢失通信的端点又已经恢复。

必须传递的参数：关联标志号，状态信息，目的端传送地址列表，出局的流的数量，入局的流的数量；

（5）COMMUNICATION LOST notification

当 SCTP 完全丢失了到一个端点的通信时，或者是检出端点已经执行了中止操作，则 SCTP 将使用该原语通知高层协议。

必须传递的参数：关联标志号，状态信息；可能传递的参数：待取回的数据的标志，对端确认的最后一个数据块的 TSN，发往对端的最后一个数据块的 TSN；

（6）COMMUNICAT10N ERROR notification

当 SCTP 从对端收到了一个 ERROR 数据块，并且确定需要向高层协议通知时，才使用该原语。

可以传递的参数：关联标志号，错误信息；

（7）RESTART notification

当 SCTP 检出其对端已经重新启动时，则可以用该原语通知高层协议。

可以传递的参数：关联标志号；

（8）SHUTDOWN COMPLETE notification

当 SCTP 已经完成了关闭程序后，则用该原语通知高层协议。

可以传递的参数：关联标志号。

2.3.6　MTP3 协议

M3UA（MTP3-User Adaptation layer，MTP3 用户适配层）协议使用 SCTP 协议来支持基于 IP 网的 No. 7 信令中 MTP3 到上层用户信令的传输，同时此协议也被用于进行 No. 7 信令网中用户和 IP 网络中 MTP3 用户对等体之间的无缝操作。在 M3UA 信令转接方式下，信令网关的功能类似于信令转接点，尽管信令网关和软交换之间仍然传递的是 M3UA 消息封装的 MTP3 层间原语，由于信令网关具备独立的信令点，所以其上的 MTP3 层可以执行类似 STP 上的 MTP3 的动作，同时呈现给用户的是信令网关和软交换之间的 SCTP 连接等同于一条传统的信令链路，而用户对这条信道的管理是标准的信令链路管理。如果信令网关从标准的 No. 7 信令接口处得到 No. 7 信令消息，则其结构如图 2-17 所示。

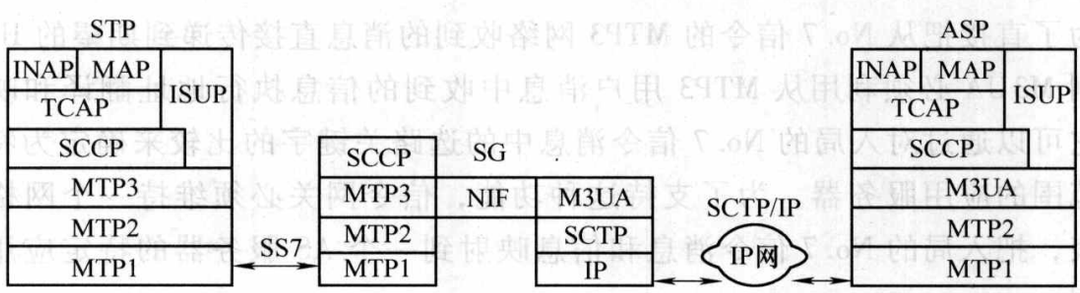


图 2-17　M3UA 协议在信令网关中的应用

其中：

1）ASP（Application Sever Process，应用服务器进程）。首先，先介绍一下

AS（Application Sever，应用服务器），AS 是为一个特定选路关键字（Routing Key）服务的逻辑实体，例如它可以是存在于某一个软交换逻辑组成部分上的虚拟交换单元，用于处理某一个特定 PSTN 中继范围内所有的呼叫进程，而这些呼叫可以由特定的 No. 7 信令 SIO/DPC/OPC 来识别。而一个 ASP 对应一个特定选路关键字。

而 AS 包含了一个或多个 ASP 的集合，ASP 是 AS 的一个进程实例。ASP 可以是应用服务器的激活态进程，也可以是备用态进程。网关控制器（MGC）、IP SCP 或 IP HLR 进程（或进程实例）就是 ASP 的例子。一个 ASP 包含一个 SCTP 端点，并可以配置成在一个以上的 AS 内处理信令流。

由特定的选路关键字标识。ASP 是 AS 的进程实例，由选路上下文（Routing Context）标识，用于处理与它的通信。AS 与 ASP 间并非映射关系，一个 AS 包含一组独立的 ASP，一个 ASP 也可为多个 AS 提供服务。每个 ASP 都对应一个 IP 信令点，为 SCTP 层提供服务。一个信令网关可以视为带有 No. 7 信令网接口的一组 AS。

2）NIF。NIF（Network Interface Fuction）为网络接口功能。对于信令网关来说，网络接口功能就是把从 MTP3 用户层选路发送到软交换侧的信令消息通过 M3UA 协议翻译解析，发送到特定的目的 IP 地址端。

3）信令网关的 SCCP 部分是可选部分。信令网关能与 No. 7 信令网 MTP3 协议层进行互通，并通过 SCTP 向软交换或 IP 本地数据库传送消息、SCCP 和/或任何其他类型 MTP3 用户协议消息以及特定的 MTP 网络管理事件。

1. M3UA 的功能

支持 No. 7 信令 MTP3 用户部分消息的传送（ISUP，SCCP，TUP）；M3UA 向上支持以下协议（仅限于这些协议）：ISUP，SCCP，TUP。TCAP 消息可以由 M3UA 协议作为 SCTP 的净荷透明地进行传送。

支持 MTP3 协议对等体的无缝操作，支持地址翻译和映射功能，对从 No. 7 信令网到 IP 网的信令传送的支持。

为了直接把从 No. 7 信令的 MTP3 网络收到的消息直接传递到期望的 IP 目的地，则 M3UA 必须利用从 MTP3 用户消息中收到的信息执行地址翻译和映射功能。它可以通过对入局的 No. 7 信令消息中的选路关键字的比较来确定为特定的业务范围的应用服务器。为了支持这种功能，信令网关必须维持一个网格地址翻译表，把入局的 No. 7 信令消息和信息映射到一个 AS 服务器的特定应用和业务范围。

No. 7 信令地址或路由信息中的路由关键字包括：OPC（源信令点编码）、DPC（目的地信令点编码）、SIO（业务信息字段）、ISUP CIC（电路标识码）和 SCCP SSN。M3UA 所使用的选路关键字应在上述关键字中选取，而具体的选取

结果取决于特定的应用和网络。当入局的 No. 7 消息不能与选路关键字匹配时，必须规定缺省的处理方法，如：提供缺省 AS，或舍弃这些消息并通告。

另外，AS 中包含用于处理业务的一个或多个 ASP 的列表。此列表是动态的，考虑到所有表中每个 ASP 的可用性状态，配置变化及可能的故障转移机制。M3UA 协议可以传送每个单独 ASP 的可用性状态信息，并可以根据此信息完成对路由的动态选取。

M3UA 本身具有以下管理能力：

1）在 SGP（信令网关进程），M3UA 提供了与 MTP3 管理功能的互操作以支持对 IP 域和 No. 7 信令网域中用户信令应用的无缝操作。

2）支持对 SGP 和 ASP 之间的 SCTP 关联的管理；

3）支持对 SG 和一个或多个软交换或是 IP 本地数据库之间的 SCTP 传送相关和业务的管理；

4）支持软交换或是 IP 本地数据库处理故障和负荷分担；

5）支持对状态改变的异步报告以进行管理；

6）支持对多个 SGP 的连接的管理，一个 ASP 可能被连接到多个 SGP 上；

7）在 SGP 的 M3UA 保持了所有配置的远端 ASP 的有效性，用以管理 SCTP 关联和 M3UA 对等体之间的业务。

2. M3UA 的服务

（1）支持传送 MTP3 用户消息

对 ASP 或是 IPSP（IP 信令点）的 MTP3 用户而言，底层消息传输具有透明性，位于 ASP 或 IPSP 的 ISUP 或 SCCP 并不知道所需的 MTP3 业务是来自远端 SGP，还是来自本地 MTP3 层。同样位于 SGP 的 MTP3 也不知道其用户是本地的，还是基于 M3UA 的远端用户部分。M3UA 提供了与 MTP3 同样的原语集和业务。当然，M3UA 在两个 IPSP 之间也可以使用点到点的信令，在特定的 MTP3 业务不是由远端 SGP 提供的情况下，MTP3 业务虽仍然被支持，但是支持这些业务的程序仅为 MTP3 的一个功能子集，这是因为 IPSP 到 IPSP 间点到点的特性所决定的。

（2）本地管理功能

M3UA 可以提供对低层 SCTP 传送协议的管理，同时还提供对收到的 M3UA 消息的相关错误指示能力，以及向本地管理或远端 M3UA 进行适当通知的能力。

（3）同 MTP3 网管功能的互通

M3UA 必须能提供同 MTP3 网管功能的互通，从而保证用户信令消息能够在 No. 7 信令和 IP 域中进行操作，这些包括：

1）向 ASP 的 MTP3 用户提供 No. 7 信令网中的远端目的地不可及的指示。

2）向 ASP 的 MTP3 用户提供 No. 7 信令网中的远端目的地可及的指示。

3）向 ASP 的 MTP3 用户提供指示，到 No. 7 信令网中的远端目的 MTP3 用户层的通信拥塞时，进行使用。

4）向 ASP 的 MTP3 用户提供 No. 7 信令网中的远端对等 MTP3 用户不可用的指示。

ASP 的 M3UA 可以启动对远端 No. 7 信令目的点的可用性和拥塞状态的查询，这些信息是从远端 M3UA 获得，同时还支持对远端和 ASP 间 SCTP 偶联的管理。

远端的 M3UA 维持所有配置的远端 ASP 的可用性状态，用于管理远端和 ASP 之间的 SCTP 偶联和业务，同样 ASP 的激活或未激活状态也由远端来维持，其中激活的 ASP 是那些正在从远端接收业务的 ASP。

3. M3UA 对信令网结构的要求

从运营角度讲，由于通信网要求具备高可靠性和高性能，所以就要求软交换的信令网关必须具备负载均衡能力，能够把从 No. 7 信令网收到的 MTP3 用户信令业务传送到多个分布的 ASP（例如软交换或 IP 数据库），以防止出现单点故障引起的系统崩溃。信令网关支持 M3UA 协议描述不能满足这种传送所要求的性能和可靠性，这时就要求物理网的网络结构具有特殊的冗余结构，其物理参考模型如图 2-18 所示。信令网关在这种特定的网络结构下，能够充分实现数据可用性和发挥物理节点的传送性能。M3UA 也提供了足够的灵活性从而允许其在不同的物理配置上操作和管理，进而满足网络运营商的性能和可靠性要求。

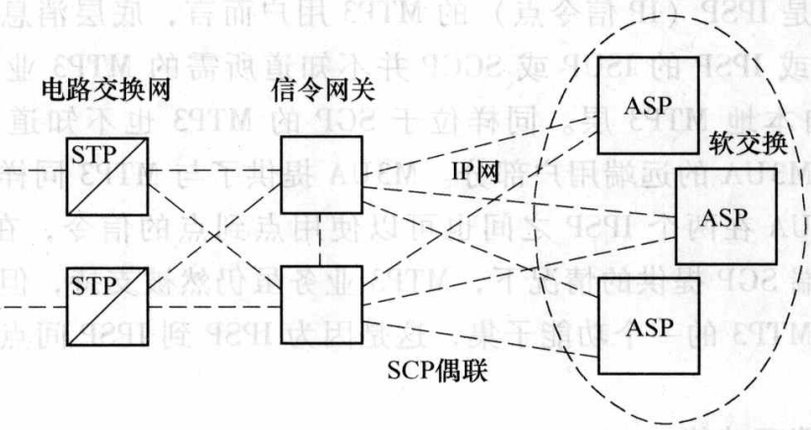


图 2-18 信令网的物理参考模型

在 No. 7 信令网中，信令网关可能要负责将 IP 域内的一系列节点在 No. 7 信令网内描述以进行路由。SG 作为 No. 7 信令网中的一个信令点，也可以使用一个 No. 7 信令网中的点码进行寻址以进行 MTP3 的管理。

信令网关可能在逻辑上被分割，以运作于多个 No. 7 信令网中。在这种情况下，SG 在每个网络中都使用一个点码进行寻址，而将在 IP 域中的一系列节点引入 No. 7 信令网。

应用服务器可以使用与 SG 同样的点码，它们自身的点码或是几个 AS 构成的 AS 组所使用的点码仅作为预留使用。仅使用一个点码来表征 SG 和在一起的所有 AS。如果 ASP 或 ASP 组通过多个 SG 与 No. 7 信令网相连，每个 SG 都有各自的点码，那么 ASP 将会以一个与其他 SG 不同的点码来标识。

4. M3UA 协议单元

M3UA 消息格式中包含一个公共消息头；之后是 0 个或多个由消息类型定义的参数，考虑到前向兼容性，因此所有消息类型都带有兼容性参数。

M3UA 协议消息的结构要求包括版本、消息类型、消息长度和消息内容。消息头对于所有信令协议适配层消息是公共的，如图 2-19 所示。

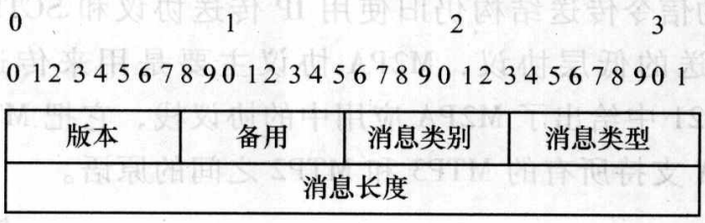


图 2-19　MTP3 消息（即公共消息头）格式

M3UA 消息由公共消息头和随后的可变长度参数构成，格式如图 2-20 所示。

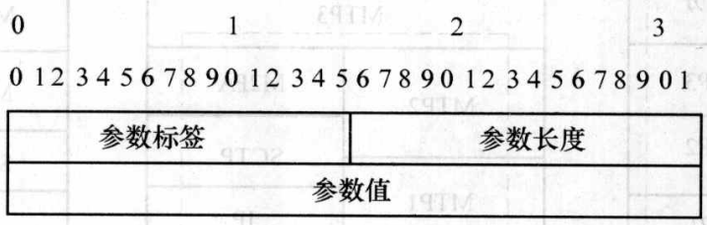


图 2-20　M3UA 消息格式

5. M3UA 边界描述

（1）M3UA 与 MTP3 用户（如：ISUP）的边界

1）MTP-TRANSFER request

2）MTP-TRANSFER indication

3）MTP-PAUSE indication

4）MTP-RESUME indication

5）MTP-STATUS indication

（2）M3UA 与 SCTP 的边界

（3）M3UA 与层管理的边界

2.3.7　M2PA 协议

1. M2PA 的协议栈位置

我们在前面已经初步提到了 M2PA（MTP2-User Peer-to-Peer Adaptationlayer，

MTP2 用户对等层间适配层）协议和 M3UA 协议的区别，M2PA 协议是一种对等适配协议，即 M2PA 和 SCTP 一起实现 MTP2 的功能。M2PA 的方式类似于 No. 7 信令网中 STP 设备的方式。使用 M2PA 协议的信令传送机制允许在任何两个 No. 7 信令点间在 IP 网络上处理 MTP3 的消息和提供 MTP 信令网网管功能。使用 M2PA 传送机制必须满足以下原则：

1）支持与 MTP3 协议在 IP 网连接上的无缝操作；

2）支持 MTP2 和 MTP3 的接口边界；

3）支持向管理异步报告状态变化；

4）支持对 SCTP 偶联和业务量的管理。

使用 M2PA 的信令传送结构仍旧使用 IP 传送协议和 SCTP（流控制传输协议）做为信令传送的低层协议，M2PA 协议主要是用来传送 No. 7 信令中的 MTP3 消息。图 2-21 中给出了 M2PA 应用中的协议栈，它把 MTP3 的消息适配到 SCTP 层上，M2PA 支持所有的 MTP3 和 MTP2 之间的原语。

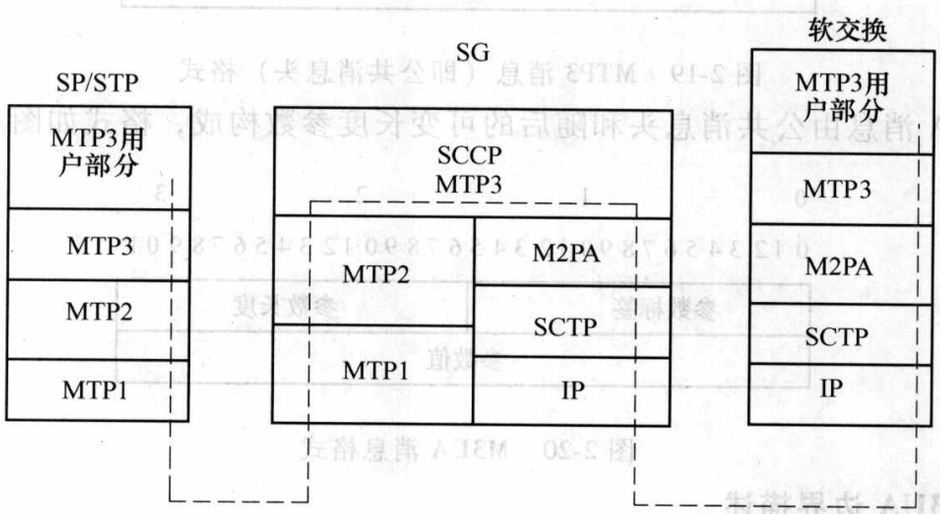


图 2-21 M2PA 的协议栈位置

2. M2PA 的业务

每个软交换上都需要包含 No. 7 信令的 MTP3 与 M2PA 边界的接口，M2PA 协议层用来为 MTP3 提供与 MTP2 所提供的等效的业务功能。

（1）支持 MTP2 和 MTP3 之间的接口边界

因为 SCTP 使用比 MPT 大的序号，所以以下的原语必须使用较大的序号：

1）BSNT（Backward Sequence Number to be Transmitted）

2）Retrieval Request and FSNC（Forward Sequence Number of last message accepted by remote level 2）

（2）支持对等层间的通信

3. M2PA 的边界描述

（1）M2PA 与 MTP3 边界的定义

由 M2PA 向上层边界提供的原语与 MTP2 向 MTP3 提供的原语相同。

（2）M2PA 和 SCTP 间的低层边界

M2PA 和 SCTP 之间的原语定义参见 SCTP 协议一节中“由高层协议向 SCTP 发送的原语”部分。

4. M2PA 与 M2UA 的异同

M2UA 协议能够实现任何 No. 7 信令网中 PSTN 或 ISDN 节点和 IP 网中 No. 7 信令节点互相传输 MTP2 的 MSU（消息信令单元），使用 M2UA 协议的信令网关其实质就是一个连接设备，只是一种传输手段。同 M2PA 相比：

（1）M2PA 与 M2UA 的相同点：

1）都用于传送 MTP3 数据消息；

2）对于 MTP3 都表现为同样的 MTP2 高层接口。

（2）M2PA 与 M2UA 的不同点见表 2-1。

表 2-1　M2PA 与 M2UA 的不同点

A	M2PA	IPSP 处理 MTP3/MTP2 原语
	M2UA	在 SG 的 MTP2 和 MGC 的 MTP3 之间通过 NIF 传送 MTP3/MTP3 原语以进行处理
B	M2PA	SG-IPSP 连接为 No. 7 信令链路
	M2UA	SG-MGC 连接不是 No. 7 信令链路，是 MTP 针对远端实体的扩展
C	M2PA	SG 是具有点码的 No. 7 信令节点
	M2UA	SG 无点码，不是 No. 7 信令节点
D	M2PA	SG 具有 No. 7 信令的较高层面，如：SCCP
	M2UA	SG 没有 MTP3 层
E	M2PA	依靠 MTP 进行管理
	M2UA	使用 M2UA 管理程序

第 3 章　H. 323 协议

在传统电话系统中，普通呼叫从呼叫连接建立到呼叫连接中断都需要一定的控制信令来配合实现。同样，在 IP 电话中，如何建立用户呼叫连接，如何按照彼此的数据处理能力发送数据，也同样需要相应的信令系统来实现，于是各种 IP 电话通信协议便相继产生。IP 电话通信协议中，H. 323 协议系列一直受到了 ITU 的推荐，它的协议体系发展非常完整，其核心体系也已基本稳定，目前在工业界广为采用，是软交换网的重要技术基础。

3.1　H. 323 协议族体系

H. 323 协议族由 ITU 的第 16 研究组提出，1996 年它的第一版被制定，并于 1998 年发展为第二版。H. 323 制定了无 QoS 保证的 PBN（Packet Based Networks，分组网络）上的多媒体通信系统标准。目前包括 TCP/IP、IPX 分组交换的以太网、快速以太网、令牌网、FDDI 技术以及使用 PPP 等分组协议通过电话交换网或 ISDN 的拨号连接或点对点连接等大多数网络技术均能在 PBN 上进行实现。PBN 的实现可以是一个单独的网段，也可以是由多个网段连接而构成的一个具有复杂拓扑结构的网络，正因为上述特点，H. 323 便在 LAN、WAN、Internet 中的多媒体通信应用中具有广阔的前景。

H. 323 协议族能够适用于包括语音、数据和视频及其相组合的多媒体通信，其中，对数据和视频的支持是可选的。另外，在整个 H. 323 的协议族中没有提供对确保通信 QoS 的定义，一般意义上讲，对 H. 323 通信服务质量的控制，是由其他的技术配合实现，例如，在 Internet 上，服务提供商和设备提供商都在不断地推出新的 QoS 产品和相关机制。

H. 323 的协议族体系结构如图 3-1 所示。

在图 3-1 中：

1）T. 120 系列协议（如：T. 122、T. 123、T. 124、T. 126、T. 127 等协议），主要用于数据应用。

2）RAS（注册、许可和状态，Registration、Admission and Status）协议，即 H. 225. 0 终端至网守信令，负责网守的 4 种必备服务：地址翻译、带宽控制、授权许可控制和区域管理。

3）H. 26 系列（如 H. 261、H. 263 等协议），主要用于视频编码。

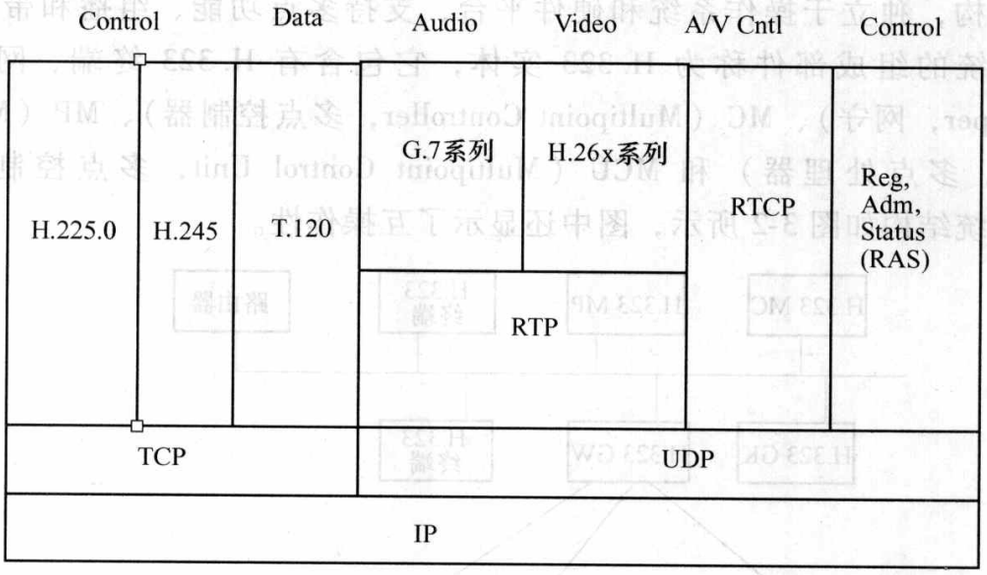


图 3-1　H. 323 的协议族体系结构

4）G. 7 系列（如 G. 729，G. 723. 1 等），主要用于语音编码。

5）RTCP 是 RTP 的一部分，能提供 QoS 服务，但只能监测 QoS 情况，而不能保证 QoS。

6）RTP，提供对需要实时传输的媒体信息提供封装传输。

7）RTCP/ RTP 是 IETF 建议的，它并非专为 IP 电话设计的，而是适用于一般的多媒体通信的技术。

8）H. 225. 0 协议，主要实现呼叫控制，及建立通信双方的呼叫联系。

9）H. 245 协议，主要实现媒体信道控制。在通信过程中每一条逻辑信道的打开、参数设定以及通话双方能力集的交互都由 H. 245 协议来完成。另外，H. 245 协议的控制信号将由一条专用的可靠信道（如 IP 网中的 TCP 网络连接）来负责传输，这条专用的可靠信道就是 H. 245 的控制信道，它必须在通信中任何逻辑信道建立之前先行建立。因此，H. 225. 0 呼叫信令协议的功能除了建立通信双方的联系外，同时还要建立 H. 245 控制信道。

10）H. 225. 0 和 H. 245 是 H. 323 系统的核心协议。

另外协议族中还使用了用于补充业务的 H. 450. 1、H. 450. 2 和 H. 450. 3，有关安全的 H. 235，与电路交换业务互操作的 H. 246 等一系列协议。实际上，H. 323 很大程度上建筑在 ITU 以前的相关协议上，同时也采用了 IETF 的 RTCP/ RTP 等协议，来解决通信中一些如 QoS 的问题。

3. 2　H. 323 系统结构介绍

H. 323 提供设备之间、高层应用之间和提供商之间的互操作性，它不依赖

于网络结构，独立于操作系统和硬件平台，支持多点功能、组播和带宽管理。H.323 系统的组成部件称为 H.323 实体，它包含有 H.323 终端、网关、GK（GateKeeper，网守）、MC（Multipoint Controller，多点控制器）、MP（Multipoint Processor，多点处理器）和 MCU（Multipoint Control Unit，多点控制单元）。H.323 系统结构如图 3-2 所示，图中还显示了互操作性。

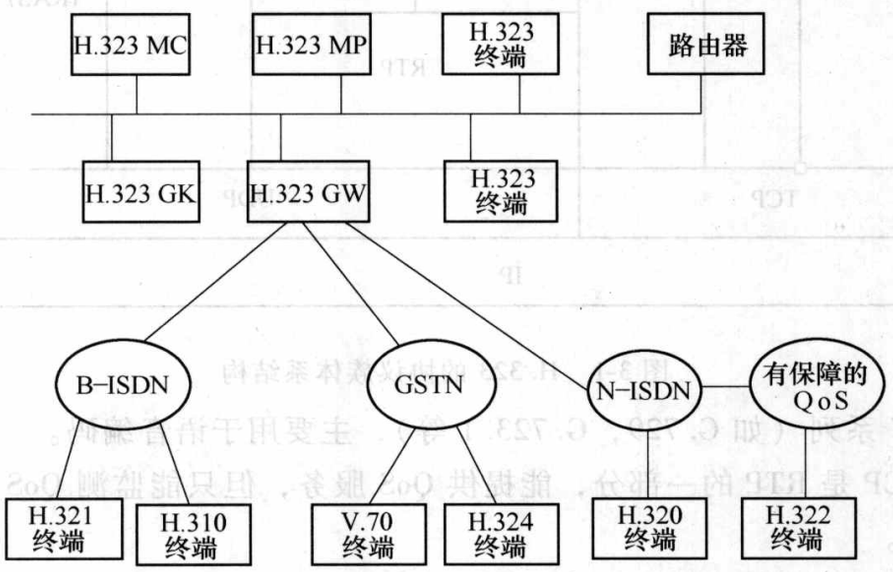


图 3-2 H.323 系统结构

图 3-2 中的 H.323 终端、H.323 网管和 MCU 是能够发起呼叫和接收呼叫的，我们将上面 3 个 H.323 实体称之为端点，媒体信息流就是在端点产生或终结。

一个 H.323 终端是一个在网络上提供实时、双向通信的端点，通信的媒体信息流包括控制、指示、音频、视频和数据信息，并在 PBN 上按照 H.323 标准进行实时通信。它可以集成到计算机上，也可以是独立的终端，如可视电话等。

H.323 网关是提供 H.323 与 SCN 终端之间、H.323 终端之间进行双方通信的端点。图 3-2 便显示了 H.323 和 GSTN（General Switched Telephone Network，公用交换电话网，包括 PSTN、ISDN 和 PLMN），N-ISDN，B-ISDN，QoS 保证的 LAN 的互联互通情况。其中 H.320 终端是 N-ISDN 的可视电话，H.324 是 GSTN 中低数据传输率的多媒体终端，H.322 是 QoS 保证下局域网的可视化终端，H.321 是 H.320 在 B-ISDN 网络环境下加上适配层功能的终端，H.310 时宽带网声像终端，它必须按 H.321 模式才能和 H.323 互通。V.70 是 DSVD（Digital Simultaneous Voice and Data，数字语音数据同步传送）终端，能同时进行语音和数据传输；GSTN 中提到的语音终端就是指普通的电话终端，而 N-ISDN 提到的语音终端则是指 ISDN 电话机。

实际上，H.323 网关的主要功能可以同 IP 网中网关的功能进行类比，IP 网中网关的功能主要是实现协议之间的转换，而 H.323 网关则主要实现媒体信息的编码转换以及不同信令之间的转换，在实际软交换的实现中，一般 H.323 网

关的功能可以由软交换设备来实现。

H. 323 网守是一个提供地址翻译和网络接入控制的 H. 323 实体。网守也向其他终端、网关、MCU 提供带宽管理和定位网关等服务。在 H. 323 协议族中，网守是一个可选部件，但在实际的 H. 323 系统使用中，作为运营商是必须用网守来进行 H. 323 系统的管理。在目前大多数软交换产品中，H. 323 网守和 H. 323 网关的功能可以集成到软交换设备中。

在 H. 323 协议族中，我们将一个由单独的网守来控制的终端、网关、MCU 的集合叫做 H. 323 区域（Zone），结构如图 3-3 所示。这个区域中可以有或者没有网关与 MCU，但至少包含一个终端，同时有且仅有一个网守。H. 323 区域不依赖网络的拓扑结构，可以由多个网段组成，这些网段之间通过路由器或其他设备连接。

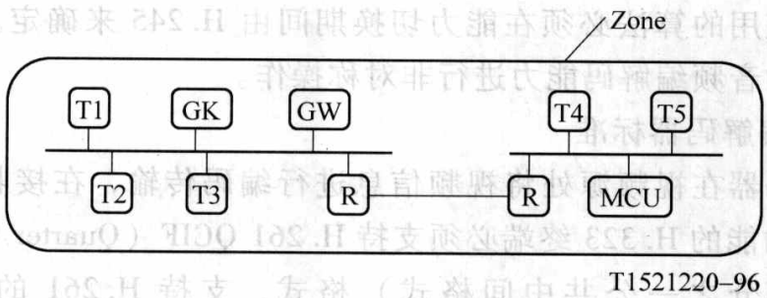


图 3-3　H. 323 区域

通常，我们将一个运营局管辖的 H. 323 的实体集合称之为一个管理域（domain）。在一个管理域中可以包括多个 H. 323 区域，不同 H. 323 区域的终端可以通过网守之间的协作来进行通信。而不同的运营局之间的互通，则通过管理域的协作来进行，在 ITU 所制定的规则中，可以通过每个管理域设定的代理网守来实现。

下面我们讲一些主要的 H. 323 的技术。

1. 信息流

信息流可以分成如下 5 种：音频、视频、数据、通信控制与呼叫控制。

音频和视频信号分别与各自的控制信号同时传输。

通信控制信号包含如下功能：能力交换、打开和关闭逻辑信道和模式控制等，即 H. 245 控制信令。

呼叫控制信号用来建立和断开呼叫及其他功能，即 H. 225. 0 呼叫控制信令。

2. 终端

终端是分组交换网络中能提供实时、双向通信的节点设备，完成两个终端之间的控制、指示、音频、视频图像或数据信息的交换这样一个通信过程。所有的终端都必须支持语音通信，对视频和数据通信的支持可选。H. 323 规定了不同的音频、视频和数据终端协同工作所需的操作模式。

所有的 H.323 终端必须支持 H.245。H.245 标准用于协商信道使用和信道性能。其他三个必需支持的部件是：用于呼叫信令和呼叫建立的 Q.931，用于和 Gatekeeper 通信的 RAS，以及顺序打包传送音视频流数据的 RTP/RTCP。

在 H.323 终端中可选组件包括图像编解码器、T.120 数据会议协议和 MCU 功能。

（1）音频编解码器标准

音频编解码器标准对从传声器输入的音频信息进行编码传输，在接收端进行解码以便输出到扬声器。H.323 支持的压缩算法都符合 ITU 标准。为进行语音压缩，H.323 终端都必须支持 G.711 语音标准，传送和接收 A 律和 μ 律压缩信号。支持其他音频编解码器的标准如 G.722，G.723.1，G.729A，MPEG-1 是可选的。

编解码器使用的算法必须在能力切换期间由 H.245 来确定。H.323 终端应能对本身具有的音频编解码能力进行非对称操作。

（2）视频编解码器标准

视频编解码器在视频源处将视频信息进行编码传输，在接收端进行解码显示。具有视频功能的 H.323 终端必须支持 H.261 QCIF（Quarter Common Intermediate Format，四分之一公共中间格式）格式，支持 H.261 的其他格式以及 H.263 标准为可选功能。在分组网络上，使用 H.261，H.263 编解码器无需 BCH 纠错和纠错帧。

解码器可接受的视频比特率、图像格式和算法选项在切换期间由 H.245 来定义。若支持多种图像解析格式，则 H.323 终端应能支持视频比特率、帧速率以及图像解析功能的非对称操作，也就是说，具备 CIF（Common Intermediate Format，公共中间格式）能力的终端可以以 QCIF 格式发送图像，而以 CIF 格式接收图像。

（3）H.245 控制功能

H.245 控制功能采用 H.245 控制信道交换端到端控制消息，从而实现 H.323 实体的运作，其中包括主从判别、能力交换、打开和关闭逻辑信道、模式参数请求、流控制消息和通用命令与指示。H.245 信令建立于两个端点之间、一个端点与一个 MC 之间，或是一个端点与一个网守之间。端点会为每个它所参与的呼叫建立唯一的一个 H.245 控制信道。由于终端、MCU、网关和网守都支持多个呼叫，这就意味着会有多个 H.245 控制信道。H.245 信道应承载于逻辑信道 0 上。在 H.245 控制信道建立到此信道被中止，逻辑信道 0 被认为始终处于打开状态的。正常的打开和关闭逻辑信道的过程不适用于 H.245 控制信道。

（4）能力交换

能力交换是指提供独立的接收和传送功能。终端通过它来描述自身能力，

以便在各种合成模式中工作。接收能力描述终端接收和处理输入流的能力，即发送方发送的信息应是接收方已经指明能够接收的。传送能力描述终端传送信息流的能力，即为接收方提供几种可能的操作模式，让接收方选择它更愿意的模式。

（5）逻辑信道信令

每个逻辑信道承载从一个发送方至另一个或多个发送方的信息，当一个逻辑信道被打开，用 Open Logical Channel（打开逻辑信道）消息全面描述逻辑信道的内容，包括媒体类型、使用的运算法则、任选项和所有接收方解释逻辑信道内容所需的信息。逻辑信道不再使用时可以关闭。打开逻辑信道使用如下流程：发起终端首先发送打开逻辑信道消息，如果逻辑信道承载的媒体类型使用 RTP，这个消息将包括含后向的 RTCP 信道传送地址的 Media Control Channel（媒体控制信道）参数。接收方终端必须响应打开逻辑信道 Ack 消息，如果终端的逻辑信道使用 RTP 承载媒体类型，这消息应包括含 RTP 传送地址的 mediaTransportChannel（媒体传输信道）参数和含前向的 RTCP 信道传送地址的媒体控制信道参数。

这个逻辑信道可以是双向的亦可以是单向的，每个逻辑信道有一个惟一的逻辑信道号。

（6）模式选择

接收方可以使用 H. 245 信道的请求模式消息请求发送方发送一个特定模式，发送方在可能的情况下都会服从。端点收到 MC 发出的多点 ModeCommand（模式命令）时，如果在它的能力范围集合之内，就必须服从这个请求。

无论在集中式还是分散式多点会议中，端点发出的 requestMode 都直接送到 MC，由 MC 决定接受还是拒绝该请求。

（7）主从决定

主从决定主要是解决冲突问题，例如两个端点都是一个会议中的 MC 或者在两个端点间试图建立一个双向信道。在这个流程中两个端点在 masterSlaveDetermination（主从决定）消息中交换随机数以决定主从端点。

（8）接收通路延迟

此功能包含了为保持同步和消除网络数据包到达抖动而加入的延迟，接收端点可以加入适当的延迟以达到音频与视频的同步。

（9）数据信道

数据信道是可选的，根据数据应用程序，数据信道可以是单向的，也可以是双向的。

H. 323 终端与其他 H. 323，H. 324，H. 320 或 H. 310 终端交互时缺省使用 T. 120 建议。

3. 网关

网关是 H.323 系统中的一个可选组件。网关能提供的众多服务中最通用的是 H.323 节点设备与其他类型终端之间的翻译转换功能。这包括传输格式（如 H.225.0 到 H.221）和通信规程的转换（如 H.245 到 H.242）。另外，在 SCN 和 PBN 之间，网关还完成语音和图像编解码器转换功能，以及呼叫建立和拆除工作功能。

通常，网关是将分组网络终端的特性映射到 SCN 终端上，或是相反过程。网关的主要应用在于：

1）通过 N-ISDN 建立与远程 H.320 兼容终端的连接；

2）通过 B-ISDN 建立与远程 H.321 兼容终端的连接；

3）通过 SCN 以及无线网络建立与远程 H.324 或 V.70 兼容终端的连接等。

终端使用 H.245 和 Q.931 协议与网关进行通信。若没有必要与其他网络建立连接，则无需网关。

采用适当的编码器，H.323 网关可支持符合 H.310，H.321，H.322 以及 V.70 标准的终端。

很多网关的功能都留给了设计者，例如，能够通过网关进行通信的 H.323 终端的实际数量并不受标准的制约。同样，SCN 连接数、支持同时进行独立会议的数量、视频或音频及数据转换功能以及多点功能的解释和细节实现都留给了制造商。

4. 网守

网守是 H.323 系统的一个可选组件，其功能是向 H.323 节点提供呼叫控制服务。两个最重要的呼叫控制功能是：

1）从终端和网关的 LAN 别名地址到 IP 或 IPX 地址的翻译；

2）带宽管制，例如，如果网络管理员设定了在 LAN 上同时举行会议数量的阈值，一旦阈值达到，网守就会拒绝再建立更多的连接，其效果是限制会议总带宽为有效带宽总量的一部分，使得能保留部分带宽给 E-mail、文件传输和其他 LAN 协议流量使用。

H.323 网守在系统中必须提供的 4 种服务包括地址翻译、带宽控制、授权许可控制和区域管理，可选的功能包括带宽管理、呼叫鉴权、呼叫控制信令和呼叫管理。

网守另一个可选的但确实有价值的特点是它有路由 H.323 呼叫的能力。一个呼叫通过网守路由后，能得到更有效的控制。业务提供商需要通过这项能力来对经过其网络的呼叫进行计费。当被叫端点失效时，这一服务也可用来将呼叫路由至另一个端点。此外，还可以用这项功能来完成基于某种路由策略的、在多个网关之间的路由流量平衡。

由单一网守管理的所有终端、网关和 MCU 的集合称为 H. 323 区域。虽然从逻辑上网守和 H. 323 节点设备是分离的，但是可将网守的功能融入 H. 323 终端、网关或 MCU 中。在 H. 323 系统中网守并不是必需的，但是如果它存在，终端必需使用由它提供的服务。这些服务在 RAS 中定义，包括地址翻译、带宽控制、授权许可控制和区域管理，即网守的 4 种必备服务。

网守也能在多点连接中发挥作用。为了支持多点会议，用户可使用网守来接收点对点会议系统两个终端的 H. 245 控制信道消息。当点对点会议切换到多点会议时，网守将 H. 245 控制信道重定向给多点控制器（MC），网守无需处理 H. 245 信令，它仅仅需要将信令传送于两个终端或终端和 MC 之间。

包含网关的分组交换网络也应该包含网守，利用它将输入的 E. 164 地址翻译成传输地址。由于区域是由网守定义的，内含网守的 H. 323 实体需要一种机制能够使内部功能失效，以便当分组网络中存在多个包含网守 H. 323 实体时，这些实体能被配置成在同一个区域内。

5. 多点控制器

MC 在一个多点会议之中提供控制功能。MC 向会议中的每个端点发送一个能力集合，这个能力集合包括了传输时使用的操作模式。在有终端加入或离开会议的时候，MC 可以修改这个能力集合。

MC 可以与网守、网关、终端或者多点控制单元位于同一物理设备上。它们的结构如图 3-4。

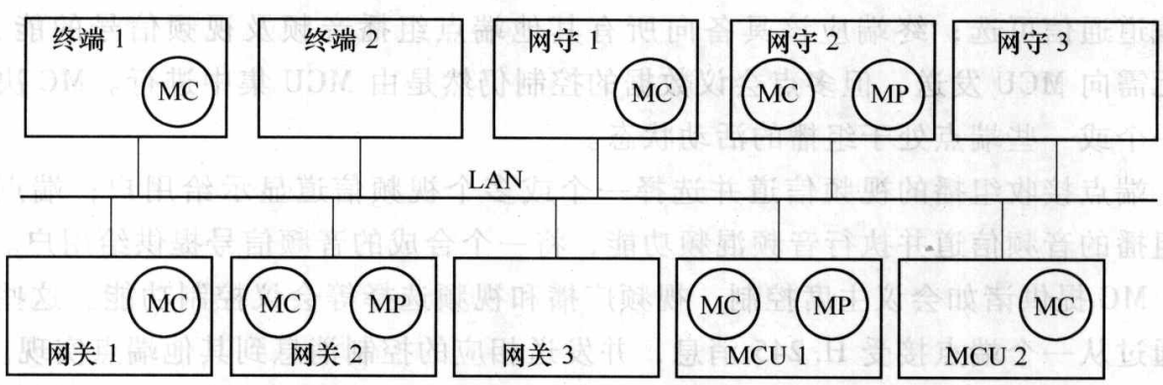


图 3-4　H. 323 系统中 MC 与 MP 的可能位置

6. 多点处理器

多点处理器（MP）在集中或异构多点会议中从终端接收音频、视频和/或数据流，处理之后再把它们回送到各个终端。

处理视频的 MP 必须提供视频交换或混频功能。视频交换是选择一个源终端视频流并将其输出至另一终端的过程；混频是将多个源视频流按照规定格式化到一个视频流的过程。

MP 是不可呼叫的，它所在的多点控制单元（MCU）是可呼叫的。MP 是一

条媒体信道的终点，也是另一条媒体信道的始点。

7. 多点控制单元

MCU 支持多点会议功能，一个 MCU 必须包含一个 MC 和 0 或多个 MP。一个典型的支持集中式会议的 MCU 包含一个 MC 和一个能够处理音频、视频和数据流的 MP；一个典型的支持分散式会议的 MCU 包含一个 MC 和一个符合 T.120 建议的 MP。

MCU 可以位于 H.323 网络侧的网关中，一个网守中也可以包含 MCU，但是，不论怎样，它们在功能上都是独立的。

8. 多点会议能力

（1）集中式多点会议

所有端点都必须支持此能力。集中式多点会议需要一个 MCU 来组织，终端与 MCU 中的 MC 以点对点的方式通过 H.245 控制信道通信，H.245 用来指定各终端的通信能力；终端与 MP 通过音频、视频和数据信道通信。MC 实现控制功能，MP 实现视频交换或混频、T.120 多点会议数据分发功能。MP 应该可以在不通模式之间转换数据，允许终端以不同的通信模式参加会议。MP 还可提供不同编码器间、数据传输速率间的转换并使用组播的方式分配已处理的视频信息。

（2）分散式多点会议

分散式多点会议利用组播技术来组织。这种模式下，终端与 MCU、网关、网守或其他终端中的 MC 以点对点的方式通过 H.245 控制信道通信，与 MP 的数据信道通信可选；终端应该具备向所有其他端点组播音频及视频信号的能力，而无需向 MCU 发送，但多点会议数据的控制仍然是由 MCU 集中进行。MC 决定那一个或一些端点处于组播的活动状态。

端点接收组播的视频信道并选择一个或多个视频信道显示给用户；端点接收组播的音频信道并执行音频混频功能，将一个合成的音频信号提供给用户。

MC 提供诸如会议主席控制、视频广播和视频选择等会议控制功能。这些功能通过从一个端点接受 H.245 消息，并发送相应的控制消息到其他端点实现。

接收终端负责处理多个进来的音频和视频流。会议终端利用 H.245 控制信道来通知 MCU 可同时解码多少个音频和视频流。一个终端的同时解码能力并不限制会议中被组播的音频和视频流数量。

（3）异构式多点会议—音频集中方式

异构式多点会议是集中和分散功能的组合。在此种模式下，MCU 和端点采用组播的方式传输视频信号，采用点对点方式传输音频信号。终端以点对点方式同 MC 通过 H.245 控制信道通信，同 MP 的数据信道可选。

终端必须具备向会议中的所有其他端点组播视频信号的能力，由 MC 决定哪一个或哪一些端点处于组播的活动状态。端点可以有选择的接受一个或多个可

用的视频信道。

会议中的所有端点传送音频信号到 MP，MP 将这些音频信号混频并将结果发送到其他端点。MP 可以为会议中的每一个端点合成一个惟一的音频流。

（4）异构式多点会议—视频集中方式

这种模式下，MCU 和端点采用分散方式传输音频信号，采用集中方式传输视频信号。终端以点对点方式同 MC 通过 H.245 控制信道通信，同 MP 的数据信道可选。

终端必须具备向会议中的所有其他端点组播音频信号的能力，由 MC 决定哪一个或哪一些端点处于组播的活动状态。端点接收组播的音频信号，并将这些音频信号混频后提供给用户。

会议中的所有端点传送视频信号到 MP，MP 进行视频交换、混合或格式转换操作并将结果送到其他端点。MP 可以为会议中的每一个端点合成一个惟一的视频流；为了减少网络中使用的带宽，也可以将一个视频流组播至所有参与的端点。

（5）级联多点控制单元

多点控制功能也可以分布在多个不同的多点控制器中，叫做级联。级联允许多个不同的多点控制器之间两两通信。一个多点控制器被定义为主多点控制器，其余的全部是从多点控制器。

3.3　RAS 协议

RAS 信令提供注册、确认、带宽改变等功能，如果没有网守就不需要 RAS。网守通过和网关或者终端之间 RAS 消息的交互，为终端提供注册服务、呼叫许可控制、地址转换、带宽管理以及呼叫管理等功能。

RAS 是端点（终端或网关）和网闸之间执行的协议，基本上是管理功能。它包含以下几个过程：

1. 网守搜寻
2. 端点登记
3. 端点定位
4. 呼叫接纳
5. 呼叫退出
6. 带宽管理
7. 状态查询
8. 网关资源指示

3.4 呼叫信令信道

呼叫信令信道基于 TCP 进行传送，用于承载 H.225.0 呼叫控制信息。它包括了呼叫的建立和拆除等流程。呼叫信令信道的功能是使用 H.225.0 消息来建立两个 H.323 端点之间的连接，呼叫信令信道独立于 RAS 信道和 H.245 信道，这是条可靠传输信道。H.225.0 中指定了 H.323 系统进行呼叫信令流程时所使用的主要消息，这些消息及功能与 ISDN 的 Q.931 十分类似。呼叫信令信道建立时间必须在 H.245 控制信道和其他逻辑信道建立之前。如果网络中没有网守，呼叫信令信道消息通过呼叫双方的呼叫信令传送地址直接在主被叫之间进行传递；如果网络中有网守，主叫端点和网守首先使用网守的 RAS 信道传送地址，交换初始允许接入消息，然后，网守在 GCF 中需指明是采用直接发送方式还是采用通过网守转发的方式。它们的具体功能在以下小节介绍。

3.4.1 H.323 通信控制的一般过程

H.323 端点建立呼叫连接一般为 3 个过程：

1）呼叫接入控制：H.323 通话双方及相关网守按照 RAS 协议进行通信交互，这是通过 RAS 信道（不可靠信道）来实现控制的。网守在同意通话双方接入的情况下，进入下一阶段，即呼叫建立阶段。

2）呼叫控制：H.323 通话双方及相关网守按照呼叫信令协议（H.225.0）进行通信交互，这是通过呼叫信令信道（可靠信道）来实现控制的。呼叫建立之后，进入下一阶段，即媒体控制建立阶段。

3）媒体控制：H.323 通话双方及相关网守按照媒体控制信令协议（H.245）进行通信交互，这是通过媒体控制信道（可靠信道）来实现控制的。然后在呼叫双方建立起一些具有一定带宽实时通信的逻辑信道（不可靠信道），呼叫双方便通过这些逻辑信道进行通信。

从呼叫的宏观角度出发，通信连接一般可分为 5 个过程：

1）呼叫建立；

2）呼叫能力协商和模式设定；

3）呼叫逻辑信道建立，包括音频和视频。

4）通信过程中的呼叫资源改变及相关服务，如带宽改变。

5）呼叫结束，及信道关闭。

3.4.2 呼叫信令信道选路

呼叫信令消息有两种传送方式：

1）网守转发呼叫信令（见图 3-5），即呼叫信令通过网守向另一个端点发送，整个通信过程必须经过转发；

2）直接端点呼叫信令（见图 3-6），即呼叫信令在端点之间直接传递。具体采用哪种方式由网守决定。

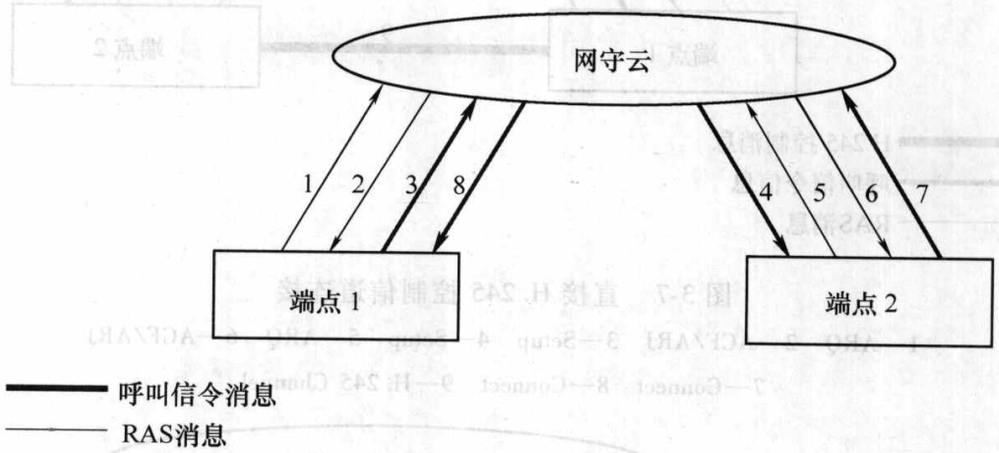


图 3-5　网守转发呼叫信令

1—ARQ　2—ACF/ARJ　3—Setup　4—Setup　5—ARQ　6—ACF/ARJ
7—Connect　8—Connect

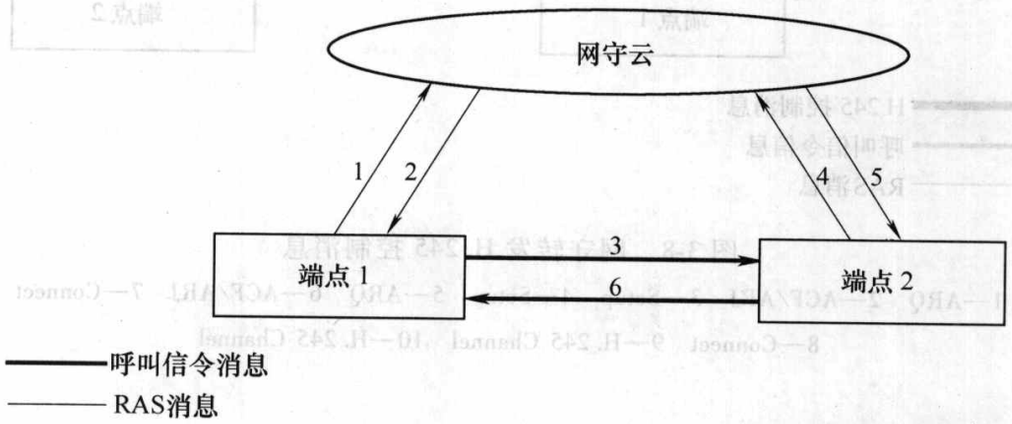


图 3-6　直接端点呼叫信令

1—ARQ　2—ACF/ARJ　3—Setup　4—ARQ　5—ACF/ARJ　6—Connect

3.4.3　媒体控制信道路由

当使用网守转发呼叫信令时，H. 245 控制信道的路由有两种方法：

1）在端点之间直接建立 H. 245 控制信道（见图 3-7）；

2）H. 245 消息由网守转发（见图 3-8）。选择方式仍由网守决定。

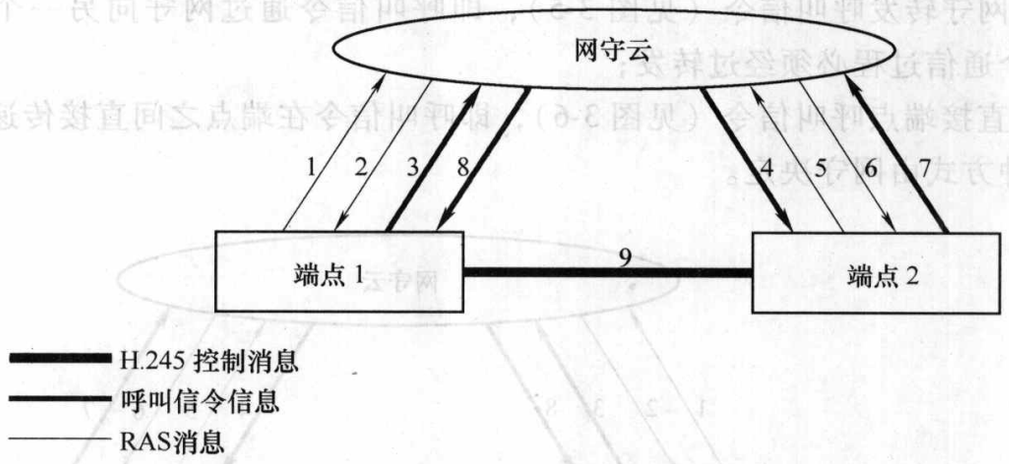


图 3-7 直接 H. 245 控制信道连接

1—ARQ 2—ACF/ARJ 3—Setup 4—Setup 5—ARQ 6—ACF/ARJ
7—Connect 8—Connect 9—H. 245 Channel

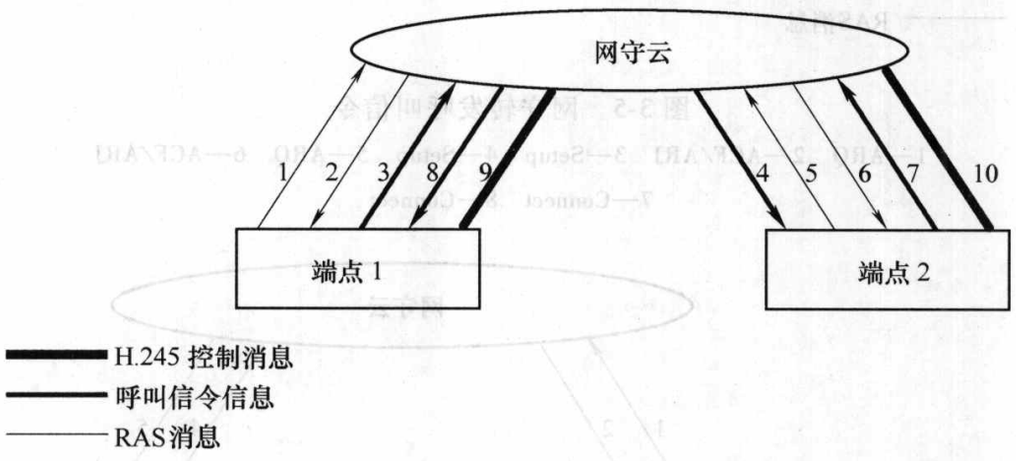


图 3-8 网守转发 H. 245 控制消息

1—ARQ 2—ACF/ARJ 3—Setup 4—Setup 5—ARQ 6—ACF/ARJ 7—Connect
8—Connect 9—H. 245 Channel 10—H. 245 Channel

3.5 呼叫信令流程

在 3.4.1 一节中我们提到，H. 323 协议中呼叫信令流程分为 5 个阶段，呼叫信令流程部分是 H. 323 协议中重要的组成部分，它是呼叫连接建立、保持与拆除的程序，同时对互操作性的实施也有实际的工程意义。在呼叫信令的交互过程中涉及的消息均在 H. 225.0 及 H. 245 中规定。

3.5.1 呼叫建立

H. 323 呼叫建立流程涉及到三条信令控制信道：RAS 信令信道、呼叫信令信道和 H. 245 控制信道。通过这三条信道的协调使得 H. 323 的呼叫得以进行。

如果别名地址和传送地址都被指定，呼叫建立使用 H.225.0 定义的呼叫控制消息。

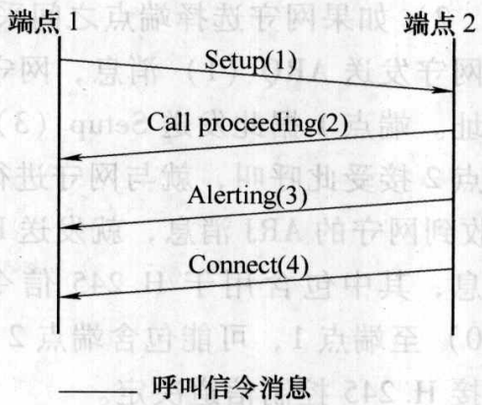


图 3-9　无网守的基本呼叫建立

3.5.2　端点均无注册

端点 1 和端点 2 都在没有网守注册情况下，两个端点进行直接通信（见图 3-9）。首先，端点 1 通过已知的端点 2 的呼叫信令信道 TSAP 标识将 Setup（1）消息发送至端点 2。端点 2 应答的 Connect（4）消息内应包含 H.245 控制信道传送地址，以便传送 H.245 消息。

3.5.3　两个端点都注册于同一网守

在这种状况下，存在两种情况：

1）如果网守选择端点之间采用直接选路信令（见图 3-10）方式。端点 1 会向网守发送 ARQ（1）消息，网守在 ACF 中返回端点 2 的呼叫信令信道传送地址。端点 1 根据得到的传送地址向端点 2 发送 Setup（3）。如果端点 2 接受此呼叫，就与网守进行 ARQ（5）/ACF（6）的消息交互。如果端点 2 收到网守的 ARJ 消息，那就向呼叫方回 Release Complete 消息。否则，端点 2 将在应答的 Connect（4）消息中包含用于 H.245 信令传送的 H.245 传送地址。

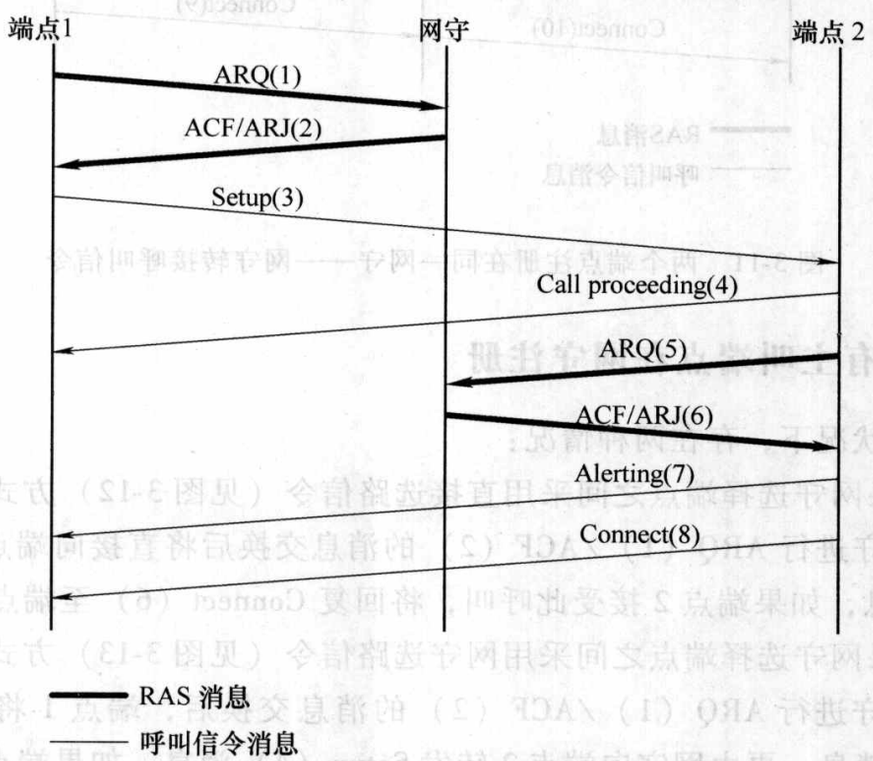


图 3-10　两个端点注册在同一网守——直接呼叫信令

2）如果网守选择端点之间采用网守选路信令（见图 3-11）方式。端点 1 会向网守发送 ARQ（1）消息，网守在 ACF（2）中返回自己的呼叫信令信道传送地址。端点 1 据此发送 Setup（3），网守再转发 Setup（4）消息至端点 2。如果端点 2 接受此呼叫，就与网守进行 ARQ（6）/ACF（7）的消息交换。如果端点 2 收到网守的 ARJ 消息，就发送 Release Complete 至网守，否则回发 Connect（9）消息，其中包含用于 H.245 信令传送的 H.245 传送地址。网守发送 Connect（10）至端点 1，可能包含端点 2 或网守的 H.245 传送地址，这由网守是否选择转接 H.245 控制信道决定。

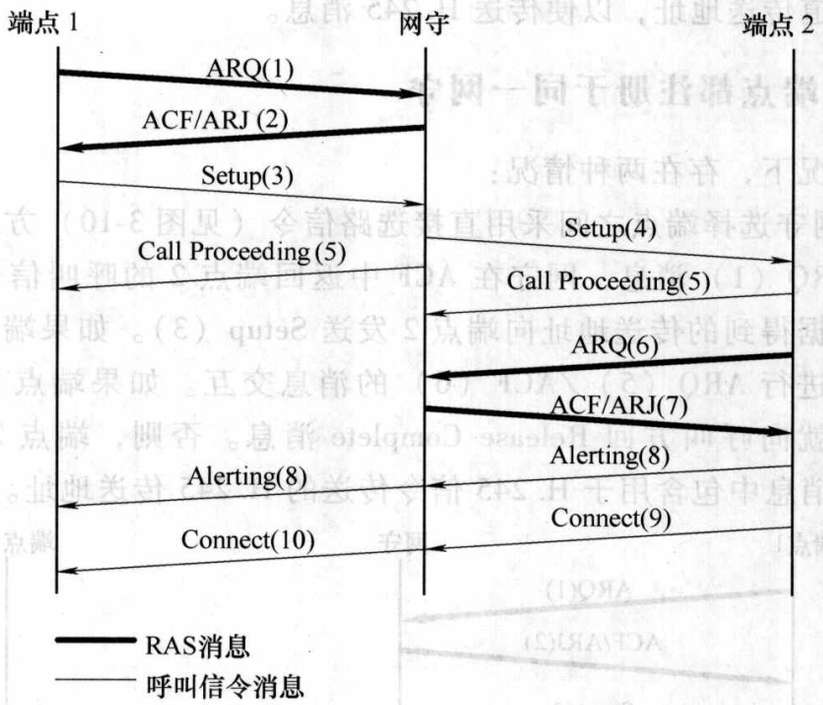


图 3-11 两个端点注册在同一网守——网守转接呼叫信令

3.5.4 只有主叫端点在网守注册

在这种状况下，存在两种情况：

1）如果网守选择端点之间采用直接选路信令（见图 3-12）方式。那么，端点 1 在与网守进行 ARQ（1）/ACF（2）的消息交换后将直接向端点 2 发送 Setup（3）消息，如果端点 2 接受此呼叫，将回复 Connect（6）至端点 1。

2）如果网守选择端点之间采用网守选路信令（见图 3-13）方式。那么，端点 1 在与网守进行 ARQ（1）/ACF（2）的消息交换后，端点 1 将向网守发送 Setup（3）消息，再由网守向端点 2 转发 Setup（4）消息，如果端点 2 接受此呼叫，将回复 Connect（6）至端点 1。

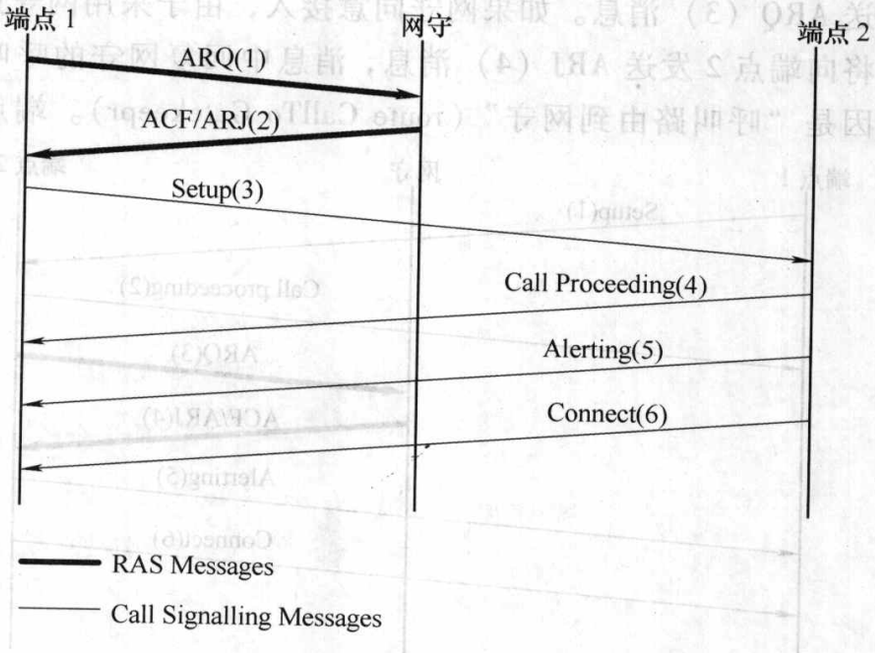


图 3-12　只有主叫端点注册——直接呼叫信令

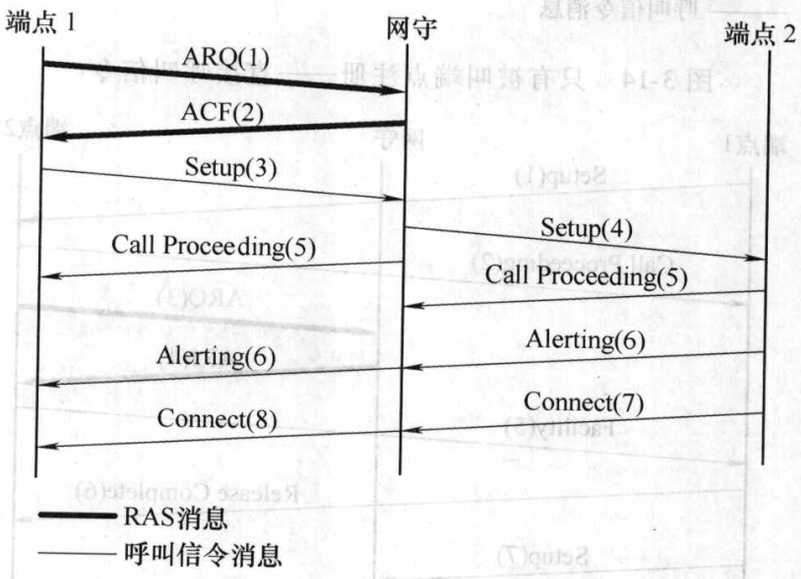


图 3-13　只有主叫端点注册——网守转发呼叫信令

3.5.5　只有被叫端点在网守注册

在这种状况下，存在两种情况：

1）如果网守选择端点之间采用直接选路信令（见图 3-14）方式。那么，端点 1 首先直接向被叫方发起 Setup（1）消息，如果端点 2 接受此呼叫，将与网守发送 ARQ（3）消息，如果网守回复 ACF（4），则端点 2 将回复 Connect（6）至端点 1，如果端点 2 收到网守的 ARJ 消息，就发送 Release Complete 至端点 1。

2）如果网守选择端点之间采用网守选路信令（见图 3-15）方式，情况较为不同。端点 1 首先直接向被叫方发起 Setup（1）消息，如果端点 2 接受此呼叫，

将与网守发送 ARQ（3）消息。如果网守同意接入，由于采用网守选路呼叫信令方式，网守将向端点 2 发送 ARJ（4）消息，消息中回复网守的呼叫信令传输地址，拒绝原因是“呼叫路由到网守”（route CallTo Gatekeepr）。端点 2 收到消息

图 3-14 只有被叫端点注册——直接呼叫信令

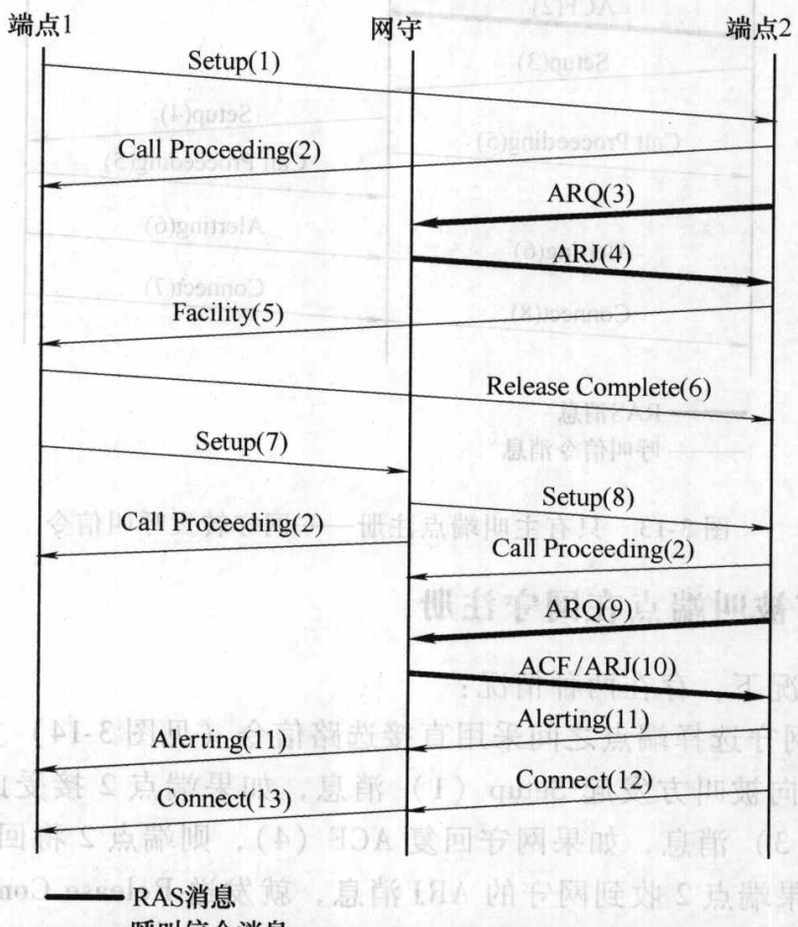


图 3-15 只有被叫端点注册——网守转发呼叫信令

后会向端点 1 发送 Facility（5）消息，该消息中把 UUIE 中的“替换地址”设置为网守的呼叫信令传输地址，“理由”设为“呼叫路由到网守”。于是，端点 1 将向端点 2 回复 Release Complete（6）消息，同时向网守发送 Setup（7），再由网守转发 Setup（7），端点 2 再与网守进行 ARQ（9）/ACF（10）的消息交换，如果网守同意接入，端点 2 将向网守回复 Connect（13）消息，再由网守向端点 1 进行转发。

3.5.6　两个端点注册在不同的网守

当两个端点是在不同的网守注册时，由于网守采用的选路信令方式的不同，同样存在两种情况：

1）如果两个端点注册在不同的网守，且两个网守均采用直接选路信令方式（见图 3-16）。端点 1 向网守 1 启动 ARQ（1）/ACF（2）消息交互，如果网守 1 可以与网守 2 通信得到端点 2 信息，网守 1 将在 ACF 消息中返回端点 2 的呼叫信令信道传送地址。然后端点 1 可以向上述地址或向端点 2 的已知呼叫信令信道传送地址发送 Setup（3）。如果端点 2 可以接受该呼叫，则与网守 2 进行 ARQ（5）/ACF（6）消息交互。最后端点 2 在 Connect（8）消息中将 H. 245 控制信道传送地址发送给端点 1。

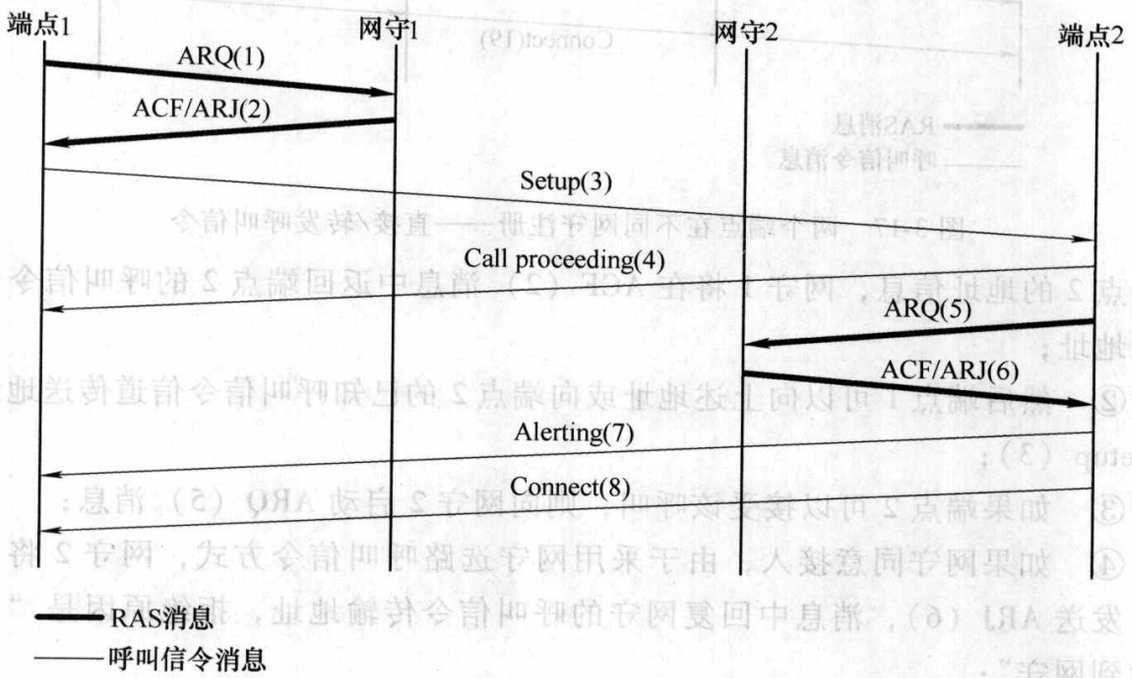


图 3-16　两个端点在不同网守注册——直接呼叫信令

2）如果两个端点注册在不同的网守，我们考虑以下 3 种情况：

网守 1 采用直接呼叫信令方式，网守 2 采用网守选路信令方式（见图 3-17）。呼叫建立过程如下：

①　端点 1 向网守 1 发送 ARQ（1）消息，如果网守 1 可以与网守 2 通信得

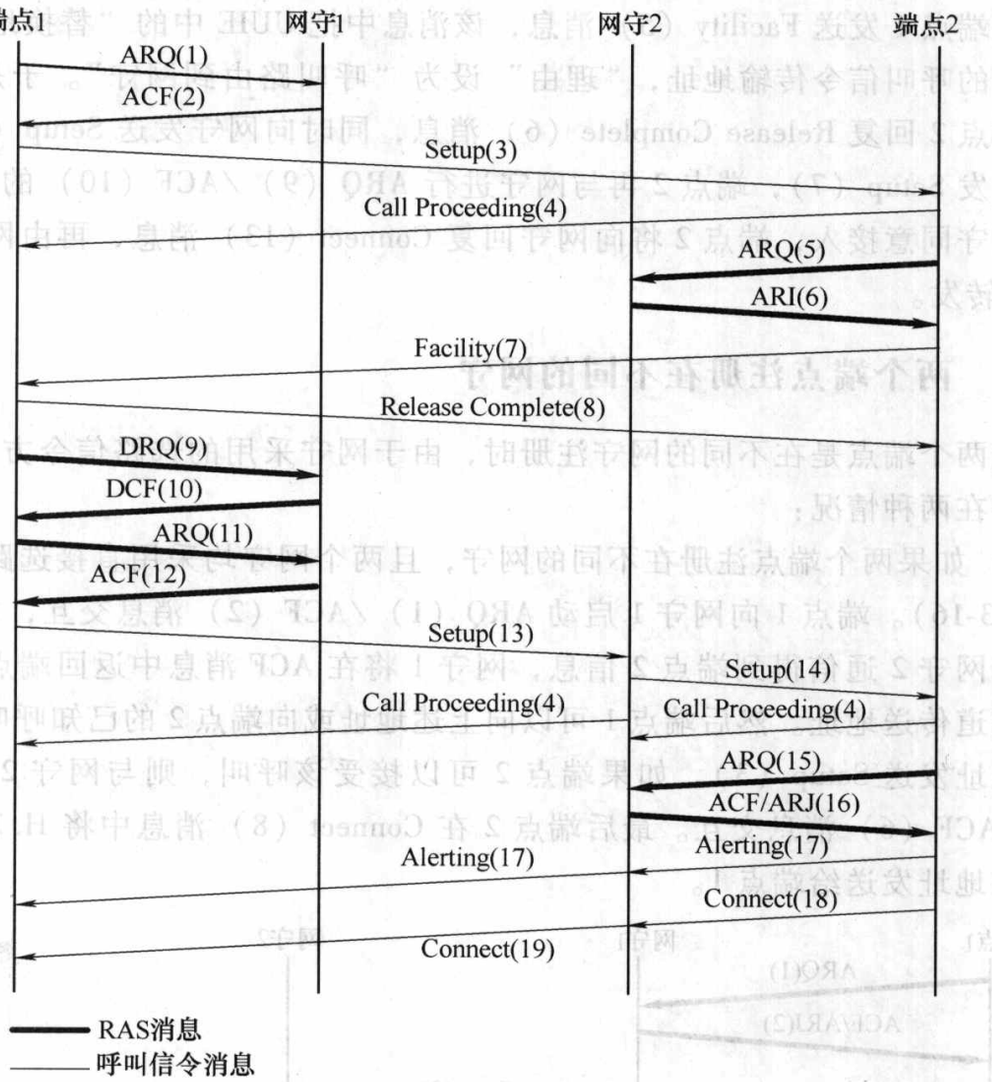


图 3-17 两个端点在不同网守注册——直接/转发呼叫信令

到端点 2 的地址信息，网守 1 将在 ACF（2）消息中返回端点 2 的呼叫信令信道传送地址；

② 然后端点 1 可以向上述地址或向端点 2 的已知呼叫信令信道传送地址发送 Setup（3）；

③ 如果端点 2 可以接受该呼叫，则向网守 2 启动 ARQ（5）消息；

④ 如果网守同意接入，由于采用网守选路呼叫信令方式，网守 2 将向端点 2 发送 ARJ（6），消息中回复网守的呼叫信令传输地址，拒绝原因是“呼叫路由到网守”；

⑤ 端点 2 收到消息后会向端点 1 发送 Facility（7）消息，该消息中把 UUIE 中的“替换地址”设置为网守的呼叫信令传输地址，“理由”设为“呼叫路由到网守”；

⑥ 端点 1 向端点 2 回发 Release Complete（8）消息；

⑦ 向网守 1 发送 DRQ（9），网守 1 响应 DCF（10）；

⑧　端点 1 重新向网守 1 启动 ARQ（11）/ACF（12）消息交互过程；

⑨　端点 1 向网守 2 的呼叫信令信道传送地址发送 Setup（13）消息；

⑩　网守 2 向端点 2 发送 Setup（14）；

⑪　端点 2 向网守 2 启动 ARQ（15）/ACF（16）程序，最后端点 2 在 Connect（18）消息中将 H. 245 控制信道传送地址发送至网守 2。网守 2 向端点 1 发送 Connect（19）消息，其中包含端点 2 的 H. 245 控制信道传送地址还是网守的 H. 245 控制信道传送地址取决于网守是否选择转发 H. 245 控制信道方式。

两个端点在不同网守注册，网守 1 采用网守选路信令方式，网守 2 采用直接呼叫信令方式（见图 3-18）。呼叫建立过程如下：

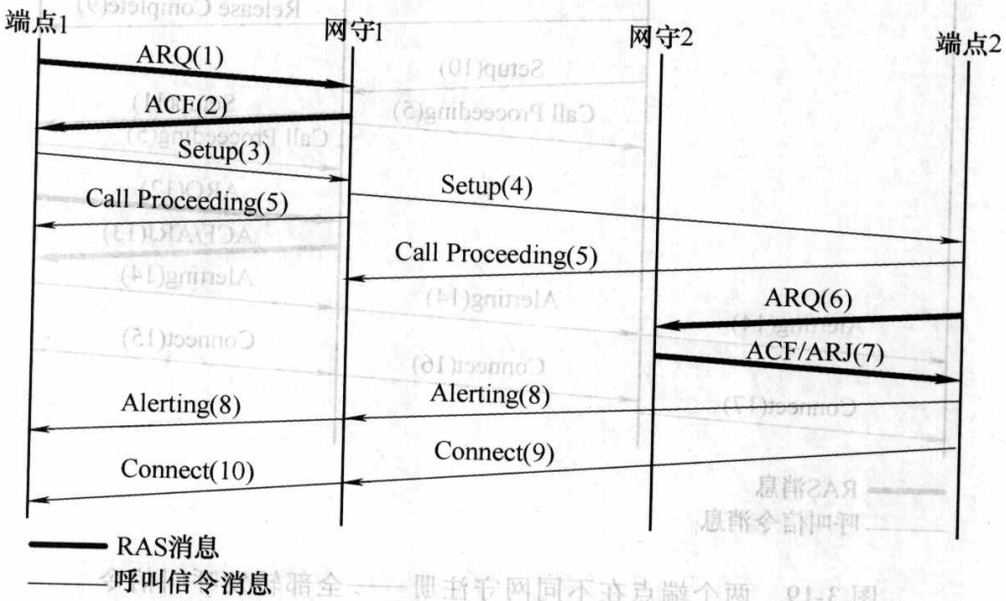


图 3-18　两个端点在不同网守注册——转发/直接呼叫信令

①　端点 1 与网守 1 进行 ARQ（1）/ACF（2）消息交互过程，网守在回送的 ACF（2）消息中将自己的呼叫信令信道传送地址送给端点 1；

②　端点 1 收到地址后，发送 Setup（3）到网守 1；

③　网守 1 通过端点 2 已知呼叫信令信道传送地址向端点 2 发送 Setup（4），同时将网守自己的呼叫信令信道传送地址传给端点 2；

④　如果端点 2 接受此呼叫，则向网守 2 启动 ARQ（6）/ACF（7）消息交互过程；

⑤　端点 2 向网守 1 发送 Connect（9）消息，其中包含自己的 H. 245 控制信道传送地址；

⑥　网守 1 向端点 1 发送 Connect（10）消息，但 Connect（10）消息中包含的 H. 245 控制信道是端点 2 的还是网守 1 的，是取决于网守 1 是否选择转发 H. 245 控制信道方式的。

两个端点在不同网守注册，两个网守均采用转发呼叫信令方式（见图 3-

19）。通信过程如下：

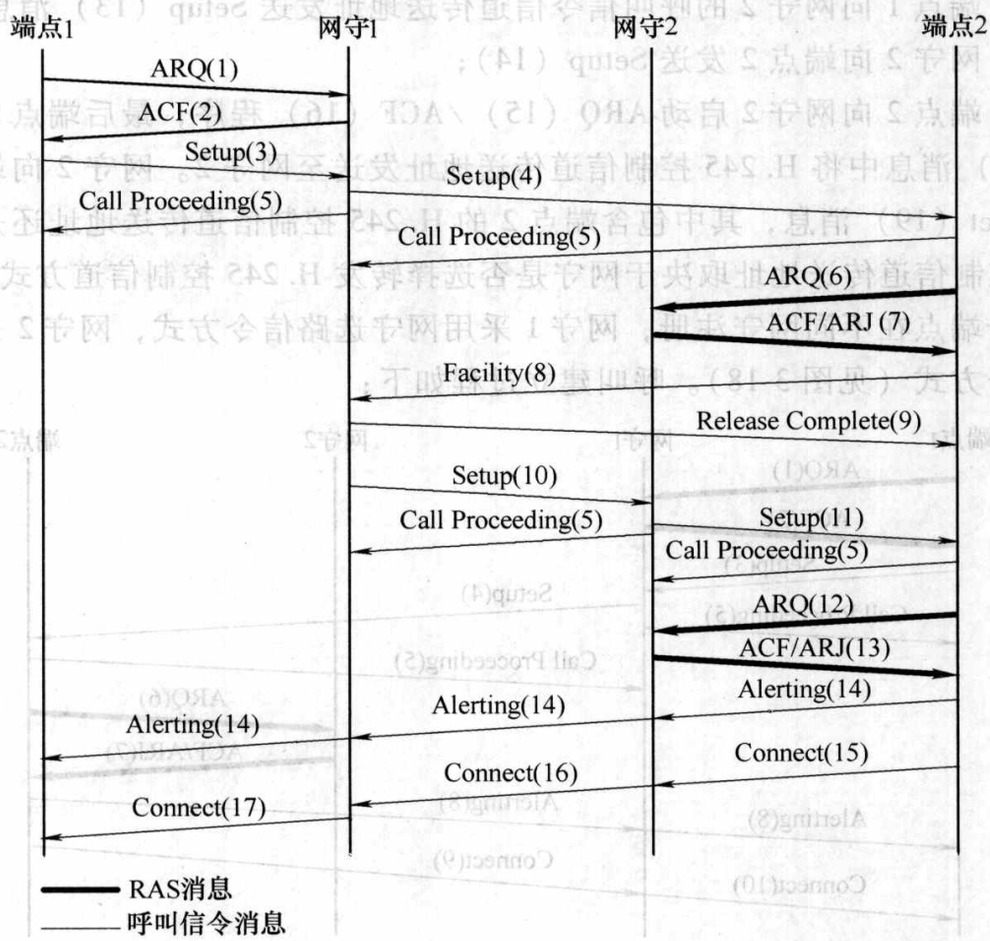


图 3-19 两个端点在不同网守注册——全部转发呼叫信令

① 端点 1 与网守 1 进行 ARQ（1）/ACF（2）消息交互过程，网守在回送的 ACF（2）中将自己的呼叫信令信道传送地址送给端点 1；

② 端点 1 收到地址后，向网守 1 发送 Setup（3）；

③ 网守 1 通过端点 2 的已知呼叫信令信道传送地址发送 Setup（4）；

④ 如果端点 2 接受此呼叫，则与网守 2 进行 ARQ（6）/ACF（7）消息交互；

⑤ 如果网守同意接入，由于采用网守选路呼叫信令方式，网守 2 将向端点 2 发送 ARJ（7），消息中回复网守的呼叫信令传输地址，拒绝原因是“呼叫路由到网守”；

⑥ 端点 2 向网守 1 发送 Facility（8）消息，包含网守 2 的呼叫信令信道传送地址；

⑦ 然后，网守 1 向端点 2 发送 Release Complete（8）消息；

⑧ 网守 1 根据网守 2 的呼叫信令信道传送地址向网守 2 发送 Setup（10）消息；

⑨ 网守 2 向端点 2 转发 Setup（11）消息；

⑩　端点 2 同网守 2 进行 ARQ（12）/ACF（13）消息交互；

⑪　然后端点 2 向网守 2 发送 Connect（15）消息，其中包含自己的 H. 245 控制信道传送地址；

⑫　网守 2 向网守 1 发送 Connect（16）消息，但 Connect 消息中包含的 H. 245 控制信道是端点 2 的还是网守 2 的，是取决于网守 2 是否选择转发 H. 245 控制信道方式的；

⑬　网守 1 向端点 1 发送 Connect（17）消息，但同样 Connect 消息中包含网守 1 的 H. 245 控制信道传送地址还是网守 2 的 H. 245 控制信道传送地址取决于网守 1 是否选择转发 H. 245 控制信道方式。

3.5.7　可选的被叫端点信令

如果主叫端点或主叫端点的网守仅仅知道被叫端点的呼叫信令信道地址（见图 3-20）。则主叫所注册的网守 1 为寻找端点 2 将向网络组播 LRQ（2）。此

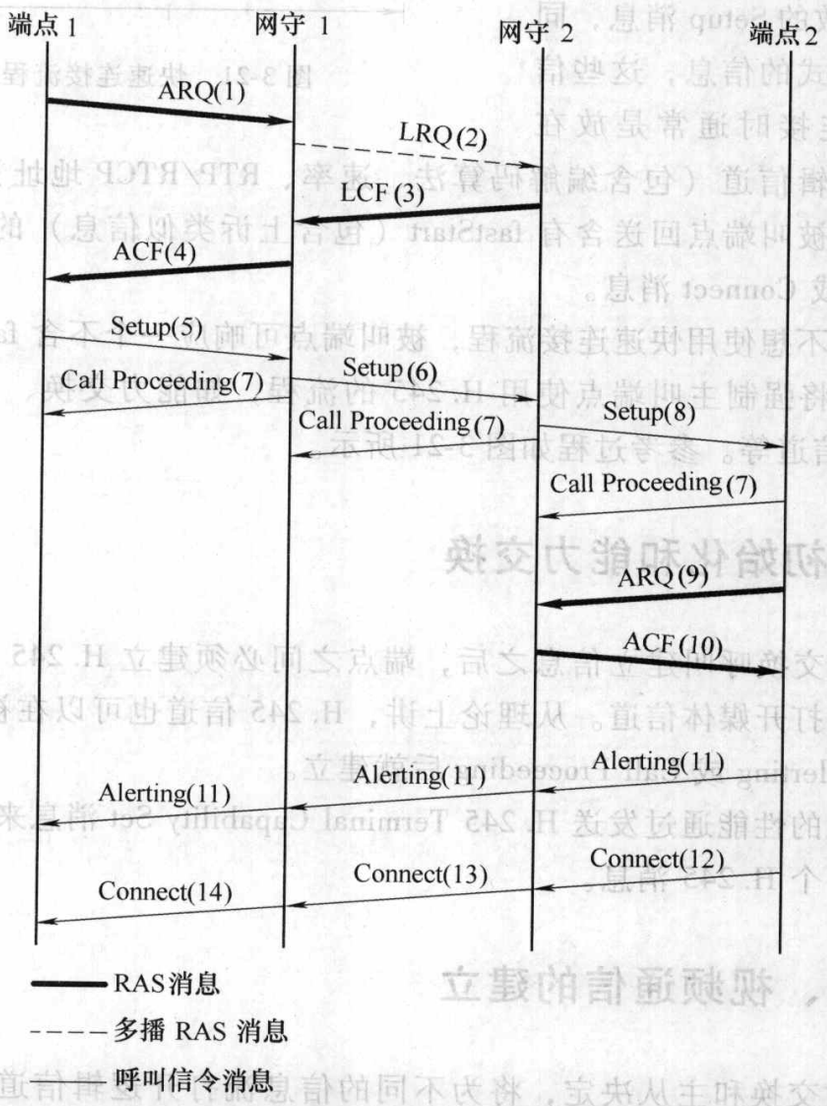


图 3-20　可选的被叫端点呼叫信令

时网守 2 响应 LCF（3），并告知自己的呼叫信令信道传送地址。这样，网守 1 可以利用这个地址同网守 2 进行消息交互了，具体消息交互过程如图 3-20 所示。

3.5.8 快速连接流程

快速连接流程是 H.323 第二版提出的一个新流程。快速连接就是一个端点呼叫另一个端点时比较快，即呼叫流程仅采用 Setup-Connect 流程。如果仅主叫端点支持快速连接方式，而被叫端点不支持，则被叫可以拒绝快速连接模式，即将呼叫降至非快速连接的方式，继续正常呼叫。当采用快速连接模式时，主叫方向被叫方发送带有 fastStart 参数的 Setup 消息，同时提供音频模式的信息，这些信息在非快速连接时通常是放在 H.245 打开逻辑信道（包含编解码算法、速率、RTP/RTCP 地址）中传送。收到此信息后，被叫端点回送含有 fastStart（包含上诉类似信息）的 Call Proceeding、Alerting 或 Connect 消息。

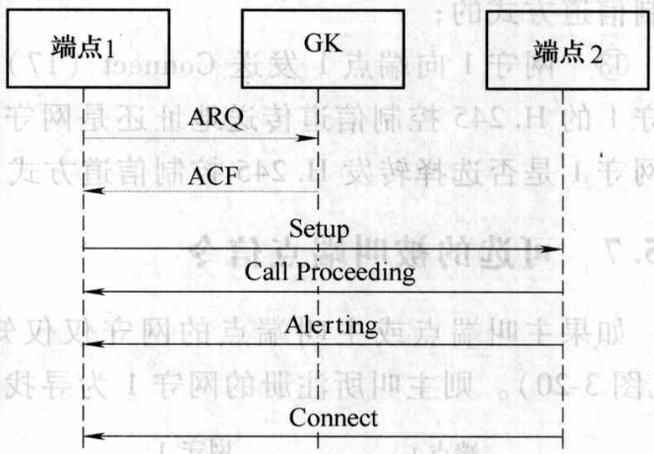


图 3-21 快速连接流程

如果被叫不想使用快速连接流程，被叫端点可响应一个不含 fastStart 的 Connect 消息。这将强制主叫端点使用 H.245 的流程，如能力交换、主从决定、打开/关闭逻辑信道等。参考过程如图 3-21 所示。

3.6 通信初始化和能力交换

一旦双方交换呼叫建立信息之后，端点之间必须建立 H.245 控制信道，进行能力交换和打开媒体信道。从理论上讲，H.245 信道也可以在被叫收到 Setup 且主叫收到 Alerting 或 Call Proceeding 后就建立。

端点系统的性能通过发送 H.245 Terminal Capability Set 消息来交换，这个消息必须是第一个 H.245 消息。

3.7 音频、视频通信的建立

随着能力交换和主从决定，将为不同的信息流打开逻辑信道。音频、视频流在建立的逻辑信道上传送，基于不可靠的信道；数据通信将在可靠的信道上

传送。

3.8　呼叫服务

在通信过程中，带宽修改流程如图 3-22 和图 3-23 所示。当端点希望改变呼叫带宽时，向网守发送 BRQ（1）消息。如果网守拒绝带宽修改请求，则向端点回送 BRJ（2）；如果网守同意带宽改变的请求，则回送 BCF（2）。

在图 3-22 中，如果端点增加一个逻辑信道的传输速率，它首先需判断带宽是否会超出。如果超出，端点 1 将向网守 1 请求改变带宽。当呼叫带宽满足要求后，端点 1 发送 Close Logic Channel（3）以关闭逻辑信道。然后再发送 Open Logic Channel（4）用于新的速率。如果被叫端点接受这个新的速率，必须首先判断带宽是否会超出。如果超出，端点 2 将向它的网守请求改变带宽。当带宽满足要求后，端点 2 发送 Open Logic ChannelAck（7），否则响应 Open Logic Channel Reject，表示不接受新的速率。

图 3-22　带宽改变请求——发送端改变

在图 3-23 中，如果端点 1 增加一个逻辑信道的传输速率，而该逻辑信道在端点 2 侧的流量控制是处于低速率门限，此时端点 1 首先需判断带宽是否会超出。如果超出，端点 1 将向网守 1 请求改变带宽。当呼叫带宽满足要求后，端点 1 发送 Flow Control Command（3），要求一个高的接收流量速率门限。如果端点 2 接受这个新的速率，它首先判断带宽是否会超出。如果超出，端点 2 将向它的网守请求改变带宽。当带宽满足要求后，端点 2 将发送 Close Logic Channel（6）以关闭逻辑信道。然后再发送 Open Logic Channel（7）用于新的速率。端点 1 接受这个新的速率，并回送 OpenLogic Channel Ack（8）。

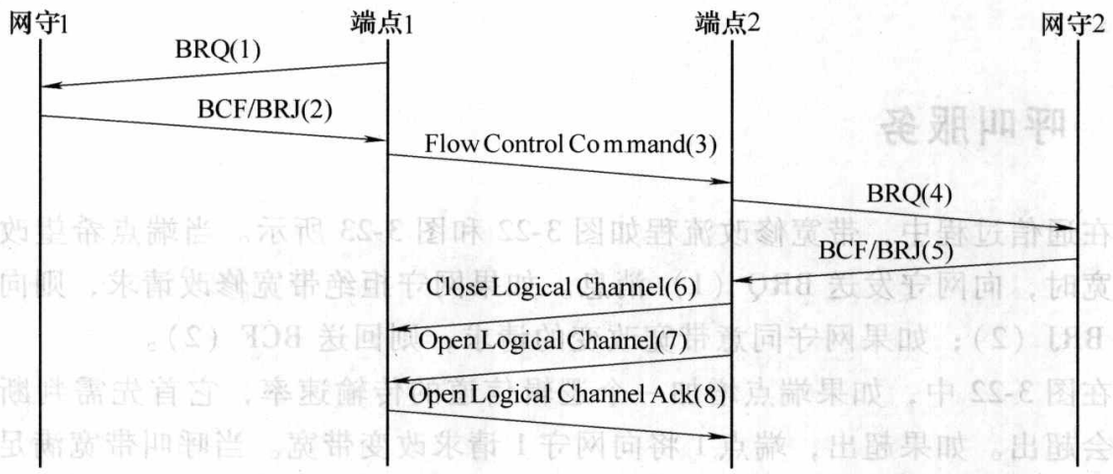


图 3-23 带宽改变请求——接收端改变

3.9 H.245 协议

H.245 作为 H.323 协议族中的媒体控制协议，定义了两类信道：

1）控制信道（H.245 信道）：位于不同 H.323 实体上的两个 H.245 对等信令实体通过该信道传送 H.245 消息，以控制媒体信道的建立和释放。是可靠的信道（TCP，端口号动态分配）。每个呼叫有且仅有一个 H.245 控制信道，它在整个呼叫期间始终存在，直到呼叫完成后才予释放。

2）通信信道（媒体信道，逻辑信道）：在其上传送用户通信信息。两个实体之间可以有多条逻辑信道，在呼叫中可以根据需要随时建立和释放。逻辑信道的开关由 H.245 控制协议完成，每个逻辑信道在打开时赋予一个标识号。可认为控制信道是一个特殊的永久逻辑信道，其信道号指定为 0。

在 H.323 中，大多数的逻辑信道都是单向信道。T.120 协议数据传输中要求双向信道，则可由一对单向逻辑信道构成，占用两个逻辑信道。而 H.245 中既支持单向信道的建立，也支持双向信道的建立，协议规定传送音视频信号的逻辑信道为不可靠信道（UDP），传送数据信号的为可靠信道（TCP），端口号动态分配。一个 H.323 呼叫的基本控制如呼叫建立、能力交换、逻辑信道打开、信道关闭和呼叫释放等过程，除呼叫建立外，基本上都是 H.245 的信令过程。H.245 主要控制过程如下：

1）能力交换，每个逻辑信道都选择传输特定的媒体信息，这些信息具有一定的带宽和算法，因此双方在通信前将对这些参数进行协商，确定参数范围；

2）逻辑信道信令过程，包括逻辑信道的打开和关闭；

3）接收方关闭逻辑信道请求，目的为当接受方发生异常情况时，能向呼叫发起方提出关闭信道的请求，但信道的关闭过程只能由呼叫发起方启动；

4）主从确定过程，主要用于会议通信时存在多 MC 情况下的仲裁，用于避免信令过程中的冲突现象；

5）往返时延确定，可用于检测远端终端是否工作；

6）环路维护，是一个常规的维护过程，经由专用消息通知对方配合进行环路测试，还包含一个环路测试结束命令消息；

7）其他命令和指示。

H. 245 的消息主要分为 4 类：请求消息、响应消息、命令消息和指示消息，主要消息如下。

1. H. 245 协议过程所用的消息见表 3-1。

表 3-1　H. 245 协议过程所用的消息

协议过程	消息名称	消息类型
能力集交换	Terminal Capability Set	请求
	Terminal Capability Set Acknowledge	响应
	Terminal Capability Set Reject	响应
	Terminal Capability Set Release	指示
逻辑信道信令过程	Open Logical Channel	请求
	Open Logical Channel Acknowledge	响应
	Open Logical Channel Reject	响应
	Open Logical Channel Confirm	指示
	close Logical Channel	请求
	close Logical Channel Acknowledge	响应
接收方关闭逻辑信道要求	Request Channel Close	请求
	Request Channel Close Acknowledge	响应
	Request Channel Close Reject	响应
	Request Channel Close Release	指示
主从确定过程	Master Slave Determination	请求
	Master Slave Determination Acknowledge	响应
	Master Slave Determination Reject	响应
	Master Slave Determination Release	指示
往返时延确定	Round Trip Delay Request	请求
	Round Trip Delay Response	响应
时延维护	Maintenance Loop Request	请求
	Maintenance Loop Acknowledge	响应
	Maintenance Loop Reject	响应
	Maintenance Loop Command off	命令

2. H. 245 基本命令消息见表 3-2。

表 3-2 H.245 基本命令消息

消息名称	作 用
Flow Control	流量控制消息的功能是限制一个逻辑信道的比特率或者所有逻辑信道的总的比特率
Send Terminal Capability Set	发送终端能力集的作用是在发生不确定问题的情况下，请求远端终端发送能力集消息
Encryption	用来交换加密能力，命令远端在信道上发送初始矢量
End Session	呼叫终止，终止 H.245 消息的传送
(miscellaneous commands)	其他命令，控制可视电话的一些命令

3. H.245 基本指示消息见表 3-3。

表 3-3 H.245 基本指示消息

消息名称	作 用
Function Not Understood	返回不理解或不支持的请求、响应或命令
Jitter Indication	向发送端告之接收端计算的逻辑信道抖动量
H2250 Maximum Skew Indicaton	向远端指示指定的两个逻辑信道之间的平均时间偏差
User Input	用于传送 DTMF 信号，供和 SCN 互通时使用
(Miscellaneous Indication)	用于可视电话的指示信号

3.10 呼叫结束

H.323 通信过程的结束可以由呼叫双方和网守的任何一方发起，我们分两种情况讨论：

(1) 任何端点可以采用如下流程终止呼叫：

1) 端点 1 在一幅完整图像的末尾停止传送视频信息，然后关闭所有视频逻辑信道，同样关闭数据和音频信道；

2) 端点 1 在 H.245 控制信道中向端点 2 传送“会话结束”(End Session Command) 消息，通知远端，发送端希望终止呼叫，然后停止 H.245 消息的传送；

3) 端点 1 等待接收来自端点 2 的“会话结束”消息，然后关闭 H.245 控制通路，此时，H.245 控制信道关闭；

4) 如果呼叫信令信道存在，可先发送 Release Complete，再关闭呼叫信令信道；

5) 端点 1 和 2 分别同各自网守进行 DRQ/DCF 的消息交互，关闭 RAS 信道。

如果该网络没有网守，则过程进行到上述步骤的第 4 步之后，呼叫就终止了，过程如图 3-24 所示。

(2) 网守可以采用如下流程终止呼叫：

1) 网守可以通过向端点 1 发送 DRQ 终止呼叫；

图 3-24　端点启动呼叫清除

2）端点 1 关闭所有逻辑信道；

3）端点 1 在 H. 245 控制信道中向端点 2 传送“会话结束”消息，通知远端，发送端希望终止呼叫，然后停止 H. 245 消息的传送；

4）端点 1 等待接收来自端点 2 的“会话结束”消息，然后关闭 H. 245 控制通路。此时，H. 245 控制信道关闭；

5）如果呼叫信令信道存在，可先发送 Release Complete，再关闭呼叫信令信道；

6）然后用 DCF 应答网守，关闭 RAS 信道。端点 2 也与网守进行 DRQ/DCF 的消息交互，关闭 RAS 信道。

另一个端点一旦收到 End Session Command，必须随之执行上述流程（见图 3-25）。

图 3-25　网守启动呼叫清除

第 4 章　MGCP 和 H.248 协议

4.1　MGCP 的简介和功能

目前，媒体网关控制器（MGC）对媒体网关（MG）的控制，是通过媒体网关控制协议（MGCP）来完成的，提出这协议的有两个标准，一个是 IETF 组织提出的 MGCP，一个是 Megaco/H.248，而媒体网关控制器之间呼叫信令协议就可以采用 SIP。

MGCP 是 SGCP 和 IPDC 相结合的产物。1998 年 Level 3 提出 IPDC，同时 Telcordia（原来的 Bellcore）提出了 SGCP，普通的基于 H.323 或 SIP 的 IP 电话网关功能复杂，为了建立 PSTN 和 Internet 之间的呼叫连接，网关不但要执行媒体格式变换，还要进行信令转换，以及控制网关内部资源，为每个呼叫建立网关内部的语音通路。这样的网关结构对 IP 电话系统的大规模部署具有相当的制约，主要表现在以下几个方面：

1）扩展性：运营商期望未来 IP 电话系统能支持数百万用户，但现有网关只能支持几千用户，原因就在于网关功能过于复杂；

2）与 PSTN 无缝综合：现有网关只相当于 PSTN 中的一个终端设备，只能提供基本的呼叫业务；

3）可用性：现有网关结构缺乏故障保护机制，难以达到电信级可靠性的要求；

4）No.7 信令能力：现在有许多网关，尤其是 ISP 提供的网关尚不能支持 No.7 信令。

1999 年上半年，IETF 专门进行有关 MGCP 的标准化工作。当时人们希望把以软件为中心的呼叫处理功能和以硬件为中心的媒体流处理功能分离开。那时 H. 323 和 SIP 不能处理这两个分离实体之间的通信，所以就产生了 MGCP。

MGCP 采用和 SIP 类似的协议结构，即文本协议。协议消息分为两类：命令和响应，每个命令都需要接收方的回送响应，采用三次“握手”方式证实。命令消息由命令行和若干参数行组成，相当于 SIP 中的请求行和头部字段行。响应消息带有 3 位的响应码。

在 MGCP 中，采用 SDP 来正式描述各种参数，以便在端点之间建立连接，例如 IP 地址、UDP 端口和 RTP 应用文档等。MGCP 采用 UDP 传送，其目的是减少信令传送时延，为了保证可靠性，必须定义相应的重发机制。目前的 MGCP

设计主要涵盖信令和媒体平面，管理和服务平面还没有广泛涉及。

MGCP 假定了一个呼叫控制结构，其中呼叫控制“智能部分”在媒体网关之外，由外部的呼叫控制部件控制。MGCP 的一个关键属性就在于它是作为分布式系统中的一个内部协议实现的，对外界好像是一个简单的 VoIP 网关。

MGCP 是软交换、媒体网关、信令网关的关键协议，它使 IP 电话网可以接入 PSTN，实现端到端的电话业务。MGCP 是无状态的协议，这个概念是整个设计方法的核心。“无状态”意味着该协议无需状态机描述两个信令实体之间的事务序列，也不保留 MGC 和 MG 之间先前的事务。这点不要与呼叫状态混淆，呼叫状态是存在于 MGC 内。MGC 和 MG 之间的基本的呼叫流存在于它们的基本构造状态机中。

MGCP 侧重的是简单性和可靠性，MGCP 本身只限于处理媒体流控制，呼叫处理等智能工作在软交换场景中由软交换设备承担，使媒体网关成为一个很简单的设备，简化了本地接入设备的设计。

MGCP 通过软交换实现对多业务分组网边缘上的数据通信设备（如 VoIP 网关、VoATM 网关、Cable Modem、机顶盒等）的外部控制和管理。其中的软交换可以分布在多个计算机平台上，从外部控制、管理多媒体网络边缘上的媒体网关，指导网关在端点之间建立连接，监控摘机之类的事件，产生振铃等信号，以及规范端点之间如何或者何时建立连接。

4.2　MGCP 的结构

如图 4-1 所示，MGCP 采用 C/S 结构，主要组成部件包括：一个 MGC（或称为 Call Agent，软交换）和一个以上的 MG，MGC 之间一般采用 SIP 互联，而 MGC 与 MG 之间就是采用 MGCP 来实现信息的传输与控制。

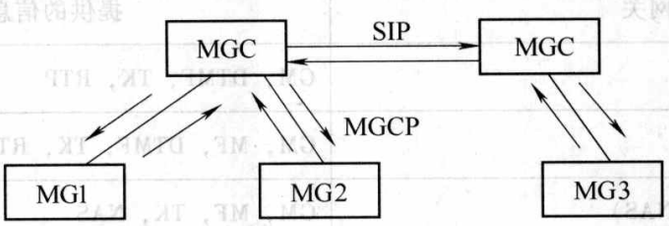


图 4-1　MGCP 采用的网络结构

其中，MG 主要作用是：

1）对 MGC 所下的命令，作出适当的处理和响应，或当有事件发生时告知 MGC；

2）音频或视频信号的压缩与解压缩；

3）在 VoIP 的 IP 网络和传统电话网络之间进行 IP 信息包和语音信号的转

换。

在图 4-1 所示网络体系中，MGC 负责控制所有的 MG，如要注意某一个端点（Endpoint）是否有 Off Hook/On Hook 的事件，通知 MG 准备侦听从端点传来的拨号音、或发出命令请网关回应每一个端点目前的状态、或命令网关对某一个端点建立（或修改、删除）通话。

总之，在 MGCP 里，MG 只要负责几个语音信号的命令，其他控制信号的部分，全部交由 MGC 专门负责。MGCP 的连接模式包括两个部分：端点（Endpoint）和连接（Connection）（下一节将详细讲述）。每一个使用者都会和一个端点相连接，然后再映射成端点之间的连接来传递信息。其示意图如图 4-2 所示。

图 4-2 MGCP 的呼叫原理

MGCP 把网关能侦测出的状态称为一种事件，例如电话拿起（Off Hook）或挂上（On Hook）即为两种事件。MGCP 也定义一些网关能产生的信号，如拨号音。由前面各种端点和网关的介绍可以得知 MGCP 在设计上是可以取代现代电话系统的一些部分，所以必须了解电话系统各个部分所应侦测的事件以及所需产生的信号。

MGCP 定义了一些信息包类型，每一种信息包皆包含一些事件和信号。每一个信息包、事件和信号都有自己的简称。各个信息包的用途以及其所包含的事件和信号都可在 MGCP 相关协议和草案中查到。MGCP 指出各种网关需提供的信息包（见表 4-1），即可以侦测出事件及可以产生的相关信号。

表 4-1 为各个网关需提供的信息包

网关	提供的信息包类型
Trunk gateway（ISUP）	GM，DTMF，TK，RTP
Trunk gateway（MF）	GM，MF，DTMF，TK，RTP
Network Access Server（NAS）	GM，MF，TK，NAS
Combined NAS/VoIP gateway	GM，MF，DTMF，TK，NAS，RTP
Access Gateway（VoIP）	GM，DTMF，MF，RTP
Access Gateway（VoIP + NAS）	GM，DTMF，MF，NAS，RTP
Residential Gateway	GM，DTMF，Line，RTP
Announcement Server	ANN，RTP

4.2.1　MGCP 的端点

MGCP 中端点（Endpoint）的结构如图 4-3 所示。

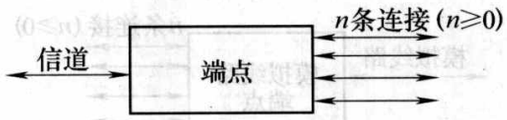


图 4-3　端点的结构图

一个媒体网关可以支持多种端点。端点的类型决定了它的功能。我们可以将端点划分成以下几种基本类型：

1）数字通道（Digital Channel（DS0））；

2）模拟线路（Analog Line）；

3）广播服务器接入点（Announcement Server access point）；

4）交互式语音响应接入点（Interactive Voice Respones access point）；

5）包中继（Packet Relay）；

6）搭线窃听接入点（wiretap access point）；

7）ATM 中继侧的接口。

端点类型并不局限于这几种，在未来还会有其他类型的端点，例如，可以用来检查网络质量的检测端点（Test Endpoint），或用来控制在帧中继电路上传播语音信号的帧中继端点（Frame Relay Endpoint）。

1. 数字通道（DS0）

数字通道提供 8kHz×8bit 的服务，这种通道可以在中继和 ISDN 接口找到。典型的是数字多路传输线路，例如 T1、E1、T3 和 E3。支持这种通道的媒体网关可以将从通道上接收的数字信号转换成语音包，这些数字信号是根据 A 律和 μ 律编码的。当媒体网关可以支持 NAS 时，网关要能既接收音频编码数据，也要能接收二进制数据（ISDN 连接），并将它们转换成数据包。DS0 端点的结构如图 4-4 所示：

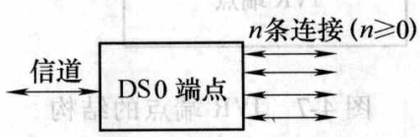


图 4-4　DS0 端点的结构

媒体网关能在端点与网络间或同一个网关的不同端点间建立多个连接。信号将根据连接的类型将附加到连接上。精确的连接数量由网关的特点决定，并根据网关中资源的配置情况而改变。

2. 模拟线路

模拟线路既可以用来作为向一个典型的电话单元提供服务的“客户端”接

口，也可以作为允许网关发送和接收模拟信号的“服务”接口。当网关支持 NAS 时，网关还可以接收音频编码数据并将它们转换成数据包。模拟线路端点的结构如图 4-5 所示。

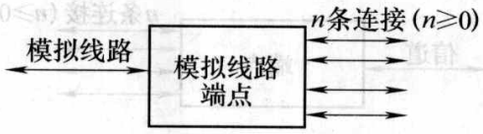


图 4-5 模拟线路端点的结构

媒体网关能在端点与网络间或同一个网关的不同端点间建立多个连接。信号将根据连接的类型将附加到连接上。精确的连接数量由网关的特点决定，并根据网关中资源的配置情况而改变。一个典型的网关每个端点应该可以支持两个到三个连接，以提供“呼叫等待”或“三方呼叫”业务。

3. 广播服务入口

广播服务入口提供广播业务的入口。在 MGC 的要求下，广播服务器将会发出一个专用的广播。广播服务入口端点的结构如图 4-6 所示。

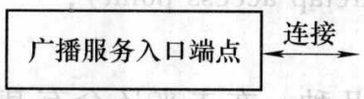


图 4-6 广播服务入口端点的结构

广播服务入口端点一次只能支持一个连接。如果同一个端点上建立了多个连接，那么相同的广播将会在所有的连接上同时发出。到广播服务器的连接是单向的。

4. 交互式语音响应接入点

交互式语音响应接入点（IVR）提供 IVR 业务的服务入口。在 MGC 的要求下，IVR 服务器将发出广播和语音，并监听用户的响应。

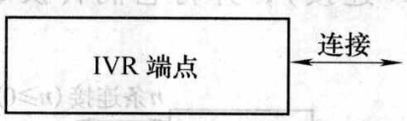


图 4-7 IVR 端点的结构

IVR 端点一次只支持一个连接。如果同一个端点上建立了多个连接，那么相同的广播将会在所有的连接上同时发出。

5. 会议桥接器接入点

会议桥接器接入点用来提供到某个会议的连接入口。会议桥接器接入点端点的结构如图 4-8 所示。

媒体网关能在端点与网络间或同一个网关的不同端点间建立多个连接。信号将根据连接的类型附加到连接上。精确的连接数量由网关的特点决定，并根

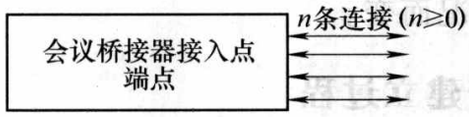


图 4-8　会议桥接器接入点端点的结构

据网关中资源的配置情况而改变。

6. 包中继

包中继端点是一种只支持两个连接的特殊的会议桥接器，它可以在保护的和开放的网络间，或在用来为两个不匹配的网关提供协作的转换代码的服务器中找到。其结构如图 4-9 所示。

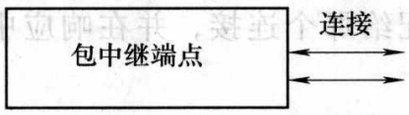


图 4-9　包中继端点的结构

7. 搭线窃听接入点

搭线窃听接入点用来提供录音和现场回放业务。其端点的结构如图 4-10 所示。

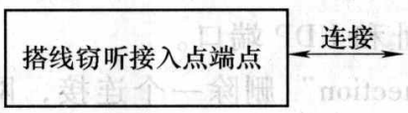


图 4-10　搭线窃听接入点端点的结构

搭线窃听接入端点一次只支持一个连接。如果同一个端点上建立了多个连接，那么相同的广播将会在所有的连接上同时发出。

到搭线窃听接入端点的连接是单向的（半双工）——广播服务器将不收听来自连接的音频信号。

8. ATM 中继侧的接口

ATM 中继侧的接口将会在一个或多个用来代替 TDM 中继交换机的 ATM 永久虚电路中找到。当使用 ATM/AAL2 时，中继线路或通道将会附加在一个电路上，每个中继与一个端点对应。ATM 中继端点结构如图 4-11 所示。

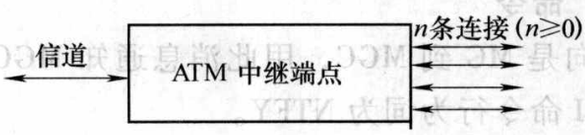


图 4-11　ATM 中继端点的结构

媒体网关能在端点与网络间或同一个网关的不同端点间建立多个连接。信号将根据连接的类型将附加到连接上。精确的连接数量由网关的特点决定，并

根据网关中资源的配置情况而变。

4.2.2 MGCP 的连接建立过程

每一个连接由一个连接标识符标识，并拥有一些连接属性。

（1）当两个端点由同一个 MGC 控制，连接的建立将通过以下三步建立：

1）MGC 让第一个网关在一个端点上建立连接，网关将资源分配给那个连接，并在创建命令的响应中携带会话描述。会话描述包括第三方要向新建立的连接发送包的必要的信息，例如 IP、UDP 端口、包的参数。

2）MGC 让第二个网关在端点上创建连接，命令中携带着第一个网关提供的会话描述，网关将资源分配给那个连接，并在响应中带上第二个连接的会话描述。

3）MGC 通过“modify connection”命令将第二个连接的会话描述提供给第一个端点，联系就是双向的了。

（2）当两个端点由不同的 MGC 控制，这两个 MGC 将通过 MGC 到 MGC 的信令协议（如 SIP）交换信息，以同步两个端点的连接建立。一旦连接建立，连接参数可以在任何时候被“modify connection”命令改变。例如，MGC 会让网关改变压缩方案，或改变 IP 地址和 UDP 端口。

MGC 通过“delete connection”删除一个连接，网关也可以通知连接不能持续。

4.2.3 MGCP 的命令种类

MGCP 采用的命令有如下的几种：

1. 端点配置（Endpoint Configuration）命令

该命令的发送方向是 MGC 到 MG，它用于指示网关某端点电路侧的编码特征。其 ASCII 命令行为词为 EPCF。

2. 通知请求（Notification Request）命令

该命令的发送方向是 MGC 到 MG，它用以请求网关监视某端点发生的某些事件，如发生则通知 MGC。其 ASCII 命令行为词为 RQNT。

3. 通知（Notify）命令

该命令的发送方向是 MG 到 MGC，用此消息通知 MGC，请求监视的某些事件已经发生。其 ASCII 命令行为词为 NTFY。

4. 创建连接（Create Connection）命令

该命令的发送方向是 MGC 到 MG，用此命令将某端点和指定的 IP 地址和 UDP 端口关联。其 ASCII 命令行为词为 CRCX。

5. 修改连接（Modify Connection）命令

该命令的发送方向是 MGC 到 MG，MGC 用此命令改变先前已建立连接的参数。其 ASCII 命令行为词为 MDCX。

6. 删除连接（Delete Connection）命令

一般情况下，该命令由 MGC 发往 MG，指示删除已有的连接。其 ASCII 命令行为词为 DLCX。

7. 审计端点（Audit Endpoint）命令

该命令的发送方向是 MGC 到 MG，MGC 用此命令获得某端点或一组端点的详细信息。其 ASCII 命令行为词为 AUEP。

8. 审计连接（Audit Connection）命令

该命令的发送方向是 MGC 到 MG，MGC 通过该命令检索某端点上某连接的信息，该连接用连接标识符识别。其 ASCII 命令行为词为 AUCX。

9. 重启动在进行中（Restart Inprogress）命令

该命令的发送方向是 MG 到 MGC，网关用此命令告知某端点或一组端点退出服务或投入服务。其 ASCII 命令行为词为 RSIP。

这些命令使 MGC 可以指挥网关在其端点上创建连接，并在网关发现事件时能得到通知。

4.2.4　返回码和出错码

所有 MGCP 命令都是有确认的，确认中带着一个返回码，指明了命令的状态。返回码是一个整数，定义了 4 种范围：

1）从 100 到 199 之间的值表示临时的响应；

2）从 200 到 299 之间的值表示命令成功完成；

3）从 400 到 499 之间的值表示暂时性错误；

4）从 500 到 599 之间的值表示永久性错误。

已具体定义的码值见表 4-2。

表 4-2　返回码值

码值	错　误　类　型
100	事务正被执行，确实的完成消息随后会发来
200	请求的事务执行正常
250	连接被撤除
400	事务由于暂时性错误不能被执行
401	电话已摘机
402	电话已挂机
403	由于端点当前没有足够的资源，事务不能被执行
404	目前带宽不足

（续）

码值	错 误 类 型
500	事务由于端点未知而不能执行
501	事务由于端点未准备好而不能执行
502	由于端点没有足够的资源，事务不能执行
510	由于发现协议错误，事务不能执行
511	由于命令包含一个无法识别的扩展，事务无法执行
512	由于网关没有配备能监测某一事件的设备，事务无法执行
513	由于网关没有配备能产生某一信号的设备，事务无法执行
514	由于网关不能发送指定的通知，事务无法执行
515	事务涉及一个错误的连接标识（可能是已被撤除的连接）
516	事务涉及一个未知的呼叫标识
517	不被支持的或非法的模式
518	不被支持的或未知的包
519	端点没有数字映像
520	由于端点正在“重启”，事务无法执行
521	端点重定向到另一个呼叫代理
522	没有这样的事件或信号
523	未知的动作或非法的动作组合
524	存在于 LocalConnectionOption 中的内部矛盾
525	在 LocalConnectionOption 有未知的扩展
526	带宽不足
527	缺 RemoteConnectionDescriptor 参数
528	不兼容的协议版本
529	内部硬件故障
530	CAS 信令协议错误
531	干线分组失败（例如功能故障）

4.2.5 原因代码

当网关撤除一个连接时，用原因代码来告知呼叫代理撤除该连接的原因。原因代码也可能用在 RestartInProgress 命令中，来告知网关重启的原因。原因代码是一个整数值，已定义的值如下：

000 端点状态正常（该码值只在响应查询请求时使用）；

900 端点故障；

901　端点停止服务；

902　失去底层的连通性（例如下游同步）。

4.2.6　MGCP 的消息格式

MGCP 消息格式和 SIP 消息类似。所有命令由命令头部和会话描述两部分组成，所有响应包由响应头部和会话描述两部分组成。头部和会话描述都由若干文本行组成，两部分之间由空行分割。会话描述为任选部分。

协议采用事务标识关联命令和响应，其取值为 1 ~ 999999999。命令完成后 3min 内，已用的事务标识不能给新的命令使用。

头部和会话描述用一组字符行编码，用回车或换行符分隔。头部和会话描述由一个空行分隔。

MGCP 用事务标识来关联命令和响应。事务标识作为头部的一部分编码，作为响应头部的一部分重复。

1. 命令头部

命令头部由以下部分组成：

1）命令行，命令行包括 4 个部分：命令行为词、事务标识、执行该命令的端点名和协议版本；

2）一组参数行，由参数名和参数值组成。

除非其他注明或其他引用的标准的规定，命令头部的每个成分对大小写不敏感的。这对动词、参数和值都适用。

2. 命令行

命令行由以下部分组成：

1）被请求的动词名；

2）事务的标识；

3）执行该命令的端点的名字；

4）协议版本。

这 4 条被编码成可打印的 ASCII 字符串，由空格或制表符分隔，建议用一个 ASCII 空格。

3. 会话描述的编码

会话描述按照 SDP 编码。MGCP 的实现要求能完全产生任意符合 SDP 的消息，能发送严格遵循 SDP 标准的会话描述。SDP 的用法取决于由“mode”参数指定的正在使用的会话的类型：

1）如果模式设置为“Data”，则会话描述部分描述数据访问业务的配置。

2）如果模式设置为其他值，则会话描述用于一个语音业务。

对一个语音业务，将考虑 SDP 提供的“audio”媒体的信息。对一个数据业

务，网关将考虑“network-access”媒体的信息。

4. MGCP 的参数行

参数行由参数名、冒号、一个空格和参数值组成。参数名大部分情况下由一个大写字符组成。可以在命令中出现的参数定义见表 4-3。

表 4-3 参数定义

参 数 名	参 数 值
ResponseAck	K
BearerInformation	B
CallId	C
ConnectionId	I
NotifiedEntity	N
RequestIdentifier	X
LocalConnectionOptions	L
Connection Mode	M
RequestedEvents	R
SignalRequests	S
DigitMap	D
ObservedEvents	O
ConnectionParameters	P
ReasonCode	E
SpecificEndpointID	Z
Second Endpoint ID	Z2
SecondConnectionId	I2
RequestedInfo	F
QuarantineHandling	Q
DetectEvents	T
RestartMethod	RM
RestartDelay	RD

4.2.7 UDP 上的传输

MGCP 消息在 UDP 上传输，命令发送到 DNS 为指定端点定义的 IP 地址中的一个，响应发回命令的源地址。

当没有为端点指定端口，命令将发送到：

1）呼叫代理发送到网关的默认 MGCP 端口 2427；

2）网关发送到呼叫代理的默认 MGCP 端口 2727。

在 UDP 上传输的 MGCP 消息可能会丢失。如果没有收到及时的响应，命令将重发。

MGCP 实体应该在内存中保存一个对最近事务的响应的列表和一个目前正在执行的事务的列表。把输入命令的事务标识符和最近响应的事务标识符比较，如果发现一个匹配，则 MGCP 实体不执行该事务，但是要重发响应。剩下的命令将与当前事务列表比较，如果发现一个匹配，则 MGCP 实体不执行该事务，简单地忽略它。

这个过程使用一个长定时器值，下面称为“LONG-TIMER”（长时计时器）。该计时器设置应该比最大地持续时间长，应该考虑最大地重发次数、重发定时器的最大值和一个包在网络上的最大传输时延。一个建议值是 30s。

事务标识符是某个小于 999 999 999 的整数。呼叫代理可以决定为它管理的每个网关指定一个整数区间，或者属于任意组的网关使用相同的整数区间。呼叫代理可以决定分担管理几个独立进程间的网关的负荷。这些进程将共享相同的事务区间。这种共享有多种实现，例如，事务标识符的集中分配，或预先分配不重叠的标识符范围给不同的进程。这些实现必须确保所有的从一个逻辑 MGC 起源的事务分配到不同的标识符。网关只要看事务标识符就能发现重复的事务。

任何命令都有响应确认属性，它携带一组“已确认的事务标识符范围”。

MGCP 可以选择删除包括在“响应确认属性”的“已确认事务标识符范围”内的事务的响应的副本。当事务标识符落在这个区间时，它们只需丢弃来自呼叫代理的进一步命令。

当网关发送最后一个响应给呼叫代理 LONG-TIMER 秒后，或者一个网关重新运行，“已确认事务标识符范围”值将不能再使用。在这种情况下，命令将被接收和处理，不用测试事务标识符。

携带“响应确认属性”的命令可能不按顺序传输。网关应该保留在最近命令收到的“响应确认属性”的合并结果。

4.2.8　状态失效修复能力和竞态情况

为了实现恰当的呼叫信令功能，呼叫代理必须跟踪端点的状态，而网关必须确保事件恰当地通报给呼叫代理。当网关或呼叫代理重启时会有特殊的情形：网关必须在失效修复过程中被重定向到一个新的呼叫代理，呼叫代理在网关离线或重启时必须采取特殊动作。

4.3 Megaco/H.248 协议的简介和功能

4.3.1 Megaco/H.248 简介

Megaco（也就是 H.248）是一项作为 ITU-T 与 IETF 合作结果的新标准，用于连接 PSTN 与 IP 网络，它使语音、传真和多媒体信号在 PSTN 与新兴的 IP 网络之间进行交换成为可能。Megaco 的含意是媒体网关控制。Megaco 框架可以使服务提供商具有提供多种电话和数据融合服务的能力。

4.3.2 Megaco 的产生

1998 年 5 月 IETF 首先提出了将网关功能分解为媒体网关控制器（MGC）、媒体网关（MG）和信令网关（SG）三部分，提出了 SGCP。

几周后，Level 3 成立 TAC，TAC 在 SGCP 的基础上进行改进，提出了 IPDC 协议。1998 年秋，IETF 综合了 SGCP 和 IPDC，提出了 MGCP，MGCP 受到广泛重视。不久，MDCP 被提交到 ITU-T 第 16 研究组。2000 年，在很大程度上借鉴了 MGCP 和 MDCP 的思想后，ITU-T 第 16 研究组和 IETF 的 Media Gateway Control（Megaco）工作组共同开发 Megaco/H.248 协议。网关控制协议的演变过程如图 4-12 所示。

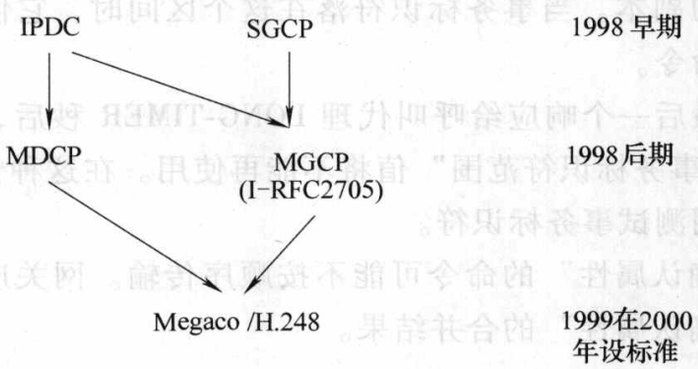


图 4-12 网关控制协议家族树

4.3.3 Megaco 的网络结构

Megaco 的网络结构如图 4-13 所示。

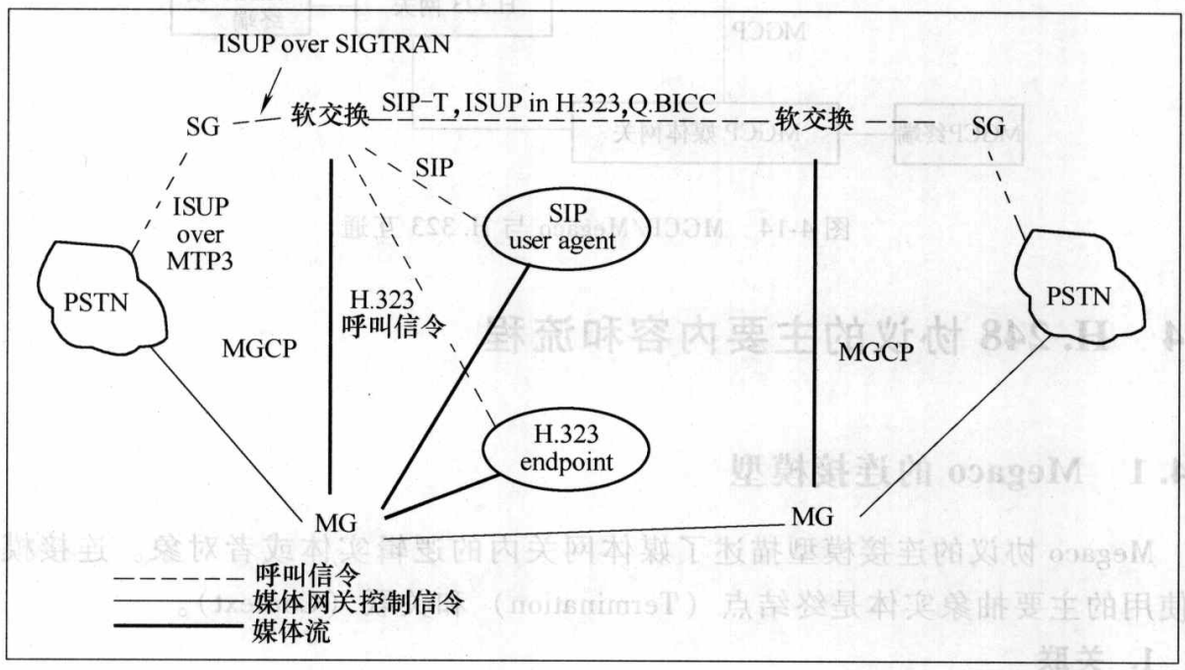


图 4-13　Megaco 的网络结构

4.3.4　Megaco 和其他协议的关系

MGCP 与 H.323 和 SIP 不在同一层面上。H.323 建议和 SIP 提出了两套 IP 电话体系结构。二者是完全独立的，它们不可能互相兼容，不可能一个包含另一个，二者之间只是存在互通问题。MGCP 不涉及 IP 电话的体系结构，只涉及网关分解问题，因而它不仅可以用于 H.323 的 IP 电话系统，也可以用于 SIP 的 IP 电话系统。其次，在网关分解方面，IETF 和 ITU-T 配合得很好。在两大组织直辖下，目前一个正式标准已经完成，这就是 H.248 建议。H.248 中引入了“context”概念（最早在 MDCP 中引入此概念），另外，增加了许多包的定义，从而将 MGCP 大大推进了一步。应该说 H.248 建议已经取代了 MGCP，成为 MGC 与 MG 之间的协议标准了。

MGCP/Megaco 适合与 SIP 或者 H.323 一起构建大规模的系统，但是 MGCP/Megaco 并不是为了取代 SIP 或者 H.323。MGCP/Megaco 不是一个对等的建立呼叫和撤消呼叫的协议。在 SIP 或 H.323 中每一个设备都是对等的，一个设备可以发起一个呼叫到另一个设备，反之依然。但是，在 MGCP/Megaco 中，有一个电话服务器，称为呼叫代理或者 MGC，通过它来建立 MG 之间的呼叫。MGCP/Megaco 与 H.323 互通如图 4-14 所示。

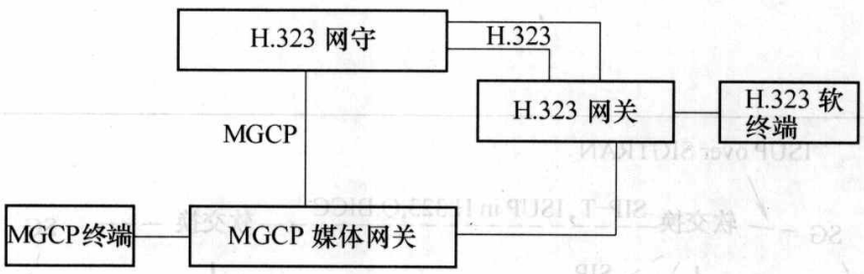


图 4-14 MGCP/Megaco 与 H. 323 互通

4.4 H. 248 协议的主要内容和流程

4.4.1 Megaco 的连接模型

Megaco 协议的连接模型描述了媒体网关内的逻辑实体或者对象。连接模型中使用的主要抽象实体是终结点（Termination）和关联（Context）。

1. 关联

关联（Context）首次提出是在 MDCP 中，它使得协议的灵活性和可扩展性更好，H. 248 建议延用了这个概念。关联是一次呼叫或一个会话中的终结点的集合，一个关联代表一次呼叫或一个会话中的媒体类型。例如从 SCN 到 IP 呼叫的关联包含 TDM 音频终结点和 RTP 音频流终结点。如果媒体网关只提供点到点连接的话，则只允许每个关联最多有两个终结点，若支持多点会议则可以有多个。

关联的属性包括：

1）ContextID：用来惟一的标识某个关联。

2）拓扑：描述同一关联中不同终结点间的媒体流向。

3）优先权：高优先级关联首先处理。

4）紧急呼叫指示器：用来处理紧急意外呼叫。

有一个特殊的关联，称作 null 关联。包括了与任何其他结点都没有联系的终结点。

2. 终结点

终结点（Termination）是 MG 中的逻辑实体，负责发起和接收媒体流或控制流。终结点是源媒体流或目的媒体流客体，终结点可以用于表示时隙、模拟线和 RTP 流等。代表物理实体的终结点有半永久的存活期。当 Add 或 Subtract 时，它被从 null 关联中取出或添入。表示临时信息流（例如 RTP 流）的终结点，则只在使用期间存在，由 Add 命令创建，Subtract 命令消除。

终结点由一系列特性描述，这些特性组成一系列包含在命令里的描述符

(Descriptor)。在 MG 创建终结点时，分配给一个惟一的标识符。

终结点有相应的信令，由 MG 生成，比如语音和提示音。终结点可通过编程来监测事件，并能触发给 MGC 的通知消息或者 MG 的动作。终结点上可以累计统计数据，并通过请求报告给 MGC。

4.4.2　Megaco 的描述符

描述符（Descriptor）是一种协议文法的元素，与属性相关。例如 MGC 通过在命令里包含适当的描述符就可以设置 MG 上的媒体流属性。命令里的参数称为描述符。例如：通过在一个命令中包含适当的描述符，软交换设备能够设置媒体网关中的媒体流的特性。一个描述符由一个名称及一组项组成，某些项还具有值，描述符也可以作为命令的输出结果。表 4-4 列出了所有可能的描述符及其描述。

表 4-4　描述符列表

描述符名称	功能描述
Modem	标识 Modem 类型和特性
Mux	描述多媒体终结点和形成输入 Mux 的终结点的复用类型
Media	媒体流规范的列表
TerminationState	与特定媒体流无关的终结点特性
Stream	对应于单个媒体流的 Remote/Local/LocalControl 描述符的列表
Local	包含对媒体流进行说明的一些特性，其中这些媒体流是媒体网关从远端实体接收到的
Remote	包含对媒体流进行说明的一些特性，其中这些媒体流是媒体网关发送给远端实体的
LocalControl	包含与媒体网关和软交换有关的一些特性
Events	描述由媒体网关监测的事件，以及当事件被监测到时，如何做出响应
EventBuffer	描述当事件缓存处于激活状态时，由媒体网关监测的事件
Signals	描述适用于终结点的信号和/或动作（如忙音）
Audit	在 Audit 命令中，标识需要何种信息
Packages	在 AuditValue 命令中返回由终结点实现的包的列表
DigitMap	在媒体网关处处理 DTMF 音的指令
ServiceChange	在 ServiceChange 处，何种业务改变发生，以及为何业务改变发生
ObservedEvents	在 Notify 或者 AuditValue 中，报告观测到的事件
Statistics	在 Subtract 和 Audit 中，报告保存在终结点上的数据

下面具体介绍每个描述符：

1. Modem 描述符

Modem 描述符指定了 Modem 的类型和参数。Modem 描述符包括下述 Modem 类型：V.18，V.22 和 V.34 等。默认情况下，并不指定终结点的 Modem 描述符。

2. Mux Multiplex 描述符

该描述符指定了多媒体终结点的多路复用类型，包括的复用类型有：H.221，H.223，H.226，V.76 以及可能的扩展。

3. Media 描述符

Media 描述符指定了所有媒体流的参数。这些参数被分别组成两个描述符：TerminationState 描述符（说明了终结点的与特定媒体流无关的特性）和 Stream 描述符（描述了单个的特定媒体流）。

媒体流由 StreamID 标识，在一个终结点里的多个媒体流应该相互同步。

4. Termination State 描述符

该描述符包括 ServiceStates 和 EventBufferControl 特性，以及在 Packages 中定义的和媒体流无关的终结点特性。

ServiceStates 特性描述了终结点的总体状态。一个终结点可以是下面几种状态的其中一种："test"、"out of service" 或者 "in service"。"test" 说明终结点正在被测试；"out of service" 说明终结点不能用来通信；"in service" 说明了终结点可以进行正常的通信。

EventBufferControl 特性说明了在监测到 Event 描述符里的事件后是否进行事件的缓冲或者立即进行处理。

5. Stream 描述符

该描述符说明了一个简单的双向媒体流。Stream 描述符包括 3 个子描述符：LocalControl，Local 和 Remote。Stream 描述符包含标识媒体流的 StreamID。通过给媒体流设置空 Local 和 Remote 描述符，并且把先前在关联里支持该媒体流的所有终结点的 LocalControl 描述符里的 ReserveGroup 和 ReserveValue 设为 "false"，删除一个媒体流。StreamID 在 MGC 和 MG 间有意义，它们由 MGC 分配。

6. Local Control 描述符

LocalControl 描述符包含 Mode、ReserveGroup 和 ReserveValue 特性。

Mode 特性的值可以是 "Send-Only"、"Receive-Only"、"Send/Receive"、"Inactive"、"Loop-Back"。"Send" 和 "Receive" 是和关联的外部特性相关的。例如 Mode Send-only 的媒体流不能将接收到的媒体传进关联。

一个终结点的 ReserveValue 和 ReserveGroup 属性表明当 MG 收到一个 Local

以及 Remote 描述符时，希望 MG 做什么。

如果 Reserve 特性值为“True”，在有资源可利用的前提下 MG 必须为在 Local 以及 Remote 中指定的所有选择预留资源。它应当对预留了资源的所有选择做出反映，如果 MG 不能支持任何一种选择，它应当对 MGC 做出一个回复，该回复包含空 Local 以及 Remote 描述符。

如果 Reserve 特性值为“False”，MG 必须选择 Local 描述符里的一种 Alternative（如果提供了）和 Remote 描述符里的一种 Alternative（如果提供了）。如果 MG 还没有为选择了的 Alternative 预留资源，它必须预留资源。最后，MG 必须发送相应的响应给 MGC。如果 MG 没有足够的资源支持任何一种可能的选择，必须以 501 进行响应。

7. Local 和 Remote 描述符

使用这两个描述符可以预留 MG 的资源，以提供给媒体的编码和解码使用。MG 应当在响应里包含额外的特性和值，即使这些值 MGC 没请求但却是必须的。例如：MGC 仅指定了负载类型，MG 却要指定详细的视频编码参数。

当文本编码协议时，描述符由 SDP 中定义的会话描述组成。当描述符从 MGC 传递到 MG，它们根据规则被解释，另外有下面的分类：

1）未指定的 Local 或 Remote 描述符被认为是一个未出现的强制参数。需要 MG 使用该描述符的最后一次指定的参数，无论参数是什么。也存在上一次没有指定值的情况，这种情况表明相关的描述符在进一步命令的处理中被忽略。

2）在从 MGC 传来的消息中的空 Local（或 Remote）描述符表示要求释放为接收到的或发送走的媒体流释放保留的资源。

3）由 MGC 发送的在一组特性内的 Underspecified 或 Overspecified 特性，要求 MG 选择一个或更多值，它能够支持的这些特性的任一特性的值。

如果 Local 描述符包含两组特性，一组指定以 G.711 A 律打包，另一组以 G.723.1 打包，MG 应当预留足够的资源以保证它可以对以 G.711 A 律或者 G.723.1 格式编码的一个音频流进行解码。MG 不用预留同时能处理两个音频流的资源，其中一个是 G.711 A 律格式，一个是 G.723.1 格式。ReserveValue 的用途类似。

如果 ReserveGroup 和 ReserveValue 值都为“true”，适用下面的规则：

1）如果 MG 没有足够的资源支持 MGC 要求的所有选择并且 MGC 在 Local 和 Remote 都要求了资源，MG 应当预留资源以支持 Local 和 Remote 里的至少一种选择。

2）如果 MG 没有足够的资源支持从 MGC 发来到的在 Local（或 Remote）描述符中至少一种选择，那么 MG 必须在响应里包含一个空至 Local（或 Remote）。

3）在给 MGC 的响应里，当 MGC 包含 Local 和 Remote 描述符时，响应里必

须包含它已经给分配了资源的所有特性组和特性的值。如果 MG 不能支持在 Local（或 Remote）描述符里的至少一种选择，MG 必须在响应里包含一个空至 Local（或 Remote）。

4）如果 LocalControl 描述符的 Mode 特性值是 Receiv-Only，Send/Recv，Loop back，MG 必须准备接受媒体，该媒体是根据 MG 对 MGC 响应里的任何一种选择进行的编码。

5）如果 ReserveGroup 和 ReserveValue 都为“False”，MG 应当用下面的规则对一个单一的“选择”解决 Local 和 Remote

① MG 选择在 Local 里的第一种“选择”，并为这种选择而支持在 Remote 里的至少一种选择。

② MG 不能支持至少一种 Local 和 Remote 的选择，其返回 510，即资源不足。

③ MG 返回在每一个 Local 和 Remote 中所选的“选择”。

如果 MGC 希望从存在的描述符中删除一些信息，它只要简单地（在 Modify 命令里）重发包括不想要的信息的描述符即可。

8. Events 描述符

Events 描述符包含 RequestIdentifier 和一系列 MG 需要检测和上报的事件列表。描述符中的每一个事件，包含事件名、可选的 StreamID、可选的 KeepActive 标志和可选的参数。事件名由包名（包含该事件定义的包）和事件的 ID 组成。默认 StreamID 是 0，表明被监测的事件与特定媒体流无关。事件识别可能导致当前活动信号的停止或者导致当前时间以及 Singals 描述符的被替代。如果 EventBufferControl 特性为 LockStep，那么任何检测到的并在 EventBuffer 描述符中出现的事件以及检测的时间将被加到 EventBuffer 的末尾（FIFO 队列），MG 必须等待新 EventsDescriptor 的载入。带有新 EventsDescriptor 的命令或者激活嵌入的 EventsDescriptor 都可以导入新 EventsDescriptor。

当 MG 收到带有 EventsDescriptor 的命令时，一个或者多个事件已经被存在 MG 的 EventBuffer 中。EventBufferControl 的值决定了 MG 如何处理缓冲的事件。

1）如果 EventBufferControl 等于 LockStep，并且 MG 收到一个新 EventsDescriptor，MG 将检查 FIFO EventBuffer，并采取以下的动作：

① 如果 EventBuffer 是空的，MG 等待监测到基于 EventsDescriptor 的事件。

② 如果 EventsBuffer 不为空，MG 从 FIFO 队列的第一个事件开始处理。

如果队列事件也出现在新 EventsDescriptor 的事件列表中，MG 对该事件做出响应，并从 EventBuffer 中除去该事件。Notify 的时间戳应该是事件实际被检测到的时间。然后 MG 等待新 EventsDescriptor 的到来。在等待过程中，任何出现在 EventsDescriptor 中并被检测的事件被置入 EventBuffer。当收到新 EventsDescriptor

时，从 step1 开始重复执行事件处理。如果事件不在新 EventsDescriptor 中，MG 必须丢弃该事件，然后从步骤 1）重复执行。

2）如果 EventsBufferControl 等于 Off，并且 MG 接收到一个新 EventsDescriptor，MG 使用新 EventsDescriptor 处理事件。如果 MG 收到一个命令，指示它设置 EventBufferControl 值为 Off，那么在 EventBuffer 中的所有事件必须被丢弃。EventBufferControl 的默认值是 Off。因为 EventBufferControl 是在 TerminationStateDescriptor 中，MG 可以接受改变 EventBufferControl 特性但不包括 EventsDescriptor 的命令。

正常情况下，识别到一个事件应当导致任何一个活动信号的停止。但是如果 KeepActive 在事件中被指定，当监测到事件的时候，MG 一定不能终止终结点上的任何活动信号。

一个事件可以包括嵌入的信号描述符以及一个内嵌的事件描述符（如果提供了），当该事件发生时，替代当前的信号或事件描述符。这种情况是可能的，例如：指定当识别到摘机事件时产生拨号音信号，或者识别到键入数字时停止拨号音信号。MGC 一定不能发送一个既包括 KeepActive 标记又包含内嵌 Signals Descriptor 的 EventsDescriptor。只允许一级内嵌，一种内嵌的 EventsDescriptor 一定不能再包含其他的内嵌的 EventsDescriptor。

9. EventBuffer 描述符

EventBuffer 描述符包含了一个带有参数的事件的列表。当 EventBufferControl 等于 LockStep 时，MG 需要监测，然后缓存这些事件。

10. Signals 描述符

它包括一组 MG 用于终结点的信号。Signals 描述符包括许多信号和/或有序的信号列表。Signals 在包中定义，由 PackageName 和 SignalID 命名，不允许使用通配符。其他可选的参数包括 StreamID，信号类型，持续时间等。

可选的参数“NotifyCompletion”表示 MGC 希望在信号播放完成后得到通知。信号被中断的可能情形是信号超时、被事件中断、Signals 描述符被代替时被中断或者信号停止或因为其他原因没有开始。

有 3 种类型的信号：

1）on/off——信号持续到被关闭。

2）timeout——信号持续到被关闭或指定的一段时间过去。

3）brief——信号持续时间非常短以至于主动停止。除非新用的信号导致它停止。

连续的信号列表包括一个信号列表标识符、被顺序播放的信号序列和信号类型。只有连续信号列表的结尾元素可以是 on/off 信号。如果连续信号列表的结尾元素是 on/off 信号，它的信号类型也必须是 on/off。如果连续信号列表中的

信号序列包括“timeout”类型信号并且结尾元素不是 on/off 类型，那么它的类型必须设为“timeout”，连续信号列表的持续时间是表中所有信号的持续时间之和。如果连续信号列表中的信号序列只包括“brief”类型信号，那么它的类型必须设为“brief”。在播放时，信号列表被视为一个单一的指定类型的信号。在一个事务范围内相同信号应用于多个终结点，MG 应当考虑同一资源产生该信号。产生在终结点上的信号被停止的原因可以是一个新 SignalsDescriptor 的应用或者在终结点上检测到事件。如果信号在替换描述符中出现而且包含 KeepActive 标记，当信号正在播放，而且没有放完时，那么要继续播放。如果替换描述符中包括和现存描述符中相同的标识符，那么在连续信号列表中的信号类型和信号序列必须忽略并且现在描述符里的信号的播放不准中断。

11. Audit 描述符

Audit 描述符说明了需要审计的信息，该描述符可以说明需要返回的描述符列表。Audit 描述符中可能包括的项是：

1）Modem；

2）Mux Events；

3）Media；

4）Signals；

5）ObservedEvents；

6）DigitMap；

7）Statistics；

8）Packages；

9）EventBuffer。

12. ServiceChange 描述符

ServiceChange 描述符包括下述的参数：

1）ServiceChangeMethod；

2）ServiceChangeReason；

3）ServiceChangeAddress；

4）ServiceChangeDelay；

5）ServiceChangeProfile；

6）ServiceChangeVersion；

7）ServiceChangeMGCId；

8）TimeStamp；

9）Extension。

13. DigitMap 描述符

DigitMap 描述符是驻留在 MG 内为检测和上报终结点收到的 Digit 事件的。

“DigitMap”可以通过管理命令将其预先调入 MG，也可以动态定义然后用名字引用，或在 EventsDescriptor 中指定其自身。在 EventsDescriptor 中对一个 DigitMap 的 Completion 事件下述的情形是允许的，用名字引用在同一个命令里由 DigitMap 描述符定义的 DigitMap，而不用考虑各自的描述符的传输顺序。

DigitMap 包括 3 种定时器：

1）开始定时器（T）。被用在任何一个已播 Digit 之前。

2）若 MG 能够确定数字串至少缺少一个以上的数字以匹配 DigitMap 中任何一个允许的模式，那么数字间定时器的值应当设置成 Long（L）持续时间。

3）若数字串匹配了一种模式，但有可能接受更多的数字而匹配另一种不同的模式，那么不是立刻报告匹配，而是运用 Short 定时器等待更多的数字。

DigitMap 语法：DigitMap 符号包括数字 0～9 和字母 A 到和信令系统有关的一个最大符号，一般不超过 K。字母“x”是一个通配符，代表 0～9 间的任一数字。符号“.”代表 0 或更多次事件位于“.”前选择器的重复。除了上述符号外，还可以包括“S”、“L”（事件间定时器指示符）和“Z”（持续时间指示符）。如果一个显式的定时指示符在一种可能的事件序列中起作用，但是在其他的任何一个候选可能中都没有，那么必须使用显式定时指示器的值。如果带有显式定时控制的所有序列从候选集合中去掉了，定时恢复到上述默认值。最后，如果在不同的可能序列中冲突的定时指示器在起作用，结果未定义。“Z”指明一个长持续时间事件，置于 symbol 前，该 symbol 指明满足给定位置的 Digit 事件。它指出仅当事件的持续时间超过了长持续时间的门限，该位置才算满足，门限值假定在 MG 中被提供。

以下几种情况都由 DigitMap 完成：

1）某个定时器超时；

2）匹配了一个可能事件序列并且没有其他可能事件序列会匹配，这是通过检测附加的事件；

3）一个事件已经被检测到，因此无论其后接收到何种事件，对 DigitMap 的可能事件序列的完成匹配都是不可能的了。

一旦完成，在提供事件映射到 DigitMap 的包中定义 DigitMap 完成事件应当生成。然后使 DigitMap 停止活动，处理包中接下来的事件。

14. Statistics 描述符

该描述符用在 Subtract 和 Audit 命令里，报告终结点的统计数据。

15. Packages 描述符

仅用在 AuditValue 命令里，该描述符返回终结点能够识别的一系列 Package。

16. ObservedEvents 描述符

ObservedEvents 描述符同 Notify 一起使用，通知 MGC 监测到的事件。如果和

AuditValue 命令一起使用，ObservedEventsDescriptor 返回存在事件 Buffer 中但还没有被 Notify 的事件。ObservedEvents 包括触发通知的 Events 描述符的 Request Identifier 和监测到的时间，监测时间精确到 1/100s。

17. Topology 描述符

该描述符用来说明关联中终结点间的流向。拓扑描述符使用在关联上而不是终结点。关联的默认拓扑是每个终结点的传输可以被其他所有的终结点接收。可以选择地实现 Topology 描述符。

4.4.3 Megaco 的命令

Megaco 提供了 8 个可以用于控制的命令：

1. 增加（Add）命令

由 MGC 发送到 MG，可以使用 Add 命令向一个关联添加一个终结点。

2. 修改（Modify）命令

由 MGC 发送到 MG，修改一个终结点的特性、事件和信号。

3. 删除（Subtract）命令

由 MGC 发送到 MG，可以使用 Subtract 命令解除一个终结点与它所处的关联之间的联系。

4. 转移（Move）命令

由 MGC 发送到 MG，可以使用 Move 命令自动地将一个终结点从一个关联转移到另一个关联。

5. 审计值（AuditValue）命令

由 MGC 发送到 MG，可以使用 AuditValue 获取与终结点的当前特性、事件、信号和统计有关的信息。

6. 审计能力（AuditCapability）命令

由 MGC 发送到 MG，可以使用 AuditCapability 获取媒体网关所允许的终结点的特性、事件和信号的所有可能值的信息。

7. 通知（Notify）命令

由 MG 发送到 MGC，媒体网关可以使用 Notify 命令向软交换设备报告媒体网关中所发生的事件。

8. 服务更改（ServiceChange）命令

MG 与 MGC 能够互发该命令。媒体网关可以使用 ServiceChange 命令来完成向软交换报告一个终结点或者一组终结点将要退出服务，或者刚刚返回服务等多种工作。

总体上讲，Megaco/H.248 提供的命令可以分为下述几类：

1）对于终结点的操作命令：Add、Subtract、Move、Modify；

2）对于事件报告：Notify；

3）用于管理的命令：AuditCapability、AuditValue、ServiceChange。

由上述可知，8 个命令中，除 Notify 和 Servicechange 命令外，其他 6 个命令的发起方都是 MGC。命令 Notify 是由 MG 通知 MGC 在 MG 中有某事件发生。只有命令 ServiceChange 是双向的，即 MG 与 MGC 互发。

Megaco 协议可以用二进制格式或者文本格式进行编码。MGC 应当支持两种编码格式，MG 也可以支持两种格式。

错误信息由 IANA 注册的错误代码和一个可选的说明性的字符串组成。当 MG 向 MGC 报告错误的时候，使用错误描述符完成这个任务。

已经确定的错误代码见表 4-5。

表 4-5　错误代码

代码	错误类型	中文名称
400	Bad Request	错误请求
401	Protocol Error	协议错误
402	Unauthorized	未授权
403	Syntax Error in Transaction	事务中的语法错误
406	Version Not Supported	版本不支持
410	Incorrect identifier	错误标志符
411	The transaction refers to an unknown ContextId	事务引用无法识别的关联 ID
412	No ContextIDs available	无可用关联 ID
421	Unknown action or illegal combination of actions	未定义的行为或非法的行为联合
422	Syntax Error in Action	行为语法错误
430	Unknown TerminationID	未定义的终结点号
431	No TerminationID matched a wildcard	没有终结点匹配通配符
432	Out of TerminationIDs or No TerminationID available	超出终结点号范围或 没有可用的终结点号
433	TerminationID is already in a Context	终结点号已经存在于一个关联中
440	Unsupported or unknown Package	不被支持或无法识别的包
441	Missing RemoteDescriptor	丢失远端描述符
442	Syntax Error in Command	命令中的语法错误
443	Unsupported or Unknown Command	不被支持或无法识别的命令
444	Unsupported or Unknown Descriptor	不被支持或无法识别的描述符
445	Unsupported or Unknown Property	不被支持或无法识别的属性

（续）

代码	错误类型	中文名称
446	Unsupported or Unknown Parameter	不被支持或无法识别的参数
447	Descriptor not legal in this command	在此命令中描述符不合法
448	Descriptor appears twice in a command	在此命令中描述符重复出现
450	No such property in this package	数据包无此属性
451	No such event in this package	数据包中无此事件
471	Implied Add for Multiplex failure	增加复用错误
500	Internal Gateway Error	内部网关错误
501	Not Implemented	未实现
502	Not ready	未准备
503	Service Unavailable	服务不可用
504	Command Received from unauthorized entity	命令来自未授权实体
505	Command Received before Restart Response	在回复之前收到命令
510	Insufficient resources	资源不足
512	Media Gateway unequipped to detect requested Event	媒体网关无法检测被请求事件
513	Media Gateway unequipped to generate requested Signals	媒体网关无法检测被请求信号
514	Media Gateway cannot send the specified announcement	媒体网关无法发送该通告
515	Unsupported Media Type	不支持的媒体类型
517	Unsupported or invalid mode	不支持或不可用模式
518	Event buffer full	事件缓冲区满
519	Out of space to store digit map	数据映射存储空间不足
520	Media Gateway does not have a digit map	媒体网关没有数据映射
521	Termination is “ServiceChangeing”	终结点服务改变
526	Insufficient bandwidth	带宽不足
529	Internal hardware failure	内部硬件错误
530	Temporary Network failure	临时网络错误
531	Permanent Network failure	永久网络错误
581	Does Not Exist	不存在

4.4.4 Megaco 中的事务

在媒体网关和软交换设备之间的一组命令组成了事务交互。一个事务交互可以由一个 TransactionID 来标识。事务交互由一个或者多个动作组成。而一个动作又由在一个关联中使用的一系列命令组成。因此，可以用一个 ContextID 来

标识一个动作。

事务交互由 TransactionRequests 发起。对 TransactionRequest 的响应放在一个独立的响应里面，这个响应可能由 TransactionPending 消息开始。

事务交互保证对命令的有序处理，即在一个事务交互中的命令是顺序执行的。但并不保证各个事务交互之间的有序处理，即对这个事务交互的处理可以以任何顺序进行，也可以同时进行。如果在一个事务交互过程之中，有一个命令执行失败，那么在这个事务处理中的所有剩余命令都停止执行。

TransactionReply 包含执行相应的 TransactionRequest 中的所有命令所得到的结果。也就是说，TransactionReply 包含所有成功执行的命令的返回值，以及所有执行失败的命令的命令名和错误描述符。TransactionPending 命令是用来周期性地向接收者报告这个事务处理正在被处理，但是没有结束。

对于每个事务处理都应实现一个应用层定时器。当定时器超时后，应当重新传输请求信息。当接收到应答后，就应当取消定时器。当接收到 Pending 后，就应当重新启动定时器。

与事务处理相关的 API 包括 TransactionRequest、TransactionReply 和 TransactionPending。

多个事务可以连接成一个消息。消息具有头部（header），包括发送者的身份。一个消息的 ID 被设成是传送消息的实体的规定名字。

每个消息包含一个版本号码，它标识了消息所遵守的协议的版本。版本号包括一个或两个数字，以 1 开头的代表了协议的当前版本。一个消息内的事务要被单独地处理。

消息结构如图 4-15 所示。

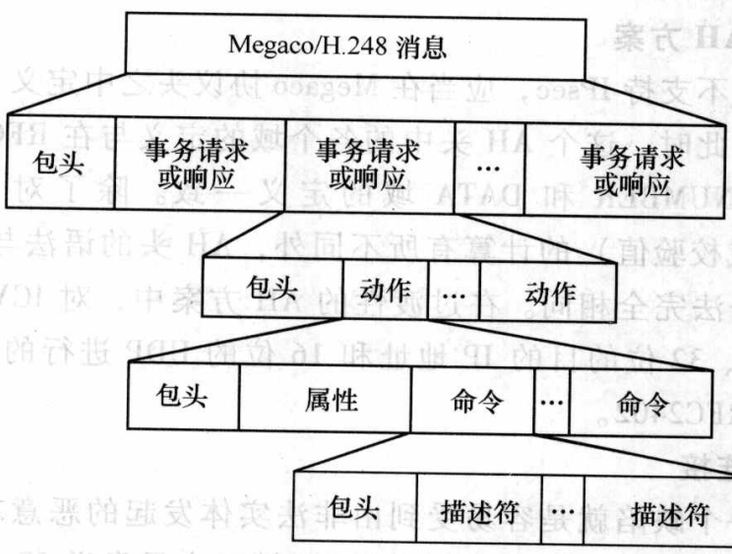


图 4-15　消息结构图

4.4.5 Megaco 的传输

本协议的传输机制应当能够支持在媒体网关和软交换之间的所有事务处理的可靠传输。如果是在 IP 上传输本协议，软交换应当实现 TCP 和 UDP，媒体网关应当实现 TCP 或者 UDP 或者同时支持两者。

应当向媒体网关提供一个主要软交换的名字或者地址（如 DNS 名字或者 IP 地址），同时还可以提供零个或者多个备用的软交换的名字或者地址，以便媒体网关能够向软交换发送消息。如果将 TCP 或者 UDP 用作协议的传输层，同时无法经过其他的方式获知将初始的 ServiceChange 请求发送到哪个端口，那么就应当将这个请求发送到默认的端口上。无论是 TCP 还是 UDP，对于文本编码的动作，这个默认端口的端口号为 2944，而对于二进制编码的动作，这个默认端口的端口号为 2945。软交换接收到来自媒体网关的包含 ServiceChange 请求的消息后，应当能够从中判断出媒体网关的地址。同时，媒体网关和软交换都可以在 ServiceChangeAddress 参数中提供一个地址，以便后续的事务交互请求都发送到这个地址。但是，所有的对请求的响应（包括对初始的 ServiceChange 请求的响应）必须发送给相应的请求的源地址。

4.4.6 Megaco 的安全考虑

1. 对协议连接的保护

为了防止未经授权的实体利用本协议建立非法呼叫，或者干涉合法呼叫，需要对本协议的传输建立安全机制。当在 IP 网络上传输本协议时，可使用 IPsec 对本协议的传输进行保护。

2. 过渡性的 AH 方案

如果低层协议不支持 IPsec，应当在 Megaco 协议头之中定义 AH 头来实现对协议连接的保护。此时，这个 AH 头中的各个域的定义与在 RFC2402 中定义的 SPI、SEQUENCE NUMBER 和 DATA 域的定义一致。除了对 ICV（Integerity Check Value，集成校验值）的计算有所不同外，AH 头的语法与 RFC2402 中传输模式中定义的语法完全相同。在过渡性的 AH 方案中，对 ICV 的计算是基于 32 位的源 IP 地址、32 位的目的 IP 地址和 16 位的 UDP 进行的。过渡性的 AH 的实现应当遵照 RFC2402。

3. 保护媒体连接

分组网络的一个缺陷就是容易受到由非法实体发起的恶意攻击。例如，非法实体可以通过向一个给定的 IP 地址和 UDP 端口大量发送 IP 包进行攻击。针对这种攻击进行保护的一种方法是仅接收来自获得授权的来源的分组，例如，仅接收在 Remote 描述符中列出的源 IP 地址和 UDP 端口号。另外一种进行保护

的方法是使用在呼叫建立过程中传送密钥对分组进行加密和认证。

4.4.7　Megaco 的包定义

包（Package）是命令控制的主要扩展方法。包定义了附加的在终结点产生的特性、事件、信号和统计。一个包应该包含以下部分：

1）Package Name：包的名字，描述性质的；

2）PackageID：包的标识符；

3）Description：描述；

4）Version：版本号；

5）Extends（Optional）：一个包可以对现有的包进行扩展。

以下是对由包定义的特性、事件、信号和统计的说明：

1. 特性

（1）Property Name：属性名字，描述性质的；

（2）PropertyID：属性标识符；

（3）Description：描述；

（4）Type：类型。下列类型之一：

1）String；

2）Integer；

3）Double；

4）Character；

5）Enumeration；

6）Sub - list；

7）Boolean。

（5）Possible Values：可能值；

（6）Defined in：表示特性定义在哪个 Megaco 的描述符里，LocalControl 描述的是与媒体流有关的特性，TerminationState 是与媒体流无关的特性，这是最普通的情况，但是属性也可以定义在其他描述符里；

（7）Characteristics。

2. 事件

（1）Event Name：事件名，描述性质的；

（2）EventID：事件标识符；

（3）Description 描述；

（4）EventsDescriptor Parameters：MGC 用它来配置事件；

（5）ObservedEventsDescriptor Parameters。

3. 信号

(1) Signal Name：信号名，描述性质的；

(2) SignalID：信号标识符，SignalsDescriptor 中使用；

(3) Description：描述；

(4) SignalType：下列类型之一：

1) OO (On/Off)；

2) TO (TimeOut)；

3) BR (Brief)。

(5) Duration：持续时间；

(6) Additional Parameters。

4. 统计

(1) Statistic Name：统计名，描述性质；

(2) StatisticID：统计标识符，StatisticsDescriptor 中使用；

(3) Descriptio：描述

(4) Units：计量单位。例如包、ms 等。

包中的标识符的定义规则：文本编码的标识符必须是最多 64 个字符的字符串，不包括空格，以 alphanumeric 字符开头，并且包括 alphanumeric 字符和/或数字，可能包括特殊的字符“_”。二进制编码的标识符是两个 8 位组长，每种标识符都应该指定值，包括用作枚举类型代表值的标识符。

附录 A 列举了一些基本包 (Basic Package)，其中列举了每个包所包含的特性、事件、信号和统计。

4.5 软交换中的 Megaco

Megaco/H. 248 协议是一个具有很好的灵活性和可扩展性的协议。协议只规定了 8 条命令，但没有规定严格的消息流程，也没有规定严格的参数数量和格式。媒体网关控制器和媒体网关之间的命令交互流程可根据具体应用而灵活使用，二者相互配合完成指定功能。命令的描述符是可嵌套的，描述符的参数封包 (Package) 也是可以不断定义和扩展的。当网络体系结构不断发展，各种新型设备出现后，媒体网关控制器与媒体网关之间可能会不断出现新的信息需要交互，Megaco/H. 248 协议通过定义新的封包以适应新的网络和设备。

Megaco/H. 248 只规定了媒体网关和媒体网关控制器之间的消息。为了充分适应网络上多种协议并存的环境，媒体网关控制器需要具有能与多种 VoIP 协议互通的能力，如，H. 223 协议、SIP 等。当用户规模达到某种程度后，网络上需要多个媒体网关控制器，它们各自控制一个区域内的若干媒体网关。不同厂商的媒体网关控制器之间需要协同工作，因此媒体网关控制器之间通信需要共同

认可的协议。SIP 可能会成为未来的发展方向。

下一代通信系统将是一个承载与控制相分离、控制与业务相分离的系统。传统交换机的功能模块分离成为独立的网络部件，部件间的接口采用国际通行的标准。Megaco/H. 248 协议可以成为控制功能部件——MGC（又称作 SoftSwitch）与承载部件进行交互的标准。

Megaco/H. 248 协议为网关分离后的控制中心部分和媒体交换部分的信息交换提供了一个标准，使得运行在通信网络上的多家厂商设备可以实现良好的互通。Megaco/H. 248 协议提供的连接模型不仅支持两方通话，而且可使系统方便地实现多方呼叫业务，适应未来的通信需求。在未来的通信网络中，分组交换技术将被大规模应用于语音通信，乃至多媒体通信，电路交换网络与分组交换网络需要实现互通；而且，通信网络也逐渐向承载与呼叫控制相分离的方向发展。Megaco/H. 248 协议在这种发展趋势下产生，因此也将会得到充分发展，并被众多通信设备厂商所支持。

软交换相当于 MGC 的功能，控制 MG 实现呼叫控制和承载连接的分离。因此 MG 必能够实现协议中规定的功能，支持软交换的控制，实现媒体流的承载连接。

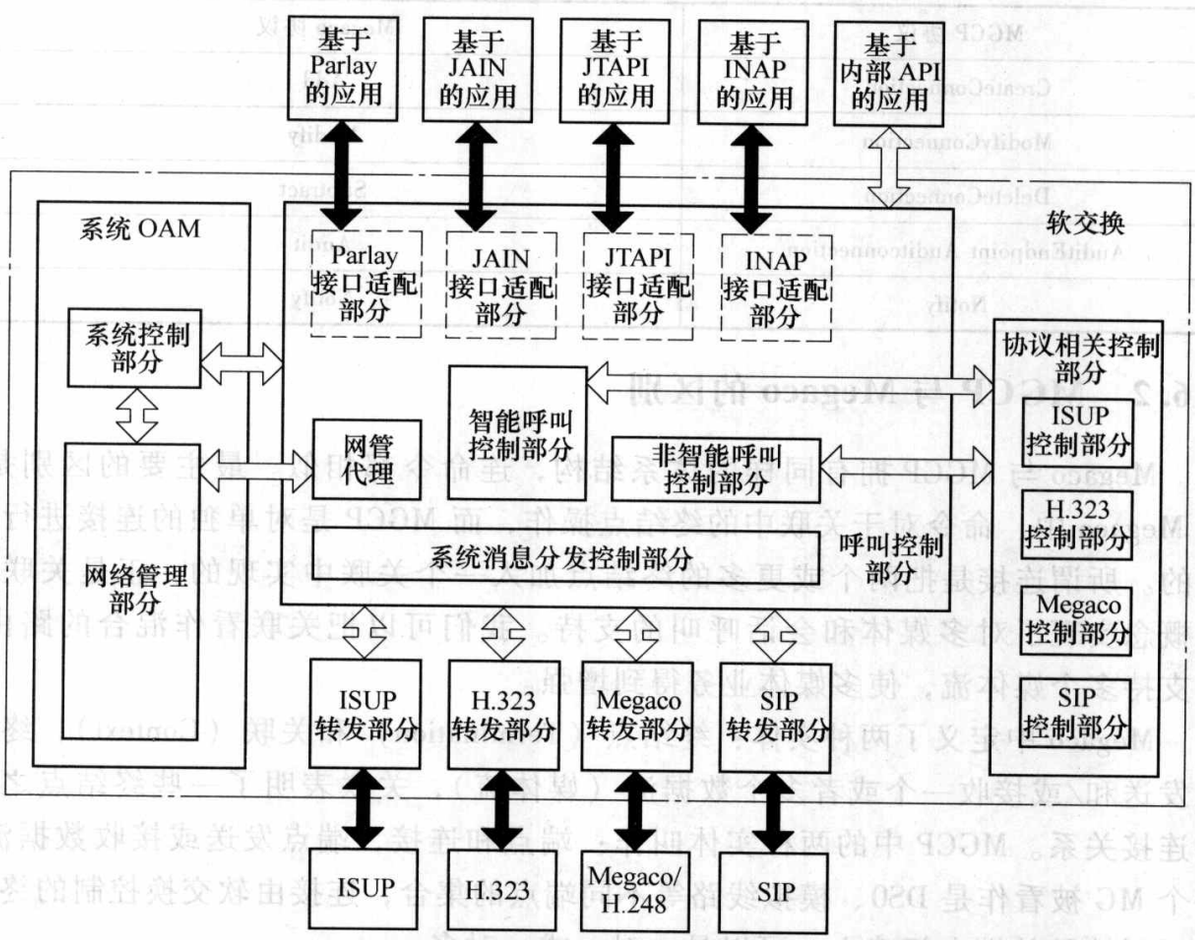


图 4-16　Megaco 在软交换中所处的位置

Megaco 在我们设计的软件换中所处的位置如图 4-16 所示。

4.6 MGCP 和 H.248 协议的比较

4.6.1 MGCP 与 Megaco 的相同和相似之处

1）都遵循 ABNF 语法规范，用 SDP（Session Description Protocol）声明媒体流的属性；

2）对媒体流中信号和事件的处理是相同的；

3）Digit Map 是指示 MG 如何收集拨号数字的方案；

4）“包”含有事件与信号的定义；

5）用 UDP 传消息的声明是相同的，MGCP 中的“三次握手”、重传计时和抑制重启造成的雪崩效应，与 Megaco 中 ALF 的定义相似；

6）可能在长时间执行命令时返回临时响应；

7）部分相关的命令见表 4-6。

表 4-6 部分相关的命令

MGCP 协议	Megaco 协议
CreateConnection	Add
ModifyConnection	Modify
DeleteConnection	Subtract
AuditEndpoint Auditconnection	Audit
Notify	Notify

4.6.2 MGCP 与 Megaco 的区别

Megaco 与 MGCP 拥有同样的体系结构，连命令都相似。最主要的区别是：在 Megaco 中，命令对于关联中的终结点操作，而 MGCP 是对单独的连接进行操作的。所谓连接是把两个或更多的终结点加入一个关联中实现的，正是关联这个概念实现了对多媒体和会话呼叫的支持。我们可以把关联看作混合的路由，它支持多个媒体流，使多媒体业务得到增强。

Megaco 中定义了两种实体：终结点（Termination）和关联（Context），终结点发送和/或接收一个或者多个数据流（媒体流），关联表明了一些终结点之间的连接关系。MGCP 中的两种实体叫作：端点和连接，端点发送或接收数据流，一个 MG 被看作是 DS0、模拟线路等不同端点的集合，连接由软交换控制的终端在呼叫涉及的端点间建立，可以是一对一或一对多。

Megaco 模型简化了 MG 内部与其外部实体间建立连接的过程：MGC 可以声

明相关的媒体流也可以声明媒体流的方向，所以 Megaco 可以提供更好的应用层支持。

Megaco 引入临时终结点的概念以适应 RTP 流，而 RTP 流是 MGCP 的连接所包含的，可以自动创建。临时终结点使表达上一致，RTP 流操作可以类似于 MG 中的其他终结点的操作，而且 MG 内部的连接与外部连接是相同的。MGCP 中外部连接实际上只是从端点到外部网络的半双工连接，内部连接可以是半双工或全双工。

Megaco 比 MGCP 的“包”概念更详细。Megaco 的“包”定义了额外的“属性“和“统计”，以及可能在终结点上产生的“事件”和“信号”，“包”是 Megaco 实现扩展的主要机制。

Megaco 中使用复用终结点和一个终结点内部的多个数据流，这样一个单独的终结点就可以有多个数据流，它们收发于不同的承载信道上。

MGCP 中建立连接后命令直接操作于端点，无需指明连接标识；Megaco 大多数的命令都是对于关联中的终结点进行操作的。模式在 MGCP 中是指连接的，在 Megaco 中指终结点。

Megaco 有事务的概念，可以将多个命令和响应放在一起，事务强制了执行顺序；MGCP 只允许单个命令在单个的消息中传输。

Megaco 和 MGCP 概念上的比较见表 4-7。

表 4-7　Megaco 和 MGCP 概念上的比较

概念比较		
	Megaco	MGCP
定义	终结点：在需要时（比如有 RTP 终结点）可创建临时终结点	端点
事务	1 个事务可以有多个动作，每个动作可以有多个命令 加入新的 TransactionPending	1 个事务只能有 1 个命令
有关终结点/端点类型	以 Local 描述的终结点 对于 RTP 端口使用单独的终结点 任何终结点都可以有 EventsDescriptor，SignalDescriptor	有参数 L 的端点 第二个端点是 RTP 端口时不使用单独的端点

（续）

概念比较		
	Megaco	MGCP
内部资源管理	内部资源可以被看作终结点； 关联可以有参数指定音、视频如何混合； 一个关联可以有的终结点数量只由 MG 决定	认为每个端点有一些可支持的资源； 不能定义混合时的参数
脚本支持	通过在命令中的 EventsDescriptor 实现	通过 NotificationRequest 实现
拨号方案	拨号方案是一种脚本，但在 DigitMapDescriptor 中处理 在事件描述符中可以引用预读入的 Digit Map Digit Map 可以定义操作于哪些终结点的范围	只是种普通表达式
会话实现方式	将多个终结点加入一个关联	将多个连接终结到同一端点上
呼叫概念	contextID 为一次呼叫建立起所有终结点间的联系	Call-id 联系起各个连接
MG 内部连接	MG 内的由一个呼叫连接起来的一组终结点为一个关联	可以创建内部连接，连接可以为半双工或全双工
MG 和 MGC 间的登记	通过 ServiceChange 命令	MG 通过 RestartInProgress 命令通报其存在
联系控制	可以使 MG 定期发送通知，以确认其可使用性（heartbeat）	
失败处理	MGC 向 MG 发送 ServiceChange 告诉 MG 一个新 MGC 进行通信	MGC、MG 之间的通信在安全层进行，任何符合条件的 MGC 都可能接管失败的通信

两者间命令的比较见表 4-8。

表 4-8　Megaco 和 MGCP 在命令上的比较

命令比较		
	Megaco	MGCP
终结点/端点信号配置	Add/Modify 中 EventsDescriptor 使用 DigitMapDescriptor 来定义 DigitMap 及要检测的事件列表	NoficationRequest 使用参数 D 定义 DigitMap 及要检测的事件列表
关联/连接	Add, Substract, Modify, Subtract	CreateConnection, ModifyConnection, DeleteConnection
媒体流改向	Move 可以将终结点从一个关联直接改向另一个	无单一命令
监听资源	AuditValue	AuditEndpoint, AuditConnection
监听资源能力	AuditCapabilities	无
MGC、MG 间联系控制	ServiceChange	依赖于安全连接
重启指示	ServiceChange	RestartInProgress

相对而言，Megaco 从一个更抽象的角度对网关分解后各实体间的接口关系进行了重新的定义，同时 Megaco 协议更充分的考虑了和 No.7 网络的互操作。Megaco 由于在基本协议中包括了用于会话的语义，因而与媒体应用结合得更紧密。对 MGCP 本身而言，用于不以媒体为中心的应用更好些，比如基于 MPLS 的会话控制。Megaco 作为 MGCP 的升级版本，虽然保留了 MGCP 的绝大部分特性，但更弥补了 MGCP 的一些不足。Megaco 可以满足更广泛的要求，通用性更好，更容易扩展，对应用层支持更好，且效率更高，为 IETF 和 ITU-T 共同支持。许多厂家、公司已经决定以后要从 MGCP 转向支持 Megaco。由于 MGCP 是先出现的，所以还会存在一段时期。

第5章　软交换的管理技术

网络管理技术经过多年的发展，已形成了两大阵营，一大阵营是以ISO和ITU-T为首，另一大阵营是以IETF为首。ISO和ITU-T着重于开放式网络的管理，注重管理的完整性和重用性，主要成果是M.3000和X.700系列建议和规范；IETF主要是由于Internet的迅速兴起，针对Internet的管理提出自己的管理方案，着重于管理的简单性，主要成果是简单网络管理协议SNMP的3个版本。但需要说明的是，IETF在提出SNMP时，很多管理的概念和技术是以ISO的网络管理技术为基础的，ISO的很多管理概念都适用于IETF的管理体系。

网络管理系统本身就是一个复杂的分布式系统，近年来，分布式计算技术得到了空前的发展，分布式技术也被应用到网络管理中来。在分布式技术中，以CORBA和Agent技术为代表，CORBA技术已经进入了实用化阶段，而Agent技术还处于研究阶段。

软交换网络环境下，所有基于IP的业务都将在同一数据传输网络上开展，传统意义上相对分离的语音业务、数据业务以及新出现的多媒体业务、视频业务将在同一个数据传输网络上运行。与现在的网络管理相比，软交换对网管系统的功能、质量、能力等不断提出新要求，对于全面发展和提升现有网管技术的要求也更加迫切。随着软交换在运营网络中的部署，软交换和TMN（电信管理网）以及运营支撑系统的融合不可逆转。

5.1　TMN概述

5.1.1　TMN的管理功能

电信管理网（TMN）是现代化电信网的一个支撑网，通过先进技术的支撑和现代化的管理手段，为电信网提供高质量、高可靠性、高效益的电信服务提供重要保证。TMN也提供自己的网络业务，拥有自己的用户。它的业务就是TMN的管理业务，这种管理业务是从使用者的角度来描述的对电信网进行的操作、组织与维护的管理活动。TMN管理业务基本可以归纳为3类：

1）通信网日常业务和网络运行管理业务；

2）通信网的检测、测试和故障处理等网络维护管理业务；

3）网路控制和异常业务处理等网络控制业务。

TMN 的用户可以是电信运营公司、电信运营公司的管理组织部门、维护部门及人员和电信业务所服务的客户。

TMN 的各类管理功能支持 TMN 的管理业务的实现，满足对被管理网络的操作、维护和管理的需要。管理人员通过人机接口与管理应用交互，通过 TMN 提供的管理功能对被管理网络进行各项管理操作活动。TMN 为电信网及电信业务提供一系列的管理功能，主要划分为以下 5 种管理功能域：

1. 性能管理

性能管理（Performance Management）是对电信设备的性能和网络单元的有效性进行评估，并提出评价报告的一组功能。包括性能测试，性能分析及性能控制。

2. 配置管理

配置管理（Configuration Management）功能包括提供状态和控制及安装功能。对网络单元的配置、业务的投入和开/停业务等进行管理，对网络的状态进行管理。

3. 计费管理

计费管理（Accounting Management）功能测试电信网中各种业务的使用情况，计算使用电信业务的应收费用，并对电信业务的收费过程提供支持。

4. 故障管理

故障管理（Fault Management）功能是对电信网的运行情况异常和设备安装环境异常进行管理，对网络的状态进行管理。

5. 安全管理

安全管理（Security Management）主要提供对网络及网络设备进行安全保护的能力。主要有接入及用户权限的管理、安全审查及安全告警处理。

TMN 的功能可以划分为不同的层次由高到底依次为：

1）事务管理层。事务管理层是最高的管理功能层，该层负责设定目标任务，但不管具体目标的实现，通常需要管理人员的介入。

2）服务管理层。服务管理层主要处理网络提供的服务相关事项，诸如提供用户与网络运营者之间的接口，与事务管理层及网络管理层的交互等。

3）网络管理层。网络管理层对所辖区域内的所有网元进行管理，主要的功能包括：从全网的观点协调与控制所有网元的活动；提供、修改或终止网络服务；就网络性能，可用性等事项与上面的服务管理层进行交互。

4）单元管理层。单元管理层直接行使对个别网元的管理职能，主要的功能包括：控制与协调一系列网络单元；为网络层的管理与网络单元进行通信提供协调功能；维护与网络单元有关的统计等数据。

5.1.2 网络管理的新要求

网络管理系统不仅仅是保证新一代网络正常运行的基本条件，也是保证网络高效、可靠、经济和安全运行的重要手段。

实现 TMN 的五大管理功能，是网络管理的基本要求，除此之外，还有以下要求：

1）对运营中的网络状态数据进行必要的趋势分析，以便对网络未来的故障和性能进行预测和决策；

2）提供端到端通道的组建、寻路方式选择、通道保护倒换、业务安排和电路调度；

3）提供必要的资源管理，对网络设备、端口、电路和通道等资源的状态参数进行统计、归类、分析、综合与使用分配；

4）提供计费信息。

5.1.3 TMN 的技术基础

支持 TMN 的主要技术有：

1. OSI 开放系统互连技术
2. OSI 系统管理技术
3. 高速发展的计算机网络技术及分布式处理环境
4. 面向对象的软件工程方法
5. 高速数据通信技术
6. 电信设备的高度智能化技术

5.1.4 TMN 的实用技术

随着现代计算机技术的飞速发展，越来越多的先进的计算机技术正逐渐在电信管理网领域内得到应用，例如 SNMP、XML、CORBA、WEB 技术等。

1. SNMP 在网络管理中的应用

SNMP 是网络管理的协议，主要通过一组 Internet 协议及其所依附的资源提供网络管理服务。SNMP 采用管理/代理站模型，提供了一种从网络上的设备中收集网络管理信息的方法。一般说来，网络管理工作站通过轮询被管理设备中的代理来收集数据，并且在控制台上用数字或图形来显示这些数据。被管理设备中的代理可以在任何时候向网络管理工作站报告错误情况。

SNMP 相对于 CMIP，其优点是设计简单、容易实现且成本低，因此它很容易在小型网络管理中实现，同时它还不会对网络信息流量带来很大压力，且具有可扩展性强的特点，可以比较轻松地进行修改以适应用户在未来的要求。

SNMP 的缺点首先是安全性差，可能给网络入侵者以可乘之机。其次是由于其设计的简化，已不太适应 20 世纪 90 年代以来网络的高速发展。虽然 CMIP 在安全性、功能等方面都大大超过 SNMP，但是由于它具有大量占用网络资源、增加管理开销的缺点，因此其应用的范围远不如 SNMP。

2. CORBA 技术在网络管理中的应用

CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）是由 OMG（Object Management Group，对象管理组织）定义的面向对象的通用体系结构，主要是为了解决分布式处理环境中系统互联而提出的一种解决方案。

要求对电信网络的管理更具开放性、灵活性，不仅支持业务对象和组件在网络内动态迁移，而且支持访问和位置透明。同时由于当前网络管理领域有多种协议同时存在的情况，如何把基于这些不同协议的系统进行综合，实现互联互通，也成为研究的重点。

作为对这些问题的一个很好的解决方案，CORBA 采用的面向对象管理体系结构，为分布式多厂商面向对象的计算环境提供了一个灵活通信的平台。通过使用 CORBA，用户可以透明地访问信息，而不用考虑信息所处的软硬件平台以及处于网络中什么位置。以 CORBA 的 ORB（Object Request Broker，对象请求代理）为核心，结合目前分布式对象体系结构、电信网络和业务管理方面的相关技术，即可以构造一个更加分布化、重用性强、可靠性高、可伸缩性好并且适用于整个 TMN 分层体系结构的新一代分布式 TMN 管理平台。具体说来，CORBA 的分布式特点可以应用在 TMN 中的不同层面，例如 CORBA 可以应用在 TMN 业务管理层，在 NE（Network Element，网元）内部采用 CORBA 技术，或者采用基于 CORBA 的 TMN OS 开发平台等。

正是由于 CORBA 带来的接入透明、位置透明、迁移透明、持久性透明、复制透明、事务透明及故障透明等优点及分布式开放环境固有的灵活性和可扩展性，CORBA 系统结构已被逐渐应用到网络管理系统当中，并成为未来网络管理的发展方向。

3. XML 在基于 Web 的网络管理中的应用

传统的网络管理都是基于专用的网络管理平台。这种专用平台价格昂贵，对网管人员要求高，人员培训和软件更新费用高，开发管理应用和集成新技术较难，而且受地理位置的限制，只能在机房进行管理。

WWW 技术为网络管理方式的革新提供了一种新思路，将 Web 技术引入到网络管理中，可以为用户提供灵活的管理方式。管理人员能够用 Web 浏览器在任何地方任何时候，接入到管理系统，监视、配置并管理他们的网络，还可以运用各种网络技术和工具来集成实现新的管理应用。基于 Web 的网络管理已经

成为网络管理的趋势和必然发展方向。

新兴的 XML（Extensible Markup Language）技术具有良好的数据存储格式、可扩展性、高度结构化、便于网络传输四大特点，为下一代网络上的互操作带来了光明前景。因此把 XML 引入到基于 Web 的网络管理中，也成为目前的一种趋势。

将 XML 应用到基于 Web 的网络管理中，主要可以应用在以下几个方面：用 XML 实现网络管理数据的发布和共享，在日志管理中应用 XML，用 XML 实现页面的多样性显示等。实践证明，利用 XML 可以使基于 Web 的网络管理更加成熟和完善。可以预计在不久的将来，XML 技术的应用会更加普及。

新一代网络管理技术已经成为当前通信领域研究的热点之一，这同时也说明了网络发展对网络管理有很大的依赖性，而在实践上，网络管理技术的研究又必须与网络结构的演进相适应。现在人们热心于寻求新一代网络管理的最佳方案，期望在下一代信息网络的发展过程中，实现网络与网络管理同步配套，更好地发挥网络资源的效能。但是要达到这一目标，特别是实现软交换的相关网络管理技术估计还有很长的路要走，还要经过很多的努力才能完成。

新一代网络管理系统应具有以下几个新技术特点：

（1）强大的管理功能

现代科技发展日新月异，采用高科技建筑的新一代网络必将具有高速度、大容量和高可靠性的特点，与之相适应的现代网络管理系统也必将具有强大的管理功能。除了 TMN 规定的配置管理、故障管理、性能管理、安全管理、计费管理等五大管理功能之外，资源管理、业务管理、故障与性能的相关性分析与趋势预测等智能化管理功能将成为网络管理系统不可缺少的管理功能。

（2）分布式处理能力

电信网是由各个时期的多厂商的各种设备组成的异质网络，大量分布在不同地域的各种 NE 协同工作，为用户提供通信服务。随着网络规模的膨胀和复杂度的增加，被管网络越来越呈现异构和动态特性。不断发展的计算机网络技术和日益复杂的网络管理应用的需求，对传统的集中式的 Manager-Agent 的模式提出了疑问。为了对这些分散的、复杂的通信设备进行有效地管理，从而更好地组织和提供业务，基于分布对象计算技术的网络管理技术将成为未来的发展趋势。采用 CORBA 等分布式技术建设的网络管理系统，不仅可以提供强大的网络管理功能，而且可以对地域上分散、结构上松散的网络进行分布式的多级管理，更好地利用现有网络资源，进行高效的管理。

（3）提供实时管理服务

电信网的基本要求是稳定、可靠的运行，提供高质量的电信服务。新时代的网络管理不仅要求为网络运行提供众所周知的五大管理功能，而且要求提供

实时的可监控功能和可管理能力，及时、迅速地反映网络资源的通信状况和运行状态，并对网络中可能出现的故障和通信质量的恶化做故障分析和预测，以电话、传呼或手机的方式提示网络维护人员在网络出现问题时应尽快进行修复和维护，实现无人职守、实时监控、人性化管理。

（4）适应跨平台环境

随着 Internet 的迅速发展，基于 Web 的网络管理系统已经成为网络发展的趋势之一。传统的集中式的专业网络管理平台由于需要专业的人员培训和软件更新，使得开发新的管理应用和集成新的管理技术非常困难。将 Web 技术引入到网络管理中，不仅可以改变以往只能在机房进行管理的限制，使用户能在任何时候任何地点使用浏览器接入到网络管理系统进行监控和管理，而且可以方便地集成和运用各种网络技术和工具实现新的管理应用，彻底实现网络的跨平台管理。

（5）具有可扩充性和自适应性

网络的不断扩大与业务能力的提升，使得网络管理变得越来越复杂，对网络级网络管理及业务级网络管理的需求越来越迫切。然而已经开通且正常运行的网络不可能终止运行，因而要求网络管理系统在运行过程中支持在线升级与系统的扩容。结构化的软件设计以及组件、插件的使用是实施可扩充性的有效手段和方法。

（6）用户界面功能人性化

随着网络管理系统复杂度的提高，用户对网络管理界面的要求也越来越高。一方面，操作简单、方便、易学易用是最基本的要求，另一方面，丰富美观、样式多变的工具条、菜单项，以及动画、音效、飞行 Tips 的运用，使得新颖美观的界面既富有吸引力，又方便了使用者。同时，由于网络管理者的操作习惯各不相同，为每个用户定制界面，提供符合操作习惯的人性化管理界面将成为网络管理用户接口的最终目标。

（7）具有多种接口

为了适应用户的组网需要，网络管理系统特别是网元级管理系统必须具有多种接口。例如 CORBA 接口、Q3 接口和厂商的私有协议接口。

5.2　软交换的网络管理

5.2.1　软交换的基本管理

软交换技术作为电信网下一代技术的核心，所以对目前已经成熟的 TMN 应该具有支持能力，但同时软交换是以分组网为承载网的系统，它必须对分组网

尤其是 IP 网的网络管理协议进行支持，因此如何将 TMN 和以 IP 为代表的分组网的网络管理技术融合起来进行软交换系统管理将成为一个亟待解决的问题。

传统 TMN 所采用的公共管理信息协议（CMIP）和 Q3 接口正在为大量的电信网络管理理提供强大的支持。而传统 IP 网络管理中应用最广泛的却是简单网络管理协议（SNMP），SNMP 已经成了事实上的 IP 网络管理标准。两者的不同点如下：

1. 管理架构

TMN 与 SNMP 在管理者与代理之间使用的通信协议不同，并且 TMN 与 SNMP 对于代理功能的要求也不相同。TMN 中代理的智能化程度较高，可以支持一些复杂的管理业务；SNMP 中的代理则比较简单。TMN 是一个相对“完美”的网络管理体系，功能完备，实现的复杂程度较高；而 SNMP 的设计较为简单，基于轮询的机制使其不适合大型网络的管理。

2. 管理协议

CMIP 和 SNMP 分别是 TMN 和 SNMP 网络管理中采用的通信协议。CMIP 提供的服务可以是有连接的，也可以是无连接的，而 SNMP 作为应用层协议通常采用无连接的 UDP 作为其传输层的协议。CMIP 功能强大，安全性较好，但是需要占用较多的系统资源；SNMP 功能简单，安全性较差，占用的系统资源较少。CMIP 支持 7 种公共管理信息服务（CMIS），如 M-GET、M-SET、M-CREATE、M-DELETE、M-ACTION、M-EVENT-REPORT 和 M-CANOE-GET、SNMPv2 提供的服务基本上包括 Get、Get-next、Get-Bulk、Set 和 Trap 等。在 TMN 中实现 IP 网络管理需要实现协议之间的映射。通常，M-GET 与 Get，M-SET 与 Set，M-EVENT-REPORT 与 Trap 之间存在简单的相似性，而 CMIP 支持的其他服务则需要通过组合 SNMP 的 Get、Set 和 Trap 等服务来实现两者之间的映射。

3. 管理信息模型

TMN 与 SNMP 都使用 ASN.1 来定义被管对象，不过 SNMP 使用的只是 ASN.1 的一个子集。TMN 使用面向对象的信息模型来定义被管对象，支持封装、继承等特性；SNMP 则使用简化的面向对象的信息模型来定义被管对象，不提供形式化的封装机制和继承等特性。从面向对象的技术角度来看，SNMP 中的 MIB 对象并不是真正意义上的对象，而是一些具有基本特征的简单变量，如数据类型、只读或读写属性。TMN 可以为资源的多个属性建模，而 SNMP 的对象只能为资源的一个属性建模。SNMP MIB 中的变量类型定义无法重用，TMN MIB 通过对象类实现了属性、动作和通报说明的重用。

TMN 和 SNMP 的 MIB 都以树形的方式构造，在 TMN 中，MIB 树的每个节点都代表一个被管对象，而 SNMP 中的 MIB 只有叶子节点才代表一个被管对象。

根据上面的分析，TMN 为电信网络管理目标的实现提供了一套整体解决方

案，能简化多厂商混合环境下电信运营企业的管理模式，降低电信运营的管理成本，这也正是电信运营商采用 TMN 的原因。为了提高效率和网络管理的能力，软交换的网络管理系统可以把 SNMP 纳入到 TMN 为出发点，进行软交换的管理体系设计。软交换管理体系系统的集成有多种方法，例如，可以通过管理者之间的通信实现集成，这种集成是松散而简单的；也可以利用 CORBA 等中间件实现集成，通过映射可以实现紧凑的集成。参考框架如图 5-1 所示。

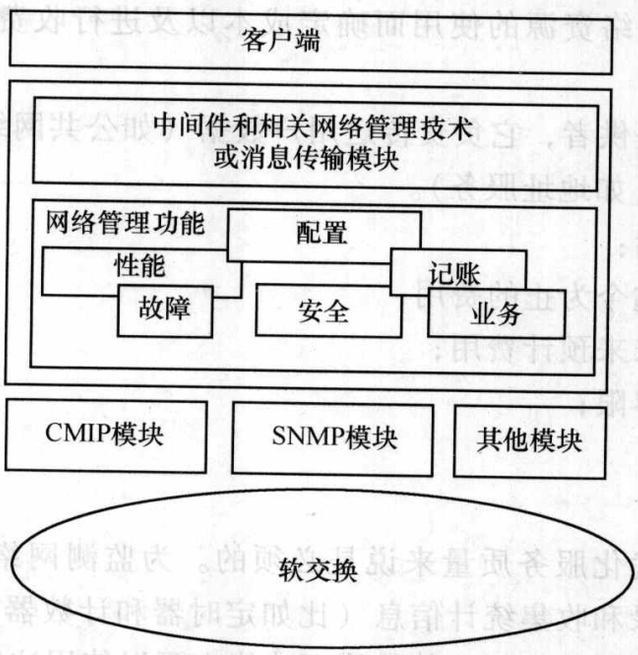


图 5-1　软交换管理体系的集成参考框架

5.2.2　软交换的基本网络管理功能

1. 故障管理

故障管理是能够监测、隔离和纠正异常运行的一组工具。可能导致异常运行出现的原因有：设计和实现上的错误、超载错误、外部干扰以及生命期终止等。故障管理包含的功能有：

（1）保存和检查错误记录；

（2）接收错误报告并且对此采取行动；

（3）追踪和确定故障；

（4）执行诊断测试；

（5）排除故障。

2. 配置管理

配置管理是如下的一组工具：

（1）记录当前配置；

（2）记录配置的变化；

（3）识别网络部件（给出 SAP 的地址和网络实体的名称）；

（4）初始和关闭网络系统；

（5）改变网络参数（如路由选择表）。

配置管理的一个重要特点是名字分配，为了强调其重要性，有时使用配置和命名管理这个术语。

3. 计费管理

计费管理是对网络资源的使用而确定成本以及进行收费的一组工具。这些资源包括：

（1）网络服务提供者，它负责传送用户数据（如公共网络）；

（2）网络应用（如地址服务）。

计费管理还包括：

（1）通知用户迄今为止的费用；

（2）通知用户未来预计费用；

（3）设置费用界限；

（4）累计费用。

4. 性能管理

性能管理对于优化服务质量来说是必须的。为监测网络性能的变化，应该附带地或周期地记录和收集统计信息（比如定时器和计数器地值）。对于这类记录的使用并不局限于性能管理，其他管理功能也可以使用这些记录：

（1）故障管理可以使用性能记录来检测故障；

（2）配置管理可以使用性能记录来决定何时需要改变配置；

（3）计费管理可以使用性能记录来调整计费账单。

5. 安全管理

安全管理是管理者用于初始化和修改防止用户对网络恶意和未经授权访问的功能的一组工具。

此外，OSI 管理框架还对管理信息的交换、被管对象、管理信息以及 MIB 等内容作了规定。

5.3 软交换的网络管理技术实现

作为软交换来讲，达到电信级的可靠性及性能要求是软交换进行运营的一个基本要求，作为软交换体系本身来讲，从宏观上分，它除了有支持普通业务的网络设备和应用服务器外，还有一个重要的逻辑上较为独立部分，即软交换的支撑或者管理系统。所以，对软交换的可靠性和性能要求不仅仅是对基本网络设备和应用服务器而言，同时还要求作为软交换的管理系统，它的管理能力

要达到 TMN 的要求，所以工程人员要将 TMN 技术和 IP 网络管理理技术结合起来，然后在软交换具体网络环境下，解决目前尚不成熟的软交换网络管理的实际问题。

下面我们将介绍一些网络管理的实现技术。

5.3.1　CORBA 技术

1. CORBA 技术融入 TMN 的背景

随着 TMN 的发展，大量的新技术涌现出来，其中最为瞩目的就是 CORBA 技术和 TMN 技术的结合，这为 TMN 的发展注入了活力，为什么 TMN 会和 CORBA 技术融合起来？首先，传统 TMN 技术有以下特点：

1）传统 OSI 系统管理所支持的传统模式是 Manager-Agent 方式，通过面向连接的 CMIP 和该协议支持的管理对象，适用于 TMN 网络层以下的管理，却不适合于业务层的管理对象，而随着 TMN 技术和运营支撑系统的融合，下一代网络技术对业务层的强大支持和管理不可或缺，因此，必须采用一种更适合的技术来解决业务层管理的技术难题。

2）TMN 在如何构造管理系统以及管理系统之间如何实现互操作上并没有较为成熟的技术；

3）相对于网元和由网元组成的网络而言，各种电信业务的异质性更强，同时业务种类多种多样；

4）每一种电信业务的实现通常要涉及不同类型的网络资源、业务组件，它还要和各种不同形式的数据发生关联；

5）每一种电信业务都是动态变化的；

6）很多电信业务还需要不同管理域之间相互配合才能完成，因而使业务的互管理、互操作性更为复杂；

7）由于 Q3 接口涉及到 OSI 的七层协议栈，CMIP 又相当复杂，实现起来代价较高；

8）TMN 网络管理平台主要提供对 Q3 接口协议栈的支持以及相应管理者和代理端应用框架的生成，缺乏特定 TMN 管理组件和支持分布式应用的基础构件，而这些恰恰是采用面向对象和软件重用技术构造大规模电信网络和业务管理应用的关键。

相比之下，CORBA 技术的优点有几下几个方面：

1）CORBA 是一种非常成熟的分布式面向对象技术；

2）CORBA 技术相对业务层管理来讲，有很强的优势。相对于 GDMO（Guidelines for Definition of Managed Objects，被管对象定义的准则）而言，IDL 非常简洁明了，更加适合于对业务对象的建模；业务对象之间可以通过 ORB 相

互调用对方提供的服务；有分布透明的支持，业务对象和组件还可以在网络内动态地迁移。因此，在 TMN 业务管理层采用 CORBA 技术将使得对电信业务的管理更具开放性、灵活性。

3）用 IDL 定义对外接口的组件通过 ORB 可以很好地支持访问和位置透明，能实现平台无关性。因此，以 CORBA 技术结合目前分布式对象体系结构和电信网络以及业务管理方面的相关技术，可以构造一个更加分布化、重用性强、可靠性高、可伸缩性好，并且适用于新一代分布式 TMN 的管理平台；

4）CORBA 具备很强的对象建模能力，对象之间可以通过 ORB 相互调用对方提供的服务，ORB 之间可以通过基于 TCP/IP 的 IIOP（Internet Inter-ORB Protocol）通信，实现起来代价较低；

5）有分布透明的支持，业务对象和组件还可以在网络内动态地迁移。

2. CORBA 技术的有关网络管理标准

ITU-T 第四研究组 2001 年对 CORBA 应用、XML/tML 应用和新一代网络管理问题进行了专题讨论。会议期间，共通过了 16 项建议，其中大部分建议都是与 CORBA 技术在网络管理中的应用有关，这些建议的通过表明 CORBA 在网络管理中应用的基础标准化工作已经完成。

与 CORBA 有关的基础标准分别是 X.780、X.780.1、Q.816、Q.816.1、M.3120、Q.821.1、Q.821.1 和 X.781，与这些标准有关的标准分别是 M.3010、M.3013、M.3100、M.3020、X.721、Q.822、X.739 和 Q.821。

2000 年版的 M.3010 和 M.3013 为 CORBA 技术引入到以 TMN 为基础的网络管理框架中铺平了道路；X.780 和 Q.816 分别规定了采用细粒度方法的基于 CORBA 技术的网络管理接口定义指南和所需 CORBA 服务；X.780.1 和 Q.816.1 分别规定了采用粗粒度方法的基于 CORBA 技术的网络管理接口定义指南和所需 CORBA 服务。

在 X.780、X.780.1、Q.816 和 Q.816.1 标准的基础之上，已有的 M.3100 和 X.721 规定的信息模型被映射为基于 CORBA 的通用信息模型 M.3120，M.3120 中既包括基于细粒度方式的 CORBA 接口信息模型，也包括基于粗粒度方式的 CORBA 接口信息模型。

基于 X.780、X.780.1、Q.816、Q.816.1 和 M.3120 标准的基础之上，Q.821 中规定的关于报警管理的信息模型被映射为基于 CORBA 的管理信息模型 Q.821.1，Q.822 和 X.739 中规定的关于性能管理的信息模型被映射为基于 CORBA 的管理信息模型 Q.822.1。

在以上基于 CORBA 技术的网络管理接口标准制订过程中，M.3020 这一项建议从规范层面上保证了采用 Q3（用 CMIP 管理协议和 GDMO/ASN.1 描述工具定义的网络管理接口）技术与 CORBA 技术定义接口信息模型时语义的一致性；

在 CORBA 接口的实现过程中，X.781 保证了接口的实现和接口规范之间的一致性。

3. CORBA 简介

CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）是为了实现分布式计算而引入的，为了解决分布式计算环境（Distributed Computing Environment，DCE）中不同硬件设备和软件系统的互联，增强网络间软件的互操作性，解决传统分布式计算模式中的不足等问题，OMG 提出了公共对象请求代理体系结构（CORBA），以增强软件系统间的互操作能力，使构造灵活的分布式应用系统成为可能。CORBA 有很广泛的应用，它易于集成各厂商的不同计算机，从大型机一直到微型内嵌式系统的终端桌面，是针对大中型企业应用的优秀的中间件。最重要的是，它使服务器真正能够实现高速度、高稳定性处理大量用户的访问。现在很多大型网站后端的服务器都运行 CORBA，其中有些网站您可能每天都在访问。

CORBA 的核心是一套标准的语言、接口和协议，以支持异构分布应用程序间的互操作性及独立于平台和编程语言的对象重用。CORBA 的底层结构是基于面向对象模型的，由 OMG 接口描述语言（OMG IDL，OMG Interface Definition Language）、对象请求代理（Object Request Broker，ORB）和 IIOP 标准协议（Internet Inter-ORB Protocol，也称网络 ORB 间协议）3 个关键模块组成。CORBA 技术有两个最大的特点：

1）与过去的面向过程的连接不同，CORBA 是基于面向对象技术的，它能解决远程对象之间的互操作问题；

2）CORBA 是真正跨平台的，平台独立性正是 CORBA 的初衷之一，它通过 IDL 能做到语言无关，也就是说，用任何语言都能制作 CORBA 组件，而 CORBA 组件能在任何语言下使用。

因此，可以这样理解 CORBA，CORBA 是一种异构平台下的语言无关的对象互操作模型。CORBA 的体系结构如图 5-2 所示。

CORBA 是一种多重绑定客户/服务器结构，CORBA 上的服务用 IDL 描述，IDL 分成两部分，在客户方叫 IDL Stub（桩），在服务器方叫 IDL Skeleton（骨架），两者可以采用不同的语言。服务器方在骨架程序的基础上编写对象实现（Object Implementation），而客户方要访问服务器对象上的方法，则要通过桩程序，而双方又要通过 ORB 总线通信。CORBA 解决了两层结构的客户/服务器模式存在的问题，例如耦合太紧，它们之间采用一种私有协议通信，服务器的改变将影响到客户方等。多重客户/服务器与此不同，两者之间的通信不能直接进行，而需要通过中间的一种叫代理的方式进行。

另外，对象实现部分要么通过 OMG IDL 产生的程序框架（Skeleton），要么

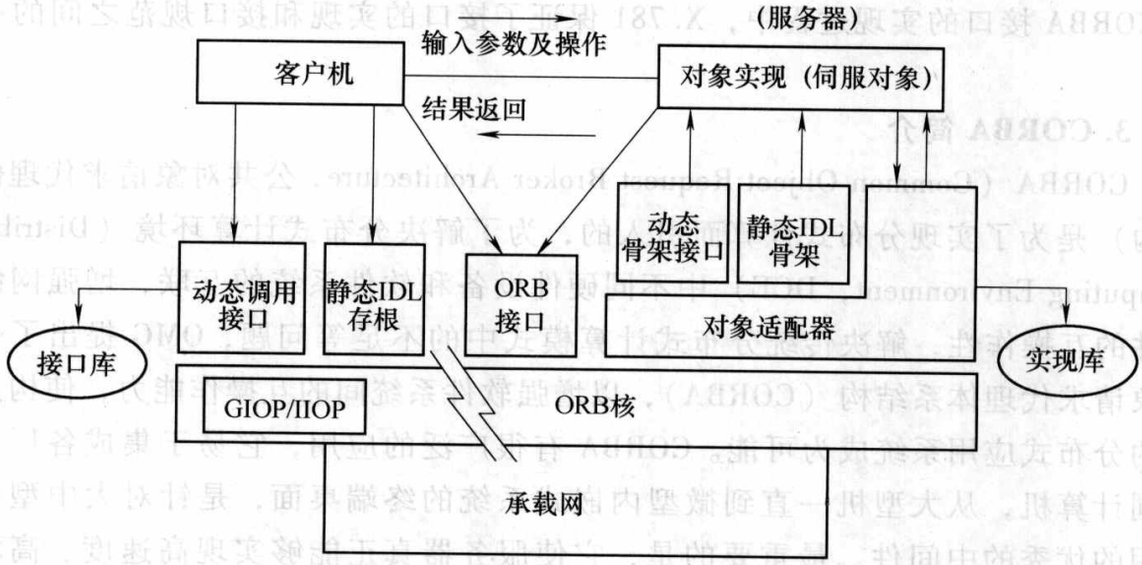


图 5-2 CORBA 的体系结构

通过动态程序框架（Dynamic Skeleton）接收作为上行调用（Up-call）的请求。在处理请求或其他任何时候，对象实现均可以调用对象适配器和 ORB。

对象接口的定义可以采用两种方法：

1）其一是用一种接口定义语言静态地对接口 IDL 进行定义；

2）另一种方法是将接口添加到接口库服务中，该服务将构成接口的组件表达为对象，并允许在运行时刻对其进行访问。

在以上任何一种 ORB 实现中，接口定义语言与接口库具有相同的表达能力。

4. 软交换的网络管理技术的 CORBA 实现方案

可以预见，CORBA 将被广泛用于开放的电信市场环境下的业务的快速构造及资源和业务的有效管理。一个参考模型如图 5-3 所示。

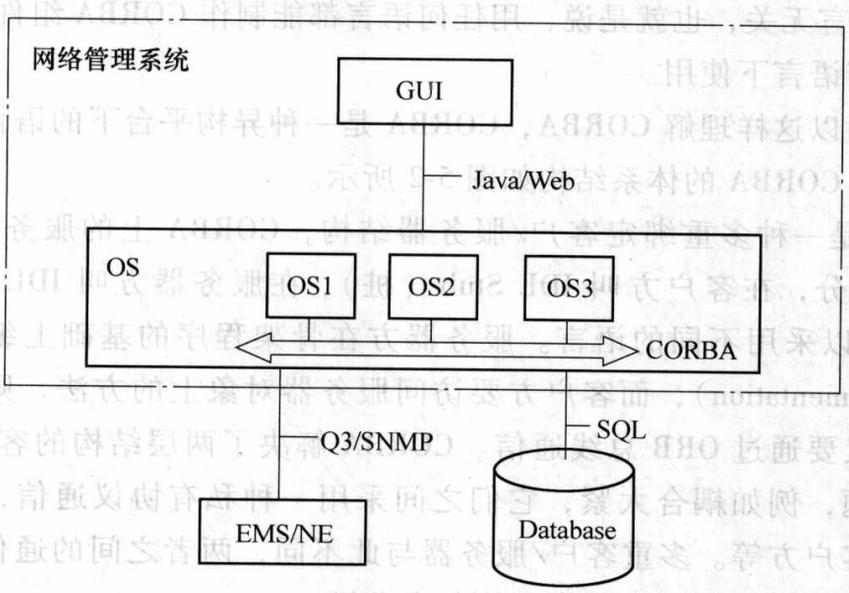


图 5-3 解决 TMN 的一个方案

电信管理网要求电信资源的互联，在一个综合平台上实现电信管理 。由于 Web 技术日趋成熟和广泛推广，以及它所拥有的以下好处：

1）维护简单，发布容易。我们只需在服务器端开发和修改应用程序，而不必在每个客户端进行修改，就可进行管理。

2）可以控制对多极用户的权限。我们只需在服务器端增加用户管理和管理注册，就可以针对不同的用户赋予不同的权限。例如可以对省局管理员赋予本省的所有权限，让一般客户拥有查看资源配置信息的权限等等。

结合上面介绍的 CORBA 分布式技术，在分布在各地的子资源上，利用原有的数据库，抽象出一系列对象，然后用 CORBA 实现这些对象，再在服务器上用 Web 来调用这些 CORBA 对象，构造主页，然后所有客户都可以通过浏览器来查看主页，并进行管理。

5. 基于 CORBA 技术的网络管理标准在应用过程中存在的问题

ITU-T 虽然已经完成了 CORBA 技术应用于网络管理中的基础性标准的制定工作，但是通过在实际软交换管理系统的应用中发现该标准要在网络管理系统的建设到实际应用还必须注意几个问题。

（1）软交换和传统 TMN 的不同给 CORBA 技术带来的挑战

软交换是支持多媒体、数据以及语音的下一代网络技术，它给传统的通信网带来了极大的革新，同时也为 TMN 技术带来了新的挑战。由于目前 CORBA 在 TMN 上的应用其规范是基于 TMN 的，所以如何面对软交换中出现的大量不同于传统网络的新特点，如何使 CORBA 技术支持软交换网络的有效管理，如何制定 CORBA 中新的适用于软交换网络管理的规范，是建设软交换为核心的 NGN 的一大挑战。

（2）软交换 CORBA 技术的一致性问题

CORBA 技术的引进扩展了 TMN 的内容，相应的关于 TMN 的一致性和顺从性也必须重新规定，目前，在软交换管理系统 CORBA 技术的应用过程中应该重点关注以下几个方面的一致性：

1）涉及 CORBA 互联互通的通信协议的一致性。这主要是指 CORBA 产品中的 ORB 与 OMG CORBA 规范版本的一致性问题。

2）管理框架的一致性。主要是指自定义规范对 CORBA 相关规范的支持情况。

3）管理信息模型的一致性。管理模型的建立要考虑同 Q3 接口管理信息模型的一致性。

（3）基于 CORBA 技术的管理接口的复杂度问题

TMN 框架要求采用 CORBA 技术实现的网络管理接口的功能，应该在语义上等同于基于 Q3 接口技术的实现，因此，为了弥补 CORBA 技术本身在实现网络

管理接口过程中存在的缺陷，ITU-T 重新引用或定义了大量 CORBA 服务，这些服务的规定使得 CORBA 接口的实现变得极为复杂，在实现过程中应该研究如何简化基于 CORBA 的 TMN 框架。

（4）符合 TMN 的 CORBA 产品化问题

目前市场上存在的 CORBA 产品种类繁多，但是由于 OMG 本身不存在对 CORBA 一致性认证的机构，因此 ORB 的互联互通存在着不完备性。

5.3.2 JMX 技术

1. JMX 技术介绍

目前，推出新业务的步伐越来越快，但是业务提供者在业务管理方面也遇到了前所未有的挑战。现在，人们可以通过网络激活、移动、使用或去激活他们所需要的资源（可能是应用程序，设备或者是服务和网络策略），这些资源可能分布在不同的网络节点甚至异构的环境下。如何才能动态有效地管理好这些资源成为业务提供者的一个难题。现有的管理技术在本质上都是静态的，它们只能管理事先已经知道的资源。这样一个静态的管理系统不能很好的管理新的业务，因为新业务的出现速度总是比系统更新的速度要快。

在这样的背景下，SUN 公司在原有技术的基础上推出了 JMX（Java Management Extensions，Java 管理扩展）。它并没有具体描述被管理对象的语义，而是一个为应用程序植入管理功能的框架，使得自描述的组件能够动态地加载进去。也就是 JMX 只定义了如何使产品可管理，而没有具体定义如何管理它。因此它可以使你的应用自由地使用任何符合 JMX 规范的管理程序。这样的描述可能很抽象，但是在了解了 JMX 的体系结构后就会对它有一个更深刻的认识。JMX 是一套标准的代理和服务，实际上，用户可以在任何 Java 应用程序中使用这些代理和服务实现管理，JMX 的前身是 JMAPI。

JMX 致力于解决分布式系统管理的问题，因此，能够适合于各种不同的环境是非常重要的。为了能够利用功能强大的 Java 计算环境解决这一的问题，SUN 公司扩充了 Java 基础类库，开发了专用的管理类库。

JMX 是一种应用编程接口，可扩充对象和方法的集合体，可以用于跨越一系列不同的异构操作系统平台、系统体系结构和网络传输协议，灵活的开发无缝集成的系统、网络和服务管理应用它提供了用户界面指导、Java 类和开发集成系统、网络及网络管理应用的规范。

管理对象是 JMX 应用程序的核心。JMX 结构包括：支持 Java 的 Web 浏览器用户接口，管理运行模块 ARM（Admin Runtime Module）和应用。这三个部件之间通过 RMI（Remote Method Invocation，远程方法调用）进行通信。这里需要说明的是，RMI 是使得一个 JVM（Java Virtual Machine，Java 虚拟机）上运行的程

序可以调用远程服务器上另一个 JVM 总的对象。

用户接口用来发布管理操作，这些操作可以间接的通过浏览器或通过单独的应用程序来激发。管理运行模块用来给应用提供实例化的管理对象。它包括 Agent 对象接口，通知接口和被管数据接口。应用指的是那些被管设备单元。

JMX 是一个完整的网络管理应用程序开发环境，它同时提供了：

1）厂商需要收集的完整的特性清单、可生成资源清单表格和图形化的用户接口；

2）访问 SNMP 的网络 API；

3）主机间远程过程调用，数据库访问方法。

JMX 这一轻型的管理基础结构，价值在于对被管理资源的服务实现抽象，提供了低层的基本类集合，开发人员在保证大多数的公共管理类的完整性和一致性的前提下，进行扩展以满足特定网络管理应用的需要。JMX 注重于构造管理工具的软件框架，并尽量采用已成熟的技术。

2. JMX 体系的层次

JMX 的层次如图 5-4 所示。

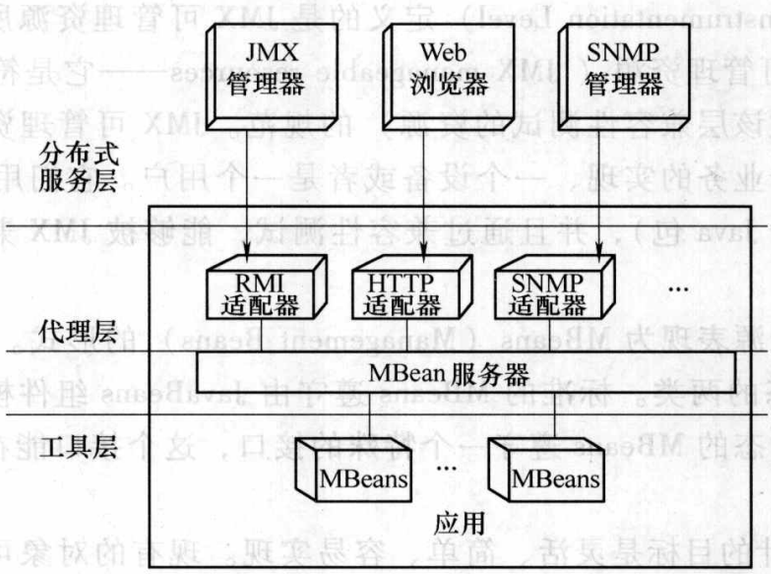


图 5-4　JMX 的层次图

总体上讲它分为 3 层：

1）表现层

2）代理层

3）分布式服务层

详细逻辑框图如图 5-5 所示。

下面我们做一下介绍：

（1）工具层

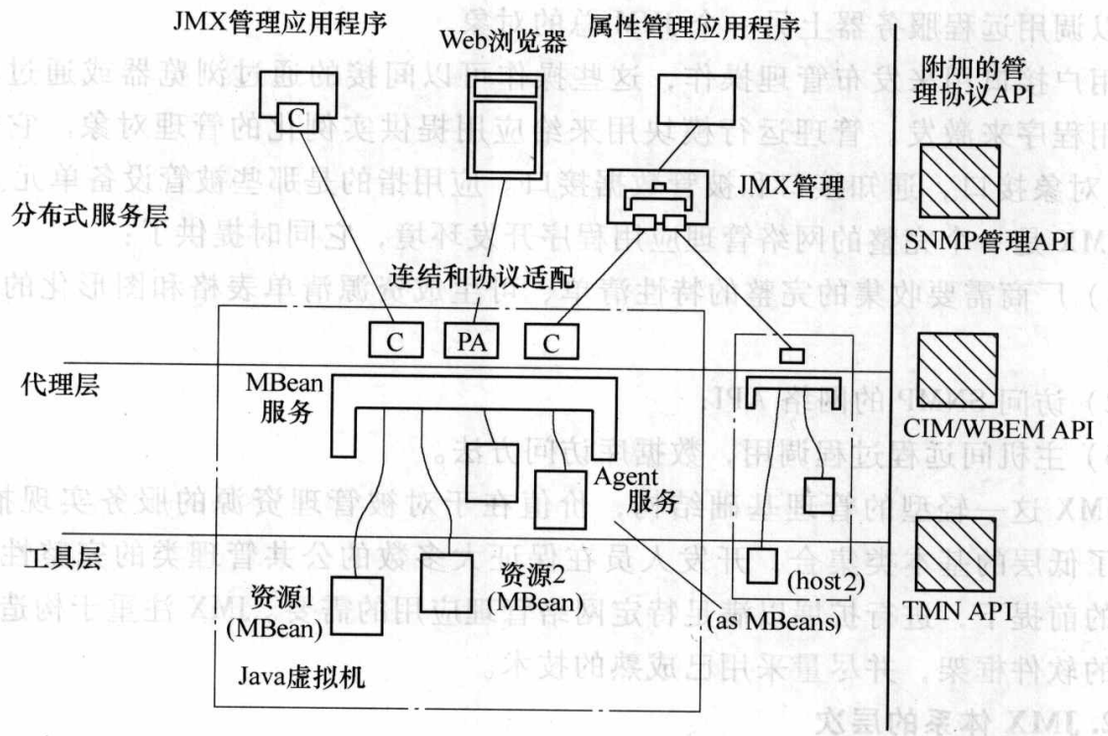


图 5-5 JMX 的详细逻辑框图

工具层（Instrumentation Level）定义的是 JMX 可管理资源所在的层。它提供了实现 JMX 可管理资源（JMX manageable resources——它是符合 JMX 的工具层规范并且通过该层兼容性测试的资源）的规范。JMX 可管理资源可以是一个应用程序、一个业务的实现、一个设备或者是一个用户。它们用 Java 开发（至少必须提供一个 Java 包），并且通过兼容性测试，能够被 JMX 兼容的应用程序管理。

工具层的资源表现为 MBeans（Management Beans）的形式。MBeans 可以分为标准的和动态的两类。标准的 MBeans 遵守由 JavaBeans 组件模型派生出来的设计模式，而动态的 MBeans 遵守一个特殊的接口，这个接口能在运行时提供更多的灵活性。

MBeans 设计的目标是灵活、简单、容易实现。现有的对象可以很容易升级成 MBeans 或是动态 MBeans 的包。

同时 Instrumentation Level 还提供了一种通知机制。这个通知机制允许 MBeans 产生事件，并将事件通知其他层的组件。

JMX 管理资源能够被兼容 JMX 的代理管理，同时它也能被支持 MBean 设计模式和接口的非 JMX 兼容的设备管理。

一般来讲，MBeans 分为 4 种类型：标准类型、动态类型、开放类型和模型类型。

（2）代理层

该层提供实现 Agent 的规范，管理代理直接控制底层的资源，使得远端的管

理应用程序能够通过它操纵底层资源。Agent 通常和它所管理的资源在同一台机器上（虽然这不是必需的）。

JMX Agent 由一个 MBean 服务器，一组操纵 MBeans 的服务组成。另外它还必须包括至少一个的协议适配器或连接器。

当机器上安装了 JVM 以后，JMX Agent 能够嵌入到包含 JMX 可管理资源的机器上去。它不需要知道它要给哪些资源提供服务，即使说任何 JMX 可管理的资源能够任意使用提供所需服务的 Agent。

远端的服务程序可以通过协议适配器或连接器访问 Agent 上的 MBean 服务器，并使用它所提供的服务，而 Agent 不需要知道使用它的管理程序的信息。

JMX Agent 能够自动与 JMX 分布式服务兼容，并且能够被支持 JMX Agent 的应用程序使用。

代理的组成：

1）一个 MBean 服务器；

2）要管理的 MBeans；

3）作为 MBeans 实现的代理服务；

4）至少一个协议适配器。

（3）分布式服务层

该层提供实现 JMX 管理程序的接口和能操纵 Agent 的组件（JMX Manager），并通过它们连接到外部的应用，如 RMI 管理应用、基于浏览器的管理控制等。这些组件有以下功能：

1）为管理程序提供一个通过连接器能和 Agent 透明交互的接口。

2）将管理信息映射到协议中去（例如 HTML 或 SNMP）。

3）将高层的管理信息分布到各个 Agent 上。

4）将底层的信息综合成用户关心的逻辑视点。

5）提供安全保证。

此外为了和现有的管理协议互通，JMX 还提供了一些附加的 API 函数。如 SNMP API、TMN API 和 CIM/WBEM API 。

应用程序的开发者可以使用这些 API 函数与现有的管理系统交互。例如，一个使用 SNMP 的系统中，如果要用 JMX 的解决方案代替原来的管理系统，那么只要开发一个 SNMP Manager，并将它封装成 MBean 就可以了。

3. JMX 的好处

1）可减少对 Java 应用实施管理的投资；

2）提供了一个可伸缩的管理框架；

3）集成现有的管理方案：如：WBEM，SNMP，TMN ；

4）使用现有的标准 Java 技术 ；

5）能使用未来的一些管理概念：如 Jini 连接技术、通用即插即用、服务定位协议（Service Location Protocol）；

6）只定义了一些可以访问的接口。

4. JMX 的发展现状

JMX 不仅仅是一个规范，它还提供了一套工具。这些工具可以帮助开发者快速开发出基于 JMX 的管理程序。它包括：

1）API 集合；

2）Java 文档参考信息；

3）相关应用；

4）兼容的测试软件。

JMX 规范目前定义工具层和代理层的规范，包括一些基本服务的接口，例如 MBeans 的注册，对已经注册的 MBeans 的查询，在资源上的操作，将事件返回给管理者，动态加载新的 MBean，创建 MBeans 之间的联系和依赖性，计时器功能和属性监测。

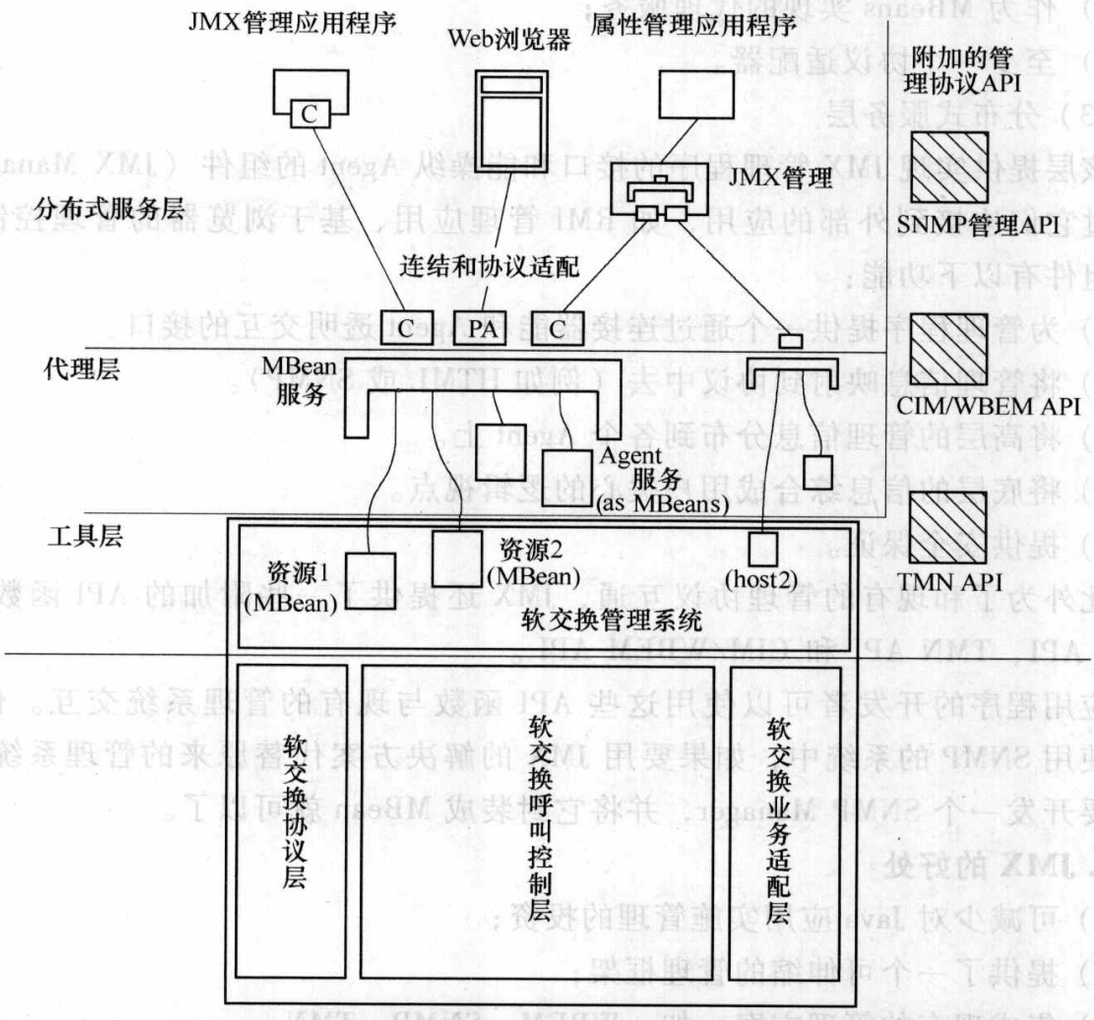


图 5-6 JMX 在软交换中的应用框图

5. JMX 在软交换的应用

在软交换管理系统中，JMX 的应用框架如图 5-6 所示。

在此框架中主要是将工具层中的 Mbeans 在软交换的管理体系中实现，并由相关的 Mbeans 实现对软交换的管理。

5.4　软交换的管理技术与下一代运营支撑系统的融合

5.4.1　电信业务运营支撑系统介绍及带给软交换技术的挑战

电信运营企业长期存在着业务环节多，流程周期长的弊病。如果要为用户提供一种业务，从用户申请、报批、安装、开通，到用户缴费，可能要经过几个月的时间，这对于一个用户数量众多、面临国际竞争的企业来说是非常不适当的，在各个环节之间流失的时间、资金、效率以及用户信任度难以衡量。因此，需要一个完善的、自动化程度高的系统去除不必要的环节，减少人为因素的影响，缩短投资回报的周期，并能够针对管理层提供决策支持的能力。

这个系统就是业务运营支撑系统（Business Operations Support System，BOSS）。BOSS 是电信运营企业的核心支撑，其主要功能有：业务规划、业务部署、网络规划、网络配置、网络管理、计费、财务管理、客户服务、客户管理、决策支持和内部信息交流等。传统的 BOSS 包括两部分的内容：一部分是 BSS（Business Support System，业务支撑系统），另一部分是狭义的 OSS（运营支撑系统），指电信网络和电信设备的运行维护支撑系统。业务支撑系统主要实现对电信业务、电信资费、电信营销的管理，以及对客户的管理和服务的过程。业务支撑系统所包含的主要系统包括：客服系统、计费系统、结算系统、经营分析系统等。狭义的 OSS 系统主要实现对电信网络和电信资源的管理，主要的应用系统包括交换网络管理系统、传输网络管理系统、数据网络管理系统、移动网络管理系统、资源管理系统等。

而在网络管理技术方面，ITU-T 最早提出了 TMN 的概念，从而促进了网络管理技术的发展。TMN 是一种电信网的管理体系结构，包括了对网络的规划、分配、安装、维护、运行和管理。但由于 TMN 模型具有高度抽象的特点，实现起来却很复杂。所以，电信管理论坛（TMF）在 TMN 的基础上，提出 TOM（Telecom Operations Map，电信运营图）模型。TOM 模型是电信业务流程框架规范，它以客户为中心，把端到端的自动业务流程放到了重要的位置。TMF 后来在 TOM 的基础上，又补充提出了 eTOM（enhanced Telecom Operations Map，增强型电信运营图）模型，为新一代运营支撑系统提出了业务流程框架规范。后来，

TMF 提出的 OSS/BSS 的新体系 NGOSS（Next Generation of Operations System and Software，下一代运营系统和软件），NGOSS 试图建立一种以构件为基础的分布式系统结构以及一套关键的系统服务，保证 OSS 具备标准化、能够逐步演化、保证互联互通、实现端到端的管理等特点。NGOSS 关注的是运营系统和软件，注重通过软件来实现业务流程的自动化，侧重的是业务流程和信息模型的定义、系统框架的定义、合作催化试点项目的实施等关键元素。

业务运营支撑系统的优势：

1）系统的引入和不断革新大幅度地提高了电信运营企业生产作业的自动化水平，使得绝大多数的关键业务和关键流程都纳入了计算机管理的范畴，增强了业务处理能力，提高了生产效率；

2）加强了运营管理的有效性，同时也使得很多关键的业务流程得以固化，减少了人为的因素和管理的随意性；

3）运营支撑系统在提升管理效率的同时也提升了电信运营企业的客户服务水平；

4）子系统之间点对点通信的模式，使其变为一个逻辑的接口环境；

5）将业务流程处理过程进行集中化管理；

6）提供通用的数据模式和实现方式；

7）减少运营维护的费用；

8）增加运营商选择软硬件提供商和系统集成商的机会。

目前，在中国电信和中国网通，网上在线运营的支撑系统主要的系统包括“九七系统”、计费账务系统、客服系统、资源管理系统、经营分析系统等。随着信息化战略以及信息化建设已经开始逐步实施，各省公司都开始以新的思路规划自身的 OSS。从趋势上看，系统的建设将逐渐由本地网集中向省过渡，功能系统之间将共享核心数据模型。

在电信北方各省和网通的南方各省，由于没有遗留系统，支撑系统的建设基本都是按照“一体化”的模式。这体现在 BSS 上，各省基本上都是建设一套综合业务支撑系统，而在里面涵盖了业务受理、计费、账务、结算、产品管理、代销商管理等功能。

中国移动的支撑系统比较有代表性的是移动 BOSS（Business & Operation Support System，业务运营支撑系统），它具有两个突出的特色：首先它是一套系统，是一体化的业务支撑，其次它是省集中的建设模式。BOSS 的功能范围包含了数据采集、计费处理、网间结算、客户服务、业务管理、综合账务、系统监控、联机指令、BOSS 网络管理和数据挖掘等各个方面。

联通的业务支撑系统联通 BSS 也是按照省集中的模式建设的。在联通的系统中营业和账务是建设在一起的，这与电信有所不同。同时在联通 BSS 中还包

含了统一客户资料管理的模块，用以实现多业务的交叉优惠。

不难看出，对于运营商来讲，运营支撑系统在企业运营中已经开始发挥核心作用，运营支撑系统的建设作用已经具有战略意义，这给下一代网络技术的核心--软交换技术带来了新的挑战，如何让软交换技术融入运营支撑系统，让软交换部署之后能够真正地在运营商现有网络资源的支持下发挥更大作用？这一直是软交换研究人员和工程人员考虑的重大问题。只有了解了运营支撑系统和软交换技术，才能将软交换的管理和业务支持与运营支撑系统更好地联系起来，发挥软交换技术更为强大的能力。

5.4.2　业务运营支撑系统的发展与组成

BOSS 的发展是伴随着通信行业和信息技术的发展而不断发展的。BOSS 在 20 世纪 70 年代侧重于大量的手工操作生成流程开始，到了 20 世纪 80 年代和 90 年代早期进入了解决支撑系统中核心系统的自动化时期，从 20 世纪 90 年代末期到现在，有关业务流程的管理和市场演变得更加复杂，这种演变的趋势引发了 BOSS 技术的重组。

但是现在的市场则大不相同，其灵活性、流动性以及不可预知性都大大提高，“游戏规则”也已不复明确。现在 BOSS 供应链上的驱动力是通过迅速的技术重组，以适应对速度、规模、灵活性、易于编程以及价格的需求。

目前国际电信行业处于一个低迷时期，国内的电信行业也开始呈现增长速度减缓的趋势，逐渐进入一个低速稳步发展的周期。电信运营商为了维持运营而缩减对硬件设备、网络设备的投资，但是对 BOSS 的投资和改造却没有放慢步伐，原因是来自多个方面的：

1）原有的 BOSS 没有跟上电信迅猛发展的脚步，要在市场上站稳脚跟就必须对原有的 BOSS 进行改造和更新；

2）BOSS 方案需要从原有的技术驱动型向当前的技术为驱动、客户为中心的模式转变，使 BOSS 成为运营商提升运营效率、提升客户服务质量、拓宽业务种类的引擎；

3）原有的 BOSS 大多数采用紧急需求驱动的模式，形成条块化建设的 BOSS，每次建设针对不同的需求，不可避免的造成不同项目引入不同的技术和某些功能模块的重复建设问题；

4）电信行业的发展和 IT 技术的发展密切相关，随着 J2EE、XML、Web Service、中间件等技术的出现，旧的 BOSS 的结构、技术面临整和与改进，因此电信运营商对新一代 BOSS 提出了很多新的期望。

从系统构成上来看，一个典型的业务运营支撑系统（BOSS）一般包括 4 部分：计费系统、营业和账务系统、客户服务系统、综合网络管理系统和决策支

持系统。

1. 计费系统

计费系统是指处理计费数据采集和核算计费两个过程的系统。计费数据采集工作包括从电信基础网络（如交换机、网关等）上收集有关原始基础数据和信息，进行相应的预处理，生成的记录用户使用网络状况的各种数据。而核算计费则是根据既定的原则和规则，对用户使用网络的情况计算费用。

2. 营业和账务系统

营业系统通常完成的是受理和处理用户的业务请求，而账务系统是将用户使用电信网络的情况汇总形成账单。

3. 客户服务系统

客户服务系统原来指的是企业的服务热线，如中国电信的“10000”和中国联通的“1001×”等，它的作用包括：

1）保证为客户提供快速方便的服务，与营业系统和账务系统进行信息交互，满足用户的查询、咨询、业务变更等业务；

2）保证在未来新业务开放的情况下，系统能及时满足多元化服务、个性化服务、交互式服务和异地服务的要求，向用户提供相应的功能。

4. 综合网络管理系统

网络管理系统主要实现网络资源数据管理、网络服务管理、网络容量和拓扑管理、网络故障分析、网络服务层测试、网络服务规划、网络性能分析、网络告警管理等功能，包括各种专业网的网络管理系统等。

5. 决策支持系统

决策支持系统的主要任务是通过动态、有选择性地采集和更新数据源的有效信息及企业外部的相关信息，进行智能化地分析、处理、预测、模拟等。最终向各级决策管理者或专业人员提供及时、科学、有效的分析报告，帮助运营商制定正确的市场应对策略。

电信业务支撑系统在整个电信系统中的位置及其外部截面如图 5-7 所示，该图所反映的业务支撑系统定位与具体运营商无关，是一种通用的系统划分。

电信管理论坛推动的 NGOSS 研究就是为了满足电信运营商的这些期望，通过对成熟的软件系统的组装和配置，快速、灵活、低成本地建设出适应不断变化的市场环境的新一代 BOSS。

5.4.3 NGOSS 概述

正是由于电信业务运营支撑系统对电信运营商的重要作用，全世界各电信运营商和设备制造商都十分重视对电信运营支撑系统的研究。ITU-T 最早提出了 TMN（Telecom Management Network，电信管理网）的概念，这大大促进了网络

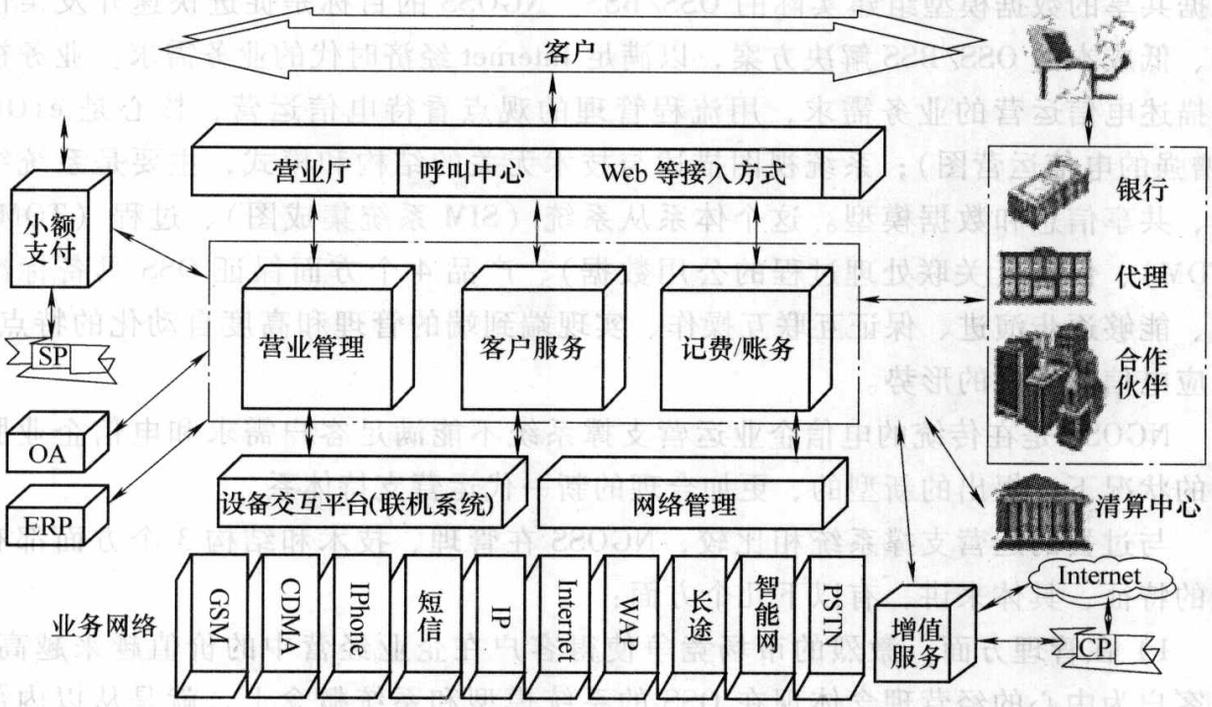


图 5-7　电信运营支撑系统定位

管理技术的发展。TMN 是一种电信网的管理体系结构，从上到下依次为商务管理层、业务管理层、网络管理层和网元管理层。

TMN 模型具有高度抽象的特点，从以往的经验来看，TMN 模型简单，但是实现起来却很复杂。在这种背景下，电信管理论坛（TMF）在 TMN 的基础上，提出 TOM 模型，TOM 模型是以客户为中心的电信业务流程框架规范。TMF 后来又在 TOM 的基础上，补充提出了 eTOM 模型，eTOM 模型是 TOM 模型的扩充，它扩展到企业的全部架构，把客户操作过程的业务开通（Fulfillment）、保障（Assurance）和计费（Billing）引入高级架构，从而强调企业关注的焦点是客户优先过程，并考虑到了电子商务的发展对运营支撑系统的影响。NGOSS 是 TMF 提出的新一代运营支撑系统框架，包括一系列的标准用来规范和指导新一代电信业务运营支撑系统的业务分析、系统设计、软件开发和部署，吸引了全球电信软件开发商和电信运营商的目光。

1. NGOSS 的含义和特征

NGOSS 是 New Generation Operation System and Software 的简写，从商务、系统、实现和运行 4 个视点描述 OSS，而且这 4 个视点分别与实现 OSS 的各个阶段相对应，这时的 OSS 不仅涵盖了 BSS 的含义，而且主要的特征是以客户为中心，因此这一阶段的系统通常被称为 OSS/BSS 或者 BOSS。NGOSS 提供了一整套的方法论来指导 OSS/BSS 的设计与建立，系统设计时首先从流程分析开始，分析自己企业的核心业务流程；然后通过分析流程，提炼出所有关键的数据，利用共享信息模型将数据进行抽象，划分为不同的管理域建立共享的数据模型；最后

根据共享的数据模型组建实际的 OSS/BSS。NGOSS 的目标是促进快速开发柔性的、低成本的 OSS/BSS 解决方案，以满足 Internet 经济时代的业务需求。业务视图描述电信运营的业务需求，用流程管理的观点看待电信运营，核心是 eTOM（增强的电信运营图）；系统视图描述与技术无关的结构和模式，主要是系统结构、共享信息和数据模型。这个体系从系统（SIM 系统集成图）、过程（TOM，eTOM）、信息（关联处理过程的公用数据）、产品 4 个方面保证 OSS 具备标准化、能够逐步演进、保证互联互操作、实现端到端的管理和高度自动化的特点，适应电信业发展的形势。

NGOSS 是在传统的电信企业运营支撑系统不能满足客户需求和电信企业服务的状况下，提出的新型的、更加合理的新一代运营支持体系。

与过去的运营支撑系统相比较，NGOSS 在管理、技术和结构 3 个方面都有新的特征，具体来讲，有以下几个方面：

1）在管理方面，激烈的市场竞争使得客户在企业经营中的价值越来越高，以客户为中心的经营理念体现在 OSS 的系统模型和系统概念上，就是从以内部管理为出发点、以技术为中心转移到以客户为出发点、以服务为中心上。以客户为中心的经营理念体现在 OSS 的系统模型和系统概念上，主要有两个方面：

① 提供统一的客户服务层，实现“一台清”、“一单清”；

② 把客户特征贯穿于整个业务流程中，为客户提供差异化服务。

2）在技术方面，为了保证迅速响应客户对新业务的需求，OSS 必须能够以最快的速度对新业务和新技术提供支持，使企业迅速将这些新的技术和业务推向市场。同时，OSS 本身使用的技术手段也在不断发生变化，为了更好地为电信运营服务，一个好的 OSS 体系必须能够支持业务、管理的任何变化，随时使用最有效的技术支持系统自身的不断升级，最大程度的保证系统的可重用性。

3）在结构方面，电信管制逐步解除，竞争日益激烈，单个运营商往往不能满足客户提出的所有需求。在 OSS 方面，各电信服务提供商和网络运营商之间的互联互操作也是不可避免的。NGOSS 试图建立一种以构件为基础的分布式系统结构，以及关键的系统服务。这些结构和服务将支持信息和通信业所需的动态商务和运营管理。系统概念的抽象使系统结构与技术无关，以便企业自由选择其适用的技术，并能应用今后出现的新系统技术。

4）从过程角度来看，为了保证不同企业能够协调。共同通过 OSS 向客户提供其需要的服务，在商务和运营方面必须有统一的概念。TMF 提出的 TOM 和 eTOM 为企业的业务重组和 OSS/BSS 的开发提供了共同的模型。在这种共同模型基础上开发出的系统，无论采用何种技术，都可以有效无缝地衔接。应当说，TOM 和 eTOM 是 NGOSS 的指导方针。

5）从信息交互的角度来看，为了保证构件之间、系统之间的互操作，TMF

提出了以合同方式定义的接口。接口的内容由构件本身所支持的操作决定。构件的产生一般遵循以下主要原则：

① 从运营过程或商务过程的需要出发，定义能够支持这些过程的管理域；

② 部署的业务构件组成能够对管理域进行实现和支持；

③ 可配置的业务构件进一步分解为事务对象。

以上是 NGOSS 的总体出发点和解决同一个问题所涉及的多个方面。为了保证其通用性和可持续的演进，NGOSS 是技术中立的也就是和技术无关的。在具体实施时，还需要结合当前可支持的技术，考虑遗留系统的无缝集成等问题，在这一框架指导下，做出切实可行的方案。

2. NGOSS 的特征

NGOSS 的目标是为即插即用的 BOSS 组件的开发、部署和修改提供框架结构、分析基础和过程方法。NGOSS 包括以下核心特征：

1）分布式体系结构，保证系统的可扩展性、高性能和高可靠性；

2）通用的集成框架结构，以支持不同的应用系统接入到整体系统框架中，并保障有效的通信互联；

3）通用的数据格式和通用的信息服务以支持 NGOSS 各系统之间的信息交互；

4）接口定义了在 NGOSS 中协同工作的各组件底层集成的接口框架；

5）剥离业务流程和具体业务流程的实现，从而保障对复杂的业务流程的支持，最大限度地减少人为协调的工作量；

6）充分利用成熟的产品组件，包括业务应用、框架组件、工作流引擎等。

NGOSS 借鉴分布式系统的“以流程为驱动、以模型为中心的”思想，是一种松耦合的分布式系统。为了适应技术的发展，支持 NGOSS 映射到先进技术上。

3. NGOSS 对软交换的支持

软交换是利用分组交换网络提供实时通信，包括电话、视/音频会议以及多媒体协同提供业务的网络。为了满足网络规模和容量的高速增长、技术和产品日新月异、电信服务全球化的运营要求，下一代网络运营支撑系统必须同软交换技术发展结合起来，以满足业务的创新与快速提供的需要，满足高质量客户服务的需要，满足商业模式调整的需要。这主要体现在几个方面：

1）软交换是数字化的分组网络，几乎所有的业务都构建于面向非连接的 IP 之上，这对于网络的性能和业务流程都有更复杂的要求，需要管理许多特殊网关以及和 PSTN 之间的互通；

2）软交换的运营支撑需要综合性和多功能的支撑软件；

3）软交换的运营商为了提高效率就必须依靠 OSS 来快速服务市场以赢得客

户的满意；

4）由于 Internet 中间件发挥了越来越大的作用，因此要求将服务器也纳入网络的统一管理中来；

5）激烈的竞争要求运营商与客户、合作伙伴之间需要进行更多、更高层次的交流。

现在我们来看 NGOSS 的功能。NGOSS 支持智能网技术，支持对业务和网络单元的通用管理，支持业务的快速开发和实时提供、实时计费，支持面向对象的接口，支持用户的自助服务和通过 Web 建立用户与运营商之间的接口，支持分布式管理，支持综合性加密安全解决方案，并且支持商业过程的重新设计。NGOSS 具备多方面的能力：

1）快速部署。新的解决方案能够在尽可能短的时间内开始运作。

2）快速集成。OSS 必须是开放式系统，能够迅速、灵活地形成综合业务的解决方案，综合业务网络运营商必须保证网络高可用性和可靠性，以及向客户承诺的 QoS。

3）容易配置。运营商能够通过软件很容易地定义各种功能和新业务。

4）基于组件。基于组件的解决方案使运营商能够方便地实时引入新用户和新业务。

5）具有灵活性和可扩展性。由于多技术、多厂家和多平台环境的日益成熟，运营商可以很容易地改造网络以满足用户和市场的需求。

由于 NGOSS 是基于组件的面向对象的分布式运营支撑系统，所以它对于新设备、新业务的引入具有天然的适应性。特别是当网络中引入了新的软交换设备后，其性能的管理仍然可以定时对于原始和统计数据的扫描，报警仍然应该可以进行分流、分级管理，监控仍然可以在各个阈值中进行。所以，网络管理层仍然可以将这些软交换设备对于网络整体的影响纳入管理范围，相应的服务、管理策略自然也就应用在新设备上。同样道理，新的用户类型、新的服务、新的业务策略也将一视同仁地通过网络管理施加在所有新老设备上。

以上可以看出，使用 NGOSS 不仅可以满足运营商当前的需求，还能够适应运营商未来的发展。NGOSS 是满足软交换的运营支撑系统，eTOM 作为电信运营业务流程向导的蓝图，是 NGOSS 的重要概念和关键组成元素。由于 eTOM 较好地代表了电信运营业的真实世界，为服务提供商提供了一个通用的框架，以便于在复杂的行业中讨论复杂的技术和复杂的业务需求，很多服务提供商已经在运用 eTOM，并把它作为核心的流程框架，通过业务流程的实施来管理企业。

第6章　软交换的计费管理

对软交换来讲，计费是个非常重要的问题，也是运营商最为关心的问题之一，同时考虑到和运营支撑系统的技术融合，因此在大多数软交换的构建应用中，计费总是作为一个单独的系统出现的。

传统意义上的计费系统主要应用于电信网，各种计费格式和协议也主要针对交换机环境设计。随着Internet的发展，IP网上的计费也日益普遍，出现了基于包格式的计费方式。随着网络融合趋势的发展，又出现了混合环境下的计费问题。对于软交换自身来讲，由于它能支持No.7信令网和各种VoIP技术，同时还实现了上层Parlay API等业务接口，实现对业务层的支持。所以，造成了软交换协议种类多，业务类型复杂的特点，从而给软交换计费带来了极大的挑战。

6.1　计费系统概述

传统的计费状况十分复杂，从支撑的网络系统的角度区分，大致有传统的电信计费、智能网计费、移动计费和IP计费；从业务的角度划分，又可分为语音业务计费、增值业务计费、智能业务计费、数据业务计费等。但没有一种统一、明确的方法真正地区分各种计费情况。传统的计费系统是一种固定线路服务模型：计费按时间与距离对语音呼叫计费。宽带应用出现之后，传统计费方式无法计算这些新业务所需的费用，例如：通过有线传送数据、VoIP，数字有线电视和视频点播业务等。新的业务模式带来了挑战，这些应用的出现都需要一种具有新结构的计费系统。随着各种新业务不断开放，对运营商服务要求不断增加，原有计费系统愈来愈不适应发展的需要，必须有适应新一代网络的计费系统来维持市场竞争和运营商的利润。

与其他事物一样，企业网络的计费系统也是从简单到复杂，随着社会需求的变化而发展，网络计费管理主要经历了下面几个发展阶段：

1）互联网接入控制阶段。方式比较简单，企业建立网络地址数据库，规定某些地址员工不能访问或者是某些地址可以访问。这一阶段基本上还谈不上计费管理。

2）初步的互联网接入管理阶段。企业网中实现一定的管理功能，从而实施对不同人员、不同部门的上网权限管理和资源分配，同时也形成网络计费管理的需求。

3）企业网络管理计费阶段。企业网实现对不同对象进行不同的网络带宽分配，建立详细的员工访问互联网的日志，提供详细的报表，能帮助企业管理层分析员工的上网行为，并对上网策略进行相关调整。网络在实现网络计费功能的同时，也能实现比较完备的上层分析功能，以帮助企业管理人员、人力资源部门和 IT 等部门解决更多的问题。

目前国内针对不同类型的网络服务，计费策略也各不相同：语音服务以时间为依据；数据服务则通常采用以时间、通信量为标准的复式计算，软交换由于本身的特点应采用混合计费的方式。一般情况来讲，软交换的网络计费系统组成大体可以分成两大部分：数据采集及处理（即对采集的数据进行分析处理）和计费管理。其中，比较关键和困难的部分是计费数据的采集和处理。对于目前的软交换网络来说，由于网络上的设备主要有软交换设备及其外围网络设备系统，因此计费数据可以从软交换设备上获得。

6.2 传统计费系统原理

整个网络计费系统主要涉及三大方面，分别是计费信息的采集、计费信息的处理及实施计费、计费结果及相关信息的发布，如图 6-1 所示。

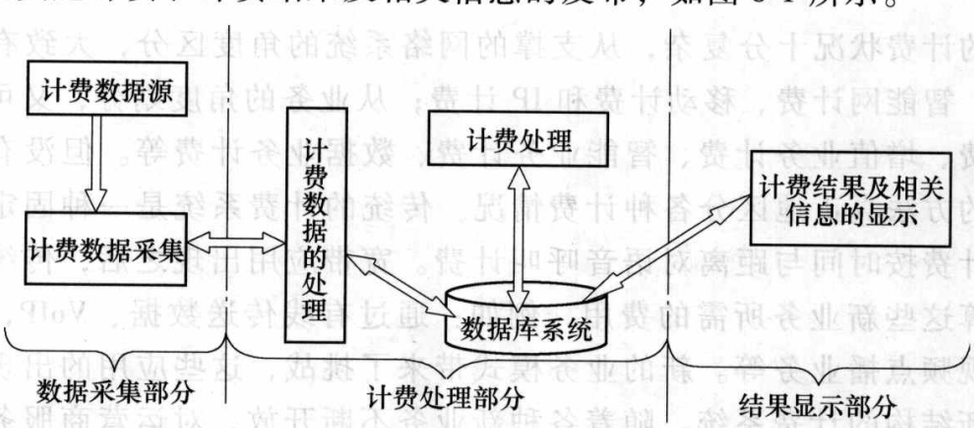


图 6-1 网络计费系统框架

其中，计费处理和显示可以合称为计费管理部分。数据采集处理和计费管理作为软交换计费系统的两个子系统，各具有一定的独立性。数据采集处理部分为计费系统提供计费所需的基本信息，这部分数据的可靠性决定了计费系统的性能；而计费管理解决如何根据这些数据进行计费，如何将这些数据以适合用户要求的形式提供给用户，完成用户的查询、打印等要求。计费管理部分直接与用户交互，而数据采集和处理部分只是作为一个服务进程，在后台运行，并不断地从软交换设备获取计费数据。

计费信息采集部分是整个系统计费的基础所在，下面我们举一个 Internet 用

户的例子。从链接到因特网的方式上来讲，用户链接因特网一般会有两种情况，一种是拥有固定 IP 地址，另一种是通过拨号链接因特网。在网络计费管理过程中，当用户拥有固定的 IP 地址时，用户所使用的计算机可能属于某个 TCP/IP 子网，或者直接作为 TCP/IP 网络上的一台主机使用网络资源，计费系统通常可以通过简单网络管理协议（SNMP）中的 snmpget、snmpset、snmpgetnext 等基本操作，从网络设备的代理处获取设备 MIB 的计费信息。当用户的计算机拥有的 IP 地址是通过 DHCP 动态分配所得时，典型的网络环境是采用 SLIP 或 PPP 与 TCP/IP 网络的一个提供接入服务的主机相连。这时可以使用相关协议记录拨号上网用户端口的连接时间和使用的端口号（IP 地址），再统计出该时间段内该 IP 地址的数据流入/流出总量，然后根据规定的计费规则进行计费。

另外，还可以根据服务器的日志信息等来获得计费所要的数据源信息。计费系统在采集计费信息后，经过一定的预处理，将计费信息按照一定的格式存入数据库。计费系统根据系统用户设计的计费方案与计费规则对数据库中的计费信息进行相应的处理，得到计费结果，并把结果存入数据库中，从而实现对用户的计费。数据库中还要存放用户的流量日志信息和其他与用户有关的资料信息。

计费管理部分中的其中一个部分就是计费结果及相关信息的显示和系统管理员对系统的设置管理。无论是系统管理员还是普通的用户都需要对系统中所关心的信息进行查询、维护与管理。如：系统管理员对计费规则和计费政策的制定，数据管理与数据维护，对普通用户的管理操作；普通用户对自己的账号使用网络资源情况的详细信息查询，网络用户根据这些信息可以计算、核对自己的收费情况。为了管理和维护上的方便，计费系统的管理以及用户的查询多半采用基于 Web 的方式，以便普通用户和管理员的远程操作。目前在软交换应用中，可用的技术有 JSP、ASP 或 PHP 等，同时还应当考虑系统的安全问题，特别是系统管理部分的安全问题。

传统的计费系统根据数据采集的方案的不同基本分为以下几类：

1. 基于数据流量集中流经的网络路由设备的计费方案

传统网络中，绝大部分路由器都处在网络的重要节点上，路由器自然成为网络中流量信息比较集中的地方，而且现在一般的路由器中都有记录流量的功能。网络流量经过路由器时，一些统计数据（如进出端口的总包数、总字节数等），还有一些 IP 的细节流量数据被记录下来。通过开发相应的数据流量采集程序，可以从路由器上采集流量数据。通过路由器收集好流量数据后，利用 SNMP 就可以从路由器获取这些数据。一般，路由器的 MIB 里包含了与计费相关的 MIB 值，因此可通过 SNMP 操作来获得这些计费数据。例如在 Cisco 路由器里该两个 MIB 值保留了对 IP 地址数据流量的一张表，其格式举例见表 6-1。

表 6-1 路由器 MIB 流量记录

源地址	目标地址	字节数/B
211.12.202.1	211.12.202.4	2048
211.12.223.112	211.12.223.144	5670

表 6-1 中，第一行数据表示从 IP 地址为 211.12.202.1 的主机到地址为 211.12.202.4 的主机的数据流量为 2048B。

使用 SNMP 从路由器获取流量信息的方法具有以下一些主要缺点：

1）必须有相关功能的边界路由器的支持。

2）由于路由器处在网络交通的枢纽位置，其计费信息将非常庞大，因此要求路由器有足够的 RAM 空间缓存数据，占用较多的路由器资源（包括存取资源和处理时间），从而影响路由器的效率。路由器流量数据的传送本身就会占用相当的系统带宽，这样就使得原本已经是网络主要瓶颈之一的路由器工作效率更为低下。

3）采集时间间隔较难确定。如太长，则可能由于某段时间计费信息的数据量太大而发生溢出，造成计费数据丢失；如太短，又使得对网络信道资源的开销增大。

4）无法区分服务类型，即不能按应用服务器的服务端口分别统计流量，从而无法针对不同的服务类型制定不同的费率。

5）由于 SNMP 建议使用 UDP 作为它的实现方式，在网络拥塞时无法保证数据传输的可靠性。

2. 基于以太网共享总线的计费方案

共享式以太网大都采用总线型拓扑结构，以太网中的各台主机采用 CSMA/CD（Carrier Sense Multiple Access with Collision Detect，载波侦听多路访问/冲突检测）技术共享以太网总线，在同一时刻网络上只有一台主机的数据帧在流动，而这个帧会流经同一网段内的所有主机，所以只要在以太网中接入一台流量采集主机，就可以侦听到网络中其他主机的数据帧。根据各种数据包的格式对这些帧进行分析，可以从中提取出数据包的大小、源 IP 地址、源端口、目的 IP 地址、目的端口等信息，再对这些信息进行统计合并，从而实现流量采集功能。

3. 存在统一流量进出口情况下的基于交换机的计费方案

当要采集流量的几台应用服务器处于同一共享网段内时，简单地在共享式集线器（Hub）上接入采集主机即可。

以上方案只适用于共享式以太网中，如果网络结构相对复杂，则需要对以上方案进行改进。例如网络通过主交换机连接路由器接入 Internet，要保证采集主机能采集到所有服务器的流量，就需要将采集主机放在网络出口处，如接在主交换机的广播口（监听口）上，若主交换机广播口不可用，可以在主交换机

和路由器间串接一个 Hub，将采集主机接在 Hub 上。这种情况下的网络结构如图 6-2 所示。

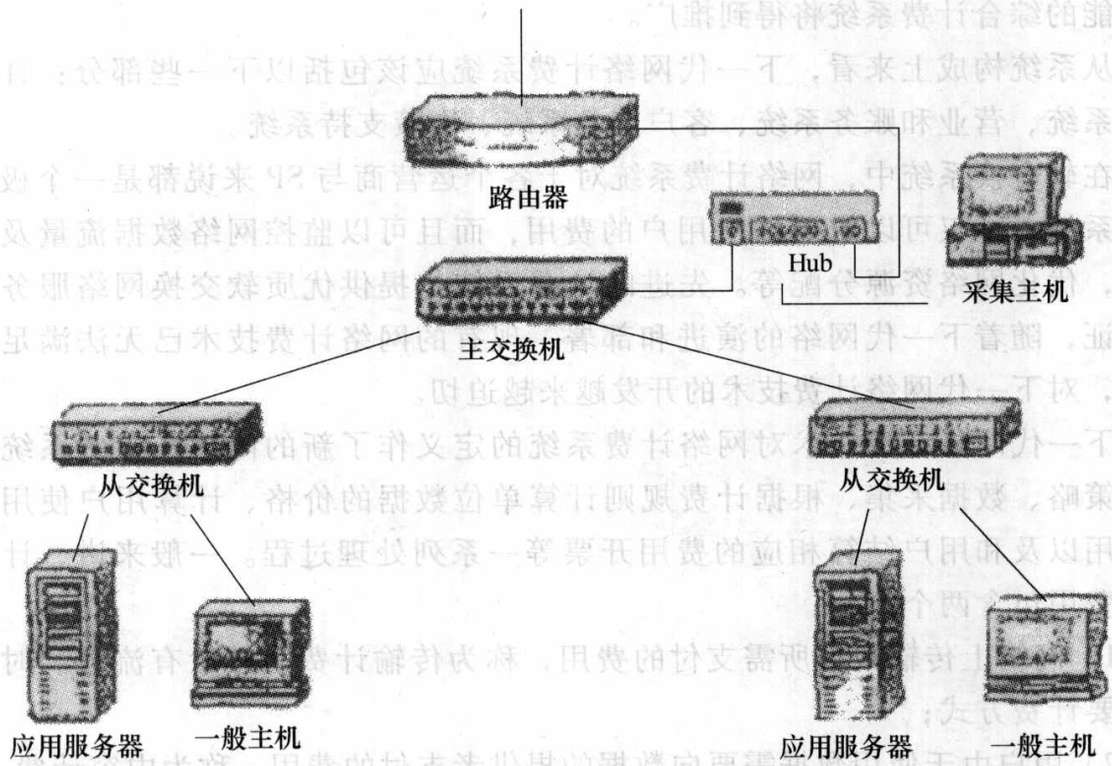


图 6-2　复杂网络结构流量采集示意

基于以太网共享总线的采集方案除需要一台额外的采集主机，并根据网络的物理结构将它接入到适当的位置外，不需要改变网络的物理结构和网络逻辑划分以及配置原有的网络拓扑结构，网络通信不受影响。不需要具有计费功能的路由器的支持，不占用网络带宽，不增加路由器的工作负荷，对网络和路由器工作无任何影响。

但这种方案中，也存在一些缺点：

1）对采集主机的性能要求比较高，而且要求其网卡能以混合模式工作；

2）该方案只适用于有统一的网络出入口的情况，对于复杂的分布式应用，例如服务提供商的几台服务器可能托管给不同的电信运营商，分别位于不同的子网内，就无法统一采集，

3）分别设置采集主机又会相应地增加硬件成本。

6.3　软交换的计费系统

软交换的计费系统最关键的特性是要有高度的系统集成化，把各式各样的分离业务系统融合起来形成统一的整体。软交换计费系统主要基于传统的电信

计费，同时，需要融合IP计费的素材。目前，计费软件正在由运营支撑系统向商业决策系统转型，今后随着BOSS体系的发展，包含客户服务、决策支持等扩展功能的综合计费系统将得到推广。

从系统构成上来看，下一代网络计费系统应该包括以下一些部分：计费及结算系统、营业和账务系统、客户服务系统、决策支持系统。

在软交换系统中，网络计费系统对于各个运营商与SP来说都是一个极其重要的系统，不仅可以统计网络用户的费用，而且可以监控网络数据流量及通话质量，优化网络资源分配等。先进的计费系统是提供优质软交换网络服务的重要保证，随着下一代网络的演进和部署，现有的网络计费技术已无法满足这一现状，对下一代网络计费技术的开发越来越迫切。

下一代网络计费技术对网络计费系统的定义作了新的阐述，计费系统包含计费策略、数据采集、根据计费规则计算单位数据的价格、计算用户使用的数据费用以及和用户结算相应的费用开票等一系列处理过程。一般来讲，计算出来的费用包含两个部分：

1）网络上传输数据所需支付的费用，称为传输计费，一般有流量和时间两种主要计费方式；

2）用户由于使用数据需要向数据的提供者支付的费用，称为内容计费。

依据用户费用支付的对象，计费系统也可以分为基于网络提供者的计费系统（provider-based accouting）和基于内容提供者的计费系统（server-based accouting）两类。一般情况而言，当前实际使用较多的是基于网络提供者的计费系统，从技术和业务的发展方向来看，基于内容的计费系统将成为软交换计费系统的一个重要方面。

6.3.1 软交换的计费系统体系结构

软交换需要提供计费原始数据的生成、采集和可能的格式转换，计费采集点逻辑结构如图6-3所示。

图6-3 软交换计费采集点逻辑结构

在下一代网络计费系统中，计费的定义包括数据的传输计费和内容计费两

方面。其中数据传输计费又可细分为通话基本费用和 QoS 费用两部分，前者建立在语音、图像传输的基础上，后者建立在通话质量（如通话时延、抖动率等）、拥塞控制等基础上。内容计费是指服务提供商给客户提供服务的相关数据的费用。为了适应新的计费定义，我们给出下一代网络计费系统的体系结构，如图 6-4 所示。

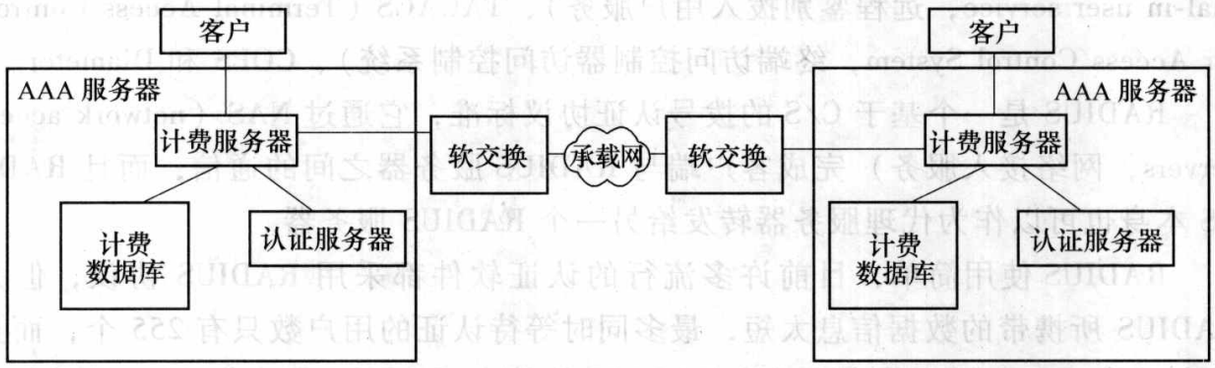


图 6-4　网络计费系统的体系结构

一个计费系统通常由计费采集点、计费中心和结算中心组成，如图 6-5 所示。

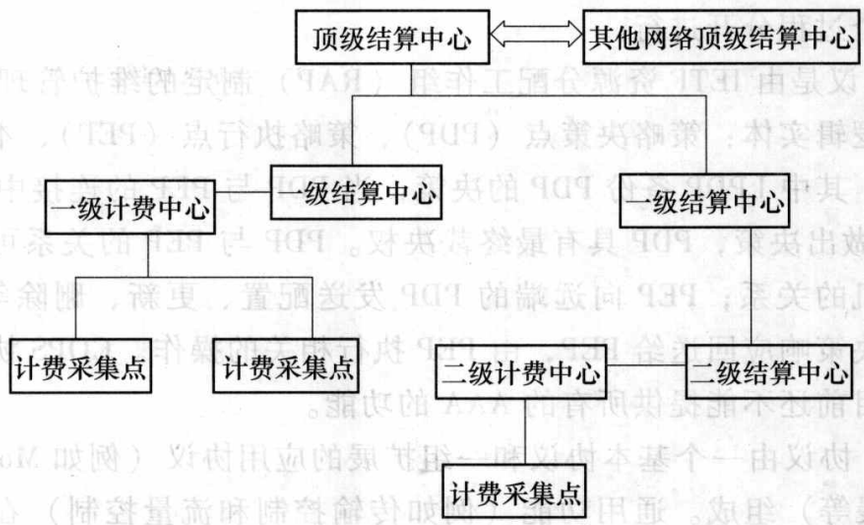


图 6-5　计费系统体系结构

1）计费采集点　提供原始话单记录。

2）计费中心　根据费率生成账单。如果是对账号计费，能够将用户余额折算成通话时长。

3）结算中心　汇总处理和费用分摊。

6.3.2　软交换的计费协议

以软交换为核心的下一代网络使用的计费协议大致可以划分 3 个类型：

1）AAA（用户身份认证、授权和计费，Authentication，Authorization and Ac-

counting)；

2）管理网络计费的协议（例如 SNMP、CMIP)；

3）传输数据的协议（例如 SMTP、FTP、HTTP 等)。

其中，AAA 协议是在计费系统的计费网关上用得最多和最重要的协议。在下一代网络计费系统中，候选的 AAA 协议主要有 RADIUS（Remote authentication dial-in user service，远程鉴别拨入用户服务)、TACACS（Terminal Access Controller Access Control System，终端访问控制器访问控制系统)、COPS 和 Diameter。

RADIUS 是一个基于 C/S 的拨号认证协议标准，它通过 NAS（network access servers，网络接入服务）完成客户端与 RADIUS 服务器之间的通信，而且 RADIUS 本身也可以作为代理服务器转发给另一个 RADIUS 服务器。

RADIUS 使用简单，目前许多流行的认证软件都采用 RADIUS 协议，但是 RADIUS 所携带的数据信息太短，最多同时等待认证的用户数只有 255 个，而且没有重传过程和错误恢复机制。TACACS 协议与 RADIUS 服务相似，不同点在于：在传输机制上，RADIUS 采用的是 UDP，而 TACACS 采用的则是 TCP；在加密处理上，RADIUS 仅对用户口令部分加密，而 TACACS 则对整个数据包加密；在认证和授权方式上，RADIUS 是将认证和授权绑在一起同时完成，而 TACACS 则是将这两个过程分开进行。

COPS 协议是由 IETF 资源分配工作组（RAP）制定的维护管理协议。COPS 定义了 3 个逻辑实体：策略决策点（PDP)、策略执行点（PEP)、本地策略决策点（LPDP)，其中 LPDP 备份 PDP 的决策，当 PDP 与 PEP 的连接中断时，LPDP 可代替 PDP 做出决策，PDP 具有最终裁决权。PDP 与 PEP 的关系可以看做是服务器和客户机的关系；PEP 向远端的 PDP 发送配置、更新、删除等请求，PDP 收到后，将决策响应回送给 PEP，由 PEP 执行相关的操作。COPS 协议只是一个建议标准，目前还不能提供所有的 AAA 的功能。

Diameter 协议由一个基本协议和一组扩展的应用协议（例如 Mobile IP 扩展、PPP 漫游扩展等）组成。通用功能（例如传输控制和流量控制）在基本协议中定义，而特定的应用功能则在相应的扩展中说明。Diameter 基本协议在提供 AAA 协议时对传输要求最少，但它不能单独使用，必须与一个 Diameter 应用扩展相结合。Diameter 目前也只是 AAA 工作组的一个建议标准，离投入实际使用还有一段距离。

6.3.3 软交换的计费策略

为了更好地管理网络资源，提高网络的 QoS，软交换计费系统的计费策略主要是围绕数据传输、业务提供商和内容的计费策略展开的。在基于非内容的计费策略中，正在研究的下一代网络计费原则主要有：

1）面向服务原则与一般性原则。面向服务的计费指计费是按使用服务的级别和质量来计算的，而一般性的计费则与服务提供的级别、质量等要求关联很少，计费更一般化。

2）静态计费原则与动态计费原则。静态计费是指是计费的价格固定，而动态计费的价格依赖于用户的需求和当时网络的负载情况。

3）简单计费原则与复杂计费原则。简单计费与复杂计费是相对于采用的计费方案的复杂程度来说的。一般说来，计费系统对 QoS 的支持度越高，计费方案的复杂程度就越高。

4）资源使用原则与资源预留原则。这两个计费原则的出发点是不同的，但无论哪一个计费原则都要求能精确指示所使用的网络资源。

5）基于使用的价值原则与基于网络资源的价值原则。基于使用的价值原则更为商业化，而基于网络资源的价值原则反映了网络资源的成本。

6）基于使用的计费原则与基于合同的计费原则。基于使用的计费原则是按实际享受的服务征收费用，而基于合同的计费是按 ISP 声明提供服务等级的合同征收费用。

6.3.4　软交换的计费方案模型

为了明确表达一个计费方案，有必要引入一个包含各个描述元素的模型。一个完整的计费方案，应当具有如下的几个部分：

1）计费函数，一个函数关系式，这个式子应当表达费率参数和计费参数的函数关系。

2）计费参数，描述所使用的服务单元的数量，例如，流量（T）、持续时间（Q）等。

3）费率参数，每服务单元的价格。一般表达形式如单位时间、单位流量的价格（元/单位流量/单位时间）。

4）计费属性，与业务相关，指明业务质量，服务类型等。计费属性的选择带有一定的任意性。时间段、网络状态相关参数、距离或距离域（如：本地、都市、国家、国际），甚至网络状态相关参数都可以作为计费方案的参量。通常情况下，计费属性是作为影响费率参数的变量出现。计费参数和计费属性的值通过测量层测量，经过收集层、记账层处理以后包含在记账层记录里。它们的采集受到服务合约控制，与用户使用的业务、QoS 等级和实现机制密切相关。费率参数是业务单元的价格，综合了技术、市场、企业策略等多方面因素。费率由专门的实体生成，是一个确定的常数。对于计费方案来说，核心任务就是使用上面提到的各种参数、属性，根据使用的业务类型选择适当的计费函数进行一次运算，也称为价格计算。

6.3.5 软交换的计费方案

计费方案是关于计算网络服务费用的算法。通过用户到 SP 接口提供的服务的计费分为基本层面的费用和服务层面的费用，如图 6-6 所示，服务层面的费用可以为内容费用，取决于所提供特定服务的内容，例如可以是 SP 通过软交换上的 Parlay API 向用户提供的某种业务，这部分费用在此不讨论，下面主要讨论基本层面的费用。

基本层面的费用，一般可以分为可分为两个部分，承载部分和会话部分：

1）承载部分，由于软交换是运营在特定承载网上，所以它在使用承载网络时应支付相关的承载费用。

2）会话部分，是跟具体的通话业务相关的，与软交换网络资源的预留和使用相关，随着业务、应用、QoS 机制等的不同而不断的变化。此部分包括了基本会话（如一次完整的 SIP 通话）的费用，和建立在基本会话之上的基本增值业务的使用费用，例如：普通呼叫的遇忙前转业务、黑白名单业务、800 业务等。

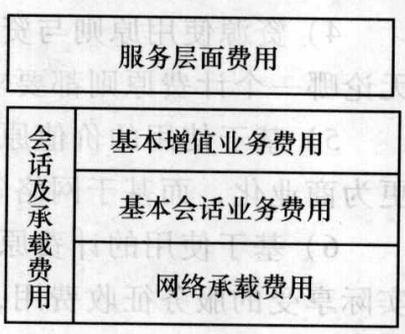


图 6-6 网络计费的组成

6.4 软交换对计费的基本要求

6.4.1 计费协议

应能支持 CMIS/FTAM 协议或 FTAM 协议、RADIUS 协议、FTP 等计费信息的传送协议，具体传送协议的使用由各运营商根据自身网络的特点决定。

6.4.2 计费方式

软交换至少应具备根据计费对象进行计费和信息采集的功能，并负责将采集到的信息送往计费中心，同时还应可选支持复式计费和立即计费等功能。

6.4.3 计费对象

对主被叫号码、会议参与和发起方、第三方以及各种业务的申请方进行计费。如果是对电话卡计费，软交换应具有计费信息传送和实时断线功能。

6.4.4 计费精度要求

采用时长计费时，一般要求其计费单位精确到 1s；当采用流量计费时，要

求其精度精确到 1B。

至于上层的服务层计费一般是根据运营商同 SP 协商进行的。

6.4.5　分组语音计费内容

仅规定分组语音计费内容的最小集：

1）日期；
2）通话开始时间；
3）通话终止时间；
4）PSTN/ISDN 侧接通开始时间；
5）PSTN/ISDN 侧释放时间；
6）通话时长；
7）卡号；
8）接入号码；
9）被叫用户号码；
10）主叫用户号码；
11）入字节数；
12）出字节数；
13）业务类别；
14）主叫侧媒体网关/终端的 IP 地址；
15）被叫侧媒体网关/终端的 IP 地址；
16）主叫侧软交换设备 IP 地址；
17）被叫侧软交换设备 IP 地址；
18）通话终止原因。

6.4.6　计费系统需要解决的问题

一个软交换计费系统通常需要关心以下问题：

1）计费方式；
2）复式计次、立即计费；
3）计费类别；
4）计费对象；
5）对主叫号计费 ；
6）对账号计费 ：在对卡号计费的情况下，需要具备实时监测和断线功能；
7）计费精度 ；
8）采用时长计费时通常精确到 s；
9）采用流量计费时通常精确到 B；

10）计费内容；

11）连接详细记录。

6.4.7 软交换计费基本内容

计费内容（CoDR）分为两种：呼叫详细记录（Call Detail Record，CDR）和包详细记录（Package Detail Record，PDR）。

CDR主要用于电路交换业务类的通话详细记录，用于记录时长相关信息。内容包括：

1）日期；

2）通话开始时间；

3）通话终止时间；

4）PSTN/ISDN侧接通开始时间；

5）PSTN/ISDN侧释放时间；

6）通话时长；

7）卡号；

8）接入号码；

9）被叫用户号码；

10）主叫用户号码；

11）业务类别等。

PDR主要用于包交换业务类的连接详细记录，用于记录流量相关信息。内容包括：

1）入字节数；

2）出字节数；

3）主叫侧媒体网关/终端的IP地址；

4）被叫侧媒体网关/终端的IP地址；

5）主叫侧软交换设备IP地址；

6）被叫侧软交换设备IP地址；

7）通话终止原因；

8）业务类别等。

6.5 软交换计费系统的特点及要求

软交换计费系统具有以下计费特点：

（1）多网融合，多种计费格式并存。

（2）软交换相连设备种类繁多，计费采集困难。

（3）随着将来基于软交换的业务的快速发展，将不断有新的计费需求。

（4）计费点可以设置在软交换、计费中心、应用服务器，计费问题相应复杂。

从性能上看，新一代网络计费系统是在多网融合中多种计费格式并存，可以支持多种业务计费要求，应具有：

1. 灵活性

1）支持多种计费策略：软交换运营商往往采用较为灵活的资费策略吸引客户，根据不同计费策略区分计费，如按时长、按流量、按内容、按 QoS 等。

2）支持多种计费对象：需要支持针对主叫计费、被叫计费和第三方计费（包括针对卡号和账号计费）。

3）支持多种计费方式：多网融合，对来自不同的 PSTN 业务、IN 业务、移动业务和多媒体业务，计费系统可以支持多种计费格式和计费方式，例如：包含同时支持复式计费、立即计费和实时计费。

4）支持多种计费协议：为提高与现有网络的兼容能力，计费系统应能够支持 RADIUS 协议、FTP 等计费信息传送协议，对未来业务考虑支持 IPDR 协议等。具体传送协议的使用应由各运营商根据自身网络的特点决定。

2. 健壮（鲁棒）性

新的计费系统还应与现有计费系统兼容，能够通过标准协议和现有计费中心互联，这样有利于现有运营商在不完全放弃现有计费系统的前提下逐渐向新的计费系统过渡。

3. 可扩展性

软交换的计费系统还应该能适应未来新业务的计费要求，便于引入新的计费方式，能够很好的适应对新业务的计费需求；能够迅速支持新的计费协议标准，才能更快地在更广泛的领域内取得应用。

4. 实时性

随着新业务的开展和运营商之间竞争的激烈，计费的实时性要求越来越高。有些业务（如计账卡类业务）必须进行实时计费；有些传统业务由于用户实时查询或运行决策方面的要求提高，也增加了实时性计费的要求。

5. 可靠性

由于计费部分是直接同用户相关联，是用户最关心的一个部分。所以软交换的计费要求数据准确，同时存在多计费系统备份和计费数据备份。

6. 与现有计费系统兼容

软交换能够通过标准协议和现有计费中心互联，符合网管要求软交换计费系统体系结构。

6.6 软交换计费系统的数据采集

在下一代网络计费系统中，为了衡量 QoS 和应用的费用，需要获取流量、时间、距离、带宽、服务优先级、服务类别以及拥塞程度、服务内容等诸多计费参数。为了采集这些参数，新的计费数据采集技术和工具应运而生，主要是 NeTraMet 和 NetFlow 这两种基于流的机制。

NeTraMet 的原理是在接入控制器中直接从数据包的头部提取上述计费参数，生成计费记录，同时按指定的服务要求和级别转发相应的数据。NeTraMet 可以完成实时流量统计、带宽分配、服务级别配置以及基于服务内容计费等多项功能，功能强大。但是，NeTraMet 的设计较为复杂，不易设定，而且所有计算均要实时完成，对设备的运算能力要求较高。

NetFlow 交换是一个高性能的网络层交换路径，可以获取丰富的统计信息。对支持 NetFlow 协议的路由器，可通过 NetFlow 协议收集与计费相关的信息，包括基于端口和 IP 地址的时长、流量等信息，最终产生有关计费的原始文件。系统通过相应的计费采集代理（agent）收集这些文件，并通过应用服务器交由计费系统处理。NetFlow 的配置非常方便、安装简单，除了需要在路由器上配置之外，只需要一台 UNIX 工作站作为流的收集工作站，所有路由器或交换机上发送的 NetFlow 都将送到此工作站集中，方便处理和分析。NetFlow 信息量特别丰富，可以为流量分布、业务分布等性能分析提供最充足的数据，非常适用于大型的网络，成本低，实施方便，而且不受速率的限制，是数据流量采集的发展方向。但是，NetFlow 需要消耗一定的路由器资源（CPU 和内存），且不能实时捕捉数据包。

6.7 软交换计费系统的计费方案

针对目前使用的计费方案存在不能区分业务、无 QoS 保证、不能控制网络拥塞等多方面的不足，根据不同的计费策略，已经提出了一系列以软交换为核心的下一代网络的计费方案，主要分为以下几类：

1）基于流的计费。

2）反转计费：这是当用户向 ISP 或服务商定购某一级别的 QoS 服务时，需要向 ISP 或服务商支付费用的计算方法。反转计费需要解决两个接入 ISP、保留客户 ISP 和服务器 ISP 与信任的第三方之间的双边协议关系。在整个计费过程中，用户只接收不发送，几乎没有用户数据在 ISP 之间交换，所以流量的检测要由 TTP 完成。

3）基于会话的计费（session-based charging）。其出发点是在网络层完成会话层的计费。这种计费方案对实时多媒体交互非常有效。它主要采集连接时间、传输的流量和文件指示信息 3 个计费参数。

4）基于策略的计费。这是目前国际上使用最广的计费模式，它建立在对包的粒度计费的基础上，通过分类、增加优先级和各种计费策略来完成计费工作。这种计费方案主要有基于拥塞控制的计费、基于区分业务的计费、基于 QoS 的计费、基于流量的计费以及上述不同计费方案的综合。

5）基于内容的计费。这是衡量内容供应商回收成本和收益的基础，其价格的制定依据主要是供应商的开发成本和客户对该产品的购买欲望。围绕内容计费，人们提出了不同的计费算法，其中最具代表性的是基于 TCP/IP 包头中 IP 地址、端口号和使用的协议来区分用户数据的类型，并根据相应的类型流量进行计费。

6.8　软交换计费的相关协议和规范

软交换常用的管理网络计费的协议如下：

1. CMIP/FTAM

1）FTAM（File Transfer Access and Management，文件传输访问和管理），由 ISO 定义（ISO 8571），是面向连接的协议，传统电话网上的计费多采用此方式传输。

2）CMIP（Common Management Information Protocol，公共管理信息协议），由 ISO 定义，是 TMN 中采用的一种管理协议，可与 FTAM 配合完成实时计费功能。

2. RADIUS

RADIUS 由 IETF 定义（RFC2138，RFC2139），采用无连接的 UDP 传输方式。是目前 IP 网上主要的一种计费协议。

3. IPDR

IPDR（IP Detail Record）由 IPDR. ORG 定义，计费格式采用 XML 描述，可以使用 FTP 或 SOAP 进行传输。IPDR. ORG 致力于为下一代数据业务提供系统间数据交换的通用框架。

我们下面对这些协议一一作详细介绍。

6.8.1　AAA 服务器的应用

AAA 即认证（Authentication）、授权（Authorization）和计费（Accounting）功能。认证是指对用户身份进行确认，允许合法用户继续使用系统提供的功能和业务，拒绝非法用户进入系统。授权是指通过身份认证的用户，需要进一步

根据用户注册的信息判断其所具有的权限和可以使用的业务种类。计费是指对系统提供的各种收费业务按照设定的费率进行结算并形成详细记录。

对用户进行认证、授权所依据的用户信息，对语音、数据业务进行计费所依据的费率信息，智能信息终端的数据信息，开展各项业务所需的业务参数，以及各种报表、统计、业务记录等信息都存储在数据库中，因此，数据库不仅是 AAA 服务器正常工作的数据依据，也是整个系统正常运行的数据基础。在整个运营平台中，数据库只与数据库代理模块和 AAA 模块进行交互。数据库代理模块能够直接访问数据库，对其进行查询、修改、删除等操作，并能执行已定义的存储过程，AAA 服务器则通过执行数据库中的存储过程完成认证、授权和计费功能。

普通用户管理的主要功能是结合路由器或交换机等网络设备，完成对用户的 AAA 接入控制。目前较为通用的 AAA 协议有 RADIUS 协议和 TACACS 协议，其中 RADIUS 协议在实际使用中应用最广。

不仅如此，在软交换中 AAA 也将发挥巨大作用。在软交换上层的应用服务器群中，AAA 服务器就是专门负责计费的服务器，它同计费中心联系起来进行计费管理。AAA 的功能主要有 3 个方面：

1. 认证

用户在使用网络系统中的资源时对用户身份的确认。在这一过程中，通过与用户的交互获得身份信息（例如：用户名—口令组合、生物特征获得等），然后提交给认证服务器；后者对身份信息与存储在数据库里的用户信息进行核对处理，最后根据处理结果确认用户身份是否正确。

2. 授权

网络系统授权用户以特定的方式使用其资源，这一过程指定了被认证的用户在接入网络后能够使用的业务和拥有的权限，如授予的 IP 地址等。认证通过的合法用户，其业务权限是用户和运营商在事前已经协商确立的。

3. 计费

网络系统收集、记录用户对网络资源的使用，以便向用户收取资源使用费用，或者用于审计等目的。以互联网接入业务供应商（ISP）为例，用户的网络接入使用情况可以按流量或者时间被准确记录下来。认证、授权和计费一起实现了网络系统对特定用户的网络资源使用情况的准确记录，这样既在一定程度上有效地保障了合法用户的权益，又能有效地保障网络系统安全可靠地运行。在 RADIUS 协议中认证和授权是集成使用的。

综上所述，系统数据库应具有以下功能：

1）控制系统数据流向；

2）保存用户信息、计费信息、终端信息、ICP 信息等；

3）确定计费方式和算法；

4）保存业务记录；

5）统计业务量。

以软交换电话卡数据业务为例，终端用户向数字接入平台发起呼叫请求后，数字接入平台首先要通过 AAA 服务器对用户身份进行认证，AAA 服务器将认证结果发回平台，如果认证通过，则可以继续其他业务，否则提示错误信息。同样，ICP 也要用户先通过身份认证才能为终端用户提供服务。服务结束后，平台向 AAA 服务器发出扣费请求，AAA 服务器通过与数据库的交互完成扣费操作。

由于所有的业务处理逻辑都在 AAA 服务器中进行，AAA 服务器通过执行存储过程从数据库中提取处理请求所需要的信息，或者直接修改数据内容，这样做一方面减轻了数据库服务器的负担，另一方面也起到了将具体业务的处理过程对外界屏蔽的作用，加强了程序的保密性。AAA 服务器对数据库的访问是通过执行存储过程来实现的，根据业务内容的不同，AAA 服务器对数据库的操作包括：对账号用户的认证和授权、对储值卡用户的认证和授权、对数据业务计费、对语音业务计费。

在 AAA 相关协议中，RFC2865（认证、授权部分），RFC2866（计费部分）中定义的 RADIUS 协议主要是用于远程访问系统中，它具体规定了在一个远程访问系统中进行认证、授权、计费的几乎所有细节。

6.8.2　RADIUS 协议

1. RADIUS 协议概述

为了提供用户信息的集中维护，通常会采用专用的验证服务器与数据库相结合的方式提供分布式的 AAA 控制。接入设备只实现客户端的验证，通过专用的协议将待验证信息传输给验证服务器进行远端验证。具体的验证过程和验证信息的维护由服务器提供，减轻了接入设备的负担并易于用户信息的统一管理。目前，RADIUS 协议是用来在验证客户端和服务端进行通信的最流行也最通用的协议。新的 RADIUS 协议标准为 RFC2865 和 RFC2866，其中 RFC2865 主要规定了认证和授权过程，RFC2866 则规定了记账过程。

通常，接入服务器连接了运营商的 RADIUS 认证服务器和 RADIUS 计费服务器。RADIUS 认证服务器带有一个含用户初始化信息的用户数据库，该数据库可以反映合法的属性值以及每个用户所享有的权限。当用户登录时，接入服务器将用户输入的用户名和口令传送到认证服务器，RADIUS 服务器将用户提交的信息和存储在数据库的与用户相关联的信息进行比较，如果匹配成功的话则验证通过，否则将拒绝用户请求。验证通过后，接入服务器将允许用户接入网络，并根据数据库中的内容提供给该用户相应的权限范围及所享有的服务。同时，

接入服务器开始收集该用户使用网络资源的情况及信息，并将该类数据交给 RADIUS 计费服务器以便对用户进行计费。接入服务器与 RADIUS 认证服务器之间通过 RADIUS 协议进行通信，从而实现了远端的验证过程和对接入用户的 AAA 控制。

RADIUS 是基于 C/S 结构的一个协议，是网络接入服务器（NAS）和后台服务器（RADIUS 服务器）之间的一个协议，NAS 作为客户端，RADIUS 服务器作为服务器端。它使得拨号和认证这两种功能放在两个分离的网络设备上：RADIUS 服务器上存放有用户名和相应鉴别信息的一个大数据库；NAS 则作为用户接入的手段。同时，RADIUS 服务器还可以为其他类型的认证服务器提供代理客户机的功能，向其提出认证请求。当用作计费时，RADIUS 提供一种计费格式定义和传输手段。计费信息由客户端（如采集点，充当网络接入服务器）传送给计费服务器（充当 RADIUS 服务器）。同时提供客户端和计费服务器之间的安全认证手段。

客户机和 RADIUS 服务器之间的认证处理是通过共享密钥来实现的，不通过网络来传送。另外，所有的用户密码均被加密传送，以防止有人在不安全的网络上窃取用户的密码。RADIUS 服务器能支持多种认证方法。当用户提交用户名和密码时，RADIUS 服务器能支持 PPP、PAP 或者 CHAP、UNIX Login 和其他认证方法。

RADIUS 在 ISP 网络中非常常见，当用户用 Modem 拨号通过某个 ISP 接入 Internet 时，ISP 需要对用户进行认证。

RADIUS 也是目前 IP 网上计费最常用的协议之一。图 6-7 是一个 RADIUS 用于计费的例子。

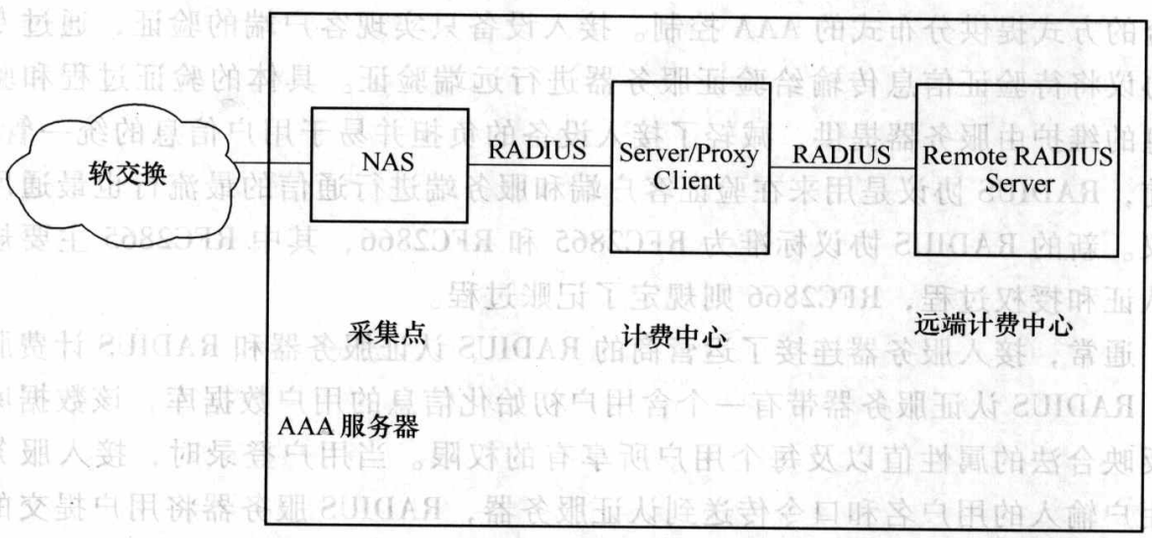


图 6-7 软交换中 RADIUS 计费

从图 6-7 可以看到，RADIUS 服务器同时可以作为其他 RADIUS 服务器的客

户代理。在 IP 电话网中，这是很常用的一种配置方式。

2. RADIUS 协议特点

（1）客户机/服务器模式

网络接入服务器通常作为 RADIUS 协议的客户端。客户端的任务是把用户信息（账号、口令等）传给指定的 RADIUS 服务器，并接收服务器的响应。RADIUS 服务器的任务主要是：

1）对用户进行连接控制；

2）对用户身份进行认证；

3）返回用户的所有配置信息；

4）另外，RADIUS 服务器还可以作为其他 RADIUS 服务器或异种认证服务器的代理客户。

（2）网络安全方面

网络安全通过一种共同的安全机制，RADIUS 服务器和其客户端的交互是处于受保护的状态，同时这种安全机制的运作方式不会被传送到网络之中。除此之外，在 RADIUS 服务器和客户端之间，所有用户的密码是被加密传输，从而防止恶意者对网络进行监听，窃取用户的密码，同时灵活的验证机制使 RADIUS 服务器能够支持许多不同的用户验证方式。当用户将原始密码（未经加密的密码）、用户名提供给 RADIUS 后，RADIUS 服务器可提供 PPP、PAP 或 CHAP 以及 UNIX 注册等验证方式，当然还可提供其他的验证机制。

（3）灵活的认证机制

RADIUS 服务器支持多种认证方法。当用户提供用户名和原始口令时，它可以支持 PPP、PHP、CHAP、UNIX Login 和其他认证机制。

（4）协议的可扩展性

所有交互都包含可变长度的属性字段。为满足实际需要，用户可以加入新的属性值而不会对原协议有任何影响。

RADIUS 的包格式中包含标识 RADIUS 包类型的代码字段、标识符字段、长度字段、验证字段（分为请求验证和响应验证）和不受限的属性字段。属性字段的使用使得 RADIUS 利用新的属性值提供新的功能而具有很好的可扩展性。RADIUS 协议的所有处理都包含可变长度的属性字段。新的属性值能够方便的被添加进去而不需干扰现有已存的协议实现，提供了良好的可扩展性。

3. RADIUS 认证过程

整个验证过程有两个阶段：

（1）客户请求阶段

如果一个客户端被配置为使用 RADIUS 协议，那么该客户端的每一个用户都会被要求提供它的验证信息给客户端，这个过程一般都含有一个注册的提示，

在提示注册的地方，用户会被要求输入用户名与密码。当然，用户也可能用其他的链路层协议比如 PPP，这些协议也含有包含此信息的验证包。一旦客户端获得了这样的信息，它可以选择利用 RADIUS 进行验证，在这样的方式下，客户端会产生一个“接入请求（Access-Request）”包到 RADIUS 服务器，这个请求包含属性、用户名、密码、用户接入的客户端 ID 和端口 ID。当密码被提交的时候，它会用 MD5 的算法将其加密隐藏起来。

通过请求将会通过网络传输提交到 RADIUS 服务器。如果在一段时间内没有任何响应，这个请求将会被重新发送。如果出现主服务器故障或者不可到达，客户端也可以将请求发送给其他服务器。

（2）RADIUS 服务器验证阶段

一旦 RADIUS 服务器接受了请求，它将验证发送到该请求的客户端。如果该客户端和 RADIUS 服务器没有使用同样的安全机制，这样的请求将被“直接丢弃”。如果客户端是有效的，RADIUS 服务器将查询用户数据库寻找数据库中是否有相同用户名的实体。在数据库中的这个用户实体包含对用户的所有要求，只有当用户满足所有这些要求，他才被允许通过。只要有任何一条要求不符合，RADIUS 服务器将向客户端发出“接入拒绝（Access-Reject）”包，表示请求无效。如果有必要，服务器还要在“接入拒绝”包中包括一条消息，用来显示给用户。“接入拒绝”包中不包含其他属性。如果所有条件都满足，RADIUS 服务器将发出一个“接入盘问（Access-Challenge）”包。如果客户端收到“接入盘问”包并支持 Challenge/Response，则可以将消息显示给用户，提示作出响应。然后客户端用一个新的请求 ID 重发原来的“接入请求”包。其中的用户口令属性由用户响应（经过加密）所取代。如果在“接入盘问”包中包含状态属性，则重发的“接入请求”包中还应包括状态属性。服务器可以根据情况响应“接入接受（Access-Accept）”“接入拒绝（Access-Reject）”或另一个“接入盘问”。

如果所有的要求都通过了，那么用户的一系列配置信息将放置在“接受通过”的响应中。这些响应包含服务种类（比如：SLIP，PPP，注册用户）和所有将要传给相应服务器的必需信息。在 SLIP 和 PPP 中，必需信息可能包含：IP 地址、子网掩码、MTU、压缩方式和过滤数据报的标志。在字符形式的用户中，这也许还包括所想得到的协议和主机。

第一字节	第二字节	第三字节	第四字节
Code	Identifier	Length	
Authenticator			
Authenticator			
Authenticator			
Authenticator			
Attnbutes...			

图 6-8 RADIUS 报文结构

客户端和服务器之间通过请求应答的方式进行协议交互，通过验证的用户会将根据服务器响应包含的用户信息给予不同的服务和控制。

4. RADIUS 报文格式

报文的总体结构如图 6-8 所示。

其中：

（1）类型

类型（Code）字段，长度 1B，用来说明 RADIUS 包的类型。如果收到的包 Code 字段无效，将被丢弃。RADIUS 类型字段分类见表 6-2。

表 6-2　RADIUS Code 字段

值	类型	中文名
1	Access-Request	接入请求
2	Access-Accept	接入接受
3	Access-Reject	接入拒绝
4	Accounting-Request	计费请求
5	Accounting-Response	计费响应
11	Access-Challenge	接入盘问
12	Status-Server	服务器状态
13	Status-Client	客户端状态
255	Reserved	保留

其中，Access-Request、Access-Accept、Access-Reject、Access-Challenge 用于认证，Accounting-Request、Accounting-Response 用于计费。

（2）标识符

标识符（Identifier）字段，长度 1B，用来帮助进行匹配请求和应答。如果 RADIUS 服务器收到的包与一段时间以前收到的包有相同的客户端 IP 地址、端口以及标识符，则认为收到了重复包，将被丢弃。

（3）长度

长度（Length）字段，长度 2B，它说明了包的长度。范围从 20 ~ 4096。包括类型字段、标识符字段、长度字段、认证符字段和属性字段。超过长度的字节将作为填充字符被忽略。如果包的长度比长度字段说明的长度小，则包被丢弃。

（4）认证码

认证码（Authenticator）字段，长度 16B。这个字段极其重要。它用来认证从 RADIUS 服务器来的应答，并用于口令隐藏算法。认证码有两种：

1）请求认证码（Request Authenticator）；

2）应答认证码（Response Authenticator）。

在请求认证码包中的认证码，被称作请求认证码。它是一个 16B 的随机数。对认证码的要求是：在同一个公开密钥下认证码必须惟一，不可预测。因为如

果公开密钥不变的情况下允许重复的请求值，攻击者就可以重发截取的认证码进行攻击。“不可预测”的原因是防止攻击者骗取服务器对预测出的请求作出的响应。然后冒充服务器对以后的接入请求作出响应。

尽管 RADIUS 协议对实时活动的窃听认证保护不了，但惟一的、不可预测的认证码可以防止受到大范围的活动攻击。

Type(类型)	Length(长度)	Value(值)

图 6-9 属性格式

(5) Attributes 域

可以为 RADIUS 请求或应答传递认证、授权或计费等信息，由 Type-Length-Value 三元组组成，属性域格式如图 6-9 所示。

RADIUS 协议属性携带认证授权信息，在请求应答中还包括配置细节。属性的结尾处由 RADIUS 包中的长度字段说明。有些属性由于特殊需要在一个包中可能出现多次。

1) 类型 (Type) 字段长度为 1B。值 192 ~ 223 保留给试验用，值 224 ~ 240 保留给特殊用途，值 241 ~ 255 保留不允许使用。常用的类型见表 6-3。

表 6-3 常见的属性

Type 值	代表类型	中文名称
1	User-Name	用户名
2	User-Password	用户密码
3	CHAP-Password	CHAP 密码
4	NAS-IP-Address	NAS 的 IP 地址
5	NAS-Port	NAS 的端口
6	Service-Type	服务类型
7	Framed-Protocol	框架协议

2) 长度 (Length) 字段长度 1B，指明了该属性的长度。包括类型字段、长度字段和属性值字段。如果收到的接入请求包中有属性长度无效，则接入拒绝包将被重发。如果收到的接入接收、接入拒绝、接入盘问包中有属性长度字段无效，则包被丢弃和视作接入拒绝。

3) 属性值 (Value) 字段长度可以是零字节和多字节。它包括该属性的特定信息。属性值的格式和长度由类型字段和长度字段决定。要注意的是 RADIUS 协议中没有数据类型以 Null 结尾。属性值的格式有以下 5 种：

① text 最短 1B，最长 253B；

② string 最短 1B，最长 253B，可以是二进制数据 (0 ~ 255)；

③ address 长度 32 位，用来表示 IP 地址；

④ integer 32 位无符号数；

⑤ time 32 位无符号数，记录从 UTC 1970 年 1 月 1 日零时开始的秒数。

为了提供对用户信息的集中维护，通常会采用专用的验证服务器与数据库相结合的方式提供分布式的 AAA 控制。接入设备只实现验证的客户端，通过专用的协议将待验证信息传输给验证服务器进行远端验证。具体的验证过程和验证信息的维护由服务器提供，这样就减轻了接入设备的负担并易于用户信息的统一管理。

通常，RADIUS 认证服务器带有一个含用户初始化信息的用户数据库，该数据库可以反映合法的属性值以及每个用户所享有的权限。当用户登录时，接入服务器将用户输入的用户名和口令传送到认证服务器，RADIUS 服务器将用户提交的信息和存储在数据库的和用户相关联的信息进行比较，如果匹配成功的话则验证通过，否则将拒绝用户请求。验证通过后，接入服务器将允许用户接入网络，并根据数据库中的内容提供给该用户相应的权限范围及所享有的服务。同时，接入服务器开始收集该用户使用网络资源的情况及信息，并将该类数据交给 RADIUS 计费服务器以便对用户进行计费。接入服务器与 RADIUS 认证服务器之间通过 RADIUS 协议进行通信，从而实现了远端通过的合法用户的业务权限是用户和运营商在事前已经协商确立的。

RADIUS 协议有以下关键特性，也是它的优点所在：

（1）C/S 模式

C/S 模式的系统具有很好的可扩展性。可以配置多个服务器以提高可靠性；某个 RADIUS 服务器可以同时作为其他服务器的代理客户端。

（2）具备一定的网络安全

客户机和 RADIUS 服务器之间的认证处理是通过共享密钥来实现的，且不通过网络来传送。另外，所有的用户密码均被加密传送，以防止有人在不安全的网络上窃取用户密码。

（3）易于协议扩展

所有消息都是 Attribute-Length-Value 三元组形式，新的属性很容易加入进来。

6.8.3　Diameter 协议

1. RADIUS 的缺陷

RADIUS 协议是在 20 世纪 90 年代初期设计的，目的是对 PPP 拨号用户提供 AAA 服务。从历史上来说，RADIUS 协议被大多数运营商所采用。然而，随着 Internet 的飞速发展，路由器和 NAS 在数目和复杂性都有了很大的提高，各种新的网络服务也不断出现，使得 RADIUS 越来越不适应现在的网络。具体体现在以下几点：

1）可携带的属性数据的长度太短。在 RADIUS 中，属性的长度段只有一个

字节，这就大大限制了一个 RADIUS 信息包的长度。随着网络的复杂度和服务的多样化，用户的认证信息愈来愈多，这就要求 AAA 协议信息包可以携带足够的信息。

2）同时等待认证的用户数最多 255 个。一个用户提出请求后，在得到响应之前有一段等待时间，在这段时间内用户处于等待状态，RADIUS 通过给用户分配一个 ID 号来识别同一时刻不同的等待用户，但是用来标示 ID 号的字段只有 8 位，这就限制了同时等待回复的用户数目只有 255 个，这在目前的大型网络中是不能满足要求的。

3）无法控制到服务器的流量。RADIUS 是使用 UDP 传输的，而 UDP 是没有流控机制的，RADIUS 对此也没有作扩展，随着用户数目的飞速增长，服务器的负担会越来越重，没有流控机制，会造成大量的认证计费请求涌向服务器，造成服务器的瘫痪，而影响网络的稳定性和可靠性。

4）无重传过程和错误响应支持。RADIUS 的客户端一段时间内没有收到响应时，它就重发原来的请求。但服务器端没有重发机制，如果服务器的响应丢失，即使客户端的请求正确到达了，也必须重新发送一个请求。另外，现在很多 NAS 配置了多个 RADIUS 服务器，当一个 RADIUS 服务器不能正常工作时可以由另外的备份服务器（即使说 RADIUS 服务器可以是多机系统）来处理 NAS 的请求，以此来提高系统的可靠性，但是这在实践中遇到很多困难，因为 RADIUS 在设计时是不支持客户端发送探测服务器是否正常工作的消息的，所以 NAS 无法知道服务器当前的工作状态，它只有不断地向一个服务器发送同一请求，当重传数目达到一个预先设定的上限数目后，NAS 认为服务器已经瘫痪，然后把 AAA 请求发送给备用的服务器。这样，问题可能是一个请求的处理时间会非常长，远远超过了用户可以等待的时间，而在网络繁忙的情况下，这种情况很容易发生，因此 RADIUS 缺少对错误的处理机制，降低了整个系统的可靠性。RADIUS 采用的是 C/S 模式，其缺点是服务器只能被动地回答客户端的请求，而不能主动发一个消息给客户端，这大大限制了客户端和服务器端的通信能力。

5）具备一定的安全性，但总体上将还较为欠缺。RADIUS 只对用户的口令部分进行加密，而且无法防止重放攻击。另外，RADIUS 支持代理功能，即允许一个 RADIUS 服务器把一个请求转发给另一个 RADIUS 服务器，但是在这种情况下，RADIUS 只支持点到点的安全性，即中间的每一个代理 RADIUS 服务器都有能力对用户的认证计费信息进行修改，这是个很大的安全隐患。

随着网络技术的发展和应用需求的增长，新的网络服务不断涌现，比如移动（Mobile）IP，漫游服务等等，这些服务需要一种安全的方式来允许用户接入网络、授权用户可以访问的资源并对用户进行计费。以前这种 AAA 服务是由 RADIUS 或 TACACS + 提供的，但是这些协议是为拨号用户设计的，无法有效地

对新的业务提供 AAA 服务。针对 RADIUS 的这些不足，IETF 的 AAA 工作组从 20 世纪 90 年代末期开始着手设计下一代 AAA 协议 Diameter。Diameter 定义了一种 AAA 的体系结构，它由一个基本协议和一组扩展协议组成（比如增强安全性安全扩展，Mobile IP 扩展和 PPP 漫游扩展等等），通用的功能（比如传输控制和流量控制）在基本协议中定义，而特定的应用功能在相应的扩展中进行说明。Diameter 基本协议提供了 AAA 协议对传输所需要的最低要求，基本协议不能够单独使用，必须与一个 Diameter 应用扩展相结合。

基本 Diameter 协议并不单独使用，它和各种应用扩展一起使用，为各种应用扩展提供服务。当前已经定义了 4 种扩展：NASREQ，Mobile IP、计费、增强安全。基本 Diameter 协议定义了报文格式、一系列原语和报文如何安全地传送。Diameter 协议的体系结构如图 6-10 所示，描述了逻辑上的 Diameter 的框架结构，Mobile IP 扩展为 Mobile IP 用户提供 AAA 服务，漫游扩展为漫游的 PPP 用户提供认证和计费支持，Mobile IP，PPP 漫游等应用的扩展协议建立在基本协议之上。

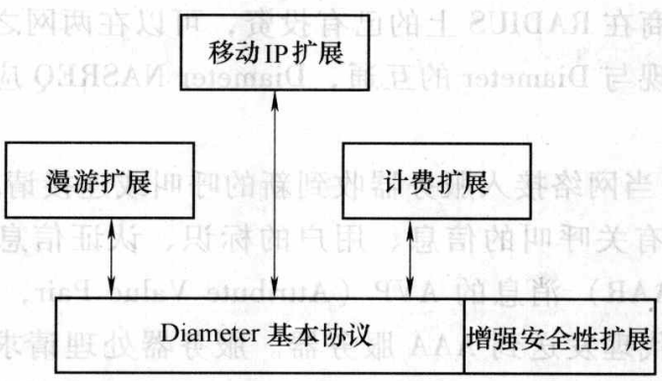


图 6-10　Diameter 协议体系结构

2. DIAMETER 的基本协议

基本 Diameter 协议使用的是对等实体通信的模型，而不是 RADIUS 使用的 C/S 模型。使用对等实体通信模型可以向 NAS 主动发送报文，带来的好处是可以根据请求主动获取计费信息，另一个好处是 AAA 服务器可主动要求结束会话。

Diameter 协议提供容量协商能力，RADIUS 没有支持容量协商和错误消息机制。当 RADIUS 客户端与服务器端不了解对方容量时，它们可能不会成功地建立一个互动的服务，甚至在某些情况下，不能确认这类服务是否已经建立。

Diameter 提供了强大的漫游支持，RADIUS 没有提供对代理的详细描述，缺乏对漫游的传输层安全认证，因此基于 RADIUS 的漫游易受外部假冒攻击。由于提供了对内部域的漫游和消息路由，传输层的安全认证，Diameter 协议可以保证安全的和可升级的漫游。

和 RADIUS 的部署一样，除了网络接入服务器对用户提供接入外，AAA 服务器还需要对用户进行认证、授权业务并收集计费信息。对于漫游用户，还需要 Diameter 代理把消息转发到用户的本地 AAA 服务器，如图 6-11 所示。

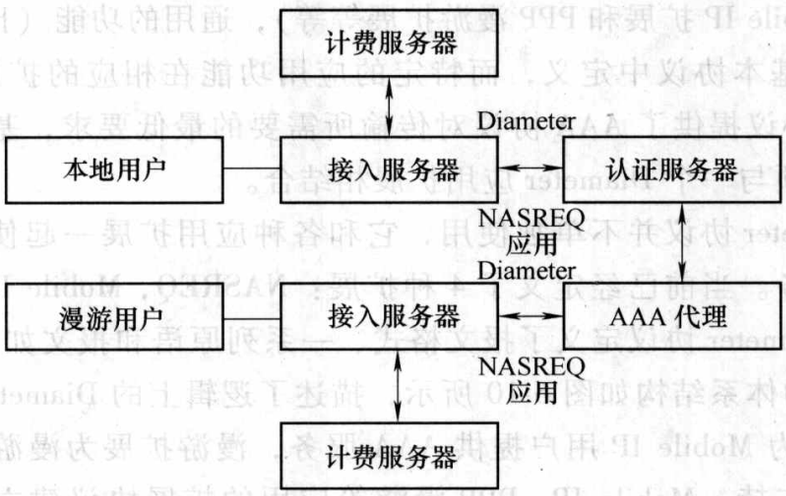


图 6-11　Diameter 服务器结构

为了保护运营商在 RADIUS 上的已有投资，可以在两网之间配置一个协议转换器，这样就能实现与 Diameter 的互通，Diameter NASREQ 应用中也定义了协议交互的一些规则。

1）协议过程中当网络接入服务器收到新的呼叫或连接请求时，就开始 NASREQ 消息的交换。有关呼叫的信息、用户的标识、认证信息等被封装到 AA 请求（AA-Request，AAR）消息的 AVP（Attribute Value Pair，属性值对）内，可以通过多个中继、代理发送到 AAA 服务器。服务器处理请求后，会发送 AA 应答（AA-Answer，AAA）消息。应答消息中包含的处理结果代码（Result-Code）AVP 除了指示成功或失败外，还能要求多次认证，这时就需要客户端和服务器之间进行多次的 AAR 和 AAA 交互，直到结果指示成功或失败。

2）与 RADIUS 不同，Diameter 支持认证和授权分开进行，也就是说可以先进行认证，然后再发送授权请求。

3）安全协议 Diameter NASREQ 应用中讨论了如何在 Diameter 协议中承载 PPP 认证协议。由于 PAP 存在安全漏洞，所以不鼓励使用 PAP，除非用在一次性密码（One-Time Password）认证的场合。在 Diameter NASREQ 应用中使用 CHAP，主要也是为了和 RADIUS 的兼容。而 EAP（Extensible Authentication Protocol，可扩展鉴别协议）较之 PAP 和 CHAP 有很多优越性：

① EAP 可以支持多种认证方案，如智能卡认证、公共密钥、一次性密码等；

② 由于认证是在 EAP 客户端和本地 AAA 服务器之间进行的，EAP 可以实现端到端的认证；

③　端到端的认证也提供了双向认证，而 PAP 和 CHAP 在漫游环境下是无法实现这一点的。

Diameter NASREQ 应用和 Diameter EAP 应用一起使用，就能很好地满足来自网络接入服务器、漫游操作等对网络接入的安全要求。

Diameter 的基本协议必须被所有的应用支持，在基本协议中定义了 Diameter 消息包的格式，消息包的传输机制和一些基本的安全服务。和 RADIUS 不同，Diameter 是传输在 SCTP（Stream Control Transmission Protocol）上或是 TCP 上的，SCTP 是一种应用层数据报传输协议，可在诸如 UDP 或 IP 之类不可靠的数据报服务器之上提供可靠的服务。SCTP 提供可靠的传输机制，大大简化了服务器端的实现。另外，SCTP 还定义了窗口机制，允许 Diameter 客户端限制到服务器的请求个数，有了窗口机制，Diameter 的客户端就可以知道当前各个 Diameter 服务器处理请求的能力，可以合理地分配发送到各个 Diameter 服务器的请求数量，有效地控制了到服务器端的流量。

Diameter 基本协议的特点有：

1）轻量级而且易于实现：Diameter 基本协议的目标是为各种应用扩展协议提高安全、可靠、快速的传输平台，因此必须是轻量级的和易于实现的。

2）大的属性数据空间：在 Diameter 中，数据对象是封装在 AVP 中的，AVP 用来传送用户的认证和授权信息、交互用以计费的资源使用信息、中继代理和重定向 Diameter 消息包等等。随着网络的复杂化，Diameter 消息包所要携带的信息越来越多，因此属性空间一定要足够大，才可以满足未来大型复杂网络的需要。

3）支持同步的大量用户请求：随着网络用户的增加，AAA 服务器应该同时处理大量用户的请求，这要求 NAS 端能保存大量等待认证结果的用户的接入信息，相对于 RADIUS 的 255 个接入请求，Diameter 可以同时支持 232 个用户的接入请求。

4）可靠的、基于 SCTP 的传输机制和错误恢复机制：Diameter 能够控制重传策略，这一点在 NAS 切换到一个备用 Diameter 服务器时更加重要。Diameter 还支持窗口机制，这要求每个会话方动态调整自己的接受窗口，以免发送超出对方处理能力的请求。虽然 Diameter 和 RADIUS 有很大的不同，但它从 RADIUS 中借鉴了很多，在保留 RADIUS 的基本模式的情况下，改正了 RADIUS 中固有的不足。Diameter 不是 RADIUS 的改进版本，而是下一代的 AAA 协议。

3. Diameter 的包格式和属性值

Diameter 的消息包格式和 RADIUS 有很大不同。Diameter 协议的数据包格式如图 6-12 所示。

1）版本号（Version）字段必须置为 1，表示 Diameter1.0。

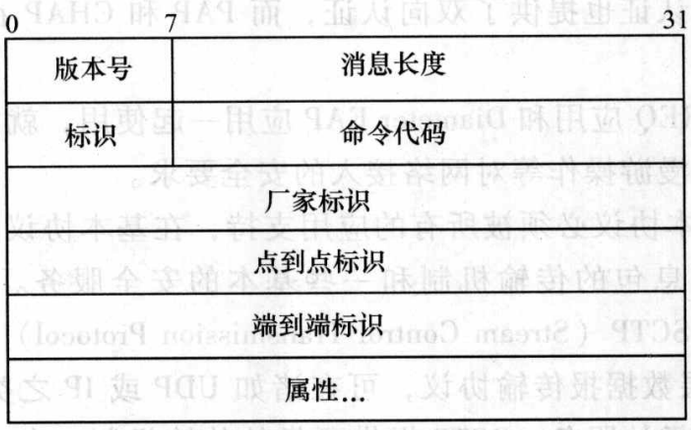


图 6-12 Diameter 协议头格式

2）长度（Message Length）字段为 3B，表示包括 Diameter 头的 Diameter 消息长度。

3）命令标识（Command Flag）字段为 1B。命令标识前 4 位为 R（Request）P（Proxiable）E（Error）T（Potentially re-Transmited message），后 4 位保留，其中：

① R 位置 1 表示为请求消息，0 表示应答消息；

② P 位置 1 表示为代理和重定向消息，0 表示本地消息；

③ E 位置 1 表示消息包含错误信息；

④ T 位置 1 表示一个链接失败。

4）命令码（Command-Code）字段为 3B，用来表示与消息相关的命令。Diameter 基本协议定义的命令码为：

① 274 Abort-Session-Request 、Abort-Session-Answer；

② 271 Accounting-Request、Accounting-Answer；

③ 257 Capabilities-Exchange-Request 、Capabilities-Exchange-Answer ；

④ 280 Device-Watchdog-Request 、Device-Watchdog-Answer ；

⑤ 282 Disconnect-Peer-Request、Disconnect-Peer-Answer；

⑥ 258 Re-Auth-Request、Re-Auth-Answer；

⑦ 275 Session-Termination-Request，Session-Termination-Answer。

5）应用标识符（Application-ID）字段为 4 个 B，用来表示与消息相关的应用，例如认证应用（Authentication Application）和计费应用（Accounting Application）。

6）逐跳标识符（Hop-by-Hop Identifier）字段为 4 个 B。逐跳标识符为无符号单调递增整数，用来防止重发攻击。

7）端到端标识符（End-to-End）标识符字段为 4B，用来监测本地链路中的复制消息。

Diameter 属性值（AVP）包含认证、计费、授权、路由与安全等信息。每个属性值字段必须被填充至 32bit，填充长度不计入属性值长度。Diameter 属性值格式如图 6-13 所示。

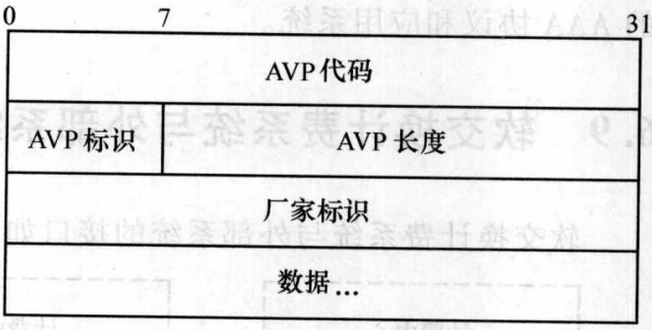


图 6-13　Diameter 属性值格式

AVP 码（AVP Code）字段与供应商 ID（Vendor-ID）字段惟一标识属性值。

1）AVP 码 1 ~ 255 为 RADIUS 保留，其他用于 Diameter 协议。

2）AVP Flag 字段为 1B。其中：

①　第一个比特为 V，置 1 表示 AVP 中包含供应商 ID 字段；

②　M 位为强制比特，用来表示对该 AVP 的支持是否必需；

③　P 位表示端到端加密。

3）AVP 长度（AVP Length）字段为 3B。AVP 长度包括 AVP 码、AVP 长度以及 AVP 标识、供应商 ID 和 AVP 数据。如果收到消息长度错误，该消息应被丢度弃。

4）AVP 数据类型有 8 种，它们为 OctetString、Integer32、Integer64、Unsigned32、Unsigned64、Float32、Float64 和 Grouped，比 RADIUS 支持的数据类型要全面、丰富的多。

6.8.4　AAA 技术的发展

随着网络技术的不断发展和服务逐渐增多，今后的 AAA 协议结构必将更加复杂，功能更加强大。总的来说未来 AAA 协议应该具有以下特点：

1）具有可扩展性，可以方便地支持新的网络技术和网络服务；

2）能够稳定地处理大量用户的请求；

3）安全，安全性对任何一个系统来说都是十分重要的，必须采用简单有效的安全机制保证用户的敏感信息；

4）支持复杂的认证和授权过程；

5）支持多种计费模式，能按各种费率对不同级别的用户进行计费。

虽然 AAA 技术是一个新兴的技术领域，目前的应用还仅限于对接入网络的用户进行管理，AAA 技术本身也有很多不足之处，但是 AAA 技术在很多领域都有重要的应用，它是很多电信业务和 IP 业务的基础，特别是下一代网络计费系统，而目前对 AAA 技术研究还远不能满足应用的需求，现在越来越多的研究机构和厂商开始重视这个问题，并积极进行商业化产品的开发工作。今后随着 AAA 应用的进一步扩展，会有更多的人进行这方面的研究，并会出现更加完善

的 AAA 协议和应用系统。

6.9 软交换计费系统与外部系统的接口

软交换计费系统与外部系统的接口如图 6-14 所示。

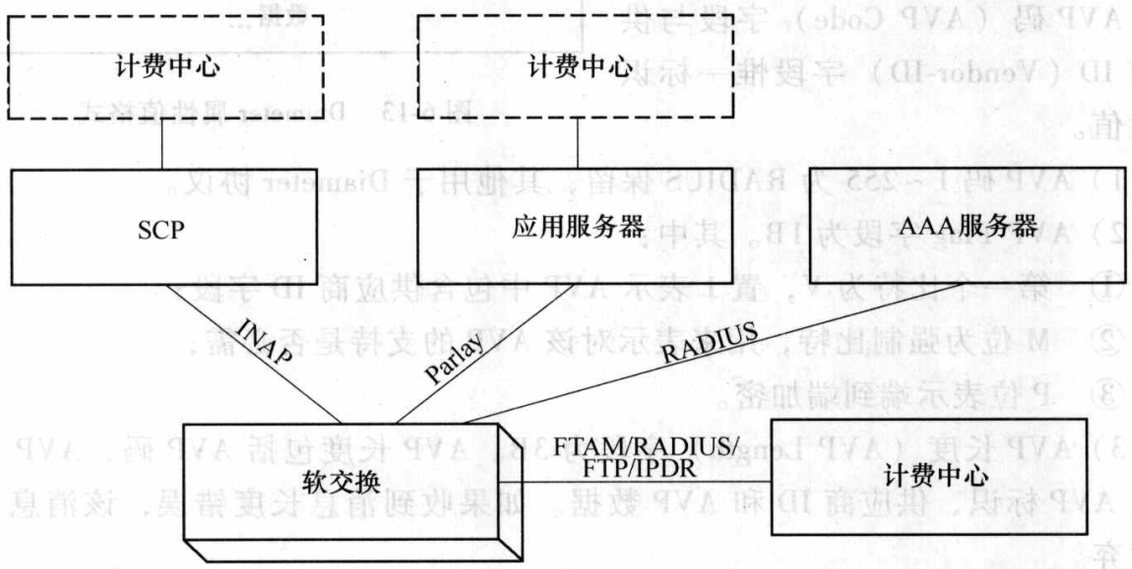


图 6-14 软交换计费系统与外部系统接口

1. 传输协议考虑

INAP 和 Parlay API 都是既定的，没有多少可以讨论的余地，我们主要考察 FTAM/FTP/RADIUS。

2. 从保护现有投资角度考虑

目前大部分计费系统支持 FTAM/RADISU 协议，从运营商保护现有投资角度出发，可以选用这两种协议。

3. 从开发角度（经济和开发难度）上考虑

FTAM 和 RADIUS 需要购买或自己开发协议栈，如果自己开发难度较大，可以考虑购买，但使用起来仍然复杂。

FTP 则相对经济，因为对于 FTP 服务器通常 UNIX 系统都提供，FTP 客户可以使用操作系统功能或者自己开发，一般难度不会太大。

4. 互联性角度考虑

RADIUS 协议定义了计费内容和格式，因此和基于 RADIUS 的计费系统可以较好的互联，有些属性仍然需要协商。

FTAM 和 FTP 则仅仅是传输协议，未对计费内容和格式做出定义。所以，软交换和计费系统之间需要较多协商，互通性不是太好。

5. 对软交换系统影响角度考虑。

1）RADIUS 协议基本上是一种实时的协议，每次呼叫结束都需要向 RADIUS

服务器上报计费信息，系统负荷较重。

2）采用 FTAM 和 FTP，CDR 在本地生成，可以定期在系统负荷轻（如半夜）的时候向计费系统传输数据。

6. 从前瞻性考虑

软交换应立足于未来网络和最新技术，在计费上也应该考虑采用新的协议和标准，如 IPDR。

7. 从支持实时监测和自动断线考虑

一般来讲，实时监测和自动断线计费业务在记账卡业务中使用较多，软交换需要具备实时监测和断线功能。可以采用 3 种策略：

1）参照 CIN-02 智能网 SCP，软交换本身具备计算功能。这就意味着软交换必须保存费率表，同时需要保存用户数据库，以便得到用户余额信息，计算最大通话时长。

2）请求应用服务器或 SCP 获得最大通话时长信息。

3）请求计费中心获得用户最大通话时长信息。

使用第 3 种方式时，需要特别指出的是，RADIUS 和 FTAM、FTP 并不是完全在同一层面的协议，FTAM 和 FTP 仅仅用作批量传输数据，无法让软交换实时得到通话时长，RADIUS 是协议则可以做到这一点。当然，引入 CMIS 协议也可以做到这一点。

6.10　实际系统中关于计费的例子

举几个实际系统中计费的例子，以便对实际系统中的计费问题有个感性认识，也是我们设计软交换计费的参考。

1. 电话用户发起呼叫

流程如图 6-15 所示。

1）网关收到 PSTN 侧发来的 REL 消息，或 IP 侧发来的 Release Complete 消息时，停止计费计数器，收集计费信息，利用“脱离请求”（DRQ）消息发送到网守。

2）软交换向网关发送“脱离确认”（DCF）消息，同时网守利用“计费请求”（Account-Request）消息将计费信息发送 RADIUS 服务器。

3）如果是漫游用户，则利用“计费请求”将计费信息发往远端 RADIUS 服务器，收到“计费响应”（Account-Response）消息后向网守回送“计费响应”。如果不是漫游用户，由 RADIUS 服务器直接进行计费信息处理，向网守发送“计费响应”。

2. PC 用户发起呼叫

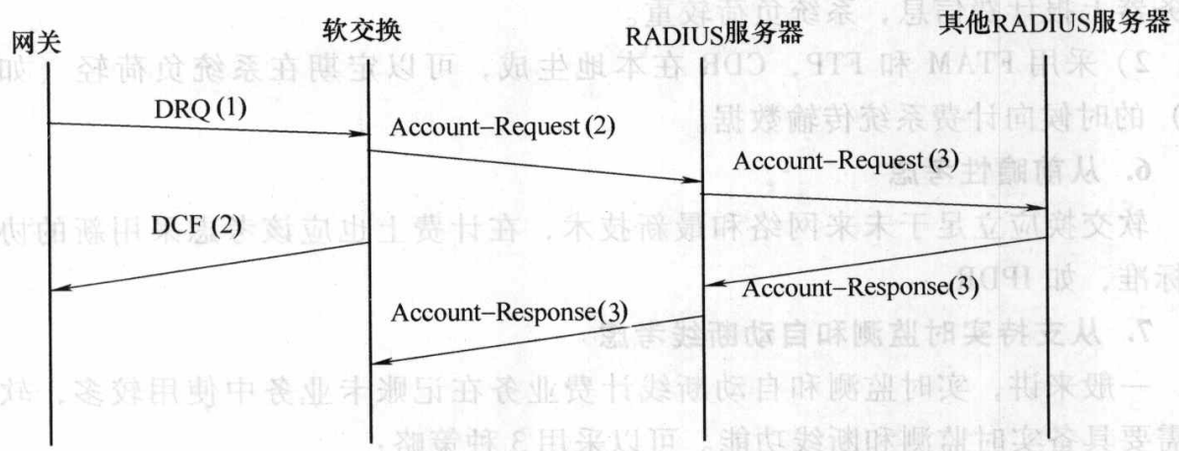


图 6-15 电话用户发起呼叫计费实例

流程如图 6-16 所示。

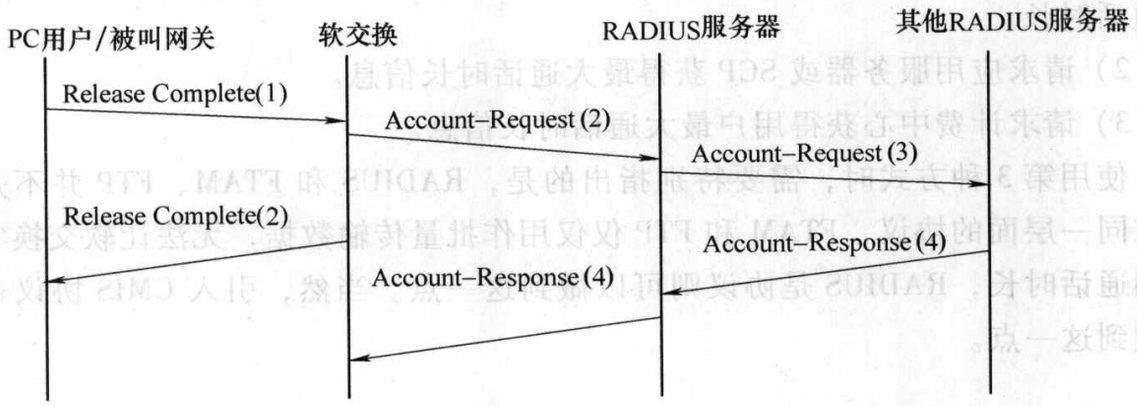


图 6-16 PC 用户发起呼叫实例

1）PC 用户或被叫网关向网守发送“释放结束”（Release Complete）消息。

2）网守利用“计费请求”（Account-Request）消息将计费信息发送 RADIUS 服务器，同时网守向被叫网关或 PC 用户回送“释放结束”（Release Complete）消息。

3）如果是漫游用户，则利用“计费请求”将计费信息向远端 RADIUS 服务器。

4）收到“计费响应”（Account-Response）消息后向网守回送“计费响应”。如果不是漫游用户，直接进行计费信息处理，向网守发送“计费响应”。

第 7 章　软交换的服务质量

7.1　QoS 的背景和定义

7.1.1　QoS 的背景

随着 Internet 规模的不断扩大，网上的实时业务量也在不断增长，软交换网络上的应用类型将会是各种各样。不同的应用对网络的需求也有所不同，应用业务发展的速度突飞猛进，也希望网络以相应的速率传输这些业务量。另外，不同的应用对网络传输的延时和延时抖动的敏感程度不一样，有的应用能容忍少量的业务量流失，有的应用则要求可靠传输。例如：实时语音传输有严格的延时要求，却能容忍少量的分组丢失；FTP 应用不能容忍分组的丢失，但对延时没有严格的要求。更进一步地说，由于实时业务对网络传输的延时、延时抖动等特性较为敏感，当突发性高的 FTP 或含有图像文件的 HTTP 等业务量在网络上传输时，实时业务会受到影响。这些高速率的多媒体实时业务（如动态图像传输）既占用网络的大量带宽资源，又不能容忍网络的延时，因此应在 IP 网络上导入 QoS 技术，以确保实时业务的通信质量，使网上资源获得最佳利用，降低成本，改善对用户的服务。

7.1.2　QoS 的定义

软交换服务质量（QoS）是指用户要求软交换网络传输系统所必须保证的关于信息传输质量的特征集，它反映服务提供者（系统）和服务使用者（用户）之间的能力和需求关系，是用来描述网络性能的。网络系统通过 QoS 体系来支持多媒体应用，提供服务质量保证。通过 QoS，网络能够对数据包进行合理的排队，对其中特定的数据包赋以较高的优先级，从而加速传输的进程，并实现实时交互。QoS 所追求的传输质量在于：数据包不仅要到达其传输的目的地址，而且要保证数据包的顺序性、完整性和实时性。

各大标准化组织也相应制定了 QoS 的定义以及一系列相关的协议和标准。

1. OSI 参考模型中的 QoS 定义

OSI 参考模型中的服务质量是用参数的形式进行定义的。即当高层协议要求低层协议提供相应的服务质量时，高层协议向低层协议发送包含 QoS 参数值的服务数据单元，低层协议按高层协议的 QoS 要求进行有关操作。

OSI 参考模型中的 QoS 参数主要针对数据传输。这些定义既未考虑服务类型，也未考虑传输错误的概率分布和传输峰值变化等因素。而且，OSI 参考模型的 QoS 定义未给出实现 QoS 控制的方法和框架。

2. ATM 的 QoS 定义

ATM 不仅把服务质量的概念引入了交换机中，而且根据应用对 ATM 的服务进行了分类。ATM 的服务可以分为 5 类：

（1）恒定数据传输率服务

（2）实时可变数据传输率服务

（3）非实时可变数据传输率服务

（4）自适应式数据传输率服务

（5）未制定数据传输率服务

3. IETF 的 QoS 定义

IETF 在 RFC2216 中对 QoS 进行了定义：QoS 是用带宽、分组延迟和分组丢失率等参数描述的关于分组传输的质量。IETF 的 QoS 控制主要是为了在 Internet 提供质量服务，例如，IETF 定义了综合服务模型以及与此相关的实现协议。

综合服务模型实质上是一个从端到端的性能到网络中元素如何控制和实现这些性能为用户提供满意的服务质量的总称。IETF 的综合服务模型出发点是网络延迟，IETF 认为网络延迟是衡量 QoS 好坏的主要标准。根据应用对延迟的要求，综合服务模型把 Internet 提供的服务分为与传输时间无关的尽力而为型服务、保证型服务以及负载受控型服务。

IETF 还制定了与 QoS 实现相关的协议和策略：主要包括资源分配和如何实现保证型服务和负载受控型服务。IP 网络的 QoS 有一套度量指标，包括带宽、延时、延时抖动、分组丢失率：

1）带宽：网络能提供给用户的资源传输速率。

2）延时：分组的第一个 bit 离开发送端与分组的最后一个 bit 到达接收端的时间间隔：

① 传输延迟（transmission delay）：设备同步一个分组所需要的时间，即接收到分组的第一个 bit 与接收到分组的最后一个 bit 的时间间隔，取决于接收速度和分组大小；

② 传播延迟（propagation delay）：一个 bit 从发送方到接收方所需要的时间，取决于介质和距离；

③ 交换延迟（exchange delay）：设备从收到分组到开始传输分组之间的时间间隔，取决于设备中的队列长度和调度策略。

3）延时抖动：分组之间的延时差异。

4）分组丢失率：一定时间间隔内，丢失的分组/传输的分组总数，它主要因

为网络拥塞时丢弃的分组。

4. ITU-T 的 QoS 定义

ITU-T 给 QoS 的定义是，QoS 是一个综合指标，用于衡量使用一个服务的满意程度。因此在简单意义上，有服务质量的服务就是能够满足用户的应用需求的服务，或者说，可提供一致的、可预计的数据交付的服务。进一步讲，QoS 描述了关于一个服务的某些性能特点，这些性能特点是用户可见的，它以用户可理解的语言表示一组参数，这些参数具有客观值或主观值。客观值刻画了系统的行为性能，如时延、抖动、包丢失率、吞吐量，主观值刻画了系统的其他服务性能，如安全性、优先级。

在网络环境中，QoS 是一组服务要求，网络必须满足这些要求才能适应不同服务级别的数据传输。不同的多媒体应用要求具有不同的服务需求，所以必须将这些服务参数化，ISO 用 QoS 定义服务需求的一组参数，使得不同的多媒体应用可以使用同一套服务。

QoS 的实施可以使实时程序最有效地使用网络带宽。由于它可以确保某个使用级别有充足的网络资源，所以它为共享网络提供了与专用网络类似的服务级别。

7.1.3　IP QoS 原理

一直以来，IP 网络都支持尽力而为的数据传送。对于一般的应用而言，现有的尽力而为的数据传送通常可以满足要求，但是现在多媒体应用的发展使得互联网对数据的传输有着不同的需求。例如，对于远距离医学，传送的准确性比时延或抖动更加重要；而对于软交换，抖动和时延是关键因素，必须将其降至最小。在传统的电话网中，路由是基于电路的概念的，即在整个通话期间首先建立起端到端的电路连接，这样，通话中的两个用户之间的任何信息总是经过相同的物理通道传送。在这种路由方式下，一旦通话建立，就可以直接保证所要求的传输特性。这种从支持电话交换中产生的模式，也被 ITU-T 用于数据通信标准的虚电路定义中。这个概念首先在 X.25 中被使用，后来又用于帧中继和 ATM 网络中。相比之下，IP 路由方式避免了为建立和维持虚电路所需要的机制。在网络层，数据报文都是逐跳传送的，即由一个路由器传到下一个路由器。每个路由器都要重新分析到达的数据报文，查找路由表，然后根据路由表中的下一跳路由把它们发送到与该路由器相连的最合适的路由器或目的主机上。一般而言，目前大多数的路由器采用先进先出的排队机制以保证公平地处理所有的数据包。正因为如此，不同数据类型的数据业务没有被区别对待，并且数据包到达的次序以及是否能够准确到达目的地都得不到保障。

QoS 研究的目的就是为用户提供有效的端到端的服务质量控制或保证。QoS

的定义就是网络单元（例如，应用程序、主机或路由器）能够在一定级别上确保它的业务流和服务要求得到满足。QoS 并没有创造带宽，只是根据各种应用对 QoS 的不同需求为其提供不同的服务质量。例如提供专用带宽减少报文丢失率、降低报文传送时延及消除时延抖动等。实际上就是要求路由设备能够：

1. 识别特定的数据流

识别特定的数据流的方法有很多种，IP 源地址、IP 目的地址、上层协议类型、源端口号、目的端口号都可以成为识别的依据，IP 头部的 TOS 域也可以成为识别的依据，特定的物理接口或 IP 隧道以及其他一些策略都可以成为识别的依据。不同的 QoS 方案的重要差别往往在于它所采用的识别特定的数据流的方法。如果将通往某一接口的数据都归为一个数据流，那么网络对外提供的就是目前 IP 网上的尽力而为的服务；如果将任意 TCP 或 UDP 连接都归为不同的数据流，那么网络对任意的用户都提供了它所要求的服务。前一种情况对路由器的要求最低，因而对用户几乎不提供任何 QoS。后一种情况对路由器的要求很高，消耗大量的 CPU 资源，同时它也提供了最贴近用户需求的服务，它也是 IETF 在 1994 年推出的基于 RSVP 的 IntServ 解决方案采用的数据流识别方法。

2. 检查特定的数据流是否符合预先声明的业务等级，实施相应控制

流量监管的目的是为了确保数据流确实以一种事先声明的方式通过网络设备。对于提供 QoS 来说，流量监管是必要的，因为不受控制的数据流能蚕食所有带宽，进而危及到为其他用户提供所需要的服务。

数据包经过沿途路由器所采取的行为：依据数据包所属的数据流，将它插入合适出口的相应级别队列中。业务管理机制根据各队列情况对输出队列的业务进行调节，以建立一种定义明确的等级行为。主要功能有分配输出带宽、针对网络中发生拥塞时如何丢弃数据包等制定规则。

3. 对所有的数据流进行排队输出

网络数据流的排队对于数据流有效通过网络是必需的。数据包的输出及拥塞丢弃都是在排队的基础上进行的。常见的排队方法有先入先出排队法、优先级排队法和加权公平排队法等。优先级排队法允许对数据流分类，然后数据流在 4 个队列中的一个中排队，按优先级从大到小的顺序，队列分为高、中、常规和低，不符合分类方案的任何数据流都默认地放在常规队列中。优先级排队法通过在服务下一个较低优先级队列前清空全部高优先级队列来分配带宽，这个动作保证高优先级队列中的任何数据流对接口有立即访问权，而低优先级队列中的任何数据流必须等待高、中和常规优先级队列中的所有数据流都空。这样，高优先级数据流以低优先级数据流等待的代价得到对接口的立即访问。

队列的特点是只在设备内部实现，没有互通性要求，不同厂家的设备可能队列调度策略实现不同，但不存在互通问题。

4. 阻塞监测和阻塞避免

我们知道 TCP 是通过超时重发、慢速启动，快速下降实现阻塞控制。当数据流的传输速率超过链路的承受能力，引起下面一连串的事件时，发生阻塞崩溃。

1）路由器缓冲区开始满。

2）分组被延时，增加了分组的传输时间，有时候引起分组地重发。

3）重发增加了数据流的量。

4）路由器缓冲区溢出，到达的数据流被丢弃。

5）丢弃的数据流引起更多地重发。

6）源站开始慢速启动或阻塞避免。

7）源站再次增加传输速率，导致缓冲区溢出。

8）更多的分组被丢弃，这种情况周而复始。

这种情况下，数据流没有充分利用带宽，网络的延迟、抖动、丢包率等技术指标急剧恶化，难以支持流式应用。通过监视输出队列深度，并按照一定的策略丢弃数据可以使得网络提供一种可以接受的服务质量。

为实现上述 4 个目的，QoS 应该提供了下述功能：

1）对报文进行分类和标记；

2）避免拥塞、管理网络；

3）流量监管和流量整形；

4）QoS 信令协议。

图 7-1 所示为路由器的一个接口在不支持 QoS 的情况下发送报文的情况。所有要从该接口输出的报文按照到达的先后顺序进入接口的 FIFO（先进先出）队列，而接口在发送报文时从 FIFO 队列的头部开始依次发送所有的报文。在发送过程中没有区别不同的报文，也不对报文传送的质量提供任何保证。在发生阻塞时，超出队列容量的报文要被丢弃。

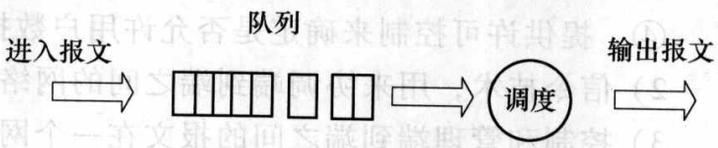


图 7-1　不支持 QoS 的路由器发送报文

图 7-2 则是一个用优先级队列来支持 QoS 时路由器接收、发送报文的情况。在报文到达接口后，首先按照报文所属的类别进行分类，然后让报文进入所属队列的尾部。在报文发送时按照队列的优先级高低来发送，只有在所有优先级高的队列发送完毕后，才发送优先级低的队列中的报文。这样，每次发送报文时总是将优先级高的报文先发出去，保证属于较高优先级队列的报文有非常低的时延和高的带宽。其报文的丢失率和通过率这两个性能指标在网络拥塞时也可以有一定的改善。

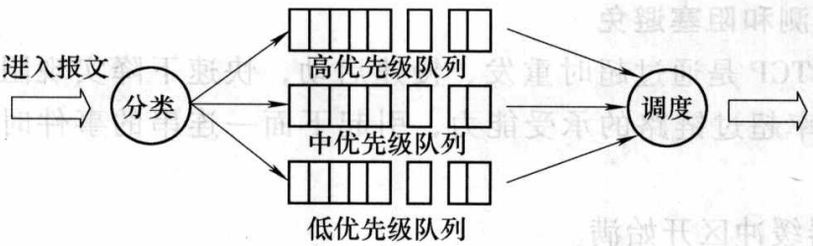


图 7-2 用优先级队列来支持 QoS 时路由器接收、发送报文

QoS 的设计的目标是可以控制各种网络应用和满足各种网络应用的要求，如：

1）可以控制资源。若可以限制网络上 FTP 使用的带宽，也可以给数据库访问以较高优先级。

2）对不同的业务可以定制 QoS。对于一个 ISP 来说，其用户可能传送语音、视频或其他实时业务，ISP 应该能区分这些不同的报文并为它们定制不同的 QoS。

3）满足多种不同需求。可以为时间敏感的多媒体业务提供带宽和低时延保证，同时也可以使其他业务在使用网络时也不会影响这些时间敏感的业务。

要在一个网络中完成端到端的 QoS，需要以下的 3 个部分来支持：

1）能够支持 QoS 的各网络元件，如路由器、以太网交换机等。这些网络元件的作用是：

① 对报文进行分类，并对不同类别的报文提供不同的处理；

② 提供队列管理和调度功能来满足不同应用所要求的不同的 QoS；

③ 提供流量监管和流量整形，限制和调整报文输入、输出的速度；

④ 提供许可控制来确定是否允许用户数据流使用网络资源。

2）信令技术，用来协调端到端之间的网络元件为报文提供 QoS。

3）控制和管理端到端之间的报文在一个网络上传送。

QoS 一般由下列参数来评测：

1）业务可用性：用户到互联网业务之间连接的可靠性；

2）传输延迟：指两个参照点之间发送和接收数据包的时间间隔；

3）延迟抖动（Jiter）：也称为可变延迟，指在同一条路由上发送的一组数据流中数据报文之间的时间差异；

4）吞吐量：网络中发送数据包的速率，可用平均速率或峰值速率表示；

5）丢包率：在网络中传输数据包时丢弃数据包的最高比率，数据包丢失一般是由网络拥塞引起的。

7.1.4 QoS 技术分类

IETF 建议了数种支持 QoS 方案，主要有：综合服务（Integrated Service，

IntServ）/资源预留协议（Resource Reservation Protocol，RSVP）、区分服务（Differentiated Service，Diff-Serv）、多协议标记交换（Multi-Protocol Label Switching，MPLS）、业务流量工程（Traffic Engineering）和基于约束的寻路。综合服务模型的主要特征就是资源预留，它使用 RSVP 作为信令协议来建立信道和进行资源预留。区分服务通过给分组打上不同的标记，把分组分成不同的类别，对不同类别的分组给予不同的服务。MPLS 是一种转发方案。分组在进入 MPLS 域时被赋予一个标签，以后就根据这个标签对分组进行分类、转发、服务。业务流量工程的目的是让网络上的业务流量更加均衡。基于约束的寻路要解决的是根据某些约束条件，如：带宽和时延要求，寻找一条合适的路径。

目前实现 QoS 有两种思想，一种是在报文转发前，先按照服务质量的要求给该业务流分配资源，网络上的节点按照分配的资源对数据报文进行转发。由于带宽资源是有限的，这种采用资源预留进行带宽分配的方法并不适合于尽力而为型的网络应用。另外一种思想是引入了优先级概念，使得优先级高的数据报文先被处理，从而达到满足对 QoS 的要求。按照以上思想，目前 IP QoS 大体可以划分为两种基本类型：

1. 基于资源预留的 QoS

网络资源按照某个业务要求的 QoS 进行分配，制定资源管理策略，IETF 提出的综合服务体系结构便是基于这种策略，其中资源预留协议是其核心部分。以 RSVP 信令向网络个节点提出业务流传输规格，并负责建立和拆除传输路径上的业务流状态。主机和路由器节点负责建立和保持业务流状态信息。

2. 基于优先级的 QoS

网络边界节点负责对业务流进行分类、标记、整形。核心节点按照资源管理策略分配资源，对 QoS 要求高的业务赋以高的优先级优先处理。IETF 提出的区分服务便是基于这种策略，在区分服务网络中，边界路由器根据用户的流规格将用户流划分为不同的级别，再聚合成流聚集，聚集信息存放在 IP 包头的 DS 标记域，称为 DSCP。核心网内部节点则根据 DSCP 提供不同 QoS 的调度转发服务。

同时，QoS 技术也可分为“保证型”和“尽力型”：

1）保证型的 QoS 在终端双方通信之前先要预留资源，确保通信过程中的网络带宽。如 ATM 业务类型、IP 网络中的 RSVP。一旦资源预留的请求被网络接受，通信中所需的带宽就能保证，与连接相关的应答时间、时延要求也能保证。如果网络不能提供所需的带宽，它就会拒绝资源预留请求。因此，这种技术的线路利用率相对要低一些。

2）在非连接型通信中，常常使用的是一种不能确保带宽的尽力型 QoS 技术。例如，在 IEEE802.1Q/p 的标记帧或 IP 分组头中的 ToS 域有 3bit 的优先级，

供 LAN 交换机或路由器进行优先级控制，由于采用这种方法不能在通信中确保带宽，只是按优先级顺序进行交换或转发，因此也称为“CoS（Class of Service）”，以区别能保证带宽的狭义 QoS 技术。这种方法的线路利用率高，但具体的效果难于预测。

软交换中，由于 IP 网是其最主要的承载网，所以上面提到的 IP QoS 主要技术将会应用于软交换网络的质量保证中。下面主要介绍最常用的综合服务模型和区分服务模型。

7.1.5 综合服务模型

综合服务（IntServ）模型使网络提供者能够提供和支持一些特殊的额外服务，以体现其与竞争对手的区别。在分组网络上支持 QoS 并不是一个新的概念，为了支持这项功能，在过去的几年中已经提出了大量的研究论文、实验和原型系统。由 Bob Braden 和 Dave Clark 等人在 1994 年 6 月正式提交给 IETF 并成为 RFC1633 的文献是第一个描述 IntServ 模型的需求和基本组成的正式文献。后来 IETF 最终成立了综合服务（IntServ）工作组，目的是完善综合服务模型，并实现对这种新的服务模型所必需的接口和需求进行定义和标准化。目前，IntServ 工作组已经完成了 IntServ 模型的定义、基本组成部分、新业务类型等方面的工作。该工作组的工作是与 RSVP 工作组协同进行的。

RFC1633 扩展了互联网体系结构和协议以支持综合服务，包括实时和非实时服务。服务模型针对两种实时服务来设计，即保证服务和预测服务，它综合了这两类服务，并提供对多播的支持。在整个互联网的不同的网络上使用同一种模型，以便使网络简单，并使得模型的实现简单化。

综合服务模型的基本思想是所有的流相关状态信息应该是在端系统上，它基于每个流单个的或是汇聚的提供端到端的保证或是受控负载的服务框架使 IP 网能够提供具有 QoS 的传输以用于对 QoS 要求较为严格的实时声音/视频业务。IntServ 使用一种类似 ATM 的 SVC 的方法，它在发送方和接收方之间用 RSVP 作为每个流的信令，RSVP 信息跨越整个网络，从接收方到发送方之间沿途的每个路由器都要为带有 QoS 请求的数据流预留资源，路径沿途的各路由器包括核心路由器都必须为 RSVP 数据流维护软状态。在 IntServ 流中定义了 3 种类型的业务：

1）保障服务（Guaranteed-Service，GS）对于 GS 业务流的最大时延是受到控制的，路由上的任何时延都会影响最大排队时延；

2）受控负载业务（Controlled-Load-Service，CLS）CLS 没有固定的时延保证，但业务流要与在网络轻负载情况下的流质量相当，实际上 CLS 要求有长期的带宽保证；

3）尽力而为的业务（Best-Effort，BE）类似当前 Internet 在多种负载环境（由轻到重）下提供的尽力而为的业务。

综合服务模型框架如图 7-3 所示。

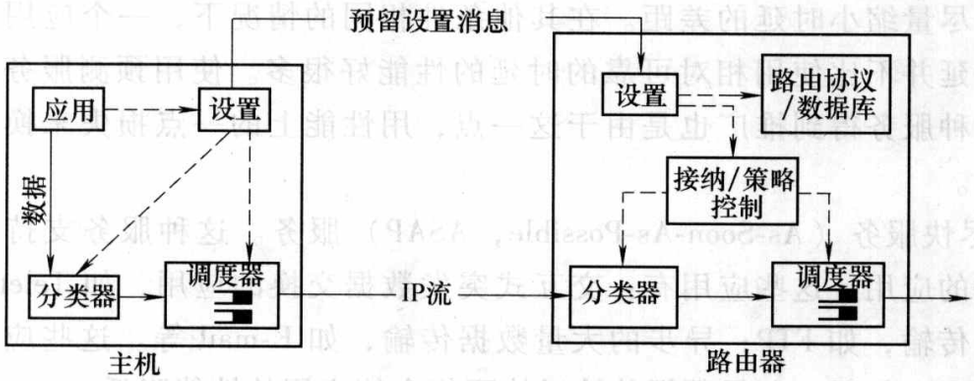


图 7-3　综合服务模型框架

模型主要由预留建立协议、流规格、流量控制器所组成。预留建立协议使主机或路由器能够根据协议动态地保留网络中的资源，以便满足不同业务流的服务需求。RSVP 就是实现预留建立协议的一种协议。流规格的作用是将一种特定业务的流映射成一种 QoS，以便在网络设备中进行处理。在 IntServ 模型中所定义的流的概念是指具有相同 QoS 的、从某个源端到目的端的一个分组报文的序列。从概念上不太严格地说，流量说明可以被看作是一个保留的网络资源的集合，这些资源是网络支持该业务流 QoS 所必需的条件。为支持特定 QoS 所必需的网络资源，在模型中还存在一些组件对流量进行控制和管理。流量控制组件的配置可以采用某种信令控制协议，如 RSVP，也可以采用人工配置。

从这个模型中可以看出，资源设置只是一个控制协议，它不承载任何数据。在 IntServ 模型中，实现 QoS 的关键组成部分是分类器和调度器。

1. 综合服务模型提供的服务

综合服务模型提供的服务主要有两种：

1）保障服务（Guaranteed Service），适用于对时间特别敏感的数据，如语音数据，通话双方对数据传输的延迟比较敏感，影视数据，同样对延迟和失真比较敏感。保障服务对某一种数据的传输的时延有着明确的要求，对每种数据流有一个固定的最小时延参数，即数据的最小时延应该不大于影响某一应用的效果的时延。现在还有一种“自适应”的保障服务，不必对每种数据流规定固定的最小时延参数，而是根据数据流传输的实际情况对最小时延参数进行调整，对于某些应用来说，这并不影响其性能。

2）预测服务（Predictive Service），适用于对时间要求不严格的数据，提供相当可靠、但不是完全可靠的时延参数，这个时延参数与确保服务的时延参数不同，不是根据某一应用在最差的情况下数据流能允许的时延来确定的，而是

根据对数据流进行保守估计而计算出来的。另外，由于多数对时间要求不严格的应用都具有一定的适应性，因此，预测服务也可以提供“最小——最大”服务，即尽量减小数据传输的最大时延。这种服务不是要减少每一个数据包的时延，而是尽量缩小时延的差距。在其他条件相同的情况下，一个应用使用绝对可靠的时延并不比使用相对可靠的时延的性能好很多。使用预测服务可以提高效率，这种服务得到推广也是由于这一点，用性能上的一点损失来换取整体效率的提高。

3）尽快服务（As-Soon-As-Possible，ASAP）服务。这种服务支持对时间延迟不敏感的应用，这些应用有：交互式突发数据交换的应用，如 Telet；交互式大块数据传输，如 FTP；异步的大量数据传输，如 E-mail 等。这些应用不在乎时间的延迟，相反，如果数据传输过快可能会使应用的性能降低。

当然这些分类方法并不是唯一的。

2. 综合服务模型的实现

这种服务模型的过程是，在发送报文前需要向所在的网络申请特定的服务，这个请求一般是通过 RSVP 来完成的。应用程序首先通知网络它自己的流量参数，并指定所需要的服务质量请求，包括带宽、时延等。应用程序一般在收到网络的确认信息，即确认网络已经为这个应用程序的报文预留了资源后才开始发送报文，同时应用程序发出的报文应该控制在流量参数描述的范围以内。网络在收到应用程序的资源请求后执行资源分配，检查许可控制，即基于应用程序的资源申请和网络现有的资源情况判断是否可以为应用程序分配资源。一旦网络承诺将满足应用程序的 QoS 需求，网络将根据两端的 IP 地址端口号协议号为每个数据流确定并维护一个状态，然后根据这个状态执行报文的分类流量监管、决策、排队及其调度来实现对应用程序的承诺。

要完成这个过程要由 4 个功能组件完成：队列调度器、访问控制器、分类器和资源预留协议。

1）分类器：为了流量控制和计费，每个要被转发的数据报文都要被映射到某一个分类中，在同一分类里的所有报文被调度器同等对待。分类是根据报文头部字段的内容，或是附加在每个报文上的分类标识号来选择的。每一个类别对应的内容可能大不一样，比如，有的类别对应着所有的视频数据报文，而有的类别可能只对应某一单一的数据报文。在不同的路由器中，同一种数据报文可能被分成不同的类别。如核心网上的网络设备上可能将多种数据流映射到同一类别中，而在边缘的设备由于数据量小而将每种报文都分成一个独立的类别。常用的分类器是多域分类器，当网络设备接收到数据包时，它根据数据报文头部的多个域（如 5 元组：传输协议，源 IP 地址，目的 IP 地址，源端口号，目的端口号），将数据包放入相应的队列中。

2）队列调度器：它将要转发的数据报文按照分类的结果放进不同的队列，具体的策略应该针对特定的输出媒质而制定。在调度器中，还包括一个组件——评估器。评估器用来测试输出数据流的能力，然后统计报文的策略和认证控制。调度服务常见的调度算法有 WFQ、WF2Q、SCFQ、VC、MD-SCFQ、WRR 等。

3）访问控制器：它使路由器或主机用户决定是否允许一个新的数据流得到它所需要的 QoS，而不影响已有的 QoS 保证。在一个主机请求实时服务时，访问控制在每个节点独立做决定。访问控制可能需要认证，它将在认证和计费方面起到很大作用。

4）资源预留建立协议：该协议是在终端主机和主机之间的路径上的网络设备上建立的，并在数据通信期间，由各个网络设备独立地为每个业务流维护流量状态。预留是指一个应用怎样协商申请一个 QoS。最简单的方式就是应用直接申请所需的 QoS，网络再做决定是允许还是拒绝。当然，现实情况要复杂的多，大多数的应用会得到能接受的服务水平。目前使用的协议是 RSVP。RSVP（RFC2205）即资源预留协议它是 Internet 上的信令协议通过 RSVP 用户可以给每个业务流（或连接）申请资源预留要预留的资源可能包括缓冲区及带宽的大小。这种预留需要在路径上的每一跳都要进行，这样才能提供端到端的 QoS，以保证 RSVP 是单向的。预留适用于点到点以及点到多点的通信环境。

3. 综合服务模型的优缺点

（1）综合服务模型的优点

综合服务模型的优点是能够提供绝对有保证的 QoS。详细的设计使 RSVP 用户能够仔细地规定业务种类，因为 RSVP 运行在从源端到目的端的每个路由器上，因此可以监视每个流以防止其消耗比它请求预留和预先购买的要多的资源。RSVP 在源和目的地间可以使用现有的路由协议决定流的通路，RSVP 使用 IP 包承载使用软状态的概念通过周期性的重传 PATH 和 RESV 消息协议能够对网络拓扑的变化做出反映，正如 PATH 和 RESV 刷新用来更改该预留的流的通路那样，没有了这些消息时 RSVP 释放与之关联的资源。设计综合服务模型开始的目的之一就是使得 QoS 能够工作在单播和多播下，RSVP 能够让 PATH 消息识别多播流的所有端点并发送 PATH 消息给它们，它同样可以把来自每个接收端的 REVP 消息合并到一个网络请求点上，该点可以让一个多播流在分开的连接上发送同样的流。

（2）综合服务模型的缺点

1）可扩展性是 IntServ 结构最致命的一个问题，因为 IntServ 采用端到端的信令，这在一运营网络中几乎无法实现。单纯从 IntServ 结构的实质来看，资源预留本身就与 IP 网络的最大特点“无连接”相冲突。另外，对保障型业务需要网

络全部使用综合业务，如果中间有不支持的节点/网络存在，虽然信令可以透明通过，但实际上对于应用来说，已经无法实现真正意义上的资源预留，所希望达到的 QoS 保证也就打了折扣。

2）对路由器的要求较高。由于需要进行端到端的资源预留。必须要求从发送端到接收端之间的所有路由器都必须支持所实施的信令协议，因此所有路由器必须实现 RSVP、许可控制、MF（Multi-Field）分类和包调度。

3）该模型不适合于短生存期的流。因为为短生存期包预留资源的开销很可能大于处理流中所有包的开销，但因特网流量绝大多数是由短生存期的流构成的。在短生存期的流需要一定程度的 QoS 保证时，综合服务模型就显得得不偿失了。

4）如何为资源预留申请授权，并确定优先权，也是 IntServ 结构本身很难克服的问题。

7.1.6 区分服务模型

由于对综合服务模型利用全程信令将原本面向无连接的 Internet 改为向面向连接的网络，所以这种方式的可实施性令人怀疑，因而希望能够出现一种新的解决问题的思想，既考虑已有网络的现状，又能达到实现服务质量的目的，这就出现了区分服务（DiffServ）模型。IETF 组织制定了 IP 网络的 QoS 标准——区分服务，为在 IP 网络上提供服务质量保证奠定了基础。在 DiffServ 标准以前，ATM 技术还是唯一能提供 QoS 的网络技术，IETF 正是借用了 ATM 的 QoS 概念，将 QoS 引入 IP 所有业务，这些业务不但可以在租用线、帧中继和 ATM 上传输，而且还可以在 SDH 等链路上传输。DiffServ 还定义了 ATM 和帧中继所不能提供的动态 QoS 服务，并增加了新的拥塞管理机制。

区分服务是由综合服务（Intserv）发展而来的，它采用了 IETF 的基于 RSVP 的服务分类标准，抛弃了分组流沿路节点上的资源预留。区分服务将会有效地取代跨越大范围的 RSVP 的使用。区分服务区域的主要成员有：核心路由器、边缘路由器、BB（Bandwidth Broker，带宽控制器）。在区分服务中，网络的边缘设备对每个分组进行分类、DS（Differentiated Service，区分服务）域，用 DS 域来携带 IP 分组对服务的需求信息。在网络的核心节点上，路由器根据分组头上的 DS 码点选择码点所对应的转发处理。带宽控制器配置了管理规则，为客户分配资源，它可以通过 SLA（Service Level Agreement，服务级别协议）与客户进行相互协调以分享规定的带宽。

与 IntServ 类似，DiffServ 也定义了 3 种业务类型：

1）最优的服务（EF）：类似于传统运营商网络的专线业务；

2）分等级的服务（AF）：这不仅仅是一种业务，而且是一个大的类别，可

以根据发展的需要定制不同的业务等级；

3）尽力而为的服务（Best Effort）：类似于 Internet 中尽力而为的业务。

IP 网络的 QoS 较之 ATM 的优越性还在于它能够提供更灵活的服务，如 DiffServ 支持基于日期的应用与动态 QoS，因此，可以给予某一特定的用户（如基于 Web 的电子商务）较高级别的 QoS 服务，而对于不那么重要的应用（如 E-mail）则给以“尽力而为的服务”。

DiffServ 比 IntServ 更具可扩展性，它可用于企业的广域网中，并在运营商网络中发挥重要的作用，因为它可以根据应用或业务类型排出不同的优先级别。IP QoS 的业务区分结构使用 IPv4 报头中的业务类型（TOS）字段，并将 8 位 TOS 字段重新命名，作为 DS 字段，其中 6 位可供目前使用，其余 2 位预留以备将来使用，如图 7-4 所示。该字段可以按照预先确定好的规则加以定义，使下行节点通过识别这个字段，获取足够的信息来处理到达输入端口的资料包，并将它们正确地转发给下一跳的路由器。这里需要注意的是在 IPv4 网中所定义的 TOS 字段与在 DiffServ 中的 DS 字段不同。TOS 字段的定义如图 7-5 所示。IPV4 报头中 TOS 字段定义 DiffServ 充分考虑了 IP 网络本身灵活，可扩展性强的特点，将复杂的服务质量保证通过 DS 字段转换为先进的单跳行为，从而大大减少了信令的工作。

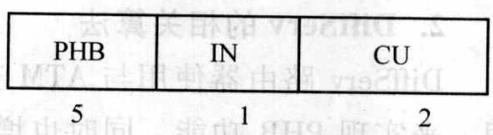


图 7-4　DS 字段的定义

PHB—单跳行为　IN—包是进入还是退出

CU—现在还未使用

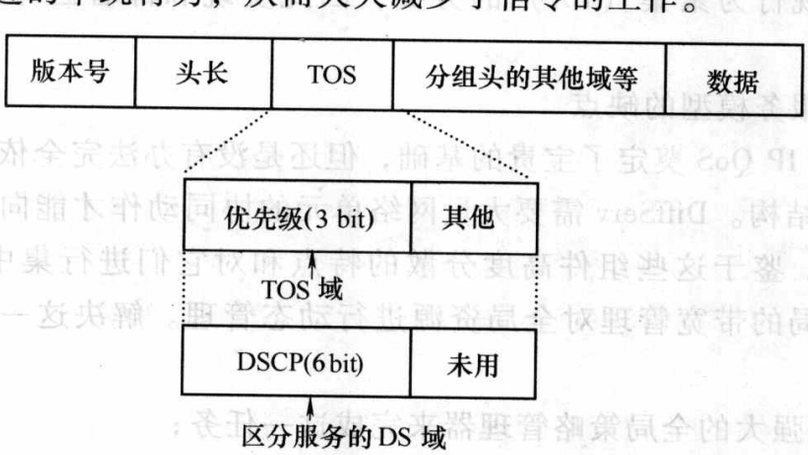


图 7-5　TOS 字段定义

1. DiffServ 优先级排列

区分服务提供一种简单粗略的方法对各种服务加以分类。不过用其他方法也可以，目前有两个单跳（PHB）的标准，其中对两个最有代表性的服务等级（业务类别）作了规定：

1）快速转发：有一个单独的码点。EF 可以把延迟和抖动减到最小，因而能提供总合服务质量的最高等级。任何超过服务范围（由本地服务策略决定）的

业务被删除。

2）保证转发：有4个等级，每个等级有3个下降过程（总共有12个码点）。超过 AF 范围的业务不会象范围内的业务那样，以尽可能高的概率传送出去。这意味着业务量有可能下降但不是绝对的，根据预定策略的标准，PHB 适用于网络入口的业务，在这点加以标记，然后根据这个标记进行路由指向，没有作标记的业务就放到了网络的出口。

2. DiffServ 的相关算法

DiffServ 路由器使用与 ATM 交换机组类似的输入管理器与输出调度器的原理，来实现 PHB 功能。同时也增加了拥塞管理机制用以处理 IP 网络交通拥塞问题，为此 DiffServ 使用了两种算法分别为随机早期检测 RED 与加权随机早期检测 WRED 算法。通过使用这两种算法达到在发生拥塞之前随机丢包以避免路由器过载的目的，这种丢包策略也能使传输控制协议（TCP）减慢发送端数据发送的速度。

3. 区分服务模型的优缺点

（1）区分服务模型的优点

区分服务模型的优点是伸缩性较好。DS 字段只规定了有限数量的业务级别，状态信息的数量正比于业务级别而不是流的数量。

便于实现只在网络的边界上才需要复杂的分类标记管制和整形操作核心路由器只需要实现行为聚集（BA）的分类，因此实现和部署区分服务都比较容易。

（2）区分服务模型的缺点

DiffServ 为 IP QoS 奠定了宝贵的基础，但还是没有办法完全依靠自己来提供端到端的 QoS 结构。DiffServ 需要大量网络单元的协同动作才能向用户提供端到端的服务质量。鉴于这些组件高度分散的特点和对它们进行集中管理的需要，必须有一个全局的带宽管理对全局资源进行动态管理。解决这一问题的方法有两个：

1）用功能强大的全局策略管理器来完成这一任务；

2）利用 MPLS 将第三层的 QoS 转换为第二层的 QoS，通过运营网中第二层的交换机来实现端到端的服务质量保证。

4. 区分服务模型与综合服务模型的互通

如果 WAN 用的是区分服务模型，而 LAN 用的是综合服务和区别服务模型的混合形式，那么当发送者和接收者之间的通路同时需要 LAN 和 WAN 时如何才能够保证端到端的 QoS 呢？IETF 建议了两种互操作方式：

1）将综合服务覆盖在区分服务网上，RSVP 信令完全透明地通过区分服务网，位于两种网络边缘的设备处理 RSVP 消息，并且根据区分服务网络中合适的

资源的可用性提供许可控制；

2）简单的并行处理，区分服务网中的每个节点可能也是具有 RSVP 功能的，采取一些策略决定哪些包用 RSVP，哪些用区分服务处理，这种模型可能适用于小型网络。

7.1.7　两种模型的比较

IntServ 对于每一种用户的需求，利用信令向沿途的每个中间路由器提出资源保留请求，可实现端到端签约服务质量保证。DiffServ 简化了信令，采用汇聚相似的业务流和逐跳的转发方式来提供一定程度上的 QoS 保证。表 7-1 详细讨论它们之间的异同。

表 7-1　IntServ 和 DiffServ 的不同

	综合服务模型	区分服务模型
QoS 保证	能够提供有绝对保证的 QoS。RSVP 运行在从源端到目的端的每个路由器上，因此可以监视每个业务流，从而防止其消耗的资源比它请求预留的资源要多	相同类别的业务分享等级服务，同一流聚集中各组成流存在一定程度的资源竞争，因此不能保证端到端的绝对质量。但在适当规划的运营网络中，该模型并不影响多种业务的提供
扩展性	扩展性不好。由于它是基于流的、与状态相关的服务模型，随着流数目的增加，状态信息的数量将成比例地增长，因此它会占用了过多的路由器存储空间和处理开销	扩展性较好。DiffServ 的服务粒度不再是每个流，而是比流更“粗”的粒度。在 DS 域中仅指明了有限服务类别，这样状态信息不再与业务流成正比，而是与业务的类的粒度成正比，因此大大提高了网络的扩展性
信令	RSVP 在源和目的地之间可以使用现有的路由协议。RSVP 可通过 IP 数据包来承载并且具有软状态的特点，通过周期性地重传 PATH 和 RESV 消息，协议能够动态地适应网络拓扑的变化。并且如果这些消息得不到及时的刷新，RSVP 将释放其预留的资源	不影响路由。与一些以虚电路方式实现 QoS 的方案（如 ATM）不同，区分服务节点提供服务的方法仅限于队列调度和缓冲管理，不涉及路由选择机制
实现复杂度	对路由器的要求较高，实现复杂。由于需要进行端到端的资源预留，必须要求从发送者到接收者之间的所有路由器都需支持必要的信令协议，因此所有路由器必须实现 RSVP、许可控制、包分类和包调度	DiffServ 可以将其大部分实现复杂度转移到网络的边缘上，而在网络核心只需实现最简单的服务保证机制。更为重要的一点是 DiffServ 模型通过约定的 DS 值来表示不同的服务等级，从而只需在网络边缘进行用户服务请求到 DS 的映射，免去了在网络的核心节点上显式预留信令的必要，从而降低了其实现复杂度。并且区分服务模型对路由器的要求较小，这些都有利于在 Internet 上实现区分服务

（续）

	综合服务模型	区分服务模型
多播支持	能够像支持单播流那样方便地支持多播流。RSVP 采用了面向接收者的方法，它能够识别多播流中的所有接收端，然后发送 PATH 消息给它们。并且它可以把来自多个接收端的 REVP 消息汇聚到一个网络汇聚点上	多播实现较复杂
适合范围	不适合用于业务量较小的流。因为这种情况下为该流预留资源的开销很可能大于处理流中有效数据的开销。但是，目前 Internet 流量绝大多数是由业务量较小的突发流构成的（如 WWW，FTP 应用），当这些流只需要一定程度的 QoS 保证时，综合服务模型的效率很低	路由器开销小、适合由网络核心使用。易于实现大吞吐量的 IP 网络。复杂的业务分类等处理功能移到了网络的边缘，大大减轻了内部路由器的负担，实现大吞吐量的业务容易得多

7.2 软交换的 QoS 分析

软交换网络中存在多种实时业务，而此刻又发生网络资源不足时，不能简单地将语音业务置于优先，这需要结合多种 QoS 技术及策略来为各种业务提供需要的服务质量。

首先，网络需要对一些大量占用带宽的非关键业务（如 FTP 等）进行带宽限制，对关键业务提供一定的带宽分配和保证。在路由器的 QoS 保障技术中，可以综合使用上面提到的两种 QoS 模型：IntServ 和 DiffServ 来提供这样的服务。

例如，可以将语音业务的 IP 地址的某一位设置为特定值，具有一定的特征，或按照上层协议端口划分语音业务，即用特定 IP 地址或上层端口来标识语音业务，网络内特定地址端口间的访问数据即为相应语音业务数据，而其他网络地址的通常就是非语音业务数据。

一般地按照实际工程中的经验值，在网络拥塞时，需要保证关键业务（如电子商务、组播等）得到主要的带宽（50% ~60%），语音业务也能得到相应的网络带宽（30% ~40%），限制 FTP 业务不超过 64kbit/s 的带宽。这样，网络就能够通过带宽资源保证，减小各种业务的拥塞及时延。另外，当关键业务流量不饱和时，该部分带宽可以被别的业务占用，从而提高整个网络的资源利用率。

另外，在骨干层，随着 MPLS 技术的日渐成熟，网络通过 RSVP，综合全网资源情况，对语音业务进行选路，保持语音业务流穿过骨干网的隧道，保证语音业务的服务质量。

值得一提的是，部署 QoS 服务策略，需要从网络的边缘接入部分就开始进行，这样才能得到全网的服务保障。在数据网的不同层次，可以采用不同 QoS 技术，保证语音端到端的服务质量，例如：在汇聚层及核心层，通过相关设备提供的 MPLS、DiffServ 等，将语音业务标识为高优先级；在骨干层，通过全网提供的 MPLS，结合 RSVP 的应用，对语音业务提供高优先级服务。

软交换网络的质量保证，除了使用上面提到的 QoS 模型进行软交换的 QoS 管理外，还应该在硬件、软件及管理策略上采用较为适用的相关技术。部分相关要求如下：

1. QoS 从技术角度来看

网络系统各种性能尺度的综合，主要包括带宽、时延、抖动和丢包率等参数，软交换的 QoS 保证主要是对带宽、时延、抖动以及丢分组率等参数的保证，具体业务不同，对 QoS 各项指标的要求也不一样，见表 7-2。

表 7-2　软交换的 QoS 业务指标

业务类型	敏感性			
	带宽	丢分组率	时延	抖动
语音	微	中	大	大
电子商务	小	大	大	小
事务	小	大	大	小
电子邮件	小	大	小	小
Telnet	小	大	中	小
偶尔浏览	小	中	中	小
连续浏览	中	大	大	小
文件传输	大	中	小	小
视频会议	大	中	大	大
多播	大	大	大	大

2. 对软交换硬件的要求，有以下三个方面：

（1）软交换网络端到端时延要求

（2）软交换网络抖动时延要求

（3）软交换网络丢包率要求

3. 应用 RTP/RTCP 控制服务质量

在 IP 包交换网络中语音及视频等实时业务都是使用 RTP 打包传送的，而 RTP 的控制协议 RTCP 可以配合 RTP 完成应用层的流量控制和拥塞控制功能。在 RTP 会话期间各参与者周期性地传送 RTCP 包，RTCP 包中含有已发送的数据包的数量、丢失的数据包的数量等统计资料。因此媒体网关可以利用这些信息动态地改变传输速率，甚至改变有效载荷类型。RTP 和 RTCP 配合使用能以有效

的反馈和最小的开销使传输效率最佳化，因而特别适合传送网上的实时数据。RTCP 将控制包周期发送给所有连接者，软交换的媒体网关通过 RTCP 的分析监控程序并结合其他网络管理程序的信息，同时根据网络的实时运行状况，随时利用各种手段以避免网络服务质量的下降。例如当通过分析 RTCP 的发送和接收报告分析监控程序，检测到包到达率逐渐下降时，即用 RTCP 包通知服务的发送方，这样就可以改变媒体的格式或者压缩算法以满足网络可提供的带宽，从而保证该业务的基本服务质量。

4. 相关的语音技术，主要有以下几点：

（1）回声抵消

（2）静音压缩

（3）舒适噪声

（4）丢包补偿

（5）静音抑制

5. COPS 策略控制协议

由于不同的软交换系统可能采用不同的 QoS 技术和模型，从而产生不同的服务质量，即使运行于同一个软交换系统上的应用，如果各个转发节点采用的 QoS 技术不相同，数据流的分类规则、优先级设置规则也各不相同，那么对软交换这个具体应用来说，就既浪费网络资源，又不能使用户获得好的服务质量，这些问题将导致无法为一个软交换具体应用提供端到端的 QoS。这种问题最先出现在 IP 网络中，为了在 IP 网络上获得高效、统一的 QoS 技术，解决 QoS 技术中存在的一系列问题，就提出了策略服务的概念，用来对网络上由企业、网络提供商提供的多数设备用一个统一的规则进行策略设置、运营、管理一体化，实现统一的软交换 QoS 管理。

策略控制是指在掌握网络中业务和设备的运行状态的基础上，通过一定的策略自动地对网络的带宽和访问进行管理。因此策略控制的关键在于对 QoS 进行的统一管理上。当用户使用 QoS 保证要求的新业务时，希望业务端到端的延迟、丢失率、抖动等性能指标处于一定范围内。因为网络资源总是有限的，所以网络首先需要根据一定的策略来决定是否能够接纳用户的传输分组。如果现有的网络资源能够满足当前用户分组的 QoS 要求，则接纳它，否则拒绝。然后根据传输分组的信息来决定该分组属于哪类数据流，不同类型的数据流对应的服务等级是不同的，并且要求使用一种认证方法来确认用户，以此来决定数据流获得何种服务等级。当服务等级确定后，网络就会按照 QoS 参数来对数据流进行 QoS 路由、缓冲区调度、排队控制，以满足数据流对服务质量的要求。

策略管理框架主要包括策略管理环境、策略库、策略决策点（PDP）和策略执行点（PEP）。功能简述如下：

1）策略管理环境为人与系统交互的接口，提供策略规则的定义、编辑和管理功能，它可以提供 GUI 和 API 两种界面；

2）策略库存储策略信息和规则，原则上它可以采用任何一种技术，如数据库（DB）或目录，但推荐使用目录的方式；

3）策略决策点（PDP）为策略服务器，具有 3 种功能：

① 响应策略事件，并锁定相应的策略规则；

② 完成状态和资源的有效性校验；

③ 将存储在策略知识库中的策略规则转换成设备可执行的格式。

4）策略执行点（PEP）作为策略系统的客户端，负责执行具体的策略操作。

在软交换中，采用策略管理可以对软交换设备进行软交换服务上的管理，当软交换设备将策略管理的命令转化为内部命令时，软交换就能提供端对端应用的 QoS 的保证了。

第8章　软交换的业务提供技术

与传统智能网相对封闭相比，使用开放的应用编程接口为业务提供商方便快捷地开发业务提供了可能，它可以根据网络的能力，向用户提供更为丰富的业务。开放式应用编程接口的目的和意义在于：将业务开发接口公共化、抽象化，使得第三方开发商也能够加入通信增值业务的开发者行列；并且屏蔽下层网络的具体协议，使得第三方开发商不需要太多的专门协议知识，最终达到快速、廉价开发和部署新业务的目的。

新一代可编程业务接口技术的开放性主要体现在两点：一是业务的可扩充性。传统的电信网络上，接入控制、媒体传输控制及呼叫处理、智能业务、应用业务和业务运营等都是靠不同的业务平台和协议完成的。这样，电信业务就可以灵活地加以扩充。二是业务的交换性。电信业务是由模块组成，用户可以自由定义，任何业务可以通过增加模块的方式实现新的业务，这样可以产生针对不同用户的个性化业务。

8.1　软交换的业务综述

1. Parlay 技术

Parlay 工作组成立于 1998 年，旨在规范面向对象的业务控制，并独立于底层的通信技术。其业务目标是为第三方业务提供商提供市场，使业务更加满足单个企业的需要和用户个性化的需要，促使较小的 IT 公司也能够开发电信业务，而电信运营商从用户对智能网平台的访问中获利。

Paraly 定义了两类接口，第一类是框架接口，它提供了信任和安全管理、事件通知、完整性管理和业务注册、业务订购、业务查找等模块。其中，信任和安全管理模块是客户访问框架的出发点，使客户可以查找感兴趣的业务，订购和选择要访问的业务，同时，在客户注册到框架之前，必须得到框架的认证。事件通知模块主要用于通知客户应用所发生的与服务相关的事件。完整性管理模块包含负载管理、故障管理和操作维护管理等模块的接口。业务注册模块使业务运营商提供的业务可以注册到框架，注册业务必须指明所提供业务的类型。业务查找模块使客户应用可以查找当前框架所维护的业务类型及其业务属性的列标。业务订购模块使企业运营商可以订购这些已经注册到框架的业务。第二类接口是业务接口，每个 Parlay 业务都实现一个业务管理的接口，当客户应用

请求使用该业务时，即发送给客户应用。Parlay 规范定义了呼叫控制、用户交互、终端能力、消息和连通性服务以及在线管理和计费服务等。Parlay API 使用户或第三方控制的业务能够与 IT 系统（如电子邮件、用户数据库等）综合起来。

2. JAIN 技术

JAIN 组织的目标是综合 Internet 和 IN，实现业务的可携带、网络的融合和安全的网络接入等功能。业务可携带是指利用 Java 的移植性，使一种业务可在所有的系统平台中运行；网络融合是指允许业务运行在 PSTN、分组网和无线网上；安全接入是指允许电信网络之外的应用可以直接接入网络资源和设备，执行规定的动作和功能。

JAIN 规范工作主要分为两个组，一个是协议专家组（Protocol Expert Group，PEG），目前，PEG 规范了 No. 7 的 TCAP、ISUP 和 MTP 的接口，以及到 SIP、MGCP 和 H. 323 的接口；另一个是应用专家组（Application Expert Group，AEG），它规范在 Java 框架中生成跨越 PEG 规定的所有协议的应用所需的 API。目前，AEG 规范了呼叫控制（Java Call Control，JCC）API 和调度及事务（Java Coordination and Transactions，JCAT）API、安全网络接入 API、业务生成环境（Service Creation Environment，SCE）和业务逻辑执行环境（Service Logic Execution Environment，SLEE）以及连通性管理。JCC/JCAT 的目的就是对业务开发者隐藏所有底层协议，提供统一的与底层网络接口的方法，使业务应用仅与 JCC 和 JCAT 接口交互，随后通过 JAIN 适配器将呼叫和数据传递到不同的网络。JAIN 的 SCE 和 SLEE 与智能网的 SCE 和 SLEE 作用相同，JAIN 业务生成环境定义了标准的 Java 业务模块，这些模块用图形工具和 Java 组件（Java Bean）描述，可以将不同的业务模块综合起来定义新的业务。业务生成后，可以在业务逻辑执行环境中测试和部署。JAIN 业务逻辑执行环境基于 Java 虚拟机、EJB 和其他 Java 应用。在 SLEE 中部署的业务都是信任域的业务，为了实现网络的安全接入，JAIN 与 Parlay 合作，规范了 JAIN 业务提供接入 API，使得与 JAIN 兼容的实体可以访问 Parlay 框架，接入框架支持的业务。

8.2　Parlay 技术

Parlay 应用编程接口（Parlay Application Programming Interface，Parlay API）是一套开放的业务接口标准规范，它基于分布式技术，使用统一建模语言（Unified Modeling Language，UML）定义了一套完整的业务应用编程接口集合。Parlay API 屏蔽了各类网络资源的操作细节，降低了业务开发的技术门槛，业务开发商使用它提供的强大的网络访问功能，可以方便地对网络进行控制，为用户

快速开发新业务。

Parlay 网关是实现 Parlay API 的物理设备，由它把底层 CAP（CAMEL Application Part，CAMEL 应用部分）、事务处理能力应用部分（Transaction Capabilties Application Part，TCAP）、移动应用部分（Mobile Application Part，MAP）、会话初始化协议（Session Initiation Protocol，SIP）等协议映射为标准的提供给运行业务逻辑程序的应用服务器使用。Parlay 网关由框架服务器和能力服务器组成，应用服务器接入框架服务器后，通过鉴权、业务发现、业务选择和业务使用条约签署等操作获得能力服务器事先向框架服务器注册的业务能力管理类对象引用。这些引用一般包括呼叫控制管理类引用、用户交互管理类引用和计费管理类引用等，这些引用都对应于能力服务器中的对象，应用服务器在分布式的环境下操作这些引用关联对象中包含的方法就可以对网关进行控制，从而实现用户业务的控制功能。

8.2.1 Parlay 组织的组成和现状

Parlay 组织成立于 1998 年 3 月，由 BT、Ulticom、Microsoft、Nortel 和 Siemens 5 家公司联合发起成立，主要研究支持外部应用访问安全网络内部资源的网络接口规范，其目的是根据下一代通信网络发展的需求，提出一系列 API 把底层通信网络的各种协议抽象为容易理解的一套接口，对于通信网络的开放和与 Internet 互通有很大作用，因此得到了业界的广泛关注，发展很快，目前其成员几乎已包括所有业内的知名企业。Parlay 组织的目标是要开放电信领域的能力，以使最大范围内的市场参与者可开发和提供先进的电信业务。为此该组织制定了一套开放的、独立于具体技术的、用于第三方业务开发和部署的 API，称为 Parlay API（Parlay 规范）。

从 1998 年 12 月推出 Parlay API 1.0 规范到目前为止已推出到 Parlay API 4.0 版本。目前 Parlay 研究组定义的 API 已得到大多数工业组织的引用。很可能成为未来网络开放的标准接口。

8.2.2 Parlay 规范定义的商业角色和商业模型

Parlay 规范定义了 5 种对象，所有商业活动围绕它们展开：

（1）网络运营商

（2）服务提供商

（3）第三方业务提供商

（4）服务

（5）提供业务的应用程序

商业模型如下：框架作为网络运营商提供的一个平台，在整个商业模型中

担当交易场和中间人的作用。服务提供商把 GCCS（General Call Control Service，普通呼叫控制服务）和 Message 等基于各种底层网络的服务注册到框架上，第三方业务提供商通过框架订购这些服务，并利用这些服务提供的 API，开发各种增值业务如预付费业务。

8.2.3　Parlay3 技术

Parlay3 与 Parlay2.1 相比有很多的不同。但是这些不同不是体系结构的大的变化，主要体现在一些服务的增加和以前的内容的修订上。

1. 在基本类型定义方面

基本的接口类型变成 IpInterface，TpResult 和 out 参数被 Exception 类和返回值代替。

2. 框架方面

Initial、Authencation、Access 的 Framework 侧的接口都为 IpXXX，接入者侧的都为 IpClientXXX；采用多种鉴权方法，对被信任方的、对非信任方的和采用非 Parlay 的机制，比如 CORBA 的安全服务。

服务的选择、协议和注册会移到新的 Ip（App）ServiceAgreementManagement 中并新增了方法 nitiateSignServiceAgreement 和删除了方法 accessCheck；Service Factory 也被服务实例的生命周期管理对象代替。

3. 呼叫控制

GCC 只是因为兼容的原因才被保留，推荐用新的 MPCCS 来代替，MMCCS 和 CCCS 都没有什么变化，由此可见，Parlay3 的重点是支持 MPCCS。

4. 新增内容计费能力

一种在线计费服务，将第三方从维护和计费相关的用户信息中解放出来，充分利用网络运营商的能力，降低第三方的开发难度。

5. 新增策略管理

允许在策略域里进行独立于网络结构和传输及应用协议的管理，策略就是一些标准。资源管理（ConnectivityManagement）将融入到这个服务中，可以想象，这个服务的内容将非常庞杂。

6. 新增 PAM 接口

一种新的服务，可以提供关于用户的出席情况的信息和管理。是对于呼叫控制服务的有益补充，提供更加丰富的控制能力。

8.2.4　Parlay X 协议综述

Parlay API 的当前版本已经发展到 Parlay 4.0，但是 Parlay 4.0 协议规范在 IT 开发人员看来，还是相当的庞杂，采用它开发创建基于电信网络的应用还是有

一定的难度。

针对此情况，3GPP 提出了 Parlay 4.1 协议规范，也成为 Parlay X 协议。Parlay X 协议在原有 Parlay API 协议的基础上，对 API 进行了进一步的抽象，并使用基于 Web 业务的 WSDL 对 API 进行描述，从而提供了 IT 开发人员一个更为清晰、简洁、抽象、易于理解的电信业务开发接口。这样 IT 开发人员无需掌握电信网络专业知识，即可以根据 Parlay X 协议接口开发、应用下一代网络业务，从而促进下一代网络业务的发展。

Parlay X 采用 WSDL 的方式对 Parlay API 进行定义。每一个 Parlay X Web 业务都是来自 Parlay API 对电信网络业务能力集的一个抽象，并且将相关的电信网络能力通过 Parlay X 协议的方式提供给开发人员。

对于 Parlay X 协议体系：

1）由 Parlay X Web 业务提供的网络能力可以是单一的（如只包括呼叫控制部分的能力），也可以是混合的（如既包括终端定位的能力，也包括用户状态的能力）。

2）使用 Parlay X Web 业务的应用和执行此项服务的服务器之间采用基于 XML 的消息交互机制。消息的交互采用同步的请求/响应模式，并且必须由应用启动消息交互，而来自 Parlay X Web 服务器的响应是可选的。然而，需要指出的是，异步的由执行 Parlay X Web 业务的服务器（又称为 Parlay X 网关）发起的到应用的请求也是需要的，如执行一个通知类型的 Web 业务。在这种情况下，消息的交互将激活一个应用侧的 Web 业务，并且使用类似的，同步方式的请求/响应机制。

3）Parlay X Web 业务之间是不相关的，并且从应用的角度看，Web 业务本身是没有状态机制的。尤其需要指出的是，如果 Parlay X 网关发起一个到应用的异步通知，不需要执行应用侧启动的激活动作，用于提供或者禁止 Parlay X 业务侧的与通知业务有关的准则。

4）Parlay X 业务尽量采用简单的语法描述机制来对协议进行描述，从而允许 IT 开发人员将精力集中在通过通用的 Web 业务编程能力技术进行电信业务的开发上。

5）尽管一个电信业务能力可能涉及到多个类型的网络，但 Parlay X 业务是和网络设备无关的，也是和具体的网络无关的。

6）Parlay X Web 采用 Web 业务技术，并且使用 WSDL 用于进行 Web 业务的描述。

8.2.5 Parlay X 提供的第三方呼叫接口

Parlay X 提供的第三方呼叫接口包括：

1）makeACall，该接口提供了应用创建一个语音呼叫的能力，在该接口中，应用提供将创建的呼叫的主叫和被叫地址（号码）。应用可以通过该接口返回的新创建应用的标识符，对该呼叫进行后续的监视，如获取呼叫的状态信息。

2）getCallInformation，该接口用于获取一个呼叫当前的状态信息。应用可以在一个通话过程中进行多次调用该接口获取呼叫状态。即使呼叫已经结束，应用也可以通过该接口对呼叫信息进行查询（呼叫结束后，呼叫信息在 Parlay X 网关中保留的时间由运营商进行确定）。

3）endCall，该接口用于应用结束一个呼叫，若呼叫仍处于创建过程中，该接口和 cancelCallRequest 实现的功能是一致的。

4）cancelCallRequest，该接口用于应用取消呼叫的创建，若呼叫已经创建，该接口对呼叫没有任何作用。

1. Parlay X 提供的网络初始的第三方呼叫接口

Parlay X 提供的网络初始的第三方呼叫接口包括：

1）handleBusy，当 Parlay X 网关发现一个呼叫的被叫忙时，触发并调用此协议接口，应用将指示 Parlay X 网关如何处理呼叫，处理的方式包括：“继续”、“结束呼叫”和“前转”。

2）handleNotReachable，当 Parlay X 网关发现一个呼叫的被叫不可达到时，触发并调用此协议接口，应用将指示 Parlay X 网关如何处理呼叫，处理的方式包括：“继续”、“结束呼叫”和“前转”。

3）handleNoAnswer，当 Parlay X 网关发现一个呼叫的被叫无应答时，触发并调用此协议接口，应用将指示 Parlay X 网关如何处理呼叫，处理的方式包括：“继续”、“结束呼叫”和“前转”。

4）handleCalledNumber，当 Parlay X 网关发现主叫号码试图呼叫一个被叫号码，而该被叫号码（如，该号码可以是一个业务号码，而不是一个实际的电话号码）尚未被接续时，Parlay X 网关触发并调用此协议接口，应用将指示 Parlay X 网关如何处理呼叫，处理的方式包括：“继续”、“结束呼叫”和“前转”。

5）handleOffHook，当 Parlay X 网关发现一个主叫号码试图发起一个呼叫时，触发并调用此协议接口，应用将指示 Parlay X 网关如何处理呼叫，处理的方式包括：“继续”、“结束呼叫”和“前转”。

2. Parlay X 提供的短消息接口

Parlay X 提供的短消息发送接口包括：

1）sendSms，该接口提供了应用发送一个短消息的功能，当网络发送短消息后，返回应用一个本次发送的标识符，用于应用后续进行短消息发送状态的查询。

2）sendSmsLogo，该接口提供了应用发送一个短消息 LOGO 的功能，当网络

发送短消息后，返回应用一个本次发送的标识符，用于应用后续进行短消息 LOGO 发送状态的查询。

3）sendSmsRingtone，该接口提供了应用发送一个短消息铃音的功能，当网络发送短消息后，返回应用一个本次发送的标识符，用于应用后续进行短消息铃音发送状态的查询。

4）getSmsDeliveryStatus，该接口提供应用查询先前进行的短消息发送的状态情况。

Parlay X 提供的短消息通知接口包括：

notifySmsReception，该接口由应用侧的 Web 业务实现，当 Parlay X 网关发现有新的短消息需要通知应用接收时，将触发此接口，通知应用有新短消息进入。该接口只有在新短消息满足应用指定的短消息过滤条件时，才触发此接口。

Parlay X 提供的短消息接收接口包括：

getReceivedSms，该接口将返回应用，Parlay X 接收到的，自上次返回后的满足过滤条件的所有短消息。

3. Parlay X 提供的多媒体消息接口

和第三方接口功能目前面临的情形类似，为了发起和接收多媒体消息，应用程序需要使用专用的协议才能使用网络提供的多媒体消息业务功能。这就要求业务开发人员有较多的网络专业知识。即使使用 Parlay API 进行开发，由于 Parlay API 对于业务开发人员而言，仍然是一个比较复杂的协议接口，因此，业务开发人员不掌握特定的网络技术知识也很难开发出基于短消息的应用。

Parlay X 协议提供一个简单的接口集合用于发送多媒体消息，从而使得业务开发人员无需掌握具体的网络知识即可以发送和接收短消息。

对于多媒体消息，包括短消息、扩展的短消息、多媒体消息、E - mail 等，分别映射到不同的网络上。因此，从协议的涉及角度出发，即可以为每个网络提供的消息能力分别定义一套单独的 Parlay X 接口，也可以定义一个统一的协议接口来为所有的网络提供消息服务。这里，Parlay X 协议采用一个统一的协议接口用于提供多媒体消息服务。

对于消息的发送和接收，采用和短消息发送和接收类似的机制。

Parlay X 协议提供的多媒体消息发送接口包括：

1）sendMessage，该接口提供了应用发送一个多媒体消息的功能，当网络发送短消息后，返回应用一个本次消息发送的标识符，用于应用后续进行短消息发送状态的查询。多媒体消息的内容采用 MIME 或者 DIME 的编码格式，作为 SOAP 附件进行发送。

2）getMessageDeliveryStatus，该接口用于应用轮询先前发送的多媒体消息的发送情况。

Parlay X 提供的多媒体消息接收接口包括：

1）getReceivedMessages，该接口用于应用轮询查询满足过滤条件的多媒体消息，Parlay X 网关返回所有发送到应用的消息的引用。

2）getMessageURI，该接口用于读取多媒体消息的不同部分，Parlay X 网关将多媒体消息不同的部分分别创建本地的文件，并返回引用到这些部分的 URI（Uniform resource identifier，统一资源标识）。应用可以根据这些 URI 简单地读取每一个文件，或者将这些应用作为链接发送给终端用户。

3）getMessage，该接口返回消息的所有内容。消息内容采用 SOAP 附件的方式进行传送。

Parlay X 提供的多媒体消息通知接口包括：

notifyMessageReception，该接口由应用侧的 Web 业务实现，当 Parlay X 网关发现有新的多媒体消息满足指定的过滤条件时，将触发此接口，通知某一应用有新短消息进入，并携带上所有的消息内容的引用。过滤条件和应用侧 Web 业务的对应关系采用离线的方式进行定义。

4. Parlay X 的优势

前面，就 Parlay X 协议及该协议所提供的接口进行了简单的描述和研究，可以看出 Parlay X 和其他协议相比：

1）提供了一个更为抽象的协议接口，用于 IT 开发人员进行下一代网络业务的开发工作；

2）提供了更为简单的接口 API，用于进行网络操作的描述；

3）采用基于 Web 业务的方式，提供业务能力的实现，并使用 WSDL 进行接口的描述。

Parlay X 协议还是一个相对比较简单的协议接口，通过对 Parlay X 协议的研究，可以看出 Parlay X 协议还有以下几方面需要增强：

1）对呼叫的持续控制能力，一个呼叫过程中，包括众多呼叫时间，如忙、无应答等，应用需要根据这些事件对呼叫进行持续的呼叫控制，而在现有的协议接口中，尚不包括这些呼叫处理能力；

2）用户交互能力的加强，在呼叫过程中，应用需要网络和用户进行语音交互，以收集用户的信息（如用户卡号、密码），指示用户下一步的动作，而现有的协议接口，尚不包括这些呼叫处理能力；

3）Parlay X 和网络协议的映射能力，随着 Parlay X 协议对接口的进一步抽象，而底层网络的多样性和复杂性，Parlay X 协议和网络协议的映射必将随之更加复杂，因此当应用 Parlay X 协议时，需要对此进行更为细致的研究。

第9章　软交换的实践

9.1　软交换的控制功能

实际软交换设备主要完成的控制功能主要有三类：呼叫控制，承载控制，媒体控制和消息映射。

1）呼叫控制（又叫会话控制）主要控制呼叫的建立、维护和释放，负责呼叫流程的控制，并根据业务的需要设置相应的检测点，向业务汇报网络事件，接受业务的控制。

2）承载控制主要负责控制承载的建立、释放、修改和协商。

3）媒体控制和消息映射完成呼叫过程中所需资源的申请、修改、释放等操作，同时把具体协议消息映射成抽象的系统内部消息，送往承载控制和信令控制中处理；接收到信令控制和承载控制传送来的消息后，把内部消息翻译为具体协议消息，发送到网络。

在软交换系统中，针对不同的网络和终端类型，需要有多个媒体操作和消息映射功能模块。会话控制支持功能负责会话控制过程中鉴权、号码分析、路由、用户属性查询等操作的具体执行，这些操作与具体协议、网络有关，在呼叫控制支持功能中根据不同网络、协议类型分别处理。采用这种控制方式后，呼叫控制、承载控制与具体的协议无关，可以向业务提供统一的呼叫控制模型，便于软交换支持 Parlay 等业务控制接口。因此从软交换具体设计实现上讲，呼叫控制功能具有核心作用。

9.1.1　呼叫模型的设计要求

软交换的基本思想是实现业务与呼叫控制相分离，呼叫控制与承载相分离。因此，作为软交换系统核心的软交换设备就应当具备屏蔽承载网上异构性的能力，同时对业务提供统一接口。所以软交换设备的核心逻辑就是完成对呼叫的控制和媒体连接的管理，并且提供底层呼叫功能的抽象。针对软交换的呼叫控制和业务开发特点，建立一个合适的呼叫模型是软交换设计实现的关键。

软交换体系的呼叫模型的设计应当有以下要求：

1）协议的融合能力：软交换要实现多种网络的接入，支持不同的网络协议。从网络协议本身的角度来讲，协议本身的元素不同，协议过程也不同，例如：MGCP 和 SIP。如果对各协议实现单独控制，不但实现不同协议之间的互通十分

困难，同时，呼叫模型本身也十分复杂。所以，一般情况下的呼叫模型，都要对不同协议的特征进行抽象，在协议层之上建立统一的呼叫控制模型，进行统一的呼叫控制。具体来讲，主要包括的协议有：MGCP、H.248、H.323、ISUP 和 SIP 等。

2）承载控制：软交换要实现呼叫与承载的分离，而这点是较为复杂的。目前的基本承载媒体有两种：一种是基于 TDM 的 64kbit/s 电路通道和基于 IP 的 RTP 实时媒体流。软交换要实现两者之间的转换，以及呼叫对两者方式的选择，同时还得保证 QoS 的需求。

3）对业务的强大支持：软交换作为下一代网络的核心，它的诞生是由业务驱动的，下一代网络的业务要求业务实现与网络底层的无关性。另外在软交换内，要实现三方呼叫、视频及多方会议等多种业务。

9.1.2　软交换的呼叫模型

从功能上讲，软交换设备的呼叫控制功能同智能网中 SSP（service switching point，业务交换点）交换机的呼叫控制能力相似，两者在本质上没有太大的差别。因此，一般情况下的软交换设备设计可以部分参照 SSP 的逻辑框架。可将软交换系统分为 4 个部分：协议部分，呼叫控制部分，业务部分，软交换支撑部分。其逻辑框图如图 9-1 所示，而软交换所采用的呼叫控制模型，可以借鉴 BCSM（Binary Synchronons Commum′ cations Macro，二进制同步通信宏指令）来进行设计。

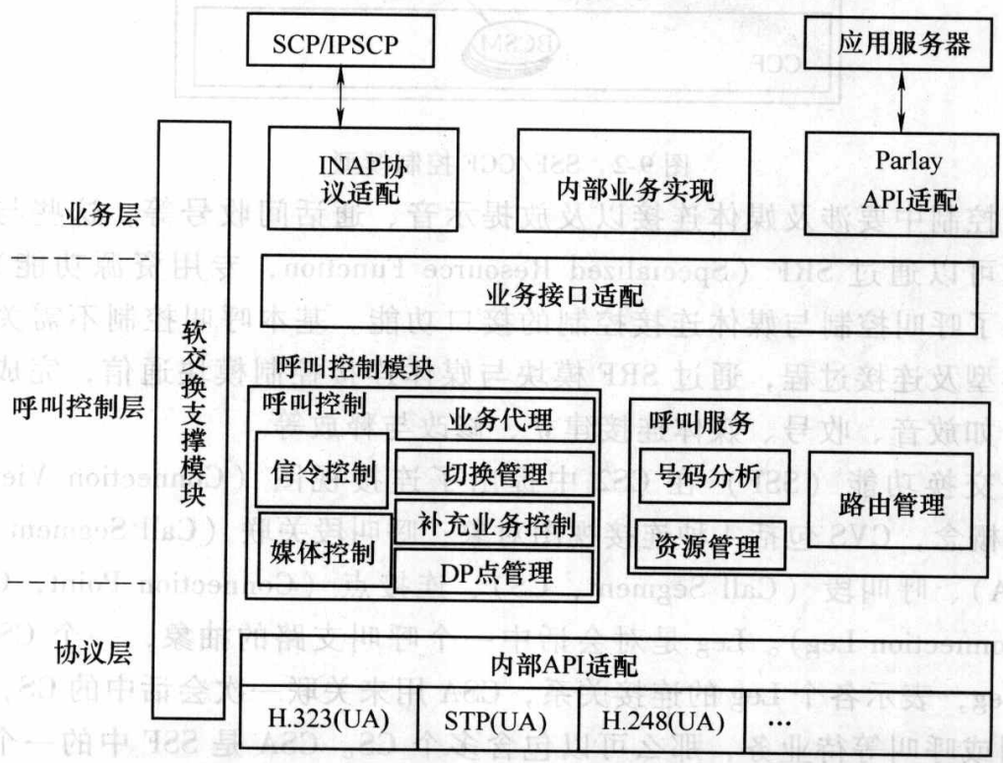


图 9-1　软交换核心体系结构

9.1.3 BCSM 呼叫控制模型

类似于智能网 SSP 交换机，基本呼叫控制模块只完成基本的呼叫处理，更复杂的业务控制流程，在基本呼叫模块中产生业务触发，由业务层实现复杂的业务逻辑。基本呼叫模型采用 CS2 描述的业务交换功能/呼出控制功能（Service Switching Function/Call Control Function，SSF/CCF）控制模型，Q.1214 为 SSF/CCF 提供了一种呼叫建模的框架，包括以下方面：基本呼叫管理者（Basic Call Manager，BCM），IN 交换管理者（IN-Switching Manager），业务交互管理者（Feature Interaction Manager，FIM）/呼叫管理者（Call Manager）等。其中，BCM 实现基本呼叫状态模型（Basic Call State Model，BCSM）和 DP（Detection Point，监测点）处理。其结构如图 9-2 所示。

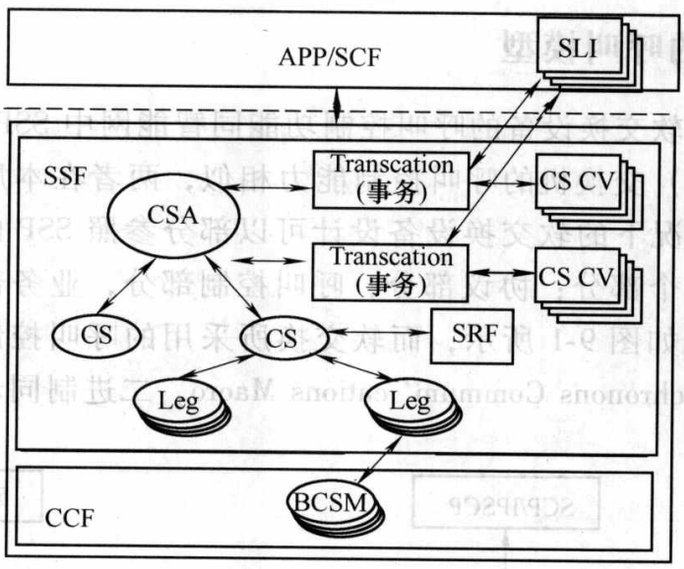


图 9-2 SSF/CCF 控制模型

呼叫控制中要涉及媒体连接以及放提示音、通话间收号等，这些与媒体相关的功能可以通过 SRF（Specialized Resource Function，专用资源功能）实现。SRF 提供了呼叫控制与媒体连接控制的接口功能。基本呼叫控制不需关心具体的媒体类型及连接过程，通过 SRF 模块与媒体连接控制模块通信，完成媒体控制需求，如放音、收号、媒体连接建立、修改与释放等。

业务交换功能（SSF）在 CS2 中提出了连接视图（Connection View State，CVS）的概念，CVS 包括 4 种连接视图对象：呼叫段关联（Call Segment Association，CSA）、呼叫段（Call Segment，CS）、连接点（Connection Point，CP）、连接腿（Connection Leg）。Leg 是对会话中一个呼叫支路的抽象，一个 CS 可以管理多个 Leg，表示各个 Leg 的连接关系，CSA 用来关联一次会话中的 CS，如果是多方呼叫或呼叫等待业务，那么可以包含多个 CS。CSA 是 SSF 中的一个子状态机，协同 BCSM 管理一次会话，共同完成呼叫处理和连接控制功能。CSA 具有

DP 触发功能，接受 BCSM 上报的 DP 事件，完成对静态配置的业务 DP 的触发，以及业务逻辑过程中业务层所动态配置的事件 DP 的触发。智能网 CS2 定义 24 个左右的 DP，对呼叫阶段的划分比较完备，软交换呼叫模型可以直接继承过来。

在连接视图概念中，把一个完整的呼叫过程分为两个半侧呼叫过程，即主叫侧与被叫侧。区分的主要目的是将源端与目的端对呼叫控制权的分离，O_BCSM 与 T_BCSM 分别对应于主叫侧与被叫侧。

智能网的呼叫模型如图 9-3 所示。

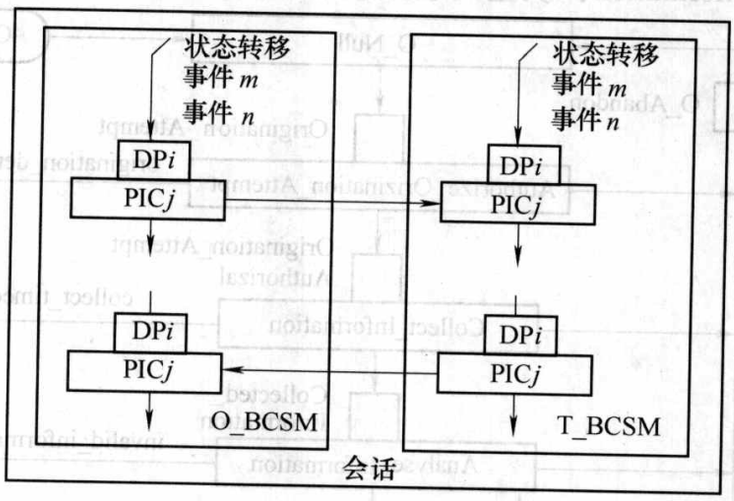


图 9-3　智能网的呼叫模型

图 9-3 中描述了 BCSM 的组件，包括：呼叫中的点（Point In Call，PIC）、DP、BCSM 转移以及事件。PIC 标识与 IN 业务逻辑实例感兴趣的一个或多个基本呼叫/连接状态有关的 CCF 活动。DP 指示基本呼叫和连接处理中的状态，在这些点，可以发生从非智能网到智能网业务逻辑的控制转移。也存在其他的状态（例如，在 PIC 里），在其由呼叫处理被中止，但智能网业务逻辑不能被调用。

BCSM 转移是从一个 PIC 到其他 PIC 的基本呼叫/连接处理的正常流程入口事件导致 BCSM 转移到 PIC，出口事件代表 PIC 的处理结果。智能网 CS-1 的 BCSM 建立的是对基本双方呼叫的现有交换处理的模型，反映源端（O 端，Originating）和宿端（T 端，Terminating）呼叫部分的功能分离，如图 9-3 所示。BCSM 只描述处理基本双方呼叫时发生的转移。PIC 入口事件一般是在处理完 DP 以后从 SSF 收到的“继续”指令，从 DP 到 PIC 的“基本”转移。PIC 出口事件是从呼叫控制代理功能（Call Control Agent Function）收到的事件的处理结果：从 PIC 到 DP/PIC 的“基本”转移。

图 9-3 中描绘主叫侧和被叫侧的基本呼叫状态（BCSM）。两者互相协作，共同控制一个呼叫的流程。也就是说两侧的状态可以相互推动，例如 O 端释放

会导致 T 端相应释放。由 O_BSCM 和 T_BSCM 共同完成的呼叫称之为会话。

BCSM 分为两个部分：发端 BCSM（O_BCSM）和收端 BCSM（T_BCSM），如图 9-4 所示。

1. O_BCSM 模型中：

1）O_NULL：此点表示呼叫发端的呼叫起始状态。

2）Authoririze_Origination_Attempt：CCF 鉴权试呼权限，主要负责对用户呼出的权限进行鉴别。

3）Collect_Information：从主叫收集拨号。

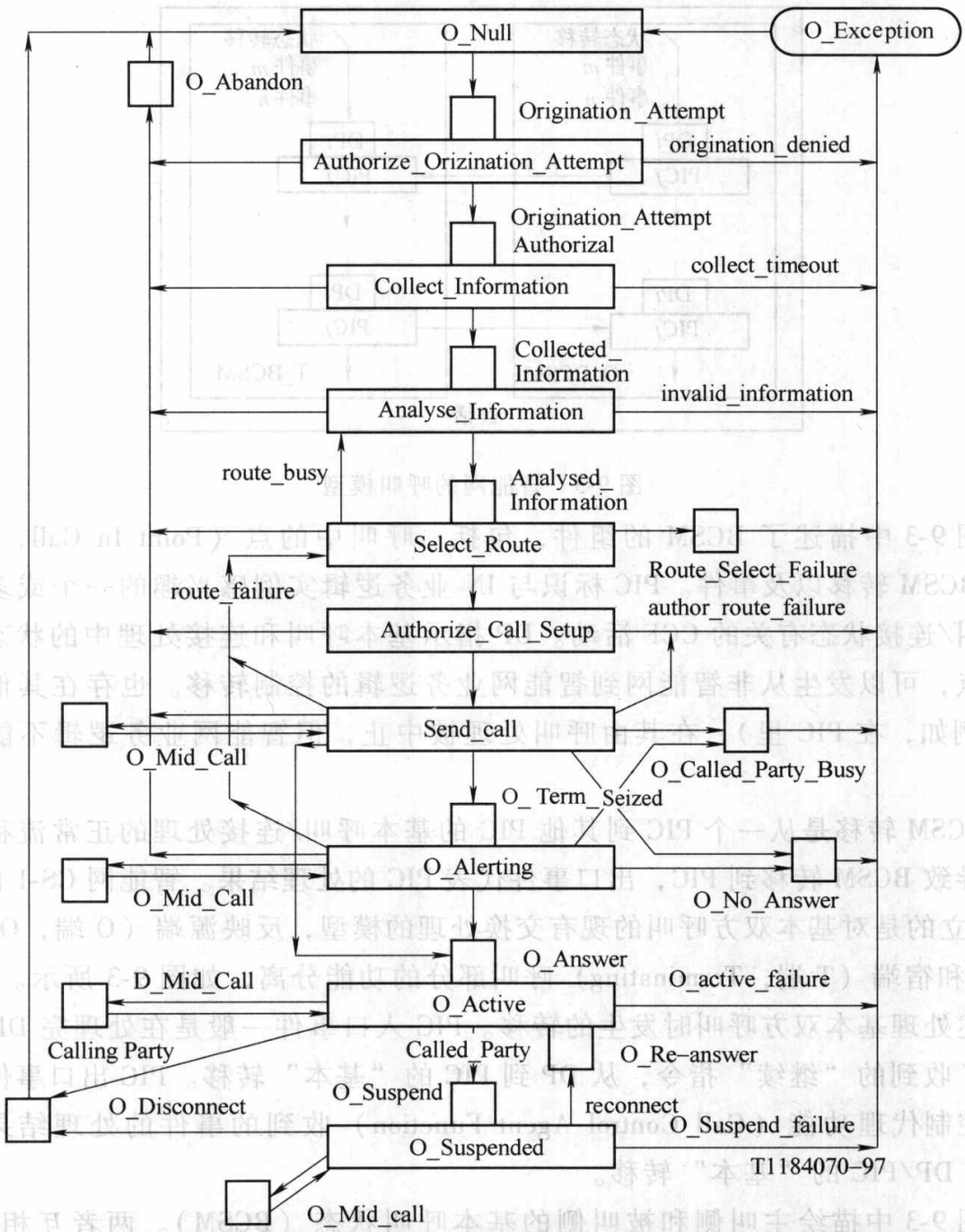


图 9-4 O_BCSM 模型

4）Analyze_Information：分析号码。

5）Select_Route：选择路由。

6）Authoririze_Call_Setup：鉴权主叫能否发起针对被叫的呼叫。

7）Send_ Call：发出呼叫建立消息，呼叫建立的控制权转移到被叫端，主叫端进行等待。

8）O_Alerting：主叫振铃。

9）O_Active：呼叫被激活。

10）O_Suspended：主叫拆线。

2. T_BCSM 模型如图 9-5 所示，其中：

1）T_NULL：此点表示呼叫被叫端的呼叫起始状态。

2）Authoririze_Termination_Attempt：验证呼叫能否传到被叫，查询呼叫是否在该线路或被叫方受限。

3）Select_Facility：选择资源。

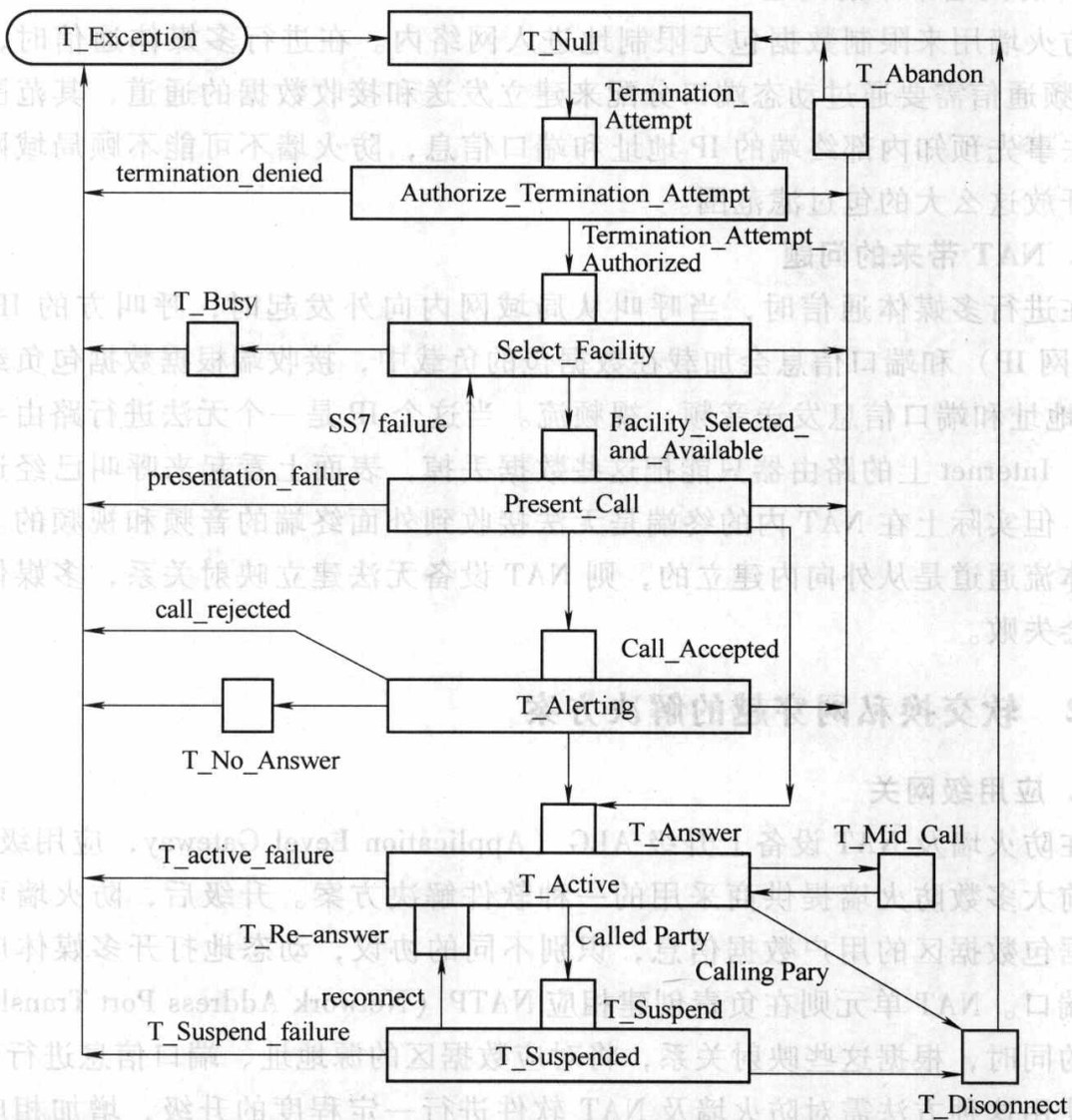


图 9-5　T_BCSM 模型

4）Present_Call：呼叫被提交到被叫端。

5）T_Alerting：被叫振铃。

6）T_Active：被叫摘机，激活呼叫。

7）T_Suspended：被叫拆线。

9.2 软交换的私网穿越

9.2.1 软交换部署时易出现的问题

通常情况下，软交换网络在局域网范围内部署时，运营商和企业出于对共享公众网 IP 资源及保护网络安全的考虑，会在网关的位置使用网络地址转换（NAT）/防火墙设备，这样就带来了互通的问题。

1. 防火墙带来的问题

防火墙用来限制数据包无限制地进入网络内。在进行多媒体通信时，由于音/视频通信需要通过动态端口分配来建立发送和接收数据的通道，其范围较大且无法事先预知内部终端的 IP 地址和端口信息，防火墙不可能不顾局域网的安全，开放这么大的包过滤范围。

2. NAT 带来的问题

在进行多媒体通信时，当呼叫从局域网内向外发起时，呼叫方的 IP 地址（局域网 IP）和端口信息会加载在数据包的负载中，接收端根据数据包负载中的源 IP 地址和端口信息发送音频、视频流。当这个 IP 是一个无法进行路由寻址的 IP 时，Internet 上的路由器只能把这些数据丢掉，表面上看起来呼叫已经连接起来了，但实际上在 NAT 内的终端是无法接收到外面终端的音频和视频的。如果多媒体流通道是从外向内建立的，则 NAT 设备无法建立映射关系，多媒体流的传送会失败。

9.2.2 软交换私网穿越的解决方案

1. 应用级网关

在防火墙及 NAT 设备上升级 ALG（Application Level Gateway，应用级网关）是目前大多数防火墙提供商采用的一种软件解决方案。升级后，防火墙可以分析数据包数据区的用户数据信息，识别不同的协议，动态地打开多媒体应用需要的端口。NAT 单元则在负责创建相应 NATP（Network Address Port Translation）映射的同时，根据这些映射关系，将对应数据区的源地址、端口信息进行改变。

使用这种方法需对防火墙及 NAT 软件进行一定程度的升级，增加相应的协议处理模块。一旦在通信中存在不能进行升级的防火墙，则无法保证多媒体通

信的正常进行。同时要分析数据包的数据信息会给防火墙带来处理能力的开销。

2. 代理网关

在与 NAT/防火墙并行的位置放置一个软交换代理，并在代理与 NAT/防火墙的外面再增加一个防火墙，从而达到使多媒体业务对 NAT/防火墙短路的目的。为这个软交换代理指定一个不同的公众网 IP 和一个局域网 IP。进行多媒体通信时使用软交换代理的 IP 地址呼叫，其他业务仍然使用原来的 IP 地址，这样，最外面一级的防火墙就可以为音、视频数据包和其他业务的数据包分配不同的路由了。软交换代理的功能像一个网关，它在两边使用相同的协议，可以将一个 IP 呼叫分成两个单独的呼叫，一个是局域网终端到代理的呼叫，另一个是从代理到公众网终端的呼叫，从而解决 NAT 带来的问题。

使用这种方法无需对原来的防火墙和 NAT 设备进行任何改变，而是让多媒体业务不经过防火墙和 NAT 设备，经过软交换代理进入局域网，不会影响原来 NAT/防火墙设备的性能。

3. RSIP 技术

RSIP（Realm Specification Internet Protocol，域限定的因特网协议）是目前 IETF 的 NAT 工作组正在研究的一种解决 NAT 问题的方案。它的解决办法是在网关上增加一个 RSIP 服务器，用于分配局域网内的应用申请的特殊的公共资源，这些公共资源可以是一个公众网 IP，也可以是一个公众网 IP 及一个特定端口。通过 RSIP，局域网地址域的客户端可以得到公众网地址域，内网的终端在向 RSIP 服务器租借资源的期间可以以公众网地址域的形式存在，但这些资源仅限于在向外的消息中使用。

公众网向内传送数据给 RSIP 服务器分配给局域网终端的公众网 IP 时，RSIP 服务器负责封装 IP 包头，由局域网终端去掉包头。多媒体终端在呼叫和传输多媒体数据时都要向 RSIP 服务器申请公众网地址、端口，并把这个地址和端口信息在确认信息中传送给发起会话的一方，可以在这些端口上双向传送数据。

4. MidCom 方案

Middlebox Communications（MidCom）是通过在第三方实体和 NAT/防火墙之间建立中间盒通信，使 NAT/防火墙设备变为可控的一种新的概念。这种方法是允许第三方成为受 NAT/防火墙信任的实体，然后代表 NAT/防火墙做出决定，强迫其开放端口传送媒体流或数据流。这些受信任的实体通过“MidCom”定义的新协议与 NAT/防火墙进行通信。受信任的第三方实体帮助 NAT/防火墙操作，而不必在其中加入智能程序去判断是否允许媒体流通过。这样，NAT/防火墙可以在对应用层内容未知的情况下继续提供安全服务。

MidCom 的框架结构是采用可信的第三方（MidCom Agent）对 Middlebox（NAT/防火墙）进行控制的机制，应用业务识别的智能也由 Middlebox 转移到外

部的 MidCom Agent 上，因此应用协议对 Middlebox 是透明的。在实际应用中，Middlebox 功能可驻留 NAT/防火墙，通过软交换设备（即 MidCom Agent）对 IP 语音和视频协议的识别和对 NAT/防火墙的控制，作为穿透 NAT/防火墙的一个解决方案。从安全性考虑，MidCom 方式可支持控制报文的加密，可支持媒体流的加密，因此安全性比较高。

5. STUN 方案

STUN（Simple Traversal of UDP Through NAT，UDP 对 NAT 的简单穿越）是一种由 IETF 研制的 UDP 流协议穿透 NAT 的协议。STUN 可以在不改变 NAT 的情况下，与任何现有的 NAT 共存。在 STUN 服务器和客户端之间可以有任意多的 NAT 设备。STUN 技术仅针对 NAT 穿越有效，对防火墙穿越没有帮助，所以，如果在整个通话话路中没有防火墙仅有 NAT，STUN 将是一种成本低廉的穿越 NAT 的选择方案。就 STUN 协议本身而言，对哪种协议并没有要求，不过到目前为止的一些成功案例，都是在基于 SIP 的系统中实现的。

STUN 即 UDP 对 NAT 的简单穿越方式。应用程序（即 STUN 客户端）向 NAT 外的 STUN 服务器通过 UDP 发送请求 STUN 消息，STUN 服务器收到请求消息，产生响应消息，响应消息中携带请求消息的源端口，即 STUN 客户端在 NAT 上对应的外部端口。然后响应消息通过 NAT 发送给 STUN 客户端，STUN 客户端通过响应消息体中的内容得知 NAT 设备的类型及其在 NAT 上对应的外部地址，并且将其填入以后呼叫协议的 UDP 负载中，告知对端，本端的 RTP 接收地址和端口号为 NAT 外的地址和端口号。由于通过 STUN 协议已在 NAT 上预先建立媒体流的 NAT 映射表项，故媒体流可顺利穿越 NAT。

STUN 方案最大的优点是无需现有 NAT/防火墙设备做任何改动，局限性在于需要应用程序支持 STUN 客户端的功能。

6. 隧道透明穿透

隧道透明传输的方案是在要进行多媒体业务的局域网内放置一个代理，经过 NAT/防火墙设备，与公众网上的服务器建立连接。局域网内的终端的呼叫和控制信令经过代理传递到服务器。代理放在防火墙内的私有网，它同时具有注册服务功能和代理功能，私有网内的终端注册到代理上，它和防火墙外的服务器创建信令和控制通道，可以把所有的注册和呼叫控制信令转发到服务器，转发信令的端口为 Well-Known，防火墙为其开放该端口。服务器放在防火墙外的公众空间，可以位于服务提供商网络或者位于企业网的 DMZ（De-Militarised Zones，非军事区）。处于公众网中的服务器根据呼叫信令中的被叫信息（别名地址）查找注册表，确定被叫终端所处的局域网的 NAT/防火墙网关。由于代理处于局域网内部，为了使服务器能穿透 NAT/防火墙将信令转发到代理，则需要在 NAT 设备中配置一条静态的 NAT 路由来实现。

信令过程完成后的多媒体数据也是由终端发送给服务器由其进行转发，为了尽量减少防火墙需要的过滤端口，从局域网内多媒体终端到服务器的所有 RTP/RTCP 数据都发往服务器的一个端口。由于 NAT 设备会改变多媒体数据包（RTP/RTCP）的源地址和端口，则代理就需要采用预先发送的探测包来让服务器区别不同的 RTP/RTCP 会话。

9.3　软交换的接入

9.3.1　媒体接入层的组网研究

媒体接入层位于 NGN 功能模型的最底层，它直接面向用户，为各类用户提供接入下一代网络的功能。其主要特点是面向不同的应用系统提供不同的接入方式，它通过各种接入手段集中用户业务，将各种信息格式转换成能够在分组网络上传递的的统一格式。媒体接入层应具备流分类、信息格式转换、接入汇聚、安全过滤以及带宽管理等功能。媒体接入层是 NGN 的重要组成部分，大量的应用系统和用户将通过它接入，可以说媒体接入层的顺利实现是 NGN 能够被推广应用的关键。

9.3.2　媒体接入层的接入方法

NGN 的引入是一个渐进的过程，现有网络应能逐渐融入 NGN，因此 NGN 的接入层应能包容目前的各种接入技术，支持现有的各种业务，其方式是采用各种各样的接入媒体网关。接入媒体网关应能提供模拟接口、V5 接口、ISDN 接口以及各种数据接口等，为传统的窄带业务和宽带业务用户提供接入的可能。另外，NGN 还采取智能终端直接为用户提供各种业务，主要包括 SIP 终端、H.248 终端和 MGCP 终端，结构如图 9-6 所示，这些终端的接入主要有两种办法：

1）直接接入承载网；

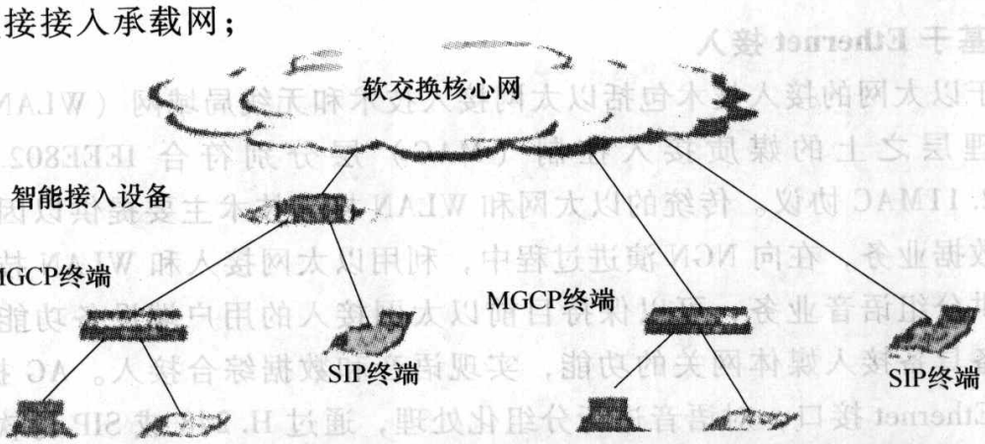


图 9-6　智能终端接入途径

2）通过媒体网关接入。

9.3.3 现阶段的接入方式

目前软交换提供的主要接入方式有：

1）IAD；

2）V5；

3）ISDN；

4）中继接入；

5）智能终端接入。

分组语音接入的实现方案

1. 基于 DSL 接入

国内的宽带接入技术发展很快，xDSL 技术就是其中之一。基于 xDSL 接入的分组语音的实现方案可以分为 VoATMoverDSL 和 VoIPoverDSL 两种。

1）VoATMoverDSL 解决方案。VoATMoverDSL 是在 ATMoverDSL 技术基础上提供语音业务的一种解决方案。整个网络是由综合接入设备（IAD）、网关和 ATM DSLAM 构成。IAD 向用户提供多路 POTS 或者 ISDN 接口、以太网接口，上行可以是 ADSL 或者 G. shdsl 接口，直接接入 DSLAM。语音业务采用 AAL2 封装，由网关进行 PVC 终结。数据业务采用 AAL5 封装，通过 ATMPVC 连到宽带接入服务器（Broad-band Access Server，BAS）。语音业务的信令通过 BLES（Broad band Loop Emulation Service，宽带循环竞争设备）协议完成，网关通过 V5. 2 或者 V5. 1 接入到 PSTN。

2）VoIPoverDSL 解决方案。与 VoATMover DSL 网络结构一样，VoIPover DSL 的网络结构也是由 IAD、网关和 DSLAM 组成。VoIPover DSL 的 IAD 与 VoATMover DSL 的类似，可以提供多路 POTS 或者 ISDN 接口、以太网（Ethernet）接口，上行可以是 ADSL 或者 G. shdsl 接口、VDSL（高速数字用户线）接口，直接接入 DSLAM。

2. 基于 Ethernet 接入

基于以太网的接入技术包括以太网接入技术和无线局域网（WLAN）接入技术，物理层之上的媒质接入控制（MAC）层分别符合 IEEE802. 3MAC 或 IEEE802. 11MAC 协议。传统的以太网和 WLAN 接入技术主要提供以因特网接入为主的数据业务，在向 NGN 演进过程中，利用以太网接入和 WLAN 技术可以方便地提供分组语音业务，可以保持目前以太网接入的用户端设备功能不变，而局端设备具备接入媒体网关的功能，实现语音和数据综合接入。AG 提供 POTS 接口和 Ethernet 接口，对语音进行分组化处理，通过 H. 248 或 SIP 与软交换进行交互，完成呼叫处理功能。

通过 IAD，同样可以实现基于 Ethernet 的分组语音接入。IAD 支持 H.248，SIP 或 MGCP，支持呼叫处理、语音处理、媒质控制处理等功能。IAD 放置在用户家中，同时提供 POTS 和 Ethernet 接口，实现语音、数据和宽带多媒体的接入。

9.4　软交换的安全问题

软交换中存在的安全问题主要有以下几个方面：

1）业务网关设备、核心控制设备和软交换支撑系统的安全；

2）信令流传输安全；

3）用户信息安全；

4）网络实体间媒体流的安全。

9.5　安全解决方案

NGN 中网络设备和业务系统是被攻击的主要目标，为了保证软交换的网络安全构架，根据软交换网络构架的层次，建立 3 条安全防线，以保证软交换网络的安全策略的实施。软交换的网络安全构架如图 9-7 所示。

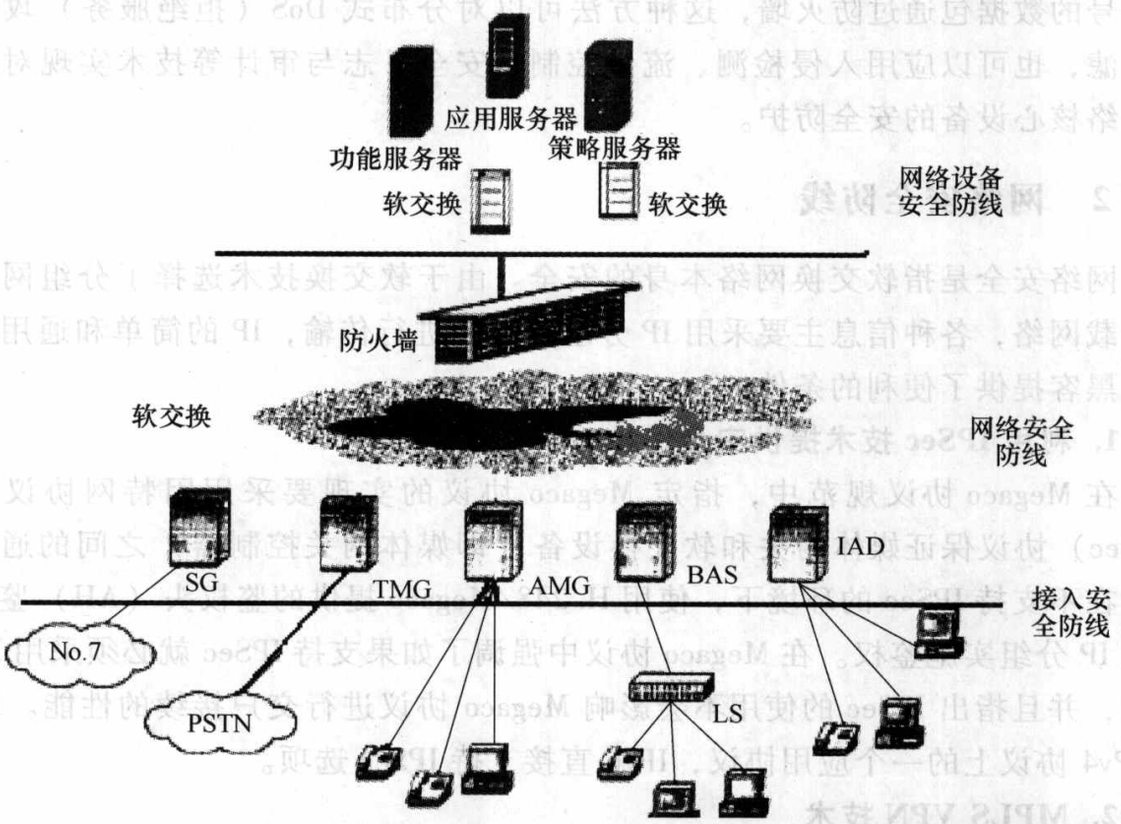


图 9-7　软交换的网络安全构架

1）网络设备安全防线：在接入口设置防火墙，保证软交换网络中的重要网络设备不受外界入侵；

2）网络安全防线：运用一系列网络安全协议，保证分组数据的安全传输；

3）接入安全防线：加密用户数据，加强认证机制，保证接入用户信息和业务的安全。

根据 3 条防线，制定一些安全方面的解决方案。

9.5.1 网络设备安全防线

网络节点设备是软交换网络中最为关键的设备，主要包括软交换设备、中继网关设备、接入网关设备、信令网关设备、应用服务器和各类服务器等。为了充分保证软交换网络的安全，首先要保证网络中核心设备的安全，如果将核心的网络设备置于开放的 IP 网络中，网络的安全性将很难保证，所以可以将网络的核心设备放在专用网络中，采用私有 IP 地址的方案。而完全采用私有 IP 地址的方案是不可行的，由于终端用户的接入方式和接入地点比较灵活，因此软交换设备要能够接入和控制终端用户，又要同时分配有对应的公有 IP 地址，这样才能保证各种方式用户的接入。为此，可以在核心网外侧设置防火墙，并将软交换网络中少数的需要与外界用户进行通信的核心设备的私有地址映射到相应的公有地址，同时利用防火墙对进入核心网的数据包进行过滤，只允许特定端口号的数据包通过防火墙，这种方法可以对分布式 DoS（拒绝服务）攻击进行过滤，也可以应用入侵检测、流量控制、安全日志与审计等技术实现对软交换网络核心设备的安全防护。

9.5.2 网络安全防线

网络安全是指软交换网络本身的安全。由于软交换技术选择了分组网络作为承载网络，各种信息主要采用 IP 分组的方式进行传输，IP 的简单和通用性为网络黑客提供了便利的条件。

1. 利用 IPSec 技术提供安全保护

在 Megaco 协议规范中，指定 Megaco 协议的实现要采用因特网协议安全（IPSec）协议保证媒体网关和软交换设备（即媒体网关控制器）之间的通信安全。在不支持 IPSec 的环境下，使用 H.248/Megaco 提供的鉴权头（AH）鉴权机制对 IP 分组实施鉴权。在 Megaco 协议中强调了如果支持 IPSec 就必须采用 IPSec 机制，并且指出 IPSec 的使用不会影响 Megaco 协议进行交户接续的性能。IPSec 是 IPv4 协议上的一个应用协议，IPv6 直接支持 IPSec 选项。

2. MPLS VPN 技术

在骨干网层面，可以采用目前相对比较成熟的技术——多协议标签交换虚

拟专用网（MPLS VPN）技术，构建相对独立的 VPN。这样可以对不同用户间、用户与公众网间、业务子网之间的路由信息进行隔离，并且非 MPLS VPN 内的用户无法访问到软交换域 VPN 内的网络设备，从而保证网络的安全。

9.5.3　接入安全防线

各种终端设备是软交换网络中最大的安全隐患，可以在软交换网络的接入边缘设置宽带接入服务器，作为用户接入网和骨干网之间的网关，终结来自用户接入网的连接，并提供接入到宽带核心业务网（主要是 IP 网和 ATM 网）的服务。宽带接入服务器一般具有网络安全模块和业务管理模块，网络安全模块包括 IP VAN 模块和防火墙模块；业务管理模块包括网络接入认证与授权模块、计费模块和统计模块，另外有些宽带接入服务器还可以提供流量管理和控制。利用宽带接入服务器可以对终端用户的接入进行控制和管理。

在软交换网络中存在大量的终端设备，其接入地点和接入方式都非常灵活，无法避免有些用户利用非法终端或设备访问网络、占用网络资源、非法享受业务和服务，并且某些用户可能利用非法终端或设备向网络发动攻击，对网络的安全造成威胁（如发送大量的 IP 数据包等）。因此，要保证软交换网络的安全，需要对软交换网络中的终端设备进行鉴权和认证，主要是 IAD 和 SIP/H. 323 等终端设备。

9.6　软交换的组网形式

控制层的层次划分有两种方式：

1. 单平面结构

所有软交换设备都处于同一个平面，软交换设备之间以网状互连。按照软交换设备路由数据设置的不同，单平面结构又可以有以下方式：

所有软交换设备均了解全网的路由设置数据（见图 9-8）。任一软交换设备的增加和减少，所有的软交换设备均需要做路由数据更改。

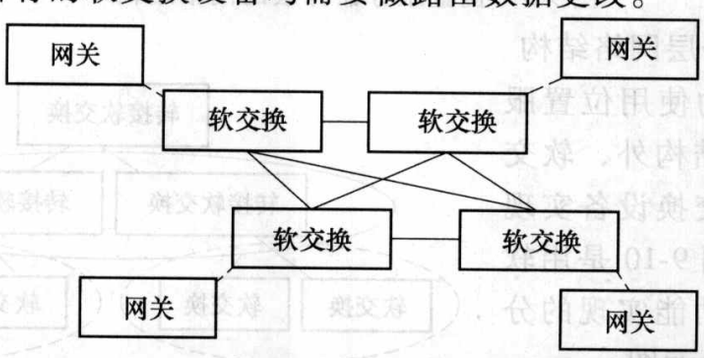


图 9-8　软交换控制层单平面全互连结构

2. 分层网络结构

（1）位置服务器分层的网络结构

在这种方式下，引入路由数据分层的概念，即软交换设备仅了解一定区域的路由设置数据，在软交换设备之上增加一层位置服务器，用于对其他区域被叫用户的寻址（见图 9-9）。位置服务器接受主叫端软交换设备的寻址请求，通过数据查询或向其他的位置服务器发出寻址请求，得到并向主叫端软交换设备返回被叫的软交换设备地址。服务器不做呼叫控制信号的传递，该方式仅在自己区域内软交换设备增加和减少时，该区域内的软交换设备需做路由数据更改，其他区域软交换设备的变动，无须对该区域软交换设备进行路由数据更改，而仅调整位置服务器中的路由数据。

此时，与呼叫相关的呼叫路由信息被存储在位置服务器上，位置服务器根据不同的地域或管理等级进行划分，形成不同层面的分层结构。软交换可配置一个或多个位置服务器，以向其发送地址解析请求。地址解析在位置服务器上完成。位置服务器之间实现路由信息的共享和互通，软交换通过位置服务器实现路由信息的获取。位置服务器之间也可采用 IP 网上的电话选路协议（TRIP），软交换和位置服务器之间可采用 TRIP 和轻型目录接入协议（LDAP）或 SIP。图 9-9 给出了利用位置服务器形成的分层网络结构。根据该网络结构，每个软交换在通过一个或多个位置服务器的地址解析后，均可直接定位到被叫的软交换系统，位置服务器用 TRIP 动态地维护它们之间的选路数据。在分层结构模式下，位置服务器的层次数量决定于实际网络容量和建设规模，运营商可根据实际情况灵活设置。从概念上看，这种分层网络结构与 H. 323 的网络结构非常相似。

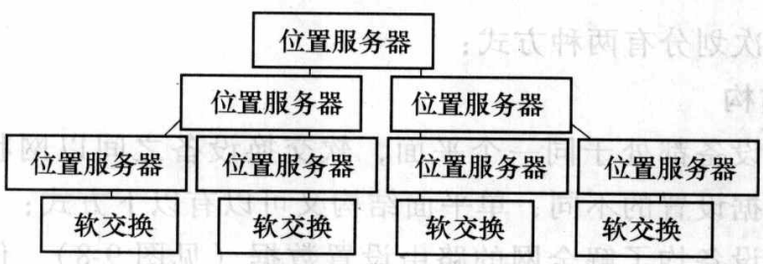


图 9-9 位置服务器分层的网络结构

（2）软交换分层网络结构

除上面提到的使用位置服务器分层的网络结构外，软交换组网亦可用软交换设备实现分层网络结构。图 9-10 是用软交换设备的路由功能实现的分层网络路由结构示意图。

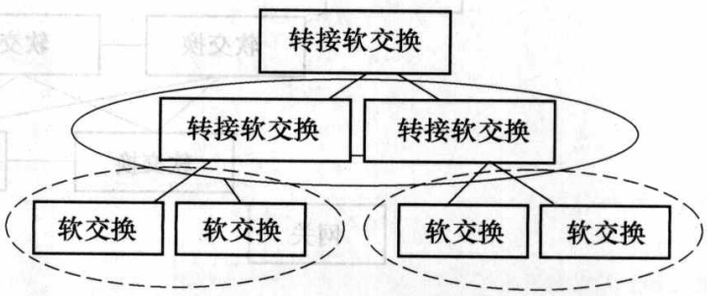


图 9-10 软交换分层的网络结构

在这种分层方式下，此方

案借鉴了现有电话网的分层思想，将软交换划分为不同层次，即软交换层面不是逻辑网状网结构，而是增加了互连软交换（代理软交换）。通常互连软交换只具备呼叫控制功能，不具备承载控制功能，即互连软交换并不控制媒体网关建立媒体流连接，但软交换之间的呼叫信令需经过互连软交换转接。互连软交换负责不同域的软交换网络之间的互通，如省间路由或者不同运营商的软交换网络之间的路由，当互连软交换的路由数据过于庞大时，可将互连软交换分成多级，如省级互连，大区级互连或国家级互连。由于这种网络结构只要求软交换设备配置和维护一定范围内的路由，即软交换只需配置和维护其同层相邻（有逻辑连接关系）的软交换及其汇接的的低层软交换和高层转接软交换的路由数据，从而大大减少每个软交换设备上路由数据。从概念上看，这种分层的网络结构与我国 PSTN 的网络结构非常相似。

9.7　软交换设备的性能要求

在 NGN 发展初期，网络规模较小时，可以采用控制层单平面结构的方式。这样，所有软交换设备处于同一平面，实施端到端的接续，控制网络结构简单。当网络发展到一定规模时，建议配置位置服务器，至于位置服务器采用独立设置方式还是综合设置方式，视当时设备的处理能力及工程具体情况而定。但在网络发展初期采用控制层单平面结构的方式时，应要求软交换设备具有支持位置服务器的能力。在两省之间以及和其他运营商互连时，采用分层软交换结构。

控制层设备的软交换设备宜采用大容量、少局所的设置原则。原则上软交换设备应以本地网为单位进行建设，但在建设初期根据业务情况，一个软交换设备可以覆盖多个本地网。原则上，软交换设备应当采取冗余备份机制，以保证网络的安全性、可靠性。软交换设备应设在自然灾害较少、供电可靠、便于维护管理的不同城市或同城市有相当距离的两个建筑物内，采用主备用或两个软交换设备相互备用的工作方式。当软交换系统规模很大时，应考虑设置位置服务器。省间或和其他运营商组网互连时，可考虑采用分层软交换结构。

一个软交换控制设备可以控制多个媒体网关设备，依据《软交换设备总体技术要求》，软交换控制设备的处理能力应为 140 万线（端局）至 300 万线（汇接局），一个电话呼叫控制器的容量应为 10 万线（端局）到 20 万线（汇接局），因此，依据现有本地电话网的规模，一个本地电话网将需要多个软交换控制设备。为此，应合理组织和建设软交换控制设备，使其方便管理、便于维护、节省投资。

第 10 章　SIP

SIP（Session Initiation Protocol）凭借其简单、易于扩展、便于实现等诸多优点越来越得到业界的青睐，它正逐步成为 NGN 和 3G IP 多媒体子系统（IP Mutimedia Subsystem，IMS）域中的重要协议，并且市场上也出现越来越多的支持 SIP 的客户端软件和智能多媒体终端，以及用由 SIP 实现的服务器和软交换设备。从 RFC2543 到 RFC3261 标准的制定，SIP 到目前已基本成熟。

本章通过对 SIP 的背景、功能、网络元素、协议框架、实现机制、以及 SIP 消息的组成等几个方面做全方位的概要性介绍，使读者能够快速全面地掌握 SIP 的精华，对今后从事与 SIP 有关的开发人员、测试人员，以及其他相关工作人员，具有实际指导意义。

10.1　SIP 介绍及协议框架

SIP 是一个比较大的体系，本小节主要介绍 SIP 产生的背景、SIP 中的一些术语、以及 SIP 在整个网络结构中的位置。

10.1.1　SIP 概述

SIP 的目的是用来实现跨越因特网的高级电话业务——IP（网际协议，Internet Protocol）电话。IP 电话正在向传统电话模式公用电话交换网（Public Switched Telephone Network，PSTN）挑战，目前对 IP 电话开放，不同的国家有不同政策，当前在我国 IP 电话没有开放，但 IP 电话以价格低廉为优势，它必将取代传统电话模式。同时 SIP 是下一代网络系列协议中重要成员之一，也是 IP 多媒体子系统的核心协议之一。

SIP 是 IETF（Internet Engineering Task Force，Internet 工程任务组）标准体系中的一部分，它是一个应用层的控制协议，可以用来建立、修改和终止多媒体会话（或者多媒体会议），例如 Internet 电话。SIP 也可以邀请第三方参加已经存在的会话，如在多方会议中可以随时邀请其他的人加入已存在的会议中，也可以在一个已经建立的会话中方便地增加（或者删除）多媒体，如可以随意在一个语音视频通信中增加或撤销视频。SIP 还支持名字映射和重定向服务，这个功能用于支持个人移动业务——用户可以使用一个惟一的外部标志符，且这个外部标志与它们所处的实际网络地点没有关系。SIP 在建立和维持终止多媒体会

话方面，主要支持下面 5 个功能：

1）名字翻译和用户定位：检查终端用户的所在位置，用于通信。无论被叫方在哪里都确保呼叫达到被叫方，执行任何描述信息到定位信息的映射，确保呼叫（会话）的本质细节被支持。

2）用户有效性：检查用户参与会话的意愿程度。呼叫中参与者能够引入其他用户加入呼叫或取消到其他用户的连接。此外，用户可以被转移或置为呼叫保持。

3）用户能力：检查媒体和媒体的参数。它允许与呼叫有关的组（这可以是多方呼叫）在支持的特征上达成一致（注意：不是所有方都能够支持相同级别的特征），例如视频可以或不可以被支持。总之，存在很多需要协商的范围。

4）建立会话："ringing"，建立会话参数在呼叫方和被叫方。

5）会话管理：包括发送和终止会话，修改会话参数，激活服务等。

SIP 不是一个垂直集成的通信系统，把 SIP 叫做一个部件更合适些，它可以作其他 IETF 协议的一个部分，用来构造完整的多媒体通信架构。比如，在一些通信中，可能会包含实时数据传输协议（RTP）（RFC1889）用来传输实时的数据，并且提供 QoS（Quality of Service，QoS）反馈，实时流协议（RTSP，Real Time Streaming Protocol）（RFC 2326）用于控制流媒体的的传输，媒体网关控制协议（RFC 3015）用来控制到公用电话交换网（PSTN）的网关，还有会话描述协议（Session Description Protocol，SDP）（RFC2327）用于描述多媒体会话。因此，SIP 应该和其他的协议一起工作，才能给终端用户提供完整的多种服务。但是 SIP 的基本功能组件并不依赖于这些协议。

虽然 SIP 本身并不提供任何服务，但是 SIP 是通信的基础，在 SIP 建立的这个通信基础上可以用来提供不同的服务。比如，SIP 可以定位用户和传输一个封装好的对象到通话的另一方的当前位置，并且如果通过 SDP 传输会话的描述，对方的用户代理立刻可以得到当前会话的参数；如果用这个传输会话描述来呼叫对方，这样就很容易建立一个基于这个"呼叫 ID"的服务。这个简单的例子说明了，SIP 是作为一个通信的基础，可以在其上提供很多不同的服务。

同时，SIP 也不提供会议控制服务（比如流量控制或者会议系统中的投票功能），并且也没有建议应该怎样来管理会议，但可以通过在 SIP 的基础上建立其他的会议控制协议来建立一个会议。虽然 SIP 可以在不同的网络结构中建立会话，并对会话进行管理，但是 SIP 并不能提供任何形式的网络资源预留管理。

安全对于所提供的服务来说特别重要，要达到理想的安全程度，SIP 提供了一套安全服务，包括防止拒绝服务、认证服务（用户到用户，代理到用户）、完整性保证、加密和隐私服务。

10.1.2 SIP 中的术语

SIP 中有两个要素：SIP 用户代理（User Agent，UA）和 SIP 用户服务器（User Server，US）。用户代理是呼叫的终端系统元素，而用户服务器是处理与多个呼叫相关联信令的网络设备。

用户代理本身又分为：客户机端（User Agent Client，用户代理客户机（UAC））和服务器端（User Agent Server，用户代理服务器（UAS））。用户代理客户机是初始呼叫的主叫端，而用户代理服务器是通话的被呼叫端，也就是应答呼叫端。一个用户代理在发起呼叫的时候，它是用户代理客户机，而当被呼叫时，它是用户代理服务器，这种通信方式继承了互联网的 C/S（Client/Server，客户端/服务器端）模式，允许点到点的呼叫通过客户机-服务器协议来完成。

SIP 服务器元素提供多种类型的服务器。有 3 种服务器形式存在于网络中：SIP 有状态代理服务器、SIP 无状态代理服务器和 SIP 重定向服务器。由于呼叫者未必知道被呼叫方的 IP 地址或主机名，SIP 服务器的主要功能是提供名字解析和用户定位。可以获得的类似 E-mail 形式的地址或与被呼叫方关联的电话号码。使用该信息，呼叫者的用户代理能够通过特定服务器来解析地址信息，这就涉及网络中很多服务器。

SIP 代理服务器接收请求，决定将这些请求传送到何处，并且将它们传送到下一代理服务器（使用下一跳路由原理）。在网络中可以有多跳。SIP 代理服务器有点像网络中的路由器，接收到一个网络包，就要判断往哪里送。

SIP 代理服务器又分为：有状态和无状态代理服务器。有状态和无状态代理服务器的区别是有状态代理服务器记住它接收的入请求，以及回送的响应和它转送的出请求。无状态代理服务器一旦转送请求后就忘记所有的信息。这允许有状态代理服务器生成请求以并行地尝试多个可能的用户位置并且送回最好的响应；无状态代理服务器可能是最快的，并且是 SIP 结构的骨干；有状态代理服务器可能是离用户代理最近的本地设备，它控制用户域，并且是应用服务的主要平台。

重定向服务器接收请求，但不是将这些请求传递给下一服务器而是向呼叫者发送响应消息，把被呼叫用户的地址告诉呼叫者。这使得呼叫者可以利用得到的用户地址，重新发起一个呼叫，直接联系在下一服务器上被呼叫方的地址。

10.1.3 SIP 在网络体系中的位置

计算机网络是一个分层结构体系，有两种流行的分层结构：ISO/OSIRM7 层模型和 TCP/IP 体系结构。

1. ISO/OSIRM7 层模型

ISO（International Organization for Standardization，国际标准化组织）是最早形式化定义连接计算机的通用方法的组织之一。它们的体系结构，称为开放系统互联（OSI）结构，如图 10-1 所示，将网络按功能性划分为 7 层，其中由一个或多个协议实现某层的功能性。按这种观点，图 10-1 中的原理图并不是一个协议图，而是一个协议图的参考模型（Reference Model）。ISO 通常与另外一个标准化组织——国际电信联盟（International Telecommunications Union，ITU）联合，一起发布一系列基于 OSI 体系结构的协议说明。这个系列通常叫做“X 点”系列，因为协议命名为 X. 25、X. 400、X. 500 等。已经有基于这些标准的一些网络，包括公共 X. 25 网络和专用网络如 Tymnet。

从底层开始向上，物理层（Physical Layer）负责通信链路上原始比特的发送，然后数据链路层（Data Link Layer）收集比特流形成一个更大的集合体称为帧（trame）。典型情况下，网络适配器和运行在节点操作系统上的设备驱动程序一起来实现数据链路层。实际传给主机的是帧而不是原始比特。网络层负责分组交换网中节点的路由，在这一层，节点间交换的数据单元通常叫做分组而不是帧，尽管事实上它们是一回事。较低的三层在所有的网络结节上实现，包括网络内部的交换机和网络外部连接的主机。传输层（Transport Layer）实现的是我们提到过的进程到进程的信道。这时，交换的数据单元通常叫做消息而不是分组或帧。传输层和更高层通常只在终端主机上运行，而不在中间交换机或路由器上运行。

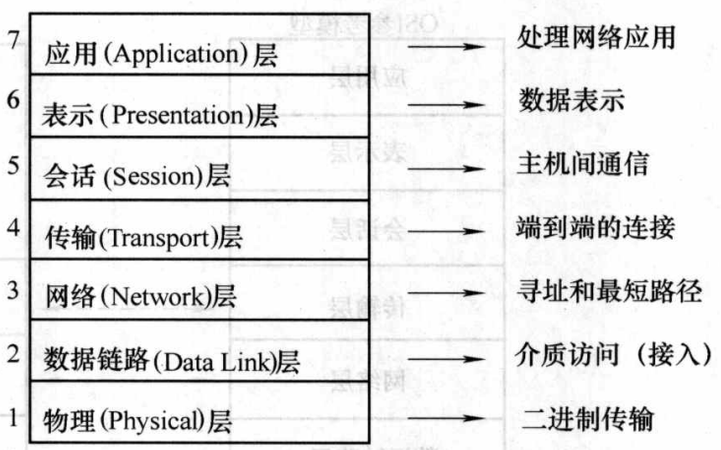


图 10-1　OSI 网络体系结构

上面三层的定义不太统一。自顶向下看，最顶层（第 7 层）是应用层（Application Layer）。应用层协议包括文件传输协议（FTP File Transfer Protocol），它定义一个协议使得文件传输应用可以互操作。接下来，表示层（Presentation Layer）关心在对等实体间交换的数据的格式，例如，一个整数是 16、32 还是 64 位长，最高有效位是最先传还是最后传，或者是一个视频流如何被格式化。最后是会话层（Session Layer），它提供一个名字空间用来将一个应用的各部分不同的传输流联系在一起。例如，在视频会议应用中，它可以同时管理一个音频流与一个视频流。

2. TCP/IP 体系结构

TCP/IP 体系结构，有时也叫做因特网（Internet）体系结构，TCP 和 IP 是它

的两个主要的协议，如图 10-2 所示。因特网体系结构是从早期的分组交换网络（ARPANET）发展起来的。因特网和 ARPANET 都是由美国高级研究规划局（ARPA）创建的，ARPA 是美国安全部的 R&D 基金会之一。在 OSI 体系结构建立之前，因特网和 ARPANET（Advanced Research Projects Agency Network，ARPA 计算机网）就已经存在了，并且来自它们的经验很大程度上影响了 OSI 参考模型。

将 OSI 模型与 TCP/IP 体系结构相比较（见图 10-2），TCP/IP 体系结构常常可用 4 层模型代替。在最低层有多种网络协议。实际中，这些协议由硬件（如网络适配器）和软件（如网络设备驱动程序）共同实现。例如，你会发现以太网或光纤分布式数据接口（FDDI）协议在这一层（实际上这些协议又可以包含几个子层，但是 TCP/IP 体系结构没有这样做）。第二层只有一个网络协议（IP）。这个协议支持多种网络技术互连为一个逻辑网络。第三层包括两个主要的协议：传输控制协议和用户数据报协议。TCP 和 UDP 为应用程序提供可选逻辑信道——TCP 提供可靠的字节流信道，UDP 提供不可靠的数据报传送信道（数据报（datagram）可认为是消息的同义词）。在互联网中，TCP 和 UDP 有时被称作端到端（end-to-end）的协议，但它们还可被认为是传输协议。

在传输层以上运行的应用协议如 FTP、TFTP（普通文件传输协议）、TelNet（远程登录）和 SNMP（Simple Network Management Protocol，简单网络管理协议），使常用的应用可互操作。

图 10-2 TCP/IP 体系结构与 OSI 模型

由于 TCP/IP 体系结构较 OSI 参考模型简单，且容易实现，所以 TCP/IP 体系结构能够得到普遍的应用，目前的 Internet 就是以 TCP/IP 体系结构为基础的网络。SIP 在网络体系结构中处于应用层，它是应用层协议，如图 10-3 所示。

从图 10-3 可以看出，SIP 是一个应用层协议，与其他应用层协议（DNS、

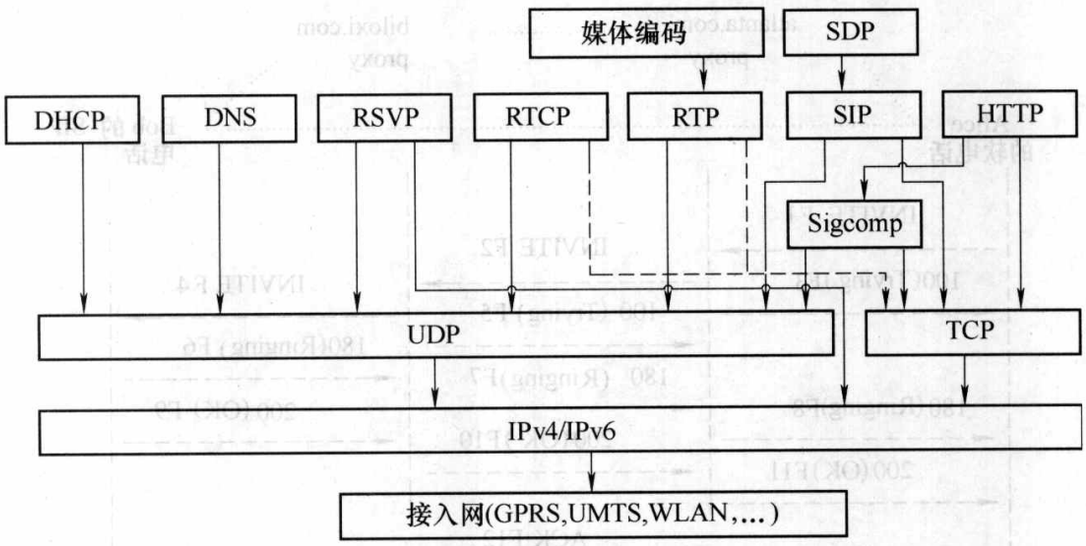


图 10-3　SIP 在网络体系中的位置

HTTP、RTCP 等）一样，可以利用 TCP 和 UDP 来传输 SIP 消息，它与底层的网络层没有直接的联系，IPv4 和 IPv6 对它都是一样。但是要注意的是，在实际的应用中，SIP 在各种消息头域中以及其他地方都会与 IP 地址有关，又由于 IPv4 和 IPv6 在地址格式等方面都有比较大的差异，因此在实现的过程中对 IPv4 网络中的应用和 IPv6 网络中的应用还是有差别的。依据 SIP 标准实现的通信软件，也只是应用层软件。

10.2　一个 SIP 通信的实例

在了解 SIP 的基本情况后，用一个简单的例子来看看 SIP 会话的建立过程。通过这个例子简单介绍 SIP 的功能，这个例子说明了 SIP 的基本功能：用户定位，单方发出通信请求，通过协商会话参数建立会话和拆卸刚才建立的会话。通话建立过程如图 10-4 所示。

图 10-4 表示一个典型的两个用户间（Alice 和 Bob）的 SIP 消息交换过程的例子（每条消息采用字母“F”和一个数字做标记）。在上述例子里，Alice 在她的 PC 上使用一个 SIP 的应用程序（比如说一个软电话），呼叫 Internet 上的另一个 SIP 电话用户 Bob，同时这个例子也说明了两个 SIP 代理服务器是怎样为 Alice 和 Bob 建立会话连接的。这个例子基本上把 SIP 的通信过程都包括了，具有代表性。

Alice 通过 Bob 的 SIP 标志“呼叫” Bob，这个 SIP 标志是统一资源标识(Uniform Resource Identifier，URI)称作 SIP URI，SIP URI 的格式在后面的章节中会介绍。一个 SIP URI 很像一个 E-mail 地址，典型的 SIP URI 包括一个用户名和一个主机名。在图 10-4 中，Bob 的 SIP URI 是 sip：bob @ biloxi. com，

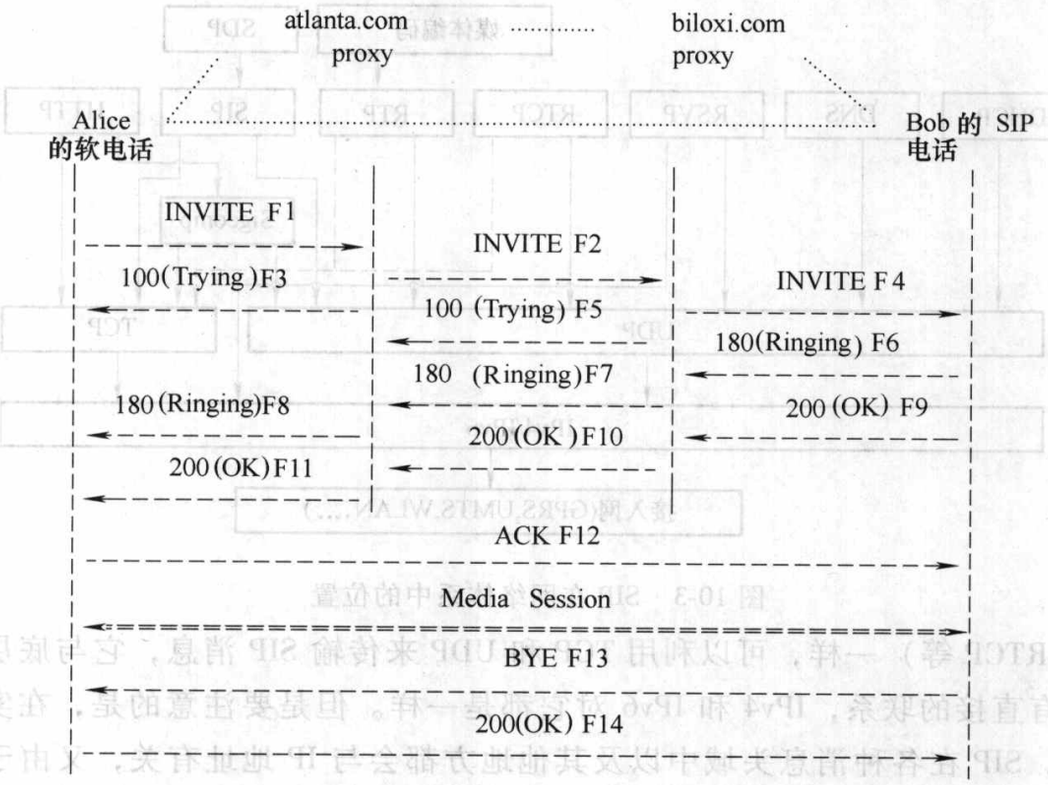


图 10-4 SIP 会话建立过程

biloxi. com 是 Bob 的 SIP 服务提供商。Alice 有一个 SIP URI 是 sip: alice@ atlanta. com，同样 atlanta. com 是 Alice 的 SIP 服务提供商。Alice 可以通过 Bob 的 URI 来实现对 Bob 的呼叫以建立通话。

Alice 的软电话发送一个含有 Bob 的 SIP URI 地址的 INVITE 请求。INVITE 是一个 SIP 请求，表示请求方（Alice）希望服务方（Bob）应答。INVTE 请求包含一系列的头域（Header Fields），这些头域中包含多个属性并且包含了其他用于传输消息的附加信息。一个 INVITE 消息中至少应该有如下的字段：呼叫的唯一标志、目的地址、Alice 的地址、Alice 和 Bob 建立会话的类型，INVITE 请求（图 10-4 中的 F1）可能像下面的形式：

INVITE sip: bob@ biloxi. com SIP/2. 0

Via: SIP/2. 0/UDP pc33. atlanta. com; branch = z9hG4bK776asdhds

Max-Forwards: 70

To: Bob <sip: bob@ biloxi. com >

From: Alice <sip: alice@ atlanta. com >; tag = 1928301774

Call-ID: a84b4c76e66710@ pc33. atlanta. com

CSeq: 314159 INVITE

Contact: <sip: alice@ pc33. atlanta. com >

Content-Type: application/sdp

Content-Length: 142

(Alice's SDP not shown)

下面简单介绍一下 INVITE 消息的一些头域，在后面的章节对这些头域还有具体的介绍：

Via 域：包含了 Alice 接收发送请求的服务器地址（pc33. atlanta. com），同样也包含了一个分支参数来标志 Alice 和这个服务器的会话事务。

To 域：包含了显示姓名（Bob）和一个 SIP 或者 SIPS URI（sip：bob@biloxi. com）请求将首先传到这个 URI 中。显示姓名（Display Names）在 RFC 2822 中描述。

From 域：也同样包含一个显示姓名（Alice）和一个 SIP 或者 SIPS URI（sip：alice @ atlanta. com），这个 URI 用来标志请求的原始发起者。这个域也包含了一个 tag 参数，这个 tag 参数是一个随机字符串（1928301774），是软电话在 URI 上增加的一个随机字符串，用来做标志用途的。

Call-ID：包含一个全局的唯一标志，用来唯一标志这个呼叫，通过随机字符串和软电话的自己名字或者 IP 地址混和产生的。通过 TO tag，FROM tag 和 Call-ID 完整定义了 Alice 和 Bob 之间的端到端的 SIP 关系，并且表示这个是一个对话性质的关系。

CSeq 或者 Command Sequence：包含了一个整数和一个请求名字。这个 Cseq 数字是顺序递增的，每当对话中发起一个新的请求都会引起这个数字的顺序递增。

Contact 域：包含一个 SIP 或者 SIPS URI 用来表示访问 Alice 的直接方式，通常由用户名和一个主机的全名（Fully Qualified Domain Name，FQDN）组成。当 FQDN 作为首选的时候，许多终端用户由于不会由名字登记（而导致不能访问 Alice 的主机），所以 IP 地址是可选的。Via 域告诉大家本请求发送到哪里并且应答到哪里，Contract 域告诉大家将来的请求将发送到哪里。

Max-Forwards：最大转发数量限制了通信中转发的次数。它是由一个整数组成，每转发一次减 1，直到为零，就不再转发。

Content-type：包含了消息正文的描述（消息正文在本范例中没有列出）。

Content-length：包含消息正文的长度（字节数）。

由于 Alice 的软电话（Soft Phone）并不知道 Bob 或者 Bob 的 SIP 服务器 biloxi. com 在哪里，所以 Soft Phone 发送 INVITE 请求到 Alice 的 SIP 服务器 atlanta. com。这个 atlanta. com SIP 服务器已经在 Alice 的 Soft Phone 中配置好了，或者通过 DHCP 获得，是一个默认的代理服务器。atlanta. com SIP 代理服务器接收 SIP 请求并且根据请求转发和应答。在图 10-4 的例子里，atlanta. com 代理服务器接收到 INVITE 请求后，代理服务器转发 INVITE 请求（图 10-4 中 F2），并且回送一个 100（Trying）（图 10-4 中 F3）应答给 Alice（100（Trying）应答表示 IN-

VITE 请求已经收到）。SIP 的应答是通过一个三位数的数字表示的，同样包含 TO、FROM、Call-ID，CSEQ 和在 Via 中的分支参数，这个参数使得 Alice 的 Soft Phone 可以把请求和应答关联起来。atlanta. com 代理服务器收到 INVITE 请求之后，通过 DNS 来找 biloxi. com，然后再把请求消息转发给 biloxi. com 代理服务器，INVITE 请求转发到 biloxi. com 或者能到达 biloxi. com 的代理服务器后，在转发请求之前，atlanta. com 代理服务器会在 Via 头上增加一个一段包含自己地址的值（INVITE 已经包含了 Alice 的地址 Via 域）。biloxi. com 代理服务器收到这个 INVITE 请求并且返回一个 100（Trying）（图 10-4 中 F5）应答给 atlanta. com 代理服务器标志它已经收到这个请求并且正在处理这个请求。这个代理服务器通过查询数据库，通常叫做地址服务，这个服务中包含了 Bob 的当前 IP 地址。biloxi. com 代理服务增加另一段包含自己地址的 Via 头域并且发送它到 Bob 的 SIP 电话（图 10-4 中 F4）。

Bob 的 SIP 电话接收到 INVITE 请求并且提醒 Bob 有一个从 Alice 的呼入，这样 Bob 可以决定是否响应这个呼入。这个意思就是：Bob 的电话响了。Bob 的 SIP 电话发送一个 180（Ringing）（图 10-4 中 F6）响应，这个响应通过两个代理服务器原路返回给 Alice。每一个代理服务器通过 Via 头域决定该把这个应答发送给哪里，并且在发送之前把自己的地址从头上拿走。虽然 DNS 和定位服务在路由最初的 INVITE 请求，180（Ringing）响应可以简单返回给发起者而不需要查找发起者在哪里，并且不需要在代理服务器保留状态，同时，每一个转发 INVITE 的代理也可以得到 INVITE 的每一个应答，这样的特性也非常有用。

当 Alice 的 Soft Phone 收到 180（Ringing）应答（图 10-4 中 F8）的时候，它提示 Alice，可能是通过一个回铃音，或者屏幕上的一个消息提示。

在图 10-4 这个例子中，Bob 决定响应这个呼叫。当他拿起电话，他的 SIP 电话发送 200（OK）回应（图 10-4 中 F9）给发送者，表示这个电话已经接起来了。这个 200（OK）包含了一个消息体，这个消息体包含 SDP 媒体描述，这个媒体描述包含 Bob 希望和 Alice 建立何种媒体连接。同样，SDP 消息也是两段交换：Alice 发送一个给 Bob，Bob 发送一个响应给 Alice。这个两段的交换提供基本的兼容性协商，并且基于简单的 SDP 提出/应答交换模型。如果 Bob 不想响应这个呼叫或者正在响应别的呼叫，一个错误的响应会代替正常的 200（OK）回送出去，这样，就不会有连接建立。Bob 发出的 200（OK）：

SIP/2. 0 200 OK

Via：SIP/2. 0/UDP erver10. biloxi. com；branch = z9hG4bKnashds8；received = 192. 0. 2. 3

Via：SIP/2. 0/UDPbigbox3. site3. atlanta. com；branch = z9hG4bK77ef4c2312983. 1；received = 192. 0. 2. 2

Via: SIP/2.0/UDP pc33.atlanta.com; branch = z9hG4bK776asdhds ; received = 192.0.2.1

To: Bob <sip: bob@biloxi.com> ; tag = a6c85cf

From: Alice <sip: alice@atlanta.com> ; tag = 1928301774

Call-ID: a84b4c76e66710@pc33.atlanta.com

CSeq: 314159 INVITE

Contact: <sip: bob@192.0.2.4>

Content-Type: application/sdp

Content-Length: 131

(Bob's SDP not shown)

应答的第一行包含了应答代码（200）和原因（OK）。剩下的行包含了其他的头域，它们分别是 Via，TO，FROM，CALL-ID，Cseq 包头域是从 INVITE 请求包中直接复制过来的。（有三个 Via 域值--一个是 Alice SIP 电话增加的，一个是 atlanta.com 代理加的，一个是 biloxi.com 代理加的）。Bob 的 SIP 电话增加了一个 tag 参数，这个 tag 参数会被参与对话的各方所使用，并且在以后的对话中被使用。Contract 域包含了一个能直接联系到 Bob 的 URI。Content-type 和 Content-Length 域包含了消息体（没有在例子中体现），这个消息体是 Bob 的 SDP 媒体信息。

除了 DNS 和位置服务之外，代理服务器可以自主决定路由，也就是说自己决定应该向哪里转发请求。在这个例子中，200（OK）应答通过两个代理并且发送到 Alice 的 SIP 电话上，Alice 的 SIP 电话收到这个应答，停止振铃，并且标志 Bob 已经接听电话。最后，Alice 的电话发送一个确认消息 ACK（图 10-4 中 F12），到 Bob 的 SIP 电话来确认接收到了这个最后的 200（OK）应答。在这个例子中，ACK 信号是直接由 Alice 的软电话发送到 Bob 的 SIP 电话上，跨过了两个代理服务器。这是因为两个端点（Alice 和 Bob）通过 INVITE/200（OK）的请求应答包中的 Contact 包头域都知道互相之间的地址了，这个地址是最开始发起 INVITE 请求的时候所不知道的。所以，不需要两个代理服务器再查找对方的地址了，所以代理服务器不参与接下来的通话流了。这就完成了一个完整的使用 INVITE/200/ACK 三方握手来建立 SIP 会话的过程。

现在，Alice 和 Bob 的媒体会话开始了，他们通过发送刚才建立会话所交换的 SDP 包中约定的互相明白的媒体包来进行会话。一般情况下，端到端的媒体包和 SIP 信号控制包都是 UDP 包，所以可能通过不同的通信路径发送到对方。

在通话结束的时候，Bob 首先断开（挂机 Hangs Up），并且发送一个 BYE 的消息（图 10-4 中 F13）。这个 BYE 的消息将直接送到 Alice 的软电话，同样是跳过代理的。Alice 通过发送 200（OK）（图 10-4 中 F14）应答来确认收到了这

个 BYE 消息，这个消息终止了会话并且应答了 BYE 的请求。ACK 在这里不需要发送一个 ACK 信号，只在响应一个 INVITE 的响应的时候被发送，这样通话就结束了。

上述详细介绍了一个 SIP 会话的全部过程，其中也提到一些消息，这些消息的格式以及在每个通信节点中的处理过程在后面的章节有更详细的介绍，图 10-4 的例子主要是为了简单说明 SIP 通信的全过程。

10.3 SIP 消息结构及基本消息体

SIP 是一个基于文本的协议，使用 UTF-8 字符集（RFC2279）。SIP 把消息分为请求消息和应答消息两种，一个 SIP 消息既可以是一个从客户端到服务器端的请求，也可以是一个从服务器端到客户端的一个应答。虽然请求消息和应答消息在字符集上和语法细节上有所不同，但无论是请求还是应答消息都基于文本的，它们都遵循 RFC2822 规定的格式，但是 SIP 允许包头域不是标准的 RFC2822 包头域。

请求消息和应答消息这两种消息类型都由一个起始行、一个或者多个包头域和一个可选的消息体组成。起始行又分为请求行和状态行。

```
generic-message  = start-line
                   * message-header
                   CRLF
                   [ message-body ]
start-ling       = Request-Line/Status-Line
```

中文意思是 一般消息 = 起始行

```
                   * 消息包头
                   CRLF
                   [ 消息正文 ]
起始行           = 请求行/状态行
```

起始行、每一个包头行，空行、都必须由回车换行组成（CRLF），即使消息值没有，也必须有一个空行跟随。

除了在字符集上的区别以外，很多 SIP 的消息和包头域的格式都同 HTTP/1.1 一样，但是 SIP 并非 HTTP 的超集或者扩展。我们在这里就不重复它的语法和语义了，具体描述参见 HTTP/1.1 规范（RFC2616）。

10.3.1 请求消息

SIP 请求是根据起始行中的 Request-Line 来区分的。一个 Request-line 包含方

法名，请求 URI 和用单个空格（SP）间隔开的协议版本号。

请求行（Request-Line）由回车换行符（CRLF）结束。回车换行符只用作行结束标志，在其他地方不允许 CR 或者 LF 出现。在此规定下，不允许在其他地方出现空白行。

Request-Line = Method SP Request-URI SP SIP-VERSION CRLF

请求行　　= 方法 SP 请求 URI SP SIP 协议版本号 回车换行

1. 方法

在 RFC3261 规范中规定了 6 种方法，分别是：REGISTER 用于登记联系信息，INVITE，ACK，CANCEL 用于建立会话，BYE 用于结束会话，OPTIONS 用于查询服务器负载。SIP 的扩展协议中追加其他方法，如为支持移动特性添加了 re-INVITE。但是在 RFC3261 标准中只要上述 6 种。

2. Request-URI

Request-URI 是一个 SIP 或者 SIPS URI，也可以是一个通用的 URI（RFC 2396）。它标志了这个请求所经过的用户或者服务的地址。使用的时候 Request-URI 也有一定的规范，它不允许包含空白字符或者控制字符，并且禁止用“ < > ”括上。

SIP 元素可以支持除了 SIP 或者 SIPS 之外所规定的 Request-URIs，比如“tel” URI 模式（RFC 2806）。SIP 元素可以用它们自己的机制来转换 non-SIP URIs 到 SIP URI、SIPS URI 或者其他什么格式的 URI。

SIP 或者 SIPS 的 URI 用来标记一个通信用的资源，就像其他所有的 URI 一样，SIP 或 SIPS URI 可以在网页上，也可以在 Email 消息里等。这些 URI 里边包含了足够的信息来发起和维持到这个资源的一个通信会话。SIPS URI 定义了对资源的访问是安全的，这表明在 UAC 和这个资源的主机之间的通信是基于 TLS 的，从资源的主机到用户之间的通信是加密安全的，也就是说安全机制是依赖于主机的实现的。任何用 SIP URI 描述的资源，只要通过加密的形式进行通信，都可以通过简单改变一下资源描述符就可以“升级”成为一个 SIPS URI。

“sip:”和“sips:”描述符是遵循 RFC2396 的规范定义的。它们使用类似 mail to URL 的格式定义，允许有 SIP 请求消息头域字段和 SIP 消息体的规范。一个 SIP URI 的通常格式是这样的：

sip: user: password@ host: port; uri-parameters? headers

SIP URI 和 SIPS URI 的格式是相同的，只是 SIPS URI 用“sips”来代替 sip。上述格式的符号，具有下列意义：

user: 是在主机中注册的有效地址资源。“主机”（host）这里通常指的是一个域名。URI 中的有关用户信息的内容包含在这个用户域，后面跟随用户口令域和一个“@”符号。其中 URI 的用户信息部分是可选的，可有可无；当目的主

机没有用户信息或者主机本身就是资源的目的地，那么这个 URI 的用户信息部分就是可以没有的。如果在 SIP 或者 SIPS URI 中有@，那么用户部分不能为空。如果主机部分可以处理电话号码地址，比如说是一个 internet 电话网关，那么根据 RFC2806 定义的电话号码域，应当出现在用户信息部分。

password：password 字段是和用户相关的。SIP 或者 SIPS URI 语法允许增加 password 这个字段，这种用法在 RFC3261 中是不推荐的，因为把身份认证信息放在用明码表示的地方（比如 URI）会带来很大的安全风险。比如，通信层在这个字段用了一个 PIN 码，那么就会暴露这个 PIN 码而带来安全隐患。

注意密码字段只是一个用户信息扩展的一部分，实现上并没有标记一个特别的密码部分，可以简单的把“user：password”当作一个简单的用户串来对待。

host：主机提供了 SIP 资源。host 部分包含了一个完整的主机名字或者是 IPv4/IPv6 的地址。RFC3261 中建议使用完整格式的主机名字。

port：端口号是请求将被送出的端口。

URI 参数：请求将使用这个 URI 来构造。

URI 参数在 host 和 port 部件之后增加，且用分号分开。

URI 参数有如下格式：

参数名'='参数值

虽然在同一个 URI 中允许有任意多个 URI 的参数，但是同一个参数名只能出现 1 次。这种扩展机制包括了 transport、maddr、ttl、user、method 和 lr 等参数。其中 transport 参数决定发送 SIP 消息的通信机制。SIP 可以使用任何网络通信协议，如 UDP（RFC 768）、TCP（RFC 761）和 SCTP（RFC2960）；对于一个 SIPS URI，transport 参数必须指向一个可靠的通信协议。maddr 参数指明了联系这个用户的服务器的地址，它会覆盖在 host 域中的地址，当给定了一个 maddr 参数，URI 中的 port 和 transport 部件将会在 maddr 中指出。maddr 字段用作简单的去掉源路由的方法来使用的，它允许一个 URI 指定一个必须经过的 proxy 来到达目的地，RFC3261 中不要求把 maddr 参数用于这个目的，实现上应当使用本文中描述的路由机制，如果有需要，则建立一个 pre-existing（预先设置的）路由集合，提供一个完整的 URI 来描述需要经过的节点。ttl 参数决定了 UDP 多点报文的生存周期，并且只能用于当 maddr 是一个多点地址并且通信协议是 UDP 的情况。例如，指定一个到 alice @ atlanta. com 的呼叫，使用多点广播到 239. 255. 255. 1，并且 ttl = 15，那么应该使用下边的一个 URI：

sip：alice@ atlanta. com；maddr = 239. 255. 255. 1；ttl = 15

从 URI 中构建 SIP 请求所需要的 method 域，可以由 method 参数指定。如果指定了 lr 参数，就标志着这个资源的拥有者是根据 RFC3261 标准来实现的路由机制。这个参数将用于 proxy 放在 Record-Route 头域的 URI 中，也可以出现在

pre-existing 的路由集合中。lr 参数是为和 RFC2543 定义中的严格路由机制向后兼容所规定的，并且 rfc2543 bis 改变为 bis-05。如果一个元素准备发送一个没有包含 lr 参数的 URI 请求，那么我们可以假定这个请求的接受方是根据严格路由的规范实现的，并且会重新规范化这个消息来保护在 Request-URI 中的内容。由于 URI 参数机制是可以扩展的，SIP 元素会跳过那些不认识的 URI 参数。

Headers：头域部分包含在请求 URI 中。在 SIP 请求的头域中，可以在 URI 中用“?”来标识。头域名（hname）和头域值（hvalue）由“=”相隔。特定的头域名“body”的头域值就是 SIP 请求的消息体。

3. SIP-Version

请求和应答消息都包含当前使用的 SIP 版本，这个遵循［H3.1］（类似 HTTP 用 SIP 替代，用 SIP/2.0 替代 HTTP/1.1）中关于版本的规定，版本依赖，升级版本号。一个应用，发出的 SIP 消息一定包含了 SIP-Version“SIP/2.0”。这个 SIP 版本串是大小写不敏感的，但是在实现中必须发送大写。不像 HTTP/1.1，SIP 把版本号当作一个文本串处理。在实现上，这个应该不会有区别。

10.3.2 应答消息

SIP 应答和 SIP 请求的区别在于在 Start-Line 中是否包含一个 Status-Line。一个 Status-Line 由协议的版本、数字表达的 Status-Code（状态码）和文本形式的其他短语组成，且每一个元素之间用一个单个空格分开，并且最后用作结束标志 CR/LF 结尾。

status-line = SIP-VERSION SP STATUS-CODE SP Reasong-Phrase CRLF

Status-Code 是一个 3 位的数字结果码，用来标志处理请求的一个结果。Reason-Phrase 是一个简短的对状态码的简要说明。Status-Code 是给自动处理机使用的，而是 Reason-Phrase 是用来给用户看得。一个客户端并不要求一定要显示或者解释这个 Reason-Phrase。RFC3261 标准建议使用 Reason-Phrase 字段，但在实现中可以选择其他文本，比如用请求包头中指定的合适语言来显示。

Status-Code 的第一个数字表示了应答的类型。接下来两个数字并不作分类使用。基于这个原因，任何 Status-Code 在 100 ~ 199 的范围里可以统称位“1xx 应答”，类似的，在 200 ~ 299 的范围里可以统称位“2xx 应答”，依此类推。SIP/2.0 允许 6 类应答：

1xx：临时应答：请求已经接收，正在处理这个请求。

2xx：成功处理：请求已经成功接收，并且正确处理了这个请求。

3xx：重定向：还需要附加的操作才能完成这个请求，本请求转发到其他的服务器上处理。

4xx：客户端错误：请求包含错误的格式或者不能在这个服务器上完成。

5xx：服务器错误：服务器不能正确的处理这个显然合法的请求。

6xx：全局错误：请求不能被任何服务器处理。

下面简单介绍一下各类应答码，并给出常见的状态码。

1. 1xx Provisional

1xx 类应答只是临时性的应答，也就是消息性质的应答，标志了对方服务器正在处理请求，但是还没有决定最后的应答。只有服务器处理请求需要花 200ms 以上时，才发送一个 1xx 应答消息，对 200ms 内能处理好的请求消息，就不再发送 1xx 应答消息。需要注意的是：1xx 应答是不可靠传输的，即用 UDP 包传送，它不要求客户端传送回一个 ACK 确认消息。1xx 类应答消息可以包含消息体和会话描述。下面简单介绍几种常见的临时应答消息：

（1）100（Trying）

这个应答消息表示下一个节点的服务器已经接收到了这个请求，但还没有执行与这个请求相应处理（比如正在打开数据库的时候）。100（Trying）消息和其他临时应答一样，是为了防止 UAC 重新传送 INVITE 请求，以免产生网络拥挤。100（Trying）应答和其他临时应答不同的是它永远不会被有状态代理转发到上行流中，换句话说，有状态代理不会把下游的 100（Trying）消息转发给上游服务器或者客户端。

（2）180（Ringing）

铃声提示用户已经收到 INVITE 请求。这个应答应当初始化为一个本地回铃。

（3）181（Call is Being Forwarded，呼叫被转发）

服务器可以用这个应答代码来表示呼叫正在转发到其他的一些目的地。

（4）182（Queued）

当被叫方暂时不能接收呼叫的时候，服务器并不是拒绝新来的呼叫，而是将呼叫排队等候，这种情况下就应当发出一个 182 应答。当被叫方一旦可以接收呼叫后，它将回复一个适当的应答消息，并可以在 Reason-Phrase 字段中，给出有关这个呼叫状态的处理信息，比如："有 5 个呼叫正在等候，大概需要等待 15min"。服务器可以多次发送 182（Queued）应答消息，告诉呼叫方排队的当前情况（比如排队靠前了等）。

（5）183（Session Progress）

183（会话进度）应答消息用于提示建立对话的进度信息。Reason-Phrase（表达原因的句子）、头域或者消息体可以用于提示呼叫进度的更消息的信息。

2. 2xx Successful

2xx 响应消息使用最多的是"200（OK）"，该响应表示请求成功，但与响应一起返回的信息还要取决于请求中的方法。

3. 3xx Redirection

3xx 系列的应答是用于提示用户的新位置信息，或者满足呼叫的其他服务器。

（1）300（Multiple Choices）

请求中的地址可以解析为几选项，每个选项都有自己的地址，用户或者 UA 可以选择合适的通信端点，并且将该请求重定向到这个位置。

请求中的消息体可能包含一个资源特征和位置的列表，用户或者 UA 可以从中选择一个 Accept 头字段中所允许的最合适的资源特征和位置，但该消息体不定义 MIME 类型。

这些选项可作为 Contact 字段列出。SIP 响应可包含几个 Contact 字段或者在一个 Contact 字段中有多个地址。UA 可以使用 Contact 字段中的值自动重定向，也可以要求用户对选项进行确认。自动选择的标准在 RFC3261 中没有定义。如果被叫可以通过多个不同位置到达，且服务器不能代理该请求，则可以使用该状态码。

（2）301（Moved Permently）

当用户已经不在 Request-URI 头字段的地址中时，请求发起用户就会重发请求至 Contact 头字段中的新地址。请求方同时更新本地姓名地址簿和用户位置并将将来的请求重定向到新地址。

（3）302（Moved Temporarily）

请求发起方向 Contact 头字段中的新地址重发的新请求中的 Request-URI 是响应中 Contact 头字段里的值。Contact 头字段中 URI 的有效期可以由 Expires 头字段或者 Contact 头字段中的 expires 参数来定义。代理服务器和 UA 可以在有效期内使用 URI。如果没有明确的定义有效期，则该地址仅可有效递归一次，而以后的事务就不可以再使用该 URI。

如果用户向 Contact 头字段中的 URI 发送请求失败，则应向重定向请求中的 URI 尝试发送请求。有效期过后该 URI 就不再使用，这时可以使用另一个新的暂时性的 URI。

（4）305（Use Proxy）

用户必须通过 Contact 头字段中指定的代理服务器接入到请求的资源，Contact 头字段指定代理服务器的 URI。接收者将通过代理服务器中继该请求，该响应只可由 UAS 发出。

（5）380（Alternative Service）

当呼叫不成功但是有另外可供选择的服务时，使用该响应消息。在响应的消息体中描述了这种另外可以选择的服务。RFC3261 没有对 380 响应消息的消息体的格式进行定义。

4. 4xx Request Failure

此类响应由服务器发出表明请求失败。客户机不应（例如增加合适的授权）将原请求不加修改并重新发送给原服务器，但将原请求发向不同的服务器也可能成功。4xx 响应消息在实际应用中最常见，主要有下面几种常见响应消息：

（1）400（Bad Request）

该响应表示请求由于语法错误而不能被理解。响应的原因短语中应详细指出语法错误。

（2）401（Unauthorized）

该响应表示请求消息需要用户鉴权。该响应由 UAS 和注册服务器发起。而后面要提到的 407 响应消息是（代理服务器要求鉴权）由代理服务器发起。

（3）402（Payment Required）

该响应消息在 RFC3261 种作为保留，供将来使用。

（4）403（Forbidden）

该响应表示服务器能理解，但是拒绝执行请求消息，即使该请求已经鉴权也不能进行中继。

（5）404（Not Found）

该响应表示服务器可以确定用户不在 Request-URI 头字段指定的域中。同时，如果 Request-URI 头字段所指定的域（Domain）和接收这个请求的 Domain 不匹配的情况下，也会产生这个应答消息。

（6）405（Method Not Allowed）

服务器支持 Request-Line 中的方法，但是对于这个 Request-URI 中的地址来说，不允许应用这个方法时，应回应 405 应答消息，同时在应答中必须包括一个 Allow 头字段来列举指定地址上所允许的方法。

（7）406（Not Acceptable）

该响应表示根据请求的 Accept 头字段，该请求所指定的资源生成响应的消息体中包含的某些内容特性是不被接受的。

（8）407（Proxy Authentication Required）

该响应类似于 401（未鉴权），不同的是它指定客户机必须首先向代理服务器鉴权自己。该状态码可用于接入到通信信道中。

（9）408（Request Timeout）

该响应表示服务器不能在允许的一段时间内，产生一个终结应答。例如当它不能及时确定用户的位置时，客户端收到该响应后，可以不加修改便重发原请求。

（10）410（Gone）

该响应表示服务器中被请求的资源不可用，并且服务器不知道转发地址，

同时这种情况是永久性的。如果服务器不知道这种情况是否为永久性的，此时则应该使用 404（未找到）状态码。

（11）413（Request Entity Too Large）

该响应表示，如果请求的消息体超出服务器能够处理的范围，服务器将拒绝处理该请求。服务器可以关闭此连接以防客户端不断发送同一个请求。但是，如果这种情况是暂时的，服务器应在响应中加入一个 Retry-After 头字段用来指定多久以后客户机可以重发该请求。

（12）414（Request-URI Too Long）

该响应表示 Request-URI 超出服务器能够处理的范围，服务器拒绝处理该请求。

（13）415（Unsupported Media Type）

该响应表示服务器不支持某请求方法的消息体格式而拒绝处理该请求。根据具体内容的不同服务器必须用响应 Accept、Accept-Encoding 或 Accept-Language 头字段返回服务器可以接收的格式列表。关于 UAC 如何处理该响应在后面的章节会介绍。

（14）416（Unsupported URI Scheme）

该响应表示由于服务器不理解 URI 的方案而不能处理该请求。关于客户端如何处理该响应也会在后面的章节介绍

（15）420（Bad Extension）

该响应表示服务器不理解 Proxy-Require 或 Require 头字段中协议的扩展规定。服务器必须在响应中的 Unsupported 字段中包含一个它不支持的扩展的列表。关于 UAC 如何处理该响应在后面的章节会介绍。

（16）421（Extension Required）

该响应表示 UAS 需要某个特定的扩展才能处理该请求，但是这种扩展没有列在请求中的 Supported 头字段中。同时该响应必须包含一个 Require 头字段列举所需要的扩展。除非 UAS 不能向客户提供任何其他所需的业务否则不应该使用该响应。如果 Supported 字段中没有所需的扩展，服务器只能用客户端所支持的扩展规定对该请求进行 SIP 的基本处理。

（17）423（Interval Too Brief）

该响应表示由于请求更新资源的间隔时间太短，服务器拒绝该请求。在注册请求的 Contact 字段中定义的有效期太短，注册服务器可以使用该响应拒绝请求。关于响应的使用以及相关的 Min-Expires 字段定义将在后面的章节介绍。

（18）480（Temporarily Unavailable）

该响应表示请求成功到达被叫方的终端系统，但是被叫方当前不可用（例如，没有登录，或者登录了但是状态是不能通信，或者有“请勿打扰”的标

记)。应答要在响应的 Retry-After 字段中指定一个合适的呼叫时间，同时在原因短语部分，应该给出一个详细的原因指明为什么被叫方不可用。这个值可以由 UA 来设置。如果重定向服务器或者代理服务器知道 Request-URI 中指定的用户但是目前并没有其有效位置，就可以返回该状态码。状态码 486 (Busy Here) 可以用来更精确的表示本请求失败的特定原因。

(19) 481 (Call/Transaction Does Not Exist)

该响应表示 UAS 收到的请求与现有的对话或者事务没有相对应的。

(20) 482 (Loop Detected)

该响应表示服务器检测到有环路，也就是请求不断在几个代理服务器间循环转发，永远达不到目的地。

(21) 483 (Too Many Hops)

该响应表示，服务器收到的请求中的 Max-Forwards 值为零。

(22) 484 (Address Incomplete)

该响应表示，服务器收到的 Request-URI 不完整并应在原因短语中提供附加信息。该状态码允许异步拨号 (overlapped dialing)。当用户使用异步拨号的方式时，客户端并不知道拨号字串的长度。因此它发送的字符串比实际的长，并提示用户输入更多的数字。直到不再收到 484 状态码为止。

(23) 485 (Ambiguous)

该响应表示请求中的 Request-URI 不明确。由于显示替代的 Request-URI 可能会破坏用户或者组织的保密性，该响应中的 Contact 字段中可以包含另外一个明确的地址。这种情况下，服务器必须可以做出 404 (未找到) 响应，或者服务器能够禁止列举可能的 URI 选项。

例子如下：

sip: lee@ example. com:

SIP/2. 0 485 Ambiguous

Contact: Carol Lee <sip: carol. lee@ example. com >

Contact: Ping Lee <sip: p. lee@ example. com >

Contact: Lee M. Foote <sips: lee. foote@ example. com >

有些电子信箱以及语音信箱系统可以提供这种功能。该状态码与 3xx 状态码语义不同，对于 300 响应，假设可以通过所提供的选项到达同一个人或者服务。自动化的选择或者连续的查找只对 3xx 有意义，而使用 485 (Ambiguous) 响应时则需要用户干涉。

(24) 486 (Busy Here)

该响应表示已经成功的和被叫终端连接，但是被叫目前不能在该终端系统执行呼叫，响应中的 Retry-After 字段可以指定一个合适的呼叫时间。该用户在

其他地方可用。如果客户机知道没有别的终端系统可以接受本次呼叫那么应该使用 600（忙）状态码。

（25）487（Request Terminated）

该响应表示，请求被 BYE 或者 CANCEL。CANCEL 请求不可以返回该响应。

（26）488（Not Acceptable Here）

该响应与 606 响应的含义相同，但是仅指 Request-URI 中指定的资源，如果在别处，该请求可能成功。

该响应中可能存在包含媒体能力描述的消息体，该消息体格式根据 INVITE 请求中的 Accept 字段（如果不存在就是 application/sdp）规定。

（27）491 Request Pending

该响应表示 UAS 收到请求但是在同一个对话中该 UAS 还有一个等待处理的请求。

（28）493 Undecipherable

该响应表示 UAS 收到的请求包含一个加密的 MIME 消息体而接收方没有合适的解码密钥。此响应可以只包含一个消息体，该消息体包含一个公共密钥用来加密发送给 UA 的 MIME 消息体。

5. 5xx Server Failure

该响应表示服务器内部出错导致失败。

（1）500（Server Internal Error）

该响应表示服务器遇到意外的情况使它不能执行该请求。客户端可以显示这种特定的出错情况，并且可以几秒钟后重发该请求。如果情况是暂时的，服务器可以在 Retry-After 字段中指定多久之后客户机可以重发该请求。

（2）501（Not Implemented）

该响应表示服务器不支持实现该请求所需要的功能。如果 UAS 无法识别该请求的方法并且不支持该方法，就可以发送该响应。如果为代理服务器，它转发请求时都不考虑请求的方法。但如果服务器识别了请求中的方法但是并不支持该方法应该发送 405（方法不允许）响应

（3）502（Bad Gateway）

当服务器作为网关或者代理服务器时，需要接入到某下行服务器来完成请求，该下行服务器发出该响应表示其为无效网关。

（4）503（Service Unavailable）

该响应表示由于服务器过载或者正在维护而导致服务器暂时不能处理该请求。服务器可以在 Retry-After 中指明何时可重发该请求，如果没有 Retry-After 则客户端必须按照收到的是 500（服务器内部错误）响应处理。

代理服务器或 UAC 收到一个该响应之后应该将原请求转发到替代的服务器

上。如果响应中存在 Retry-After 字段，则在该字段定义的时间之内该客户机不能再向原来的服务器发送任何请求。服务器也可以不必发送该响应，而直接拒绝连接或者丢弃原请求。

(5) 504 (Server Time-out)

该响应表示服务器接入到一个外部服务器来处理请求，但是没有及时收到该外部服务器的响应。如果在 Expires 字段中规定的时间之内没有收到上行服务器发来的响应，就应该使用 408 (Request Timeout) 响应。

(6) 505 (Version Not Supported)

该响应表示服务器不支持请求中的协议的版本。

(7) 513 (Message Too Large)

该响应表示由于消息体的长度超过服务器的处理能力限制，服务器不能处理该请求。

6. 6xx Global Failures

该响应表示服务器对于某一特定用户的确定的信息。

(1) 600 (Busy Everywhere)

该响应表示与被叫终端连接成功，但是被叫因为忙而不能接收呼叫。可以在该响应中的 Retry-After 字段指定过多久之后可以重新呼叫。如果被叫不希望给出拒绝本次呼叫的原因则应该使用 603 (拒绝)。该响应只用于客户端知道没有别的终端可以接受该请求的情况，否则应该返回 486 响应。

(2) 603 Decline

该响应表示与被叫已经成功连接，被叫用户明确表示不能参与此呼叫。该响应中的 Retry-After 字段可以指定过多久之后可以重新呼叫。只有客户端知道没有别的终端可以接收该请求才可以使用该响应。

(3) 604 (Does Not Exist Anywhere)

该响应通知服务器 Request-URI 中的用户根本不存在。

(4) 606 (无法接受)

该响应表示与用户代理已经成功连接，但是会话描述中如请求的媒体、带宽或者地址形式等都不可接受。此响应可以包含一个 Warning 头字段来详细说明不支持该会话描述的原因。在该响应中可以有一个包含媒体能力描述的消息体，它的格式依据 INVITE 请求中的 Accept 头字段，如果没有该字段，则依据 application/sdp。

通信中不希望有频繁的协商，但如果一个新用户被邀请参加原有的一个会议根据邀请发起者是否发送 606 响应来决定是否需要协商。此响应只用于客户机知道没有别的终端可以响应该请求。

以上给出各类响应消息，在 RFC3261 标准中都有定义，其他没有给出的应

答消息编码，是目前还没有明确使用的，它们可以在以后 SIP 的扩展标准中使用，代表特定的应答消息。

10.3.3　头字段

SIP 头字段的语法和语义定义与 HTTP 头字段定义基本相同，有关 HTTP 头字段的定义和 SIP 头字段多行扩展的定义规则参见 RFC 2616 。RFC 2616 中定义的多行扩展可使用隐含的空格和折叠字符，而本规范定义的多行扩展规则只能使用显式空格和折叠字符，且空格和折叠字符作为消息的组成部分。

同样，RFC 2616 也定义了将具有多个参数值的同一字段扩展为具有相同字段名称的多个字段行的规则。该规则同样适用于本协议，但具体应用时规则会有所不同。实际上，任何 SIP 的包头语法都是基于如下范式的：

header = “header-name” HCOLON header-value * (COMMA header-value)

如上所示，SIP 头字段允许一个头字段可以定义多个参数值，且多个参数值之间用“,”字符进行分隔。当属性值不为“*”时，Contact 头字段允许属性值之间用“,”字符进行分隔。

1. SIP 头字段格式

SIP 消息的头字段格式遵循 RFC2822 所定义的通用头字段格式定义规则。头字段格式如下所示，头字段名和字段值之间用字符“:”进行分隔。

field-name: field-value

RFC3261 标准中规定，消息头字段中允许出现多个空格字符。但是，在具体实现过程中，RFC3261 标准建议字段头部右侧，即字段名和“:”字符之间应避免出现空格字符，而字段头部左侧，即字段值和“:”之间用一个空格字符进行分隔，如下所示：

field-name: field-value

头字段部分可以通过增加多个空白字符或者 TAB 字符而扩展成多行，RFC3261 标准规定扩展行首位的多个空白字符以及分行字符都应作为一个空白字符对待。依照以上规则，以下两个头字段是相同的。

Subject: I know you're there, pick up the phone and talk to me!

Subject: I know you're there,
pick up the phone
and talk to me!

消息头字段内不同头字段名的顺序可任意排列，但是 RFC3261 标准规范建议与代理服务器处理相关的字段名（如 Via、Route、Record-Route、Proxy-Require、Max-Forwards、Proxy-Authorization 等）应尽可能排在消息头字段的前几位以加快代理服务器对消息的处理速度。相同头字段名的排列顺序也非常重要，

RFC3261 标准规范规定，当且仅当字段值为多个字段值列表且字段值列表遵循上面所定义的语法规则（即多个字段值之间用“,”分割）时，则一个消息内允许同时存在多个相同的字段名行。同理，多个相同字段名的字段行可以组成一个单一的字段行，其字段值为一个用“,”字符分隔的值列表。WWW-Authenticate、Authorization、Proxy-Authenticate 和 Proxy-Authorization 字段不能遵循以上规则。由于以上字段值不允许出现列表值，因此当消息中出现多个以上字段行时，多个字段行不能组成一个字段行。

具体实现过程中，消息接收实体应按照同样的规则处理字段名相同的多个字段行或一个包含字段列表的组合字段行。按照以上规则，以下两个字段行是等价的。

1） Route：<sip：alice@atlanta.com>

Subject：Lunch

Route：<sip：bob@biloxi.com>

Route：<sip：carol@chicago.com>

2） Route：<sip：alice@atlanta.com>，<sip：bob@biloxi.com>

Route：<sip：carol@chicago.com>

Subject：Lunch

3） Subject：Lunch

Route：<sip：alice@atlanta.com>，<sip：bob@biloxi.com>，

<sip：carol@chicago.com>

以下几个例子符合消息定义规则，但彼此并不等价。

1） Route：<sip：alice@atlanta.com>

Route：<sip：bob@biloxi.com>

Route：<sip：carol@chicago.com>

2） Route：<sip：bob@biloxi.com>

Route：<sip：alice@atlanta.com>

Route：<sip：carol@chicago.com>

3） Route：<sip：alice@atlanta.com>，<sip：carol@chicago.com>，

<sip：bob@biloxi.com>

RFC3261 标准规范规定，头字段值由头字段名进行标识。通常头字段名由 UTF8 字符集所包含的文本字符组成。除此之外，头字段还允许包含空格、破折号、分隔符和引号字符。多数头字段名和头字段属性值之间用“:”进行分隔，且头字段中还可包含参数名和参数值，格式如下所示：

field-name：field-value *（；parameter-name = parameter-value）

RFC3261 标准规范对头字段中所包含的参数名数目不作限制，但规定在一

个头字段内，同一参数名能且仅能出现一次。SIP 标准 RFC3261 规定，头字段名对大小写字符不敏感，即相同字段名不区分大小写字符。如未特别指出，头字段中所包含的字段参数值、参数名和参数值也对大小写字符不敏感、除此之外、头字段名中所包含的“-”字符对大小写字符不敏感，但是引号字符内的字符内容对大小写字符敏感。例如，以下头字段分别相同：

1）Contact：<sip：alice@ atlanta. com>；expires = 3600

CONTACT：<sip：alice@ atlanta. com>；ExPiReS = 3600

2）Content-Disposition：session；handling = optional

content-disposition：Session；HANDLING = OPTIONAL

以下头字段则不同：

3）Warning：370 devnull“Choose a bigger pipe”

Warning：370 devnull“CHOOSE A BIGGER PIPE”

2. SIP 头字段分类

SIP 标准中定义了几十种头字段，但是在一条 SIP 消息中不是所有的头字段都同时出现，有些头字段仅在请求或响应中的时候才有意义。只能存在于请求消息中的字段称为请求字段，只能存在于响应消息中的字段称为响应字段。当消息头字段出现了不匹配的字段类型时，则消息接收实体应忽略该字段。下面简单介绍有关的主要头字段。

SIP 的头域字段在 RFC3261 标准规范中就定义了 44 种，随着对标准协议的扩展标准的增加，头域字段的数量也不断增加。下面用表 10-1 列出了 RFC3261 标准中定义的所有头域字段。

表 10-1　头域概览

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Accept	R		—	o	—	o	m*	o
Accept	2xx		—	—	—	o	m*	o
Accept	415		—	c	—	c	c	c
Accept-Encoding	R		—	o	—	o	o	o
Accept-Encoding	2xx		—	—	—	o	m*	o
Accept-Encoding	415		—	c	—	c	c	c
Accept-Language	R		—	o	—	o	o	o
Accept-Language	2xx		—	—	—	o	m*	o
Accept-Language	415		—	c	—	c	c	c
Alert-Info	R	ar	—	—	—	o	—	—
Alter-Info	180	ar	—	—	—	o	—	—

（续）

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Allow	R		—	o	—	o	o	o
Allow	2xx		—	o	—	m*	m*	o
Allow	r		—	o	—	o	o	o
Allow	405		—	m	—	m	m	m
Authentication-Info	2xx		—	o	—	o	o	o
Authorization	R		o	o	o	o	o	o
Call-ID	c	r	m	m	m	m	m	m
Call-Info		ar	—	—	—	o	o	o
Contact	R		o	—	—	m	o	o
Contact	1xx		—	—	—	o	—	—
Contact	2xx		—	—	—	m	o	o
Contact	3xx	d	—	o	—	o	o	o
Contact	485		—	o	—	o	o	o
Content-Disposition			o	o	—	o	o	o
Content-Encoding			o	o	—	o	o	o
Content-Language			o	o	—	o	o	o
Content-Length		ar	t	t	t	t	t	t
Content-Type			*	*	—	*	*	*
Cseq	c	r	m	m	m	m	m	m
Date		a	o	o	o	o	o	o
Error-Info	300-699	a	—	o	o	o	o	o
Expires			—	—	—	o	—	o
From	c	r	m	m	m	m	m	m
In-Reply-To	R		—	—	—	o	—	—
Max-Forwards	R	amr	m	m	m	m	m	m
Min-Expires	423		—	—	—	—	—	m
MIME-Version			o	o	—	o	o	o
Organization		ar	—	—	—	o	o	o
Priority	R	ar	—	—	—	o	—	—
Proxy-Authenticate	407	ar	—	m	—	m	m	m
Proxy-Authenticate	401	ar	—	o	o	o	o	o
Proxy-Authorization	R	dr	o	o	—	o	o	o

（续）

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Proxy-Require	R	ar	—	o	—	o	o	o
Record-Route	R	ar	o	o	o	o	o	o
Record-Route	2xx, 18x	mr	—	o	o	o	o	—
Reply-To			—	—	—	o	—	—
Require		ar	—	c	—	c	c	c
Retry-After	404, 413, 480, 486		—	o	o	o	o	o
Retry-After	500, 503 600, 603		—	o	o	o	o	o
Route	R	adr	c	c	c	c	c	c
Server	r		—	o	o	o	o	o
Subject	R		—	—	—	o	—	—
Supported	R		—	o	o	m *	o	o
Supported	2xx		—	o	o	m *	m *	o
Timestamp			o	o	o	o	o	o
To	c (1)	r	m	m	m	m	m	m
Unsupported	420		—	m	—	m	m	m
User-Agent			o	o	o	o	o	o
Via	R	amr	m	m	m	m	m	m
Via	rc	dr	m	m	m	m	m	m
Warning	r		—	o	o	o	o	o
WWW-Authenticate	401	ar	—	m	—	m	m	m
WWW-Authenticate	407	ar	—	o	—	o	o	o

注：1. “where” 列是可以使用该头字段的请求和响应的类型。每个值的含义如下：

R 表示该头字段只能用于请求；

r 表示该头字段只能用于响应；

2xx, 4xx 等表示该头字段可以用于的响应代码；

c 表示该头字段从请求中复制到响应中。

2. “proxy”列讲述的是代理服务器可以在该头字段上进行的操作：

a 表示如果没有该字段，代理服务器可以增加或者连接该字段；

m 表示可以修改头字段原有的值；

d 表示可以删除一个头字端的值；

r 表示代理服务器可以读该字段，因此该字段一定不能加密。

3. 剩下的六列涉及的是某方法中可以存在的头字段：

c（Conditional）表示该头字端的要求依赖于消息的上下文；

m（mandatory）表示该头字段是强制使用的；

m * 表示该头字段应该被发送，但是客户机/服务器可以接收没有该字端的消息；

o（optional）表示该头字段是可选的；

t 表示该头字段应该被发送，但是客户机/服务器可以接收没有该字端的消息，如果使用基于流的传送协议（例如 TCP），必须含有该头字段；

* 表示消息体不为空时，该头字段是必需的；

- 表示不可使用该字段。

“可选的”意味着在请求中或者响应中可以包含该头字段，UA 也可以忽略它，但是 Require 字段例外。“强制的”要求请求中或者响应中必须含有该头字段，收到该请求的 UAS 或者 UAC 必须能够理解该字段。“不可使用”要求请求消息不可以含有该头字段，UAS 必须忽略含有这些字段的请求消息；同样，对于响应，标注了“不可使用”的头字段意味着 UAS 不可以将该字段放于响应中，并且 UAC 必须忽略响应中的该字段。

UA 应该忽略它所不理解的扩展的头字段参数。当整个消息太大时，某些普通的头字段可以使用缩略形式。Contact、From 以及 To 字段都包含一个 URI 。如果该 URI 包含逗号、问号或者分号，则该 URI 必须用三角括号括起来。任何 URI 参数也都包括在这些三角括号内。如果某 URI 没有在三角括号内并且参数使用分号分隔，那么这些参数就认为是头字段参数，而不是 URI 参数。

（1）Accept 头字段

Accept 头字段参照 HTTP 的语法规定，语义也基本相同，唯一不同的是如果 Accept 头字段不存在，服务器应该默认其值为 application/sdp 。如果 Accept 字段为空，意味着没有可接受的媒体类型。举例如下：

Accept: application/sdp; level = 1, application/x-private, text/html

（2）Accept-Encoding 头字段

Accept-Encoding 字段类似于 Accept，限制响应中可接受的内容编码 Accept-Encoding 字段允许为空。它等价于 Accept-Encoding: identity，也就是说只有 identity 时允许没有编码。如果没有 Accept-Encoding 字段，服务器应该假设默认值是 identity。

（3）Accept-Language 头字段

Accept-Language 头字段用来在请求中指定响应中首选使用的语言，用于响应中作为消息体的原因描述、对话描述或者状态响应。若无该字段，服务器应该假定客户机接受任何一种语言。Accept-Language 的语法遵照 HTTP，基于“q”参数排序的规定也适用于 SIP。举例如下：

Accept-Language：da，en-gb；q = 0.8，en；q = 0.7

（4）Alert-Info 头字段

Alert-Info 存在于 INVITE 请求时，该字段对 UAS 定义了另外一个可用的铃声。当 Alert-Info 头字段位于 180（Ringing）响应中时，该字段对 UAC 定义了另外一个可以使用的回铃声。代理服务器插入该字段是可以提供一种独特的铃声。但是 Alert-Info 字段会引入安全隐患，后面将会介绍。

另外，用户应该也可以修改铃声，并能够使铃声可用。这有助于防止由于不信任的实体使用了该字段而导致通信中断。举例如下：

Alert-Info：<http：//www. example. com/sounds/moo. wav >

（5）Allow 头字段

该字段列出请求发起 UA 所支持的方法。UA 能理解的所有方法就必须列于该头字段中。消息中若无该头字段，则意味 UA 未提供任何关于它所支持的方法的信息，并不意味着 UA 不支持任何方法。响应消息中的 Allow 头字段包含 OPTIONS 头字段以外其他的方法的，因此可以减少所需消息数量。例如：

Allow：INVITE，ACK，OPTIONS，CANCEL，BYE

（6）Authentication-Info 头字段

Authentication-Info 头字段为通信双方提供 HTTP 分类鉴权信息。如果某请求已经基于 Authorization 头字段完成了鉴权，那么 UAS 可以在该请求的 2xx 的响应中包含一个 Authentication-Info 字段。语法和语义的规定见 RFC2617。例如：

Authentication-Info：nextnonce = “47364c23432d2e131a5fb210812c”

（7）Authorization 头字段

Authorization 头字段中含有一个 UA 的鉴权证书。关于该头字段的用法和 HTTP 鉴权的语法和语义，在后面的 SIP 安全策略中将进行介绍。Authorization 和 Proxy-Authorization 头字段不遵循多头字段值的一般规则。虽然多个数值之间没有逗号分隔，该头字段名仍可以出现多次。并且不可以按照前面讲到的规则将其合并成一个头字段行。

下例中的 Digest 参数没有引号：

Authorization：Digest username = “Alice”，realm = “atlanta. com”，
nonce = “84a4cc6f3082121f32b42a2187831a9e”，
response = “7587245234b3434cc3412213e5f113a5432”

（8）Call-ID 头字段

Call-ID 头字段唯一的标识某个客户端的某个特定的邀请或所有的注册请求。一个多媒体会议可以发起几个 Call-ID 不同的呼叫，例如，某用户可以多次邀请某人参与同一个会议。Call-ID 区分大小写并逐字节比较。缩写形式为 i，代表 Call-ID。例如：

Call-ID：f81d4fae-7dec-11d0-a765-00a0c91e6bf6@ biloxi. com

i：f81d4fae-7dec-11d0-a765-00a0c91e6bf6@ 192. 0. 2. 4

(9) Call-Info 头字段

Call-Info 头字段若在请求中则提供主叫的附加信息，若在响应中来则提供被叫附加信息。其中的“purpose”参数描述了 URI 的用途；“icon”参数指定主叫或者是被叫的表示图像；“info”参数则为主叫或者被叫的一般描述；“card”参数用于提供一个商务名片。其余的参数可以通过 IANA 注册，具体在 SIP 的安全策略中介绍。

需要提醒的是，使用 Call-Info 字段可能引起安全隐患。如果某被叫收到一个恶意主叫提供的 URI，那么该被叫可能会显现一些不恰当或者攻击性的内容，或者一些危险的或者非法的内容。因此 RFC3261 建议仅当 UA 能够证明发出 Call-Info 字段的实体是真实并可信的，该 UA 才能使用 Call-Info 的信息。这一要求是对等的 UA，代理服务器也可以在请求中插入该字段。例如：

Call-Info：<http：//wwww. example. com/alice/photo. jpg>；purpose = icon，

<http：//www. example. com/alice/>；purpose = info

(10) Contact 头字段

Contact 字段的值含有一个 URI，其含义取决于该字段所在的请求或者响应的类型。

Contact 头字段值中还可以包含一个显示名称、含有 URI 参数的 URI 和头字段参数。在 RFC3261 中定义了 Contact 参数“q”和“expires”。这些参数只用于 REGISTER 请求及其响应以及 3xx 响应。

当 Contact 头字段包含一个显示名称的时候，带有所有的 URI 参数的 URI 应放于三角括号中，否则，URI 后面的参数都认为是头字段参数而不是 URI 参数。即使“display-name”为空，只要“addr-spec”包含逗号、分号或者问号，也必须使用“name-addr”中的格式。名字和“<”之间的空格可有可无。

解析显示名称、URI、URI 参数以及头字段参数的规则同样适用于 To 和 From 头字段。Contact 头字段的作用类似于 HTTP 中的 Location 头字段，但是 HTTP 的 Location 头字段只允许一个不用引号标注的地址。由于 URI 中可以包括逗号和分号，它们可能被误认为头字段或者参数的分隔符。Contact 头字段的缩写是 m（“moved”）。例如：

Contact：“Mr. Watson”<sip：watson@ worcester. bell-telephone. com>；

q = 0.7; expires = 3600, "Mr. Watson" < mailto: watson@ bell-telephone. com > ; q = 0.1

m: < sips: bob@ 192. 0. 2. 4 > ; expires = 60

(11) Content-Disposition 头字段

Content-Disposition 头字段描述了 UAC 或者 UAS 如何翻译消息体或者多部分消息体中的每部分消息体。该头字段扩展了 MIME Content-Type。RFC3261 协议在 Content-Disposition 头字段中定义了几个新的类型。"session" 表示该消息体部分描述了一个呼叫时或者呼叫前的媒体会话。"render" 表示该消息体部分应被显示，否则回传给用户。在这里使用 "render" 而不是 "inline" 参数值是为了避免 MIME 消息体的隐藏部分作为整个消息的一部分而被显现出来。为了后向兼容，如果 Content-Disposition 头字段丢失，服务器应该假设 Content-Type application/sdp 的消息体是所要描述的 "session"，其他消息体的值为 "render"。"icon" 表示该消息体包含一个图像作为表示主叫或者被叫的图标，当某个消息被接收时由 UA 递交或者在某个对话期间保留。"alert" 指明该消息体部分包含一些信息应该由 UA 通知用户接收请求的时候，回传给用户，通常一个请求发起一个对话。例如，180 临时性响应发出之后，报警该消息体应该作为一个铃声信号回传给电话机。

任何含有 "disposition-type" 的 MIME 消息体只有在该消息被鉴权之后才能被处理。参数 handling-param 指定了如果 UA 不理解所收到的某消息体的内容类型或者处理类型，UAS 应该如何处理。该参数定义了 "optional" 和 "required" 两个值，如果没有该参数，默认值为 "required"。关于该参数参见 RFC3204。例如：

Content-Disposition: session

(12) Content-Encoding 头字段

Content-Encoding 头字段是 "media-type" 的修正。它的值指定了适用于该实体的编码以及为了获得 Content-Type 指定的媒体类型所需要使用的解码机制。Content-Encoding 头字段主要用于消息体压缩而且不丢失底层媒体类型标识。如果某个实体可以使用多个编码方式，则编码方式按其使用的顺序排列。

Content-Encoding 头字段的所有值都不区分大小写，其符号值必须在 IANA 上注册。客户端可以在请求中的消息体中使用 Content-Enconding 头字段指定的编码方式。服务器端则可以在响应中的消息体中使用这些编码方式。服务器端在请求中 Accept-Encoding 字段中列举的编码方式。Content-Encoding 的缩写形式为。举例如下：

Content-Encoding: gzip

e: tar

(13) Content-Language

在 SIP 的安全机制中的 HTTP 相关部分进行介绍。举例如下：

Content-Language：fr

(14) Content-Length

Content-Length 头字段用十进制数指定发送的消息体的大小（字节数）。具体实现时，应该使用该字段指示所传递消息体的大小而不考虑该实体的媒体类型。RFC3261 协议规定，使用基于流的传输协议时必须使用该字段。

本字段所指示的消息体的大小不包括分隔头字段和消息体的 CRLF 符。该字段的有效值为大于或等于零的数。如果消息中无消息体，那么该字段值必须设为零。省略 Content-Length 字段简化了动态的产生响应的类 CGI 脚本程序。该字段的缩写形式为 l。例如：

Content-Length：349

l：173

(15) Content-Type

Content-Type 头字段指定消息体的媒体类型。RFC 协议规定，如果消息体不为空，Content-Type 头字段必须存在。如果消息体为空且消息中包含 Content-Type 头字段，表明所指定的媒体类型长度为零（例如一个空的音频文件）。该字段缩写形式为 c。例如：

Content-Type：application/sdp

c：text/html；charset = ISO-8859-4

(16) Cseq 头字段

Cseq 头字段位于请求消息中，包含一个十进制数字序列和一个请求方法。该数字序列必须用一个 32bit 的无符号整数来表示。Cseq 头字段的请求方法部分区分大小写。该头字段用于把某对话中的事务进行排序且提供了一种唯一标识某事务的方法，并能够区分某请求是新的请求还是重发的请求。如果两个 Cseq 的数字序列以及方法都相等那么这两个 Cseq 就是等价的。例如：

CSeq：4711 INVITE

(17) Date 头字段

Date 头字段包含日期和时间。与 HTTP/1.1 不同，RFC3261 协议只支持最新的 RFC 1123 的日期格式。SIP 中的时区限制为“GMT”，而 RFC 1123 允许任何时区。RFC 1123 的日期区分大小写。

Date 头字段反映了请求或者响应第一次发送的时间。没有后备电池的终端系统可以利用 Date 字段获得当前时间。然而，使用 GMT 格式的时候，要求客户机知道本地与 GMT 时间的偏移。例如：

Date：Sat，13 Nov 2010 23：29：00 GMT

（18）Error-Info 头字段

Error-Info 头字段用来提供出错状态码的附加信息。UAC 具备用户接口功能，包括 PC 软客户的弹出窗口接口、音频接口、通过网关连接的音频电话或端点的接口。当服务器产生错误信号的时候，它可以发送一个带有详细原因描述的出错状态码，或者可发送一个音频记录，而且 Error-Info 头字段可以同时发送这两种信号。UAC 可以选择任何一种方式通知主叫。

UAC 可以把 Error-Info 字段里的 SIP 或 SIPS URI 当作重定向消息的 Contact 头字段，发起一个新的 INVITE 请求并建立一个已记录的会话。非 SIP URI 可以回送给用户。例如：

SIP/2.0 404 The number you have dialed is not in service

Error-Info：<sip：not-in-service-recording@ atlanta.com>

（19）Expires 头字段

Expires 头字段给出消息（内容）的有效期时间，其含义因请求方法不同而不同。INVITE 的有效时间并不影响它所引起的会话的实际持续时间。会话持续时间的表达方法由会话描述协议（SDP）规定。

该字段的值是一个十进制整数，单位为秒（s），其值介于 0～231 之间。从收到请求的那一刻开始计时。举例如下：

Expires：5

（20）From 头字段

From 头字段用于指示请求的发起者，这可能与对话的发起者并不同。被叫发送给主叫的请求在 From 字段中使用被叫的地址。

“display-name”参数可选，主要用于人机接口。如果某用户需要隐藏其标识，则本字段应为“Anonymous”。即使本字段为空，如果“addr-spec”包含一个逗号问号或分号，那么必须使用“name-addr”的格式。

如果两个 From 字段的 URI 一致并且参数一致，则这两个字段是等价的。如果其中一个字段中有扩展的参数而另外一个没有，那么比较的时候应该忽略这个参数，这意味着名字和三角括号并不影响其等价性。From 的缩写形式是 f。例如：

From：“A. G. Bell”<sip：agb@ bell-telephone.com>；tag = a48s

From：sip：+12125551212@ server.phone2net.com；tag = 887s

f：Anonymous <sip：c8oqz84zk7z@ privacy.org>；tag = hyh8

（21）In-Reply-To 头字段

In-Reply-To 头字段列举了本次呼叫涉及或者返回的所有的 Call-ID。客户端可以缓存这些 Call-ID，并放在所返回的呼叫的 In-Reply-To 字段中。这允许自动呼叫分配系统（ACD）将返回的呼叫路由给第一个呼叫发起者，这也允许被叫过

滤这些呼叫，仅返回那些被接受的呼叫。这个头字段并不可替代请求鉴权。例如：

In-Reply-To: 70710@ saturn. bell-tel. com, 17320@ saturn. bell-tel. com

(22) Max-Forwards 头字段

在 RFC3261 规范中，Max-Forwards 头字段必须和任何方法一起规定向下游转发消息的代理服务器和网关的个数。当某客户端沿着某条链路发送请求消息的时候，使用该字段可以有效的防止链路中出错或者发生回环。

Max-Forwards 头字段的值是一个 0 ~ 255 的整数，表示该请求还允许被转发的次数。转发该请求时每经过一个服务器该值就减一。RFC3261 规范建议起始值为 70。实体如果无法保证环路检测则应该插入该头字段，如 B2BUA 。例如：

Max-Forwards: 6

(23) Min-Expires 头字段

Min-Expires 头字段定义了由服务器控制的软状态实体更新间隔的最小值。这也包括注册服务器所存储的 Contact 字段。该头字段的值是一个 0 ~ 231 的十进制整数，单位为秒 (s)。该字段在 423 响应中的用法参见前面的章节。例如：

Min-Expires: 60

(24) MIME-Version 头字段

该字段的用法参见 HTTP/1. 1 规范，在后面的章节也会再提到。举例如下：

MIME-Version: 1. 0

(25) Organization 头字段

Organization 头字段给出发出请求或者响应消息的实体所属的组织的名称。该字段可以被客户端软件用来过滤呼叫。举例如下：

Organization: Boxes by Bob

(26) Priority 头字段

Priority 头字段定义客户端请求的紧急程度。该头字段描述了对于接收用户或其代理，请求消息所具有优先级。例如，它可以影响呼叫路由的选择以及是否决定接受。请求中若无该头字段则应认为优先权为 “normal”。Priority 头字段并不影响通信资源的使用，比如其不会影响路由器中数据包转发优先级或者接入到 PSTN 网关中的情况。该字段的值可以为 “non-urgent”，“normal”，“urgent” 和 “emergency”。例如：

Subject: A tornado is heading our way!

Priority: emergency

或者 Subject: Weekend plans

Priority: non-urgent

(27) Proxy-Authenticate 头字段

Proxy-Authenticate 头字段包含一个鉴权质询口令。关于本字段在 SIP 的安全机制中进行详细介绍。示例：

Proxy-Authenticate: Digest realm = "atlanta. com",
domain = "sip: ss1. carrier. com", qop = "auth",
nonce = "f84f1cec41e6cbe5aea9c8e88d359",
opaque = "", stale = FALSE, algorithm = MD5

(28) Proxy-Authorization 头字段

Proxy-Authorization 头字段用于用户对要求鉴权的代理服务器标识自己。Proxy-Authorization 头字段的值中包含一个证书，其中包含 UA 对于代理服务器的鉴权信息和所要请求的资源的域。关于本字段在 SIP 的安全机制中进行详细介绍。

关于多字段名的一般规定并不适于 Proxy-Authorization 和 Authorization 头字段。虽然多个数值之间没有逗号分隔，该头字段名仍可以出现多次，但是不可以将它们结合成一个单独的头字段行。示例如下：

Proxy-Authorization: Digest username = "Alice", realm = "atlanta. com",
nonce = "c60f3082ee1212b402a21831ae",
response = "245f23415f11432b3434341c022"

(29) Proxy-Require 头字段

Proxy-Require 头字段指定代理服务器所必须支持的某些特征。举例如下：

Proxy-Require: foo

(30) Record-Route 头字段

Record-Route 头字段由代理服务器插入请求消息中，这样可以使该对话中将来的请求仍能经过该代理服务器。举例如下：

Record-Route: <sip: server10. biloxi. com; lr>,
<sip: bigbox3. site3. atlanta. com; lr>

(31) Reply-To 头字段

Reply-To 头字段中包含一个逻辑返回 URI，该 URI 与 From 头字段可能不同。如果用户需要保持匿名，则该字段必须从请求中省略掉，或者使用一种不显现任何个人信息的方式。

即使“display-name”为空，但如果“addr-spec”包含一个逗号、问号或者分号，那么必须使用“name-addr”的格式。示例如下：

Reply-To: Bob <sip: bob@ biloxi. com>

(32) Require 头字段

UAC 通过 Require 头字段告知 UAS 处理请求时需要支持的选项。该字段为可选，但不可以被忽略。该字段包含一系列选项标签，选项标签指定了一系列

的扩展头字段，这些字段在处理请求时必须被理解。举例如下：

Require：100rel

（33）Retry-After 头字段

Retry-After 头字段用在 500（服务器内部错误）或 503（服务不可用）响应中，可以对请求发起客户端指示多久以后服务不可用，用于 404（未找到）、413（请求过大）、480（暂时不可用）、486（正忙）、600（忙）或者 603（下降）响应中表明被叫何时可用。该值用一个十进制的正整数表示，单位为秒（s），从响应发出开始计时。

本字段中可以使用一个注释项标明回应时间的附加信息，该注释项为可选。可用“duration”参数指明被叫从空闲的开始保持空闲的时长，该参数为可选。举例如下：

Retry-After：18000；duration = 3600

Retry-After：120（I’ m in a meeting）

（34）Route 头字段

Route 头字段中有一个代理服务器列表，用来指定请求消息的路由。举例如下：

Route：<sip：bigbox3. site3. atlanta. com；lr>，

<sip：server10. biloxi. com；lr>

（35）Server 头字段

Server 头字段包含 UAS 处理请求所使用的软件信息。如果显示服务器特定的软件版本信息，一旦该软件包含安全缺口的话，就容易使该服务器受到攻击。具体实现中，应把 Server 字段作为可配置的选项。举例如下：

Server：HomeServer v2

（36）Subject 头字段

Subject 头字段包含呼叫的概述或者呼叫的性质，过滤呼叫时，可以不必解析会话描述，会话描述不必与邀请使用同一个对象。该字段的缩写形式为 s。举例如下：

Subject：Need more boxes

s：Tech Support

（37）Supported 头字段

Supported 头字段列举了 UAS 或者 UAC 支持的所有扩展，该头字段中包含一个 UAS 或者 UAC 所支持的选项标签的列表。如果此头字段为空表示则没有可以支持的扩展定义。该字段的缩写形式为 k。举例如下：

Supported：100rel

（38）Timestamp 头字段

Timestamp 头字段给出UAC 向 UAS 发出请求的时间。RFC3261 规范中没有定义该头字段的用法，但可以使用某些扩展来获得 RTT（TCP 往返传输时间）估计值。举例如下：

Timestamp：54

（39）To 头字段

To 头字段指定请求的逻辑接收者。“display-name”参数用于人机接口，为可选；“tag”参数一般用于标识对话。To 头字段等价性的判定同 From 头字段。该字段的缩写形式为 t。举例如下：

To：The Operator ＜sip：operator@ cs. columbia. edu＞；tag＝287447

t：sip：＋12125551212@ server. phone2net. com

（40）Unsupported 头字段

Unsupported 头字段列出 UAS 不支持的特征。举例如下：

Unsupported：foo

（41）User-Agent 头字段

User-Agent 头字段含有与请求发起 UAC 有关的信息。该字段的语义参见 HTTP/1.1。如果显示 UA 的特定的软件版本信息，一旦软件存在安全缺口的话，该UA 就容易易受到攻击。应把 User-Agent 字段作为可配置的选项。举例如下：

User-Agent：Softphone Beta1. 5

（42）Via 头字段

Via 头字段指定目前请求消息经过的路径，同时指定响应也要按该路径返回。该字段值中的 branch ID 参数是一个事务标识符，代理服务器用它来检测环路。

Via 头字段包含一个用来发送消息的传送协议和客户端的主机名或者网络地址，该头字段还可能包含一个接受响应的端口号，本字段还可以包含下列参数：maddr 、ttl 、received 和 branch。具体实现时，branch 参数的值必须从“z9hG4bK”这个字符串开始。本字段定义的传送协议有“UDP”、“TCP”、“TLS”和“SCTP”。举例如下：

Via：SIP/2. 0/UDP erlang. bell-telephone. com：5060；branch＝z9hG4bK87asdks7

Via：SIP/2. 0/UDP 192. 0. 2. 1：5060；received＝192. 0. 2. 207；branch＝z9hG4bK77asjd

Via 头字段的缩写形式为 v。上面例子中，一个拥有两个地址（192. 0. 2. 1 和 192. 0. 2. 207）的多穴主机发出消息。发送方可能用错了网络接口。Erlang. bell-telephone. com 发现不匹配，就在前一跳的 Via 字段中插入一个包含实际发送该数据包的地址的参数。

主机或者网络地址以及端口号不需要遵循 SIP URI 的语法。“:”或者“/”的任何一边的 LWS 都是允许的。如下所示：

Via: SIP / 2.0 / UDP first. example. com: 4000; ttl = 16

; maddr = 224. 2. 0. 1 ; branch = z9hG4bKa7c6a8dlze. 1

虽然 RFC3261 规范要求所有的请求中都包含 branch 参数，该字段的 BNF 仍然规定该参数为可选的。这样可增强与 RFC 2543 定义的实体之间的互操作性。

如果两个 Via 头字段的 sent-protocol 和 sent-by 参数相等（有同样的参数组并且相同的参数值）则两个头字段等价。

(43) Warning 头字段

Warning 头字段中是与响应状态有关的附加信息。该字段值放于响应中发送，其中包含一个三位报警码、主机名以及报警文本信息。

“warn-text”应为可被接收用户所理解的自然语言。可以根据以下几种情况选择：用户的位置、请求中的 Accept-Language 字段、响应中的 Content-Language 字段。默认的语言是 i-default。

以下列举了已定义的“warn-code”和英文的 warn-text。这些报警是由会话描述所引入的失败。报警码的第一位是 3 则表明为 SIP 报警。300 ~ 329 表明是因为会话描述中的关键词引起的错误；330 ~ 339 表示错误与会话描述要求的基本网络业务有关；370 ~ 379 表明错误与会话描述要求的 QoS 参数有关；390 ~ 399 是除上述情况之外的错误。

300 网络协议不兼容：会话描述中的一个或多个网络协议不可用。

301 网络地址格式不兼容：会话描述中的一个或多个地址格式不可用。

302 传送协议不兼容：会话描述中的一个或多个传送协议不可用。

303 带宽单位不兼容：会话描述中的一个或多个带宽度量单位不被理解。

304 媒体类型不可用：对话描述中的一个或多个媒体类型不可用。

305 媒体格式不兼容：对话描述中的一个或多个媒体格式不可用。

306 媒体特征不被理解：对话描述中的一个或多个媒体特征不被支持。

307 对话描述参数不被理解：除上述几种参数之外的参数不被理解。

330 组播不可用：用户站点不支持组播。

331 单播不可用：用户站点不支持单播通信（通常是由于防火墙的存在）。

370 带宽不足：对话描述中定义的或者媒体定义的带宽超出可用带宽。

399 混合报警：该报警表示用户存在的任意一种错误，收到该报警的系统不可以采取任何自动的动作。

其余的“warn-code”可以由 IANA 定义。举例如下：

Warning: 307 isi. edu “Session parameter ’ foo’ not understood”

Warning: 301 isi. edu “Incompatible network address type ’ E. 164’”

(44) WWW-Authenticate 头字段

WWW-Authenticate 包含一个鉴权质询。举例如下：

WWW-Authenticate: Digest realm = "atlanta. com",

domain = "sip: boxesbybob. com", qop = "auth",

nonce = "f84f1cec41e6cbe5aea9c8e88d359",

opaque = "", stale = FALSE, algorithm = MD5

上面介绍了在 RFC3261 协议标准中规定的 44 个头域字段的用法和对应的规范。目前对 SIP 出现了许多扩展协议，其中也定义了很多头域字段，其具体用法参见相应扩展协议的标准。

10.3.4　消息体

RFC3261 协议规定 SIP 请求消息可包含消息实体部分，消息实体部分的解释与消息请求方法相一致。对于 SIP 响应消息，请求方法和响应状态码可以识别消息实体的类型。RFC3261 协议规定所有 SIP 响应消息应包含一个消息实体部分。

1. 消息实体类型

消息实体的类型必须由 Content-Type 头字段进行定义，且如果消息实体采用了压缩编码方式，则相应地应在 Content-Encoding 头字段中定义其所采用的压缩编码算法。否则，如果消息实体未采用压缩编码方式，则 Content-Encoding 头字段的内容应被忽略。在具体应用过程中，消息接收实体应将消息实体的内容作为 Content-Type 头字段值来对待。

RFC3261 协议规定消息实体可以承载使用“multipart” MIME 方式进行编码的信息单元，有关 MIME（参见 RFC2046），也会在第 11 章 SIP 的安全机制中进行介绍。具体实现中，当请求发送方需要发送消息实体部分包含 MIME 信息单元的请求消息时，且接收方在 Accept 头字段指示下其不能接受 MIME 消息实体，则消息请求发送方应在消息中附加一个 SDP 部分作为非 MIME 消息实体进行发送。SIP 消息可以包含二进制编码的消息实体。如果未明确指出，默认的消息文本编码类型是 UTF-8 。

2. 消息实体长度

消息实体的长度由 Content-Length 头字段进行定义，单位为 B。RFC3261 协议规定 SIP 消息实体不能采用 HTTP1. 1 中所定义的“chunked” 传送编码机制。在“chunked” 传送编码机制中，每一块消息实体的长度由相应的标识符进行标识。

3. SIP 消息帧

SIP 可以使用 UDP 或者其他不可靠的报文协议进行承载传送，且每一个报文携带一个请求或响应消息。对于 UDP 来说能够处理的最大数据包就是 65535B

并包括 IP 和 UDP 字头。消息尺寸和 MTU 之间的 200B 的“缓冲区”适应了 SIP 中响应消息一般比请求大的情况。例如，可以在 INVITE 响应中附加 Record-Route 头字段。有了这个额外的缓冲区，响应可以比请求大 170B 而在 IPv4 中不必被分块（假设没有 IPSec 时，IP/UDP 消耗大约 30B）。当路径 MTU 未知时，基于以太网的 MTU 是 1500B 来将 MTU 设为 1300B。

如果一个实体由于消息大小的缘故才需要通过 TCP 发送某请求，否则该请求可以通过 UDP 传输，如果建立该连接时引起下面两种情况：ICMP 不支持和重新建立 TCP 连接，此时该实体就应该使用 UDP 重新发送请求。这将兼容 RFC2543 中不支持 TCP 的要求。

具体实现时，如果 SIP 消息采用面向流的方式进行传送，则 SIP 消息起始行前的任何 CRLF 字符应忽略。

Content-Length 头字段值用于定义一个 SIP 消息在流中的结束位置。当 SIP 消息采用面向流的方式进行传送时，该头字段不能被省略。

10.4 用户代理的行为

首先声明，我们这里讨论的用户代理客户端（UAC）和用户代理服务端（UAS），是一般用户代理中的客户端和服务端，而非代理中的客户端和服务端。

用户代理（UA）表示一个终端系统。它包括两部分：UAC 和 UAS，前者产生请求，后者产生对应的响应。UAC 可以根据某些外部激励产生请求消息并且处理这些请求的响应消息。UAC 外部激励可以是用户在某一图形界面上的按钮点击、也可以是某一 PSTN 链路上的信令信号等。UAS 是一个能够接收请求，并且产生应答的部件或者功能模块，它可以根据用户输入、外部输入、程序执行结果或者其他什么机制来产生应答。

当一个 UAC 发送一个请求，这些请求可能通过一些 PROXY（代理服务器）传递到 UAS 上。当 UAS 产生一个应答，那么这个应答就会同样的被传送到 UAC。UAC 和 UAS 的处理由两个特点：第一，基于请求或者应答是否在一个对话里，第二，基于请求的方法。会话描述了点对点的用户代理之间的关系，并且通过一些 SIP 方法建立会话，比如 INVITE 方法等。

在本节，我们将讨论在处理对话外的请求时，即包括 UAC 和 UAS 的方法无关的规则，也包括用于建立会话的请求。其中建立会话的安全问题，将在第 11 章讲述对在对话外的请求和应答的安全处理，特别是 UAS 和 UAC 之间的互相认证的机制。通过用 S/MIME 加密的消息体可以提供有限的隐私保证。

10.4.1　用户代理客户端

本节讨论 UAC 行为时，把它分成两部分来讲：对话（dialog）外行为和对话内行为。

对于对话外的行为。在对话外发送的请求消息包括：用 INVITE 消息建立起一个会话，用 OPTIONS 消息进行能力查询等。一次会话的建立，在 UAC 端有两个处理过程：生成请求消息和发送请求消息。下面以 INVITE 消息为例，针对这两步进行详细讲解。

1. 生成（产生）请求消息

RFC3261 协议标准规定，由 UAC 产生的一个有效的 SIP 请求消息必须至少包含下列头字段：To、From、CSeq、Call-ID、Max-Forwards 和 Via 头字段，它们在所有的 SIP 请求消息都是必选的。这 6 个头字段是构建 SIP 消息的基本单元，它们共同提供了大部分关键的消息路由服务，包括消息的寻址、响应的路由、消息传播距离限制、消息排序以及事务交互的唯一性标识等。另外，请求行（request line）也是必选的，它包含了请求方法、Request-URI、SIP 版本信息。

初始化 INVITE 请求消息时，其中的 To、From、CSeq、Call-ID、Max-Forwards 和 Via 等头字段应遵循下面的规则，下面分别进行详细介绍：

（1）Request-URI 头字段

初始化 Request-URI 头域时，其最开始的值应该是 To 头域的值（需要提出的是在 REGISTER 方法中有点不同）。但是出于安全性考虑，UA 有时可能不希望将这两个头字段设为同样的值，尤其是当发送端 UA 知道 Request-URI 可能会在传输中改变的时候。

某些情况下，由于预设路由集（Route Set）的存在，可能会影响到消息的 Request-URI。预设路由集是一个有序的 URI 集合，它标识了一个服务器链，UAC 向这个服务器链发出对话外请求消息。预设路由集通常由用户或服务供应商在 UA 上手工配置，或者通过某些其他的非 SIP 机制配置。RFC3261 中建议，在给某个 UA 配置一个外拨代理服务器时，服务供应商所提供的预设路由集中只有一个 URI，即外拨代理服务器的 URI。当存在预设路由集时，必须提供 Request-URI 和 Route 头域（在下节有详细介绍），并且把 Request-URI 当作远端目标 URI。

（2）To 头字段

To 头字段指定请求消息的逻辑接收者或者是用户或资源的注册地址，该地址同样是作为请求消息的目标地址。To 头字段所指示的不一定为请求消息的最终接收者，它可能包含一个 SIP 或 SIPS URI，或其他的 URI 方案（如 RFC2806 中的 tel URL）。RFC3261 协议规定，所有的具体实现过程都必须支持 SIP URI 方

案，任何支持 TLS 的实现必须支持 SIPS URI 方案。To 头字段中允许包含一个显示名称。

UAC 可通过多种方式来构造一个请求消息的 To 头字段，通常经过人机接口，由用户输入或者从地址簿中选取。用户通常不会输入一个完整的 URI，而只是提供一个由数字或字母组成的字符串（比如“bob”），由 UA 判断并解释该字符串。如果这个字符串被用来构造一个 SIP URI 的用户部分，则用户名在@ 符号右侧所示的主域中被解析；如果这个字符串被用来构造一个 SIPS URI 的用户部分，则用户希望进行安全的通信，同时用户名将在@ 符号右侧所示的主域中被解析。SIP URI 中@ 符号右侧，通常是请求发起方的主域，它使本地主域能够处理外发请求。此外，如果用户输入的是一个电话号码，且 UA 不会指定由某个主域来解释该号码，这时可以使用 tel URL，从而使请求消息所经过的每一个主域都可以处理它。例如：一个来到机场的用户可能登录机场的外发（outbound）代理服务器并通过它发送请求，如果用户输入“114”，那么该号码应由机场的外发代理服务器来分析和处理，而不是用户的本地主域。此时正确构造的 URI 应该是“tel：411”。

请求消息的 To 标签标识了一个对话中的对端，如果对话没有建立，标签就不应当出现。对话之外的请求消息中不可以包含 To 标签（tag）。如下列是一个有效的 To 头字段：

To：Carol <sip：carol@ chicago. com >

（3）From 头字段

From 头字段是指示请求发起方的逻辑标识，它可能是用户的注册地址。From 头字段包含一个 URI 和一个可选的显示名称。SIP 实体用它来决定如何处理一个请求（如呼叫自动拒绝）。由于不是逻辑名称，因此 From URI 不包含 IP 地址或 UA 所在主机的全称域名（FQDN）。

From 头字段中允许包含一个显示名称。如果一个 UAC 需要隐藏自己的身份，它可以使用“Anonymous”作为显示名称和一个语法正确但没有任何意义的 URI，如 sip：this@ anonymous. invalid 。通常，某个 UA 产生的请求消息中的 From 头字段值是由用户或用户本地主域的服务器预先设置的。

如果 UA 被多个用户所共用，那么它可以有多个可切换的用户配置文件，每个文件中含有某一用户的 URI。请求消息的接收者要对发送者进行鉴权，以确认发送者身份与 From 头字段是否一致。From 头字段中必须包含一个新的由 UAC 选定的“tag”参数。例如：

From：“Bob” <sips：bob@ Biloxi. com > ；tag = a48s

From：sip：+12125551212@ phone2net. com；tag = 887s

From：Anonymous <sip：c8oqz84zk7z@ privacy. org > ；tag = hyh8

（4）Call-ID 头字段

Call-ID 头字段是用来将消息分组的唯一性标识。RFC3261 协议规定，在一个对话中，UA 发送的所有请求消息和响应消息都必须有同样的 Call-ID。一个 UA 每次注册所用的 Call-ID 也应是一样的。

当 UAC 产生一个新的对话外请求时，除非被某些方法指定，否则它必须为这个请求消息选择一个在空间上和时间上都是全局唯一的 Call-ID 头字段。所有的 SIP UA 都必须保证它所产生的 Call-ID 不会与其他 UA 所产生的相同。当 UA 收到某些失败的响应后，请求会根据响应的内容修改并重发，这些重发的请求不作为新请求处理，因而也就不需要新的 Call-ID 头字段。

RFC3261 规范建议使用 RFC1750 中的加密随机标识符生成方法来生成 Call-ID。具体实现可采用 localid@ host 的形式。Call-ID 对大小写敏感，需逐字节地进行比较。

加密随机标识符生成方法在一定程度上能防范黑客的会话攻击，并降低了无意中产生 Call-ID 冲突的可能性。Call-ID 的选择不需要通过人机接口和预先设定值来实现。如下例是一个有效的 Call-Id：

Call-ID：f81d4fae-7dec-11d0-a765-00a0c91e6bf6@ foo. bar. com

（5）Cseq 头字段

CSeq 头字段用于标识事务并对事务进行排序。它由一个请求方法和一个序列号组成，请求方法必须与对应的请求消息类型一致。对话外的非 REGISTER 请求，序列号值可以是任意的，但它必须可被表示成一个 32 位的无符号整数，且小于 2^{31}。只要符合以上规则，客户端可以用任何方式来选择 Cseq 头字段值。如下是 Cseq 头字段：

CSeq：4711 INVITE

（6）Max-Fowords 头字段

Max-Fowords 头字段限定一个请求消息在到达目的地之前允许经过的最大跳数。它包含一个整数值，每经过一跳，这个值就被减一。如果在请求消息到达目的地之前该值变为零，那么请求将被拒绝并返回一个 483（跳数过多）错误响应消息。

UAC 必须在它发起的每个请求中都插入值为 70 的 Max-Fowords 头字段。在任何不出现回路的 SIP 网络中，选择该值为 70 足够大地保证一个请求消息不被丢弃，且在有回路的情况下，这个值也不会因太大而过分浪费代理服务器的资源。UA 只有知道网络的拓扑结构时，才可以谨慎地选择更小的跳数值。

（7）Via 头字段

Via 头字段定义 SIP 事务的下层（传输层）传输协议，并标识响应消息将要被发送的位置。只有当到达下一跳所用的传输协议被选定后，才能在请求消息

中加入 Via 头字段值。

RFC3261 协议规定，当 UAC 生成请求消息时，它必须在其中插入一个 Via 头字段。Via 头字段的协议名称和协议版本必须分别为“SIP”和“2.0”。Via 头字段中必须包含一个 branch 参数，该参数用来标识由当前请求所建立的事务，该参数既用在客户端也用在服务器端。

对于某个 UA 发出的所有请求，它们的 branch 参数值在空间和时间上必须全局唯一的。但有两种情况例外：一是 CANCEL 请求，以后会说明 CANCEL 请求的 branch 参数与它所要取消的那个请求的 branch 参数是一样的；另一个是对非 2xx 响应的 ACK 请求，这种情况下 ACK 请求与相关的 INVITE 请求有着同样的 branch ID，它所要确认的就是该 INVITE 的响应。

branch ID 参数的唯一性特点使它能被用作事务的 ID。

SIP 实体在插入 branch ID 时，必须以“z9hG4bK”开头（7 个字符足以保证不会与旧的 RFC2543 实体的选取值相同），这样 SIP 服务器在收到请求消息时，就能确定现在的 branch ID 是全局唯一。另外，branch ID 参数的准确格式由具体的实现定义。

Via 头字段的 maddr、ttl、以及 sent-by 部分在传输层对请求消息进行处理时进行设置。

（8）Contact 头字段

Contact 头字段指定一个 SIP 或 SIPS URI，后续请求可以用它来联系到当前 UA。任何能够建立对话的请求消息中都必须有 Contact 头字段，并且该头字段中只能含有一个 SIP 或 SIPS URI。在 RFC3261 协议标准定义的请求方法中，只有 INVITE 能建立对话。对这些能建立对话的请求，Contact 的作用范围是全局的，也就是说，Contact 头字段值中包含的 URI 是 UA 希望用来接收请求的地址，即使用在任何对话外的后续请求消息中，该 URI 也必须有效。

如果请求消息的 Request-URI 或顶端 Route 头字段值中包含了 SIPS URI，那么在 Contact 头字段中也必须包含一个 SIPS URI。

（9）Supported、Proxy-Require 和 Require 头字段

如果 UAC 支持某些 SIP 的扩展，并且这些扩展可被服务器用来构成请求的响应，那么 UAC 应在请求消息中包含一个 Supported 头字段，列出这些扩展的选项标签。只有在 RFC 规范中定义的扩展所对应的选项标签能够在 Supported 头字段中列出。在实验性和信息性 RFC 中定义的扩展禁止在 Supported 头字段中使用。

如果 UAC 要求 UAS 必须理解某项扩展以便处理它所发出的请求，它必须在请求消息中插入一个 Require 头字段，并列出此扩展的选项标签。如果 UAC 希望在请求中使用某项扩展，并要求请求消息经过的所有代理服务器都能理解此

扩展，它必须在请求消息中插入一个 Proxy-Require 头字段，并列出此扩展的选项标签。

同 Supported 头字段一样，Require 和 Proxy-Require 头字段中的选项标签所对应的扩展也必须是在标准 RFC 的后续规范中定义的。

以上 9 种头字段是 USC 在 INVITE 请求必须有的字段，但由于初始化的 INVITE 请求是一个对话外的请求，它除了要遵循上述规则外，还有专门针对 INVITE 的附加处理规则。

（10）Allow 头域

在 INVITE 中应当包括一个 Allow 头域（10.3.3 节），它用来标志在这个 INVITE 建立的对话（dialog）后，在对话过程中还有那些方法可以接受。比如，一个 UA 可以在对话中接收和处理 INFO 请求，那么在 INVITE 请求的 Allow 头域中应当列出这个 INFO 方法。在 INVITE 请求中，可以被支持的头域应当包含在这个 INVITE 请求消息中，并在 Allow 头域中列出 UAC 支持的所有扩展部分。

（11）Accept 头域和 Expire 头域

在 INVITE 中可以包含一个 Accept 头域（10.3.3 节）。它标志 UA 在后续建立的对话中，能兼容的接收和发送的 Content-Type。当在 Accept 头域支持不同会话描述格式（session descrioption format）的时候特别有用。

UAC 可以通过包含一个 Expire 头域（10.3.3 节）来限制邀请的有效期限。如果 Expire 头域的时间到了还没有接收到 INVITE 的终结应答，UAC 处理核心应当产生一个对 INVITE 请求的 CANCEL 请求。

UAC 还可以根据需要，还可以增加 Subject（10.3.3 节）、Organization（10.3.3 节）和 User-Agent（10.3.3 节）头域。这些头域都包含了 INVITE 请求消息的相关资料。

（12）Content-Type 头域字段

UAC 可以给 INVITE 增加一个消息体，由 Content-Type 头域字段来描述消息体。但是对于这种消息体，有一些特别的规定——它们是基于某种协商机制的，它们的描述的内容就是“session（会话）”。UAC 发出一个会话描述，UAS 回应一个对应的会话描述，SIP 通过这种请求/应答模型来进行协商。请求方提供那些通信形式（比如 audio、vidio 和 game）可以被接受，以及这些通信形式的那些参数（比如解码器等等）也可以接收，并且从应答方接收媒体信息的地址。对方 UA 会回应另外一个相应的会话描述，标明那些通信形式以及这些通信形式的那些参数可以被接受。请求/应答的交换是通过对话的上下文来进行的，所以如果一个 SIP INVITE 请求产生了多个对话，那么每一个对话都要分别独立的通过请求/应答方式来交换信息。请求/应答模型定义限制了请求和应答不能被修改（比如在上一个协商请求尚未处理完成的情况下不能发起下一个协商请求）。

这就限制了需要协商的通信方式和对应参数出现在 SIP 消息体中。所以这些需要协商的信息只能出现在请求和应答消息的 ACK。它们的用法进一步受到限制。在初始化一个 INVITE 事务时，应按照如下规则：

1）初始化会话协商描述请求必须在 INVITE 中，如果不在 INVITE 请求中，就必须在 UAS 回送给 UAC 的第一个非失败的可靠消息中。在 RFC3261 中，这个应答就是 2xx 应答。

2）如果在 INVITE 中初始化了会话协商描述，UAS 必须发送回一个与 INVITE 对应的可靠的非失败的应答消息给 UAC。符合这个规定的应答消息，只有 2xx 应答消息对应这个 INVITE 请求。同样相同的应答可能在之前发送的临时应答中存在。UAC 必须把它接收到的第一个有关会话描述应答消息当作是对 INVITE 应答，而且忽略任何后续的应答消息中的会话描述。

3）如果当会话协商描述请求出现在由 UAS 回送给 UAC 的第一个可靠的非失败的消息中时，应答必须是在这个消息的确认消息中（就是 2xx 应答消息所对应的 ACK 确认消息）。

4）在已经发送或者接收到会话协商描述请求的消息之后，UAC 可以同样依据这样的请求/应答的方式产生后续的请求。但是只能在收到所有的会话协商描述请求的应答之后，才能发出下一个会话协商描述请求，不能在上一个会话协商描述请求尚未收到应答时，又发出下一个会话协商描述请求。

5）一旦 UAS 发送或者接收到初始化的会话协商描述请求后，就禁止再给协商会话描述请求产生后续的应答消息。这就意味着基于本规范的 UAS 在完成初始化的事务之前，不会产生任何会话描述请求。

具体来说，上边的规则分别定义了 UA 之间两种信息交换的方法。会话协商请求在 INVITE 中，而应答在 2xx 中（相同的值也可能存在于 1xx 中）；或者会话协商请求在 2xx 中，应答在 ACK 中。这个意思是说，两个 UA 之间建立连接的时候，首先需要协商一下两个 UA 能够支持的消息体正文，那么这个协商关系是通过请求/应答的方式来完成的，这个媒体协商的请求既可以在 UAC 发起的 INVITE 请求中，也可以在 UAS 回应的 2xx 应答中；同样的，媒体协商的应答既可以在 UAS 的 2xx 应答或者 1xx 应答中，也可以在 ACK 确认请求中。所有的支持 INVITE 请求的 UA 都必须支持两种交换方式。

会话描述协议（Session Description Protocol，SDP）（RFC 2327）在所有的 UA 中都必须得到支持，并且它的用法和请求/应答的构造必须遵循 RFC2327 协议。

上边讲述的请求/应答模型，只能适用于当消息体中的包头域 Content-Disposition 的值是“session”的情况。因此，有可能 INVITE 和 ACK 都包含一个消息体。如，INVITE 包含一个相片（Content-Disposition：render）并且 ACK 包含一

个会话描述（Content-Disposition：session）。

如果 Content-Disposition 头域字段不存在，Content-Type 的 application/SDP 就等同于 Content-Disposition 的“session”，其他 Content-Type 的就等同于“render”。

2. 发送请求

INVITE 请求的创建必须按照上面的规则来进行，当 INVITE 请求创建成功后，UAC 遵循对话外请求发送的规则进行发送。这也就是创建一个客户事务并且由这个客户事务发送请求并且处理应答。下面对发送过程进行介绍。

当 INVITE 请求生成后，首先就开始查找请求发送的目的地。RFC3261 规定除非有其他特殊说明，否则目的地址必须按照 RFC3263 中的 DNS 过程来确定。如果 Route 头字段中路由集的第一个实体是个严格的路由，RFC3263 中的过程必须被应用于请求消息的 Request-URI；否则，RFC3263 中的过程应用于请求消息中的第一个 Route 头字段值，若没有 Route 头字段出现则 RFC3263 中的过程仍然应用于请求消息的 Request-URI。其输出结果是个可以发送的目标地址有序集，其中每一项是由地址、端口号以及传输协议构成的三元组。不论用哪个 URI 作为 RFC3263 过程的输入，如果 Request-URI 指定一个 SIPS 资源，UAC 就必须把这个 URI 当作 SIPS URI 来解析。

本地策略可以指定一套额外的目的地用于发送。如果 Request-URI 包含一个 SIPS URI，那么任何额外的目的地都必须用 TLS 来表达；如果请求没有包含 Route 头域，那么就没有对额外的目的地有什么其他的限制了。这个就提供了一个简单的外发（outbound）代理服务器的事前路由的选择。但是，用这样的方法配置一个外发代理服务器是不推荐的；应该由单个 URI 规定的预先设定的路由集来指定外发代理服务器。如果请求包含了 Route 头域，请求应该发送到 Route 头域最上边的一个位置，但是请求也可能被发给由本文档约定的 Route 或者 Request-URI 所指定的服务器（同 RFC2543 定义的相反）。特别的，一个配置了外发 proxy 的 UAC 应该首先尝试把请求发送给由第一个 Route 头域值指定的位置，而不是采用把所有消息都发给外发代理服务器的策略。这就保证了外发的代理服务器通过不增加 Record-Route 头域而不参与后续请求的路径。这个也允许让不能分析第一个 Route URI 的终端，把请求交给外发代理服务器来发送。UAC 应当遵循 RFC3263 中定义的过程尝试目标地址集中的每个地址，直到联系上某个服务器。每次尝试都会建立新的事务，因而每个新请求的最顶端 Via 头字段值都不同，其中包含着新的 branch 参数。此外，Via 头字段中的 transport 值也要根据不同的目标服务器来设置。

3. 处理应答

应答首先是被 transport 层处理，并且被 transport 层发送给上一层的 transac-

tion 层处理。transaction 层处理完成之后将应答发送给更上一层 TU 处理。TU 对响应消息的处理主要依赖于请求方法，但是一些基本处理方法则同具体的请求方法无关。

（1）transaction 层错误（事务层错误）

在某些情况下，从 transaction 层返回的应答不一定是一个 SIP 消息，而是一个 transaction 层的错误。当从 transaction 层收到一个 timeout 错误的时候，必须将这个 timeout 错误当作是收到了一个状态码是 408（请求 timeout）的应答。如果 transport 层报告了一个严重错误（通常取决于 UDP 传输中的严重的 ICMP 错误，或者是 TCP 连接中的错误），必须把这个错误当作是状态码 503（服务未提供）的错误。

（2）无法识别的响应消息

对无法识别的最终响应，UAC 必须将等价于所属响应类别的 x00 响应码，且 UAC 必须能够处理所有响应类别的 x00 响应码。例如：如果 UAC 收到了一个无法识别的响应码 431，那么它能断定自己发出的请求消息出错，因而对该 431 码的处理同 400（错误请求）响应码。但对于任何无法识别的非 100 临时响应的处理必须同 183 响应（会话处理中）。UAC 必须能够处理 100 和 183 响应。

（3）多 Via 头字段值

如果某个响应消息中出现了多个 Via 头字段值，UAC 应当丢弃该响应。在标识请求发起者的 Via 头字段值之前出现了其他的 Via 头字段值，表明消息在传送过程中发生了路由错误，或者已被破坏。

（4）3xx 响应码处理

由于接收到一个重定向的应答（比如，状态码是 301 的应答），客户端应该用在 Contact 头域中的 URI（s）来组织一个或者多个基于重定向以后的新请求，这个处理过程同代理服务器处理一个 3xx 类别的应答很类似，相关资料在 10.5 节中有描述。客户发起请求的时候只有一个目标 URI，就是原始请求中的 Request-URI。如果客户端想在这个请求基础上重构一个基于 3xx 类应答的新请求，那么就需要把这个需要尝试的 URI（s）放到目标集合中去。UAC 在遵循 RFC3261 规范定义的限制条件下，可以选择 Contact 头字段中一些 URI 放入目标地址集。同代理服务器进行递归处理时一样，客户端在处理 3xx 类响应时不可以将任何 URI 多次加入到目标地址集中。如果初始请求消息的 Request-URI 为 SIPS URI，客户端可选择到非 SIPS URI 地址的递归处理，同时应当通知用户其请求被重定向到不安全的 URI。

任何新请求的发出都可能收到 3xx 响应，这些响应又包含了原始请求的 Request-URI。这是因为两个地址有可能被配置成互为重定向目标。所以为了防止出现重定向循环，规定任何 URI 只能在目标地址集中放置一次。随着地址集中

条目的增长，客户端可按任何次序来产生到这些 URI 地址的新请求。通常是按 Contact 头字段值中 q 值的大小对地址集中的条目进行排序。请求消息可以串行或并行生成：一种做法是按 q 值的降序分组后串行处理，同组的 URI 并行处理；另一种方法是只按 q 值的降序串行处理，q 值相等的则次序任选。

如果与目标地址集中的某个地址联系失败，UA 应继续试下一条地址，直到所有的地址都用完。如果所有的地址都已用完，即告请求失败。请求失败应当通过失败响应码来检测，响应码应大于 399。对于网络错误，客户端事务将向事务用户（TU）报告传输层失败。有些响应码指示请求可以重发，重发的请求不应视为失败。当客户端与一个地址的联系失败后，它应继续联系下一个地址。这将创建一个新的事务来发送新的请求。

如果 UAC 基于 3xx 响应码中的某个 Contact 地址来创建新请求，就必须从目标地址集中把相应的 URI 除了 method-param 和 Header 以外全部复制到 Request-URI 中。Header 参数可以用于创建新请求的某些头字段值并重写重定向请求的某些头字段值。某种情况下，Contact 地址中所携带的头字段可附于最初的重定向请求的现有头字段之后。RFC3261 协议规定，如果某个头字段能接受逗号分隔的值列表，那么新的头字段值可被附于原始请求的现有值之后；

如果某个头字段不接受多值，则原始请求的现有值可被 Contact 地址中携带的头字段值重写。例如：如果 3xx 响应码中返回的 Contact 地址带有下列值：

sip：user@ host? Subject = foo&Call-Info = 〈http：//www. foo. com 〉

那么原始请求的每个 Subject 头字段值都将被重写，但那个 HTTP URI 只是被附加到现有的 Call-Info 头字段值之后。RFC3261 建议，UAC 可以重用原始请求的 To、From 和 Call-ID 头字段。

如果新的请求消息构建完成，它将在一个新的事务中发送。顶端 Via 字段中必须有一个新的 branch ID 参数。RFC3261 协议规定，对于所有未述及部分，在收到重定向响应之后发送的新请求应重用原始请求的头字段和消息体。

某种情况下，UAC 会根据收到的响应码以及 Contact 头字段的有效期来临时或永久地缓存某些 Contact 头字段值。具体规则参见 10. 3. 3 节。

（5）4xx 响应消息处理

某些 4xx 响应码需要特定 UA 的处理，处理方法与请求方法无关。RFC3261 协议规定，UAC 在收到 401（未鉴权）或者 407（代理服务器需要鉴权）响应码时应重发携带鉴权证书的请求。

如果收到的是 413（请求消息过大）响应（参见 10. 3. 2 节），说明请求中包含的消息体超过了 UAS 所能接受的长度。在可能的情况下，UAC 应当去掉消息体或减小它的长度。

如果收到的是 415（不支持的媒体类型）响应（参见 10. 3. 2 节），说明请

求中包含了 UAS 不能支持的媒体类型。UAC 应重发请求，并只用响应消息的 Accept 头字段中列出的类型、Accept-Encoding 头字段中列出的编码方式和 Accept-Language 头字段中列出的语言种类。

如果收到的是 416（不支持的 URI 方案）响应（10.3.2 节），说明请求中的 Request-URI 使用了 UAS 不支持的 URI 方案。UAC 应当将其改为 SIP URI 后重发该请求。

如果收到的是 420（错误的扩展）响应，说明请求的 Require 或 Proxy-Require 头字段中列出了某个选项标签，该标签所对应的扩展特性不能被 UAS 或代理服务器所支持。此时，UAC 应当去掉在响应消息的 Unsupported 头字段中列出的所有不支持的扩展，然后重发请求。

所有上述情况下，请求消息重发都是通过对原请求作适当修改之后创建一个新请求来完成的。新请求建立了一个新的事务，并且与原请求有相同的 Call-ID、To 和 From。但 CSeq 应当包含一个新序列号值，它比原请求中 CSeq 的序列号大 1。

对其他 4xx 响应，包括那些将要定义的，请求消息重发是否可能要依赖于具体的请求方法和应用场合。

对于对话（dialog）内行为。对话在两个 UA 之间建立之后，任一个 UA 都可根据需要在对话中发起一个新的事务。在该事务中，发送请求消息的 UA 作为 UAC，接收请求消息的 UA 作为 UAS。这种情况下，UA 的角色可能和建立对话时的事务中的有所不同。

对话中的请求消息可包括 Record-Route 和 Contact 头字段。但是即使修改了远端目的 URI，这些请求仍然不会改变对话的路由集。另外，非目的刷新的请求消息不会修改对话的远端目的 URI，而目的刷新的请求消息则会修改它们。由 INVITE 请求建立的对话，唯一的目的刷新请求定义为 Re-INVITE。针对由其他方式建立的对话，有相应的目标刷新请求的扩展。注意 ACK 不是一个刷新 target 的请求。目标刷新请求消息仅是更新对话的远端目的 URI，而不是更新 Record-Route 头字段的路由集，因为这样会引入与 RFC2543 系统的兼容性问题。

4. 产生请求

在对话中的请求是通过许多对话的状态部分来构造的。在 To 头域中的 URI 部分必须设置成为对话状态中的 remote URI。To 头域的 tag 参数必须设置成为 dialog ID 中的 remote tag 部分。请求的 From URI 必须设置成为对话状态中的 local URI。From 头域的 tag 参数必须设置成为 dialog ID 的 local tag 部分。如果 remote 或者 local tag 是空值，那么 tag 参数必须分别从 From 或者 To 头域中去除。

在请求序列中的原始请求的 To 和 From 头域的 URI 的使用方法是为了向下兼容 RFC2543 协议的，在 RFC2543 协议中，使用 URI 作为对话的标志。在

RFC3261 协议中，只有 tag 用于区分对话。有可能会在 RFC3261 协议的后续版本中，在对话中的请求必须强制反应原始请求的 To 和 From 头域的 URI 将会被去除。

请求的 Call-ID 必须设置成为对话的 Call-ID。在对话中的请求必须严格遵循单个递增的 Cseq 序列号（每次增加 1，当然要除了 ACK 和 CANCEL，这两个请求中的 Cseq 必须和原始的请求或者确认请求一样）。因此，如果本地序列号（local sequence number）不为空，那么本地序列号必须依次增加 1，并且这个数值要存放到 Cseq 头域中。如果本地序列号是空的，那么起初约定的初始值必须填写进去。在 Cseq 头域中的 method 字段必须和请求的方法一致。通过使用 32 位的长整数，使得即使每秒种产生 1 个请求，也会要 136 年才会用完这个整数出现重复。这个序列号的初始值的选取是为了让对话中后续的请求序列号不会重复。非 0 的初始值可以考虑采用时间来作为初始的序列号。一个客户端可以用 31 位有符号整数或者 32 位无符号整数来存放时间作为初始化的序列号。

UAC 使用 remote target 和路由集来构造请求中的 Request-URI 和 Route 头域。如果路由集是空的，那么 UAC 必须把 remote target URI 放到 Request-URI 中，并且 UAC 不能添加 Route 头域到请求中。

如果路由集不为空，并且路由集的第一个 URI 包含 lr 参数，那么 UAC 必须填写 remote target URI 到 Request-URI，并且必须包含 Route 头域，这个 Route 头域按照顺序填写路由集和路由集的参数。

如果路由集不为空，并且路由集合的第一个 URI 没有包含 lr 参数，那么 UAC 必须把第一个 URI 放在 Request-URI 中，并且拆去所有不被 Request-URI 允许的参数。UAC 必须增加一个 Route 头域顺序包含所有剩下的路由集元素及其参数。UAC 接着必须把 remote target URI 放在 Route 头域的最后一项。

例如，如果 remote targe 是：sip：user@ remoteua 并且路由集包括：

< sip：proxy1 >，< sip：proxy2 >，< sip：proxy3；lr >，< sip：proxy4 >

那么请求应该有下列的 Request-URI 和 Route 头域：

METHOD sip：proxy1

Route：< sip：proxy2 >，< sip：proxy3；lr >，< sip：proxy4 >，< sip：user@ remoteua >

如果路由集的第一个 URI 不包含 lr 参数，那么对应的说明代理服务器并不能支持本文档所约定的路由机制，而是支持 RFC2543 文档所约定的路由机制，那么在发送信息的时候需要通过替换 Request-URI 为接收到的第一个 Route 头域的值。将 Request-URI 的值放在 Route 头域的目的是为了保护 Request - URI，使得它经过严格路由的时候不丢失。

在对话内的任何一个刷新 target 的请求中，都应当包含一个 Contact 头域，并且这个 URI 除非有必要，否则都应当是和对话内上次请求的 URI 值一样。如

果“secure”标志设置成为 TRUE，那么 URI 也应当是 SIPS URI。

在刷新 target 请求中的 Contact 头域会更新 remote target URI，这就允许 UA 提供一个新的联系地址（Contact Address），表明它在对话中改变了自己的地址。不过，如果不是刷新 target 的请求，那么不会影响对话中的 remote target URI。请求中剩下的部分请按照上节描述的填写。一旦请求被创建了，它将按照对话外请求发送标准来解析服务器的地址并且发送请求。

一般把请求发送到 Route 头域的最上一个地址，或者如果没有 Route 头域，那么就发送到 Request-URI 地址。由于受到特定的限制，这些步骤也允许把请求发送到另外一个地址（比如在 route set 中没有的默认的外发代理服务器）。

5. 处理应答

UAC 从事务层收到对请求的响应消息。如果客户端事务层返回一个超时信息，则对其的处理与 408 响应相同，对话过程中，UAC 收到 3xx 响应的处理与对话外响应的处理相同，具体规则参见上一节的内容。

但当 UAC 进行位置选择时，它还会使用对话的路由集去构建请求消息的 Route 头字段。当收到对目的刷新请求消息的 2xx 响应时，UAC 必须将对话的远端目的 URI 设置为响应消息中存在的 Contact 头字段的 URI 值。如果响应为 481（呼叫/事务不存在）或者 408（请求超时），UAC 应该终止对话；在在对方无响应的时候 UAC 也应终止对话，客户端事务处理将会通知 TU 超时。

以 INVITE 请求消息发起的对话，UAC 通过发送 BYE 来终止对话。

10.4.2 用户代理服务器端

UAS 在处理会话外请求时，其处理规则与请求方法无关。

请求消息的处理具有不可分割性。如果请求被接受，那么与之相关的所有状态的改变都必须被执行；如果请求被拒绝，则所有状态变化全部禁止。

UAS 应当按下述步骤处理请求：即从鉴权开始，然后检查请求方法，检查各个头字段等。

1. 请求方法检查

在请求的鉴权完成后，UAS 必须检查请求的方法。如果 UAS 认识但不支持某个请求方法，它必须产生一个 405（不允许的请求方法）响应。UAS 还必须在此 405 响应中加入一个 Allow 头字段，其中列出所有它所支持的请求方法。如果 UAS 能够支持请求的类型，处理继续进行。

（1）消息头检查

如果 UAS 不理解请求消息中的某个头字段（即该头字段未在 RFC3261 规范或任何它所支持的扩展中定义），它必须忽略这个头字段并继续进行处理。此外，UAS 应当忽略任何不影响请求处理的格式错误的头字段。

（2）To 和 Request-URI 头字段

To 头字段标识了请求消息的原始接收者，它由 From 头字段所标识的用户指定。由于呼叫转发或其他代理服务器的操作，请求的原始接收者不一定是最终处理该请求的 UAS。当 To 头字段所标识的接收者不是自己时，UAS 可应用任何策略来决定是否接受该请求。RFC3261 建议，即使 UAS 不能识别请求消息中 To 头字段的 URI 编码方案，或不知道该头字段所要寻址的用户，或该用户并非它的当前用户，UAS 仍然要接收这些请求消息。另外，如果 UAS 决定拒绝某个请求，那么它应当产生一个 403（禁止）响应，并将该响应传递给服务器端的事务发送。

请求消息的 Request-URI 标识了将要处理该请求的 UAS。如果请求消息的 Request-URI 使用了 UAS 所不支持的 URI 方案，UAS 应用一个 416 响应（不支持的 URI 方案）拒绝该请求。如果请求消息的 Request-URI 标识的不是 UAS 想要用来接收请求的地址，UAS 应当用一个 404（未找到）响应拒绝该请求。一般来说，UA 用 REGISTER 请求把自己的注册地址与某个特定的联系地址绑定到一起，其后请求消息的 Request-URI 等于该联系地址。另外，Request-URI 也源于 UA 建立或更新对话时所发出的请求或响应中的 Contact 头字段。

（3）请求消息的合并

如果请求消息的 To 头字段中没有标签，此时，UAS 必须针对该请求检查正在进行的事务。如果请求的 From 标签、Call-ID 和 Cseq 与某个正在进行的事务的有关信息相匹配，但依据规则，该请求又确实与该事务不匹配，则 UAS 应产生一个 482（检测到路由循环）响应并传递给服务器端的事务进行处理。

同一个请求沿着不同的路径多次到达某个 UAS，最大的可能是发生了分叉代理现象。这时 UAS 正常处理第一个请求，其余的回送 482 响应。

（4）Require 头字段

当 UAS 认为自己可以处理某一请求时，它将检查请求消息中出现的 Require 头字段。UAC 用 Require 头字段来通知 UAS 它所希望该 UAS 能够支持的 SIP 扩展，以便 UAS 能正确处理它所发出的请求消息。如果 UAS 不能理解 Require 头字段中所列出的扩展选项标签，它必须回以 420（错误的扩展）响应。同时必须在该响应中加入一个 Unsupported 头字段，以列出请求的 Require 头字段中它不支持的那些扩展选项。

RFC3261 协议规定，Require 头字段和 Proxy-Require 头字段不能用在 CANCEL 请求中，也不可用在非 2xx 响应的 ACK 请求中。如果这些请求中出现了这两个头字段，该请求必须被忽略。2xx 响应的 ACK 请求中只能包含对应的原始请求中出现的那些 Require 和 Proxy-Require 值。

例如：

UAC- > UAS: INVITE sip: Watson@ bell-telphone. com SIP/2. 0

Require: 100rel

UAS- > UAC: SIP/2. 0 420 Bad Extension

Unsupported: 100rel

UAS 的这种处理方式保证了当会话双方都能理解所有的扩展选项时，客户与服务器间交互可以无延迟地顺利进行，而如果扩展选项不被理解则会导致交互过程减慢（如上例所示）。对于一个配合良好的客户机—服务器对，理解扩展选项之后省去了能力协商机制，因而交互可以快速进行。此外，当 UAC 对 UAS 提出其不支持的特性时，它使双方的交互和表达能够准确清晰地进行。

2. 消息内容处理

如果 UAS 理解客户端要求的任何扩展，它将检查消息体和描述消息体内容的头字段。如果 UAS 不理解消息体中任何由 Content-Type 指示的类型、由 Content-Language 指示的语言或由 Content-Encoding 指示的编码，且对该部分消息体的处理是必选的（如 Content-Disposition 头字段所示），那么 UAS 必须用一个 415 响应拒绝请求。如果请求中包含了 UAS 不支持的消息体类型，响应消息中必须包含一个 Accept 头字段，其中列出它能理解的所有消息体类型。如果请求中包含了 UAS 不理解的编码方式，响应消息中必须包含一个 Accept-Encoding 头字段，其中列出它能理解的编码方式。

如果请求中包含了 UAS 不理解的语言，响应消息中必须包含一个 Accept-Language 头字段，其中列出它能理解的语言。除此之外，消息体的处理与请求与方法以及消息体的类型有关。

3. 扩展的应用

如果 UAS 在响应时要应用某些扩展，那么相应请求的 Supported 头字段中必须已指明客户端支持这些扩展；如果客户端不支持需要的扩展，UAS 只能依赖基本的 SIP 和客户端支持的扩展来达到目的。极个别情况下如果没有所需的扩展支持则请求无法处理，此时 UAS 可发出 421（需要扩展支持）响应。该响应指出如果不支持某一特定扩展就无法产生正确的响应，该响应的 Require 头字段中必须包含需要支持的扩展选项。由于可能破坏互操作性，因此 RFC3261 不建议采用这种方式。

对非 421 响应要用的任何扩展也必须在该 421 响应的 Require 头字段中列出，并且不能使用请求的 Supported 头字段中未列出的扩展。这样，响应消息的 Require 头字段将只能包含标准 RFC 后续规范中定义的扩展选项标签。

4. 请求的处理

如果上述检查全部通过，接下来 UAS 的处理就与具体的请求方法相关了。在 RFC3261 协议标准中列举了 UAS 对 REGISTER、OPTIONS、INVITE 和 BYE 4

个请求的处理过程。下面以 UAS 处理 INVITE 消息的过程为例。

UAS 核心从事务层收到 INVITE 请求，首先进行处理请求。

假设上述处理顺利完成，同时也没有产生应答消息，那么 UAS 核心将根据如下步骤进行额外处理，这个规则不仅适用于对话外的请求消息还适用于对话内的请求消息。

1）如果 INVITE 请求消息中，包含一个 Expires 头域，那么 UAS 核心就设置一个时钟计数作为这个头域的值。如果时钟到了，则这个邀请就认为过期了。如果在 UAS 尚未产生终结应答前就超时了，那么当产生 487（请求终止）应答消息发送给 UAC。

2）如果请求是一个对话内的请求，与方法无关的处理步骤，将优先进行处理。但这种处理可能会影响到会话。

3）如果请求的 To 头域包含了一个 tag，但是对话的 ID 与现存的任何一个对话都不匹配，这种情况可能是由于 UAS 崩溃或者重新启动而引起的，或者是由于接收到了本应当发送给另外 UAS 的请求。

从这开始的处理将假定这个 INVITE 是在对话外的，并且 INVITE 请求的目的是建立一个新的会话。

INVITE 请求可能包含一个会话描述，在这种情况下是希望和 UAS 进行会话媒体的协商。即使 INVITE 请求是对话外发出的，这个 INVITE 参与的用户也有可能正是那个会话中的参与方。这个是由于在多方会议中，某个正在会议中的用户被其他参与方邀请参加。如果这样情况需要鉴别，UAS 可以使用会话描述来检查是否重复邀请。比如，SDP 包含了会话的 ID 和版本号。如果这个用户本身就是会话中的一方，并且会话参数包含的会话描述没有改变，UAS 可能在不提示用户的情况下接受这个邀请（就是说发送 2xx 应答）。

如果 INVITE 并没有包含某个会话描述（session description），UAS 就是被邀请创建一个会话，并且 UAC 已经要求 UAS 来提供与这个会话相关的参数，UAS 必须在它给 UAC 的第一个非失败的可靠消息中提供这些参数信息，在 RFC3261 协议中规定，给 INVITE 请求发送一个 2xx 应答的应答消息。

UAS 根据实际情况，可以有提示进度、接受邀请、转发请求、拒绝邀请等操作。但它们都按照上面所介绍的规则进行处理，生成对应的应答消息。

（1）提示进度

如果 UAS 不能马上接受或者拒绝邀请，那么它可以提示某种形式的进度给 UAC（比如提示一个回铃声等）。这是通过一个 101 到 199 的临时应答实现的，这些临时应答建立了早期对话（early dialog）。如果需要，UAS 可以发送多个临时应答，每一个临时应答都必须包含相同的对话 ID。同时这些临时应答都是使用 UDP 的非可靠传输发送的。

如果 UAS 打算延长一点时间来响应这个 INVITE 请求，它需要请求一个“extension”来防止代理服务器取消这个 INVITE 事务。代理服务器有权利来取消超过 3min 未完成的事务。要防止这种取消，UAS 必须每分钟发送一个非 100 临时应答，防止由于 1xx 临时应答的非可靠传输导致临时应答丢失。

如果呼叫出于等待状态（比如用户设置成呼叫等待）或者这个呼叫正在和 PSTN 电话系统进行通信，由于 PSTN 系统允许呼叫没有应答，所以 INVITE 事务是可以被延长处理时间的。

（2）INVITE 请求转发

如果 UAS 决定转发这个呼叫，就需要发出 3xx 的应答。300（多重选择）、301（永久转移）和 302（临时转移）应答中应当包含一个 Contact 头域，这个头域包含了一个或者多个表明需要重试的 URI 新地址。这个应答交给 INVITE 服务端事务层，由服务端事务层负责应答的重发。

（3）INVITE 请求的拒绝

拒绝 INVITE 请求的常见情景是被叫方不想或者不能在终端系统上接收这个呼叫，这样的情况下返回 486（用户忙）应答消息。如果 UAS 知道没有其他终端系统能够响应这个呼叫，就应当返回一个 600（全忙）。不过，通常情况下 UAS 是不会知道这种情况的，并且这个应答也很少见。这些应答消息都是交给 INVITE 服务端的事务层进行发送的，由这个事务层来保证应答的重发机制。如果 UAS 拒绝的是 INVITE 请求，而且包含的媒体协商，UAS 应当返回一个 488（此处不接受）应答，这个应答应当包含一个 Warning 头域来解释为何媒体协商被拒绝。

（4）接受 INVITE 请求

UAS 核心产生一个 2xx 应答，通过这个应答，就建立了一个对话，然后进行其他处理。响应 INVITE 请求的 2xx 应答包含 Allow 头域和 Supported 头域，并且还可以包含 Accept 头域。包含 Accept 头域的目的是为了让 UAC 不需要再次请求就能够知道 UAS 的特性以及 UAS 的扩展支持。

如果 INVITE 请求包含了一个媒体协商请求，并且 UAS 还没有发送应答，2xx 应答中必须包含针对这个媒体协商请求的应答。如果 INVITE 请求没有包含媒体协商请求，而且 UAS 也尚未发出媒体协商请求，2xx 应答也必须包含这个媒体协商。

当应答构建好了以后，它会交给 INVITE 的服务端事务层进行发送。由于收到这个终结应答，INVITE 的服务端事务层将会交给通信层进行发送并销毁。因此，有必要在没有收到 ACK 的时候，每隔一定的时间就直接交给通信层进行发送。2xx 交给通信层进行发送的时间间隔是从 T1 秒开始，并且每次发送后就加倍，直到到达 T2 秒的时间间隔（T1 和 T2 的时间间隔定义见 RFC3261 协议 17

节）。当收到这个应答的相应 ACK 请求之后，重发就终止了。这个与使用什么通信协议来发送这个应答消息无关。

由于2xx 的重发是端到端的，并且在 UAS 和 UAC 之间采用 UDP 通信，所以要保证通过这些节点进行可靠的传送，就必须采用间隔时间重发的机制，哪怕 UAS 本身的通信机制是可靠的。

如果服务端对 2xx 应答的重发时间，超过了 64 × T1 秒还没有收到 ACK 请求，那么对话就认为是成功建立，但在这时应该用 BYE 请求来结束会话。

5. 产生响应

当 UAS 要遵循一些流程来要构造某个请求的响应，可能还需要某些特定响应码进行相关的处理，但在本节不作描述。

UAS 在完成创建响应的所有流程之后，应把响应消息传递给发起请求的服务器端的事务。

（1）发送临时响应

UAS 不应考虑对非 INVITE 请求发送临时响应，而是应尽快对其产生最终响应。当 100（正在尝试）响应产生后，请求消息中出现的任何 Timestamp 头字段都必须被复制到该响应中去。如果响应产生时出现了延迟，UAS 应当在响应的 Timestamp 值上附加一个延迟值，这个延迟值必须是从收到请求开始到发出响应为止的以秒计算的时间间隔。

（2）头字段和标签

响应消息的 From 头字段、Call-ID 头字段和 Cseq 头字段必须分别与被响应的请求的 From 头字段、Call-ID 头字段和 Cseq 头字段相等。RFC3261 要求 Via 头字段不但值必须要相等，各个值之间的顺序也必须保持一致。

如果请求消息中包含 To 标签，那么响应消息的 To 头字段必须与请求消息的 To 头字段相等。然而，如果请求消息的 To 头字段不包含标签，那么响应消息中 To 头字段的 URI 必须与请求消息中 To 头字段的 URI 相等；此外，UAS 必须在响应消息的 To 头字段中添加一个标签（100 响应例外，因为 100 响应中可以没有标签），这是为了标识正在应答的 UAS，也许会成为一个对话 ID 的组成部分。同一个请求的所有响应必须使用同一个标签，包括临时响应和最终响应（100 响应仍然例外）。

10.5　SIP 代理服务器

10.5.1　概述

代理服务器是将请求消息路由到 UAS 以及将响应消息路由到 UAC 的实体。

一个请求消息在到达 UAS 之前可能要经过若干个代理服务器的转发，每个代理服务器都要进行路由决策，并在将请求消息转发到下一个实体之前对其进行修改。响应消息将遍历请求消息所经的那些服务器，但顺序却完全相反。

代理服务器是一个逻辑 SIP 实体。当一个请求消息到来时，一个能做为代理服务器的 SIP 实体首先决定是否需要由自己来应答这个请求，例如请求消息中可能有格式错误，或者在执行代理功能之前需要先获得客户端的鉴权证书等，而该实体也可用任何适当的错误码来响应。SIP 实体直接应答一个请求时，它承担的角色就是 UAS，因而必须按 10.5.9 节所述来执行相应的处理功能。对每个新请求，代理服务器既可以在有状态模式下工作，也可以在无状态模式下工作。当以无状态模式工作时，代理服务器只是作为一个简单的消息转发实体，它根据请求消息来做转发目的地和路由决策，然后把请求转发到下游的某个实体；对于响应消息，则只简单地将其往上游方向转发。一旦消息转发完毕，无状态代理服务器将丢弃所有与此消息相关的信息。有状态代理服务器会记住它所收到的每个请求的信息，如事务状态，以及作为某一请求的处理结果而发送的任何请求的信息。这些信息将影响它对后续的、与先前接收的某一请求相关的消息的处理。有状态代理服务器可能选择“分叉”（fork）转发一个请求，即将一个请求向多个目的地路由。任何被转发到多个地点的请求都必须在有状态模式下处理。

有些情况下，代理服务器可能采用有状态的传输方式（比如 TCP）来转发请求，这时不必保留事务状态。例如，代理服务器可能在不保留事务状态的情况下将一个请求从一个 TCP 连接转发到另一个 TCP 连接，只要它在请求消息中放置足够的信息，使它能把对应的响应消息沿着接收请求所用的那条 TCP 连接往回转发。当请求消息在不同类型的传输方式之间转发时，代理服务器必须采用有状态模式，代理服务器的事务用户必须保证消息的可靠传输。

一个有状态的代理服务器可能在请求处理过程中的任何时候转到无状态工作模式下，只要它之前无任何禁止这种转换的动作，如分叉代理或产生了 100 响应等动作，在做这样的转换时，所有的事务状态信息都将被丢掉。代理服务器不应主动发起 CANCEL 请求。

在作为无状态的或者有状态的时候，处理请求的步骤是一样复杂的。接下来以有状态的代理服务器为例讲述代理服务器的处理过程。最后在 10.5.9 节讲述无状态代理服务器与有状态代理服务器的区别。

10.5.2 有状态代理服务器

一个工作在有状态模式下的代理服务器就是一个 SIP 事务处理引擎。有状态代理服务器的行为模型可根据客户端事务和服务器端事务的有关定义而建立。

一个有状态代理服务器包含一个服务器端事务，一个或多个客户端事务，以及将它们关联起来的一个被称为代理服务器核心的（见图 10-5）高层处理模块。从外部收到的请求由服务器端事务来处理，处理后的请求送给代理服务器核心层，核心层决定将请求路由到何处，选择一个或多个下一跳位置。发往每个下一跳位置的请求消息都由一个它自己的客户端事务进行处理。核心层从客户端事务收集响应，并依据这些响应来向服务器端事务发送响应。

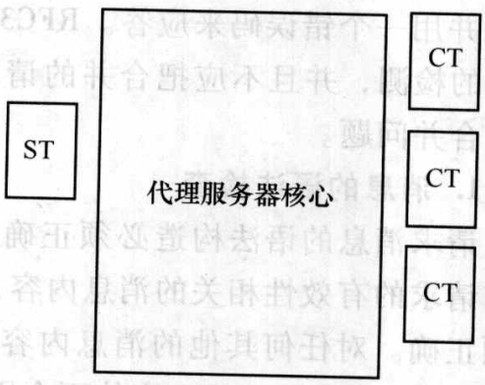


图 10-5　有状态代理服务器模型

ST—服务器端事务　CT—客户端事务

有状态代理服务器为每个收到的请求创建一个新的服务器端事务。任何重传的请求由服务器端事务的处理过程（见 10.6 节）。当立即回送临时性响应消息（例如 100）到服务器端事务时，代理服务器核心必须作为 UAS 进行处理（参见 10.4 节）。有状态代理服务器不应当对非 INVITE 请求产生 100 响应。

本节描述的是代理服务器行为的模型，不是代理服务器软件的模型。具体实现时可以采用任何方法来完成该模型定义的外部行为。

对所有新请求，包括任何方法未知的请求，一个执行代理功能的实体必须具有以下功能：

1）确认请求的有效性，即验证请求

2）预处理路由信息

3）确定请求发送的目的地

4）将请求向每个目的地转发

5）处理所有的响应消息

10.5.3　确认请求的有效性

一个实体在进行代理服务之前必须确认请求的有效性。一个有效的请求消息必须通过如下的检查：

1）合理的语法

2）URI 方案

3）最大转发数

4）回路检测（可选）

5）Proxy-Require

6）Proxy-Authorization

如果任何一项检查失败，代理服务器必须像 UAS 那样处理（参见 10.4.2

节）并用一个错误码来应答。RFC3261 协议规定，代理服务器不要求进行请求合并的检测，并且不应把合并的请求当作错误处理，接收请求的端点将会解决请求合并问题。

1. 消息的语法检查

请求消息的语法构造必须正确，这样服务器端事务才可以处理该请求。任何与请求的有效性相关的消息内容，或者与请求的转发相关的消息内容，其构造必须正确。对任何其他的消息内容，在消息转发时应忽略其语法构造是否正确并保持原样。例如，SIP 实体不会因为一个格式错误的 Date 头字段而拒绝一个请求。同样，代理服务器在转发一个请求之前也不会删除一个格式错误的 Date 头字段。

RFC3261 协议是可扩展的，将来的扩展协议可能定义新的请求方法和新的消息头字段。一个实体不可以因为某个请求包含了它不认识的请求方法或者消息头字段而拒绝代理该请求。

2. URI 编码格式检查

如果一个请求的 Request-URI 使用了代理服务器不能理解的 URI 方案，代理服务器应当用一个 416（不支持的 URI 方案）响应拒绝该请求。

3. 最大转发数检查

Max-Forwards 头字段用来限制一个 SIP 请求消息所能经过的实体的最大数目。

如果请求消息中不包含 Max-Forwards 头字段，本项检查通过；如果请求消息中包含 Max-Forwards 头字段，并且其值大于零，本项检查通过；如果请求消息中包含 Max-Forwards 头字段，并且其值等于零，那么 SIP 实体不可以转发此请求。

如果请求方法是 OPTIONS，那么 SIP 实体可作为消息的最终接收者，否则，SIP 实体必须返回一个 483（跳数过多）响应。

4. 路由循环检测（可选）

SIP 实体在转发一个请求前可能进行转发循环的检测。如果 SIP 实体收到的某个请求中包含了一个 Via 头字段，其 sent-by 参数值与该实体作为代理服务器时在以前的请求消息中加入的值相同，则检测到该请求已经被该实体转发过。这种情况可能是请求消息被循环路由，或者是合法地“螺旋路由”地经过这个 SIP 实体。SIP 实体可以根据该规则的后续处理步骤来计算该消息的 branch 参数，并将结果与上述 Via 头字段中的 branch 参数进行比较。如果两个参数相匹配，那么请求消息被循环路由；如果它们不同，那么请求消息被“螺旋路由”，且处理继续。如果检测到路由循环，SIP 实体可能返回一个 482（检测到路由循环）响应。

5. Proxy-Require 检查

在将来 RFC3261 协议的扩展协议可能会引入一些要求代理服务器所具有的特殊的处理特性。终端在使用这些新特性的请求消息中加入一个 Proxy-Require 头字段，代理服务器如果不理解指定特性将不能进行请求消息的处理。

如果请求消息中包含了一个 Proxy-Require 头字段，其中列出的某项或某几项扩展能力选项标签不能为 SIP 实体所理解，SIP 实体必须返回一个 420（错误的扩展）响应。该响应中必须包含一个 Unsupported 头字段，其中列出那些不能被理解的选项标签。

6. Proxy-Authorization 检查

如果 SIP 实体在转发一个请求之前需要鉴权证书，则需要先进行消息检查。

10.5.4　路由信息预处理

代理服务器必须检查请求消息的 Request-URI。如果请求消息的 Request-URI 是它以前插在 Record-Route 头字段中的值，代理服务器则必须用 Route 头字段的最后一个值替换 Request-URI，并且从 Route 头字段中删除该值。在这之后，代理服务器必须按修改过的请求来进行处理。

上述情况只有当发送请求到代理服务器（可能是一个端点）的 SIP 实体为一严格路由器（代理服务器）时才会发生。为了获得与这类实体的后向兼容性，此时对接收消息的修改是必要的。这也使得遵从 RFC3261 协议的实体能够在通过严格路由的代理服务器时保留 Request-URI，并不要求代理服务器为了检测它以前插入到 Record-Route 头字段中的那些 URI 而保留状态信息。代理服务器只需要在那些 URI 中加入足够的信息，使其在以后出现时能够被识别出来就可以。

如果 Request-URI 中包含 maddr 参数，代理服务器必须检查其值是否在自己所负责的地址集或主域集中。如果在其中，且对应的请求消息是通过 Request-URI 中指定或默认的端口和传输方式接收的，代理服务器必须去掉 maddr 参数和所有非默认的端口及传输方式参数，并按照没有这些参数的请求消息来进行处理。

代理服务器收到的某个请求消息中可能有与该服务器相匹配的 maddr 参数，但接收该消息时所用的端口和传输方式却与 Request-URI 中指定的不同。这样的请求需要被转发到使用 Request-URI 中指定的端口和传输方式的代理服务器。

如果请求消息中 Route 头字段的第一个值指示的就是本代理服务器，那么它必须将该值从请求消息中删掉。

10.5.5　确定请求的发送目的地

代理服务器计算请求消息的发送目的地。发送的目标地址集可以由请求消

息的内容预先决定，也可以是从某个抽象的定位服务获得，地址集中的每个目的地都被表示成一个 URI。如果请求的 Request-URI 中包含一个 maddr 参数，Request-URI 必须被作为唯一的目标 URI 放在目标地址集中。如果 Request-URI 所指示的域不是代理服务器所负责的域，Request-URI 必须被作为唯一的发送目的地放进目标地址集中，然后该实体必须对该消息进行转发，具体过程见后面的章节。

许多情况下代理服务器都可能收到 Request-URI 所指主域不归自己负责的请求消息，比如处理外拨呼叫的防火墙代理服务器（HTTP 代理服务器处理外发请求的方式）就是这种情况可能发生的一个例子。

如果目标地址集不能通过上述的方法确定，当前 SIP 实体则负责 Request-URI 中的主域，并且该实体可以用任何机制来决定向何处转发请求。不论是哪种机制，其模型都可以化为访问一个抽象定位服务的过程。可能的机制有：从 SIP 注册服务器创建的某个定位服务获得信息；读取数据库；查询一个 presence 服务器；使用其他协议；执行一个简单的算法对 Request-URI 进行替换等。当访问一个由注册服务器构建的定位服务时，必须首先对 Request-URI 进行规范化，然后才能把它作为查询索引。这些机制的输出结果被用于构造目标地址集。

如果 Request-URI 没有提供足够的信息使得代理服务器能够决定目标地址集，代理服务器应当返回一个 485（不明确）响应。响应中应当包含一个 Contact 头字段，其中列出了可供尝试的新地址的 URI。例如，如果某代理服务器的定位服务中有多个 John Smith，则一个构造为 sip：John. Smith @ company. com 的 INVITE 请求对该代理服务器是不明确的。

任何在请求消息中，或者与请求消息相关的，或者与实体的当前环境相关的信息都可能用于目标地址集的构建。例如，根据某些头字段和消息体是否出现及其内容如何，或请求消息到达的时间、或接收请求所用的接口、或以前请求消息的失败情况，甚至实体当前的可用性级别等，都可能会构造出不同的目标地址集。

通过上述定位服务确定了可能的目标地址之后，它们的 URI 就被加入到目标地址集中。目标地址只能被放置到目标地址集一次。根据 URI 相等的定义，一个目标 URI 已经在目标地址集中出现，那么它就不能被再次加入该地址集。

如果原请求的 Request-URI 所指示的资源位置不在本代理服务器负责的范围之内，代理服务器不得在目标地址集中加入其他地址，代理服务器只有在请求的 Request-URI 是由自己所负责的情况下才能在消息转发时改变这个 URI。

如果代理服务器对某个 Request-URI 不负责，它将不能按如下所述进行 3xx 类响应或 416 响应的递归处理。

如果代理服务器负责原始请求的 Request-URI 所指定的资源，则在请求转发

开始之后它可能继续向目标地址集加入地址。它可以使用处理过程中获得的任何信息来决定新的目标地址。比如，代理服务器可以将一个重定向响应（3xx 类响应）中获得的 Contact URI 并入目标地址集中。如果代理服务器在构造目标地址集时使用了动态信息源（比如查询一个 SIP 注册服务器），它应当在请求处理过程中监视所用信息源的变化。如果有新的可用地址，那就应当将其加入目标地址集中。同样的地址决不能被多次加入，允许一个 URI 只被加入到目标地址集中一次可减少不必要的网络流量。在将重定向响应中的 Contact 地址合并到目标地址集中时还可以防止无限递归的出现。例如，“无操作”也是一个简单的定位服务，此时目标 URI 就等于接收请求的 Request-URI。请求消息将被发送到特定的下一跳代理服务器以进一步处理。用 SIP 或 SIPS URI 表示的下一跳地址标识被作为 Route 头字段最顶端的值插入到请求消息中。如果在 Request-URI 中指定的某个资源不存在，代理服务器必须返回一个 404（找不到）响应。

按上述所有的方法处理之后，目标地址集仍然为空，代理服务器就必须返回一个错误响应，并且应当是 480（临时不可用）响应。

10.5.6　请求转发

只要目标地址集非空，代理服务器就可以开始转发请求。有状态代理服务器可以按任何次序来处理目标地址集。代理服务器可以对多个目标地址采用串行处理的办法，使得一个客户端事务完成之后才开始下一个；也可以同时开始所有的目标地址处理，即并行处理相应的所有客户端事务；它还可以将目标地址集任意分组，组与组之间串行处理，组内的目标地址并行处理。

一种常用的排序机制是利用目标地址的 qvalue 参数，这些参数来自 Contact 头字段。目标地址按 qvalue 值的大小从最高往最低处理，qvalue 值相同的可能会同时处理。

有状态代理服务器必须有目标地址集的维护机制，这样当收到各个转发请求的响应后就能把它们与原始请求联系起来。本模型中，这是“响应上下文”的机制（response context），它由代理服务器层在转发第一个请求之前创建。对每个目标地址，代理服务器按如下步骤转发请求。

1. 请求消息的复制

代理服务器首先对收到的请求做一份备份。复制最初必须包含所收到的请求的所有头字段。在下面描述的处理过程中未提及的字段也不能删除。复制过程中各个头字段的顺序应当与所收到的请求保持一致。代理服务器决不能将头字段名相同的头字段值重新排序，不能添加、修改、或删除消息体。请求消息复制的具体实现不需要进行真正的消息复制。其基本要求是对每个下一跳的处理都要从同一个请求消息开始。

2. 更新 Request-URI

复制过的请求消息起始行的 Request-URI 必须被替换为当前目标地址的 URI。如果该 URI 中包含任何在 Request-URI 中不允许出现的参数，这些参数必须被删掉。代理服务器通过这种机制把一个请求消息路由到它的目的地。有些情况下，所收到的请求消息的 Request-URI 没有经过任何修改就被放到了目标地址集中。对这样的目标地址，上述的替换实际上是一种“操作”行为。

3. 更新 Max-Forwards 头字段

如果复制的消息中包含一个 Max-Forwards 头字段，代理服务器必须将其值减 1。如果复制的消息中不含 Max-Forwards 头字段，代理服务器必须加入该字段，其值应当为 70。现有的某些 UA 在请求消息中不提供 Max-Forwards 头字段。

4. 添加 Record-Route 头字段值

如果代理服务器希望自己留在后续请求消息的传输路径上，而这些后续的请求消息属于由当前请求创建的一个对话，那么即使请求中已经出现了 Route 头字段，代理服务器也必须在请求消息的复制件中任何已有的 Record-Route 头字段值之前插入一个 Record-Route 头字段值。建立对话的请求消息中可能包含预置的 Route 头字段。

如果当前请求已经属于某一个对话，那么代理服务器要想留在该对话中后续请求消息的传输路径上，它应当在请求中插入一个 Record-Route 头字段。正常情况下，这些 Record-Route 头字段将不会对终端所用的路由集产生任何影响。

对于已经属于某个对话的请求消息，即使代理服务器不在其中插入 Record-Route 头字段值，它也能留在后续请求的传输路径上。然而，在端点发生了错误后重建对话时，这个代理服务器将被从以前的传输路径上移除。

代理服务器可能在任何请求中插入 Record-Route 头字段值，如果请求不是用来发起对话的，那么终端将会忽略这个 Record-Route 头字段值。

请求传输路径上的每个代理服务器都独立决定是否在请求消息中添加一个 Record-Route 头字段值。请求消息中已经出现的 Record-Route 头字段，并不要求代理服务器一定得添加新值。

在 Record-Route 头字段中放置的 URI 必须是一个 SIP URI 或 SIPS URI。这个 URI 必须包含一个 lr 参数。对请求转发的每个目的地，这个 URI 都可能不同。该 URI 不应包含传输方式参数，除非代理服务器明确相邻的下游实体支持该参数所示的传输方式，同时该下游实体将出现在后续请求的传输路径上。

当前代理服务器提供的 URI 将被其他 SIP 实体用于路由决策。通常来说，代理服务器无法知道其他实体的能力，所以它必须满足 SIP 实现的必要条件：支持 SIP URI 以及 TCP 或 UDP 传输方式。应用 RFC3263 中所述的服务器定位过程对 Record-Route 头字段中加入的 URI 进行解析，其结果必须是插入该 URI 的 SIP

实体，这样使得后续的请求能够到达同一个 SIP 实体。如果请求消息的 Request-URI 包含了一个 SIPS URI，或者最顶端的 Route 头字段值在路由信息后处理以后包含了一个 SIPS URI，那么代理服务器在 Record-Route 头字段中加入的 URI 也必须是 SIPS URI。另外，如果请求消息不是通过 TLS 方式接收的，那么代理服务器必须插入一个 Record-Route 头字段；如果请求消息是通过 TLS 接收的，但转发的请求消息中 Request-URI 或最顶端 Route 头字段值在路由信息后处理以后不是 SIPS URI，那么代理服务器必须在转发的请求消息中插入一个非 SIPS URI 的 Record-Route 头字段。一个处于某一安全范围内的代理服务器，在对话当中必须保留在该安全范围内。

当插在 Record-Route 头字段中的 URI 通过响应消息又被传回来时，如果代理服务器需要重写该 URI，则该 URI 必须清晰明确以便被准确查找（请求消息可能通过“螺旋路由”多次经过一个代理服务器，导致代理服务器加入多个 Record-Route 头字段值）。代理服务器可能在 Record-Route 头字段值中包含一些参数，这些参数会在请求的某些响应消息中被送回，如 INVITE 请求的 200（OK）响应。如果在消息中而不是代理服务器中保存了一些状态信息，便可以使用这些参数。

如果不论对话是什么类型，代理服务器都需要留在它的消息传输路径上，代理服务器则应当在每个请求消息中都加入一个 Record-Route 头字段值，即使它不认识请求的方法，也可能会建立会话或属于某个对话。

代理服务器加到某一请求的 Record-Route 头字段中的 URI 只在相应的对话生命期中有效，这个对话由该请求所在的事务创建。例如，一个保持对话状态的代理服务器在对话终止之后，可能会拒绝接收 Request-URI 的某个 URI 的后续请求消息，因为该 URI 的有效期已过。当然，一个没有对话状态的代理服务器无从得知对话是否终止，但它可以在插入的 URI 中添加足够的信息，再将其同后续请求的对话标识相比较，其对话标识与信息不相匹配的请求将有可能被拒绝。终端不可以在某个对话外使用在该对话中得到的 Record-Route 头字段的 URI 。

Record-Route 操作可能是某些特定业务所要求的，这些业务需要代理服务器察看一个对话中的所有消息。但是，这种操作降低了系统的处理速度和系统的可扩展性，所以只有在特定的业务要求时，代理服务器才应进行 Record-Route 操作。

Record-Route 的操作是处理能够发起一个对话的 SIP 请求消息，RFC3261 所涉及这类请求只有 INVITE 。

5. 其他消息头字段的加入（可选）

代理服务器可以在请求消息的复制件中加入其他合适的头字段。

6. 路由信息后处理

代理服务器可以有本地策略，强制某个请求消息在被发到目的地之前必须经过某些特定的代理服务器，但它必须保证这些特定的代理服务器都是松散路由器。通常来说，只有同一个管理域中的代理服务器才能确切知道彼此是否为松散路由器，这些特定的代理服务器由一系列的 URI 表示，每个 URI 都含有 lr 参数。这些 URI 必须被加到请求消息复制件中的 Route 头字段的任何已有的值之前。如果请求复制件中没有出现 Route 头字段，那么需要加入一个新的 Route 头字段，该字段中包含那些必经的代理服务器的 URI。

如果代理服务器有一个本地策略，要求请求消息必须经过某一个特定的代理服务器，可以简单的直接按该代理服务器的地址、所用的端口以及传输方式把请求发送给它。但如果请求消息中有 Route 头字段，那么代理服务器就不能采用这种方法，除非它能确定那个下一跳必经代理服务器是个松散路由器。

采用上述在请求消息中插入 Route 头字段值的办法，有着更好的鲁棒性、灵活性、通用性和操作一致性，因此 RFC3261 建议，应优先考虑这种方式实现。此外，如果请求的 Request-URI 中包含的是个 SIPS URI，那么在与那个必经代理服务器通信时必须使用 TLS。

如果请求消息的复制件中包含 Route 头字段，代理服务器必须检查其第一个值所含的 URI，如果那个 URI 没有包含 lr 参数，那么代理服务器必须按如下方式修改请求消息的复制：

首先，代理服务器必须将 Request-URI 移至 Route 头字段中，并将其作为 Route 头字段的最后一个值。

然后，代理服务器必须把 Route 头字段中的第一个值作为 Request-URI，并将其从 Route 头字段中删除。在请求消息通过严格路由实体时，需要一种机制来保证 Request-URI 所携带的信息不被丢失，将 Request-URI 附在 Route 头字段之后则是这种机制的一部分。将 Route 头字段的第一个值“弹出”到 Request-URI 中使得一个严格路由实体在接收请求时可以得到它所希望的消息格式（请求消息的 Request-URI 即为自己的 URI，请求消息需要访问的下一个位置由 Route 头字段的第一个值给出）。

7. 确定下一跳的地址、端口、传输方式

代理服务器可以有一个本地策略，即将请求消息发往一个特定的 IP 地址，使用特定的端口和传输方式，而与请求消息的 Route 头字段和 Request-URI 值无关。如果代理服务器不能确定对应于特定地址、端口、以及传输方式的服务器是一个松散路由器，那么它就不能使用这种策略。在 RFC3261 标准协议中不建议采用这种机制发送请求消息到特定下一跳位置。

如果没有这种本地策略，代理服务器则按如下方式应用 RFC3263 中所列的解析过程来决定向何处转发请求：代理服务器为将请求消息发送到一个严格路

由实体而对其作了修改，那么它必须将相应的解析过程应用于解析请求消息的 Request-URI。否则，代理服务器必须将其用于解析 Route 头字段的第一个值，如果无 Route 头字段，则仍然用于解析 Request-URI。解析过程将产生一个有序的结果集，其中的每个元素是一个由 IP 地址、端口和传输方式构成的三元组。不管使用哪个 URI 作为 RFC3263 中解析过程的输入，如果请求消息的 Request-URI 是一个 SIPS URI，代理服务器都要把输入 URI 当作 SIPS URI 来处理。

如 RFC3263 所述，代理服务器必须尝试按解析结果集的第一个三元组发送请求消息，然后依序继续进行，直到发送成功为止。在尝试发送的每个三元组，代理服务器必须对消息作适当地修改，并用一个新的客户端事务来发送请求。

既然每次发送尝试都用一个新的客户端事务，即代表一个新的对话分支（branch）。因此，插入的 Via 头字段中的 branch 参数对每次发送尝试都必须不同。

如果客户端事务报告发送失败，或其状态机超时，代理服务器将按次序尝试解析结果集中的下一条地址。如果该集合中的地址全部试完，则说明请求消息无法转发到目标地址集中当前 URI 所指的实体。此时代理服务器不需要在响应上下文中放置任何信息，就如同该实体返回了一个 408（请求超时）响应。

8. 添加一个 Via 头字段值

代理服务器必须在请求消息的复制件中已有 Via 头字段值之前插入一个新值。这意味着代理服务器将计算它自己的 branch 参数，该参数为全局唯一的，来标识相应的对话分支，并包含必需的 magic cookie。这也意味着对于循环路由或“螺旋路由”多次经过某个代理服务器的请求消息而言，每次经过该代理服务器时它得到的 branch 参数是不同的。

如果代理服务器选择进行路由循环检测，那么它们用于构造 branch 参数的值将有更多的限制。这些代理服务器创建的 branch 参数应当可以分为两部分：第一部分必须满足上述 10.3.1 节有关 Via 的要求；第二部分用于执行路由循环检测以及区分循环路由和“螺旋路由”。

路由循环检测的方法是：当一个请求消息返回到代理服务器时，如果那些对请求消息处理有影响的头字段都没有改变，则认为出现了路由循环。branch 参数中用于路由检测部分的值应当体现出所有这些头字段的信息（包括任何 Route、Proxy-Require 以及 Proxy-Authorization 头字段）。这就保证当请求消息被再次路由到代理服务器时，如果这些头字段中有一个发生改变，就可认为请求消息发生了“螺旋路由”而非循环路由，参见本规范 10.5.3 节。创建路由检测部分的值有一个常用方法，即计算一个包括所有相关头字段相关信息的加密哈希表（hash），包括 To tag、From tag、Call-ID 头字段、转换前的原始接收请求的 Request-URI、最顶端 Via 头字段、Cseq 头字段的序号值、以及任何可能出现的

Proxy-Require 头字段和 Proxy-Authorization 头字段。哈希表的产生算法依赖于具体的实现，比较常用的是用十六进制表示的 MD5 算法（RFC1321）。

如果代理服务器要检测循环路由，那么它提供的 branch 参数必须依赖于所有对请求处理有影响的信息，包括原始接收请求的 Request-URI 以及任何影响请求消息的接收或路由的头字段。这样作对区分发生路由循环的请求和返回到当前服务器之前路由参数发生改变的请求时是必需的。

计算 branch 参数时不能包括请求的方法。对于非 2xx 响应的 CANCEL 和 ACK 请求与对应的被取消或被确认的请求必须有相同的 branch 参数值。在处理这些请求的服务器上，branch 参数用于将这些请求相互关联起来。

9. 必要时加入 Content-Length 头字段

如果要采用流传输协议将请求消息发往下一跳，请求消息中没有 Content-Length 头字段，那么代理服务器必须在其中插入一个 Content-Length 头字段，该头字段值等于消息体的长度。

10. 请求消息转发

有状态代理服务器必须为转发请求创建一个新的客户端事务，并指示这个事务使用由指定的方法确定请求发送的地址、端口和传输方式。

11. 设置定时器 C

事务用户使用了一个定时器 C，用来处理无法得到最终响应的 INVITE 请求。当 INVITE 请求被代理转发时，每个客户端事务都必须设置这个定时器。定时器 C 的值必须大于 3min。关于收到临时响应后如何更新这个定时器，以及关于定时器超时后如何处理，分别在 10.5.7 节和 10.5.8 节介绍。

10.5.7 应答处理

当一个 SIP 实体收到响应时，首先它会查找与响应消息相匹配的客户端事务。如果没有找到，那么该实体必须作为一个无状态代理服务器（下文中有关描述）来处理这个响应（即使它是一个信息性响应）。如果事务相匹配，那么这个响应由所匹配的事务来处理。

转发找不到匹配事务的响应消息提高了系统的鲁棒性。特别是可以保证 INVITE 请求“迟到”的 2xx 响应能被正确转发。客户端事务把响应消息传给代理服务层之后，必须执行下述处理：

1）查找正确的响应上下文；

2）对临时响应更新定时器 C；

3）删除最顶端 Via 头字段值；

4）把响应消息加到响应上下文中；

5）检查响应是否需要立即转发；

6）需要时，从响应上下文中选取最优的最终响应；

如果同响应上下文相关的所有客户端事务都已终止，但还没有送出最终响应，代理服务器必须从它已有的响应中选取一个“最优的”转发出去。

对每个要转发的响应消息都必须作下述处理。每个请求都会至少有一个临时响应和一个最终响应将要转发。

7）必要时汇聚所有的 Authorization 头字段值；

8）重写 Record-Route 头字段值（可选）；

9）转发响应消息；

10）生成任何必要的 CANCEL 请求。

1. 查找正确的响应上下文

代理服务器使用关键字查找与响应匹配的响应上下文，该上下文是它在转发原始请求之前创建的。接下来的处理都在这个响应上下文中进行。

2. 对临时响应更新定时器 C

对于 INVITE 事务，如果响应消息是一个响应码在 101 到 199 之间的临时响应，代理服务器必须重置相应客户端事务的定时器 C。定时器 C 可以被设为不同的值，但必须大于 3min。

3. 删除最顶端 Via 头字段

代理服务器必须从响应消息中将最顶端的 Via 头字段值删除。删除之后，响应消息中无 Via 头字段值，则该响应消息就是发给当前实体的，不能再被转发。然后实体按照 UAC 的方式进行处理。这种情况在实体生成 CANCEL 请求时就会发生。

4. 将响应消息加入响应上下文中

收到的最终响应被存在响应上下文中，直到与这个上下文相关的服务器端事务生成了一个最终响应。当前加入的响应可能是“最优”最终响应之一，这个“最优”最终响应将由相关的服务器端事务返回。即使当前加入的响应不是最优的，它所携带的信息也可能用于构造“最优”响应。

如果代理服务器要对 3xx 响应中的任何 Contact 地址进行递归处理，从而需要将这些地址加入到目标地址集中。在响应加入到响应上下文之前代理服务器必须把这些地址从响应消息中删除。如果原始请求的 Request-URI 是一个 SIPS URI，那么代理服务器不应对非 SIPS URI 进行递归处理。如果代理服务器对 3xx 响应的所有 Contact 地址都进行了递归处理，那么它就就不应当把由此产生的无 Contact 地址的响应加入到响应上下文中去。

在将 3xx 响应加入到响应上下文之前把被递归处理的 Contact 地址从其中删除，可以防止上游下一个实体重复尝试这个地址。3xx 响应消息中可能同时包含 SIP、SIPS 以及非 SIP URI。代理服务器可以对 SIP 和 SIPS URI 进行递归处理，

把其他类型的 URI 放在响应上下文中，它们以后可能会在最终响应中返回。

如果代理服务器收到了一个 416（不支持的 URI 方案）响应，该响应对应的请求的 Request-URI 方案为非 SIP 类型，而所收到的原始请求消息的 Request-URI 却是 SIP 或 SIPS 方案，即代理服务器在代理转发一个请求消息时将其 Request-URI 从 SIP 或 SIPS 改成了其他类型，那么代理服务器应在目标地址集中加入一个新的 URI，这个 URI 应是该非 SIP URI 的 SIP URI 版本。如果那个非 SIP URI 是 tel URI，那么 URI 转换规则如下：把 tel URI 的 telephone-subscriber 部分作为 SIP URI 的 user 部分；把前一请求发送的主域作为 SIP URI 的 host 部分。

代理服务器通过 SIP 或 SIPS URI 来“递归处理”416 响应，同 3xx 响应，这个 416 响应也不应当被加到响应上下文中。

5. 检查响应是否需要立即转发

以下响应都必须立即转发，直到服务器端事务发送最终响应：除 100 响应之外的临时响应。

2xx 和响应有状态代理服务器收到的 6xx 响应不必立即转发。但是它应取消所有进行中的客户端事务，并且不可以在响应上下文中再创建任何新的对话分支。

同 RFC2543 要求代理服务器对 6xx 响应必须立即转发的要求不同。INVITE 事务中，在其他对话分支上可能会收到 2xx 响应，此时代理服务器必须转发这个 2xx 响应。如果要求立即转发 6xx 响应，会导致某个 UAC 收到了 6xx 响应之后又接着收到 2xx 响应。RFC3261 标准协议要求，代理服务器在收到 6xx 响应后将发出 CANCEL 请求，通常这会使所有未决的客户端事务都收到 487 响应，这样代理服务器才能往上游方向转发 6xx 响应。

在服务器端事务发出最终响应之后，代理服务器必须立即转发 INVITE 请求的 2xx 响应。除此之外，有状态代理服务器决不可以立即转发其他任何响应，特别是 100 响应。那些以后可能作为“最优”响应被转发的响应应放在响应上下文中。

有状态代理服务器对非 INVITE 请求恰好只转发一个最终响应；对 INVITE 请求转发一个非 2xx 响应或者一个及多个 2xx 响应。

6. 选择最优响应

经过上述步骤，没有最终响应消息被立即转发，并且响应上下文中的所有客户端事务都已终止，那么有状态代理服务器必须给响应上下文的服务器端事务发送一个最终响应。

有状态代理服务器必须从已收到并存放于响应上下文中的响应中选出一个“最优”的最终响应。如果响应上下文中没有最终响应，代理服务器必须发送一个 408（请求超时）响应给服务器端事务，否则，代理服务器必须转发一个存在

于响应上下文中的响应。服务器首先选择响应上下文中的 6xx 响应，如果上下文中没有 6xx 响应，服务器就应当选择响应上下文中保存的最低类别的响应，并可以在该类别中选择任意一个响应。代理服务器应优先考虑那些对重发请求提供有用信息的响应，如选择的是 4xx 类响应，就应该优先考虑 401、407、415、420 以及 484 响应。

除非代理服务器能确定所有被转发的后续请求都会产生 503 响应，否则它就不可以向上游转发所收到的 503（业务不可使用）响应。如果代理服务器转发 503 响应则表示自己不能再处理之后的任何请求消息。如果所收到的唯一响应是个 503 响应，代理服务器则应产生一个 500 响应并向上游转发。

例如，如果代理服务器向 4 个地点转发了一个请求，并且分别收到了 503、407、501 和 404 响应，那么它会转发 407（代理服务器需要鉴权信息）响应。

对话建立过程中可能产生 1xx 响应和 2xx 响应。当请求中不包含 To 标签时，UAC 就用响应中的 To 标签来区分创建对话的请求消息的多个响应。如果请求中不含 To 标签，代理服务器就不能在 1xx 或 2xx 响应的 To 头字段中插入标签，同时，代理服务器也不能修改 1xx 或 2xx 响应中的 To 标签。

由于代理服务器不能在没有 To 标签的请求消息的 1xx 响应中插入 To 标签，那么它自己也不能发送非 100 临时响应。然而，它可以把该请求转给一个与自己共享同一实体的 UAS，该 UAS 可以返回自己的临时响应，从而与请求的发送者进入对话的初始状态。该 UAS 的处理不需要与代理服务器分开，它可以是一个虚拟 UAS 并与代理服务器在同一代码空间中实现。

3xx 到 6xx 的响应是逐跳发送的。当发送这些响应消息时，转发实体作为 UAS，基于从下游实体收到的那些响应发出自己的响应。如果一个请求消息中不含 To 标签，那么当实体简单地转发该请求的 3xx 到 6xx 响应时，应当保存响应消息的 To 标签。请求中如果包含 To 标签的，代理服务器在转发其任何响应时不能修改它们的 To 标签。

代理服务器是否替换被转发的 3xx 到 6xx 响应的 To 标签，对上游实体来说没什么不同，但保留原始标签有助于调试错误。当代理服务器收集了几个响应的信息时，选择其中哪个响应消息的 To 标签是任意的，也可产生一个新的 To 标签，这样可能使错误调试更容易。例如：合并 401（请求未经授权）和 407（代理服务器需要鉴权）响应的认证质询以及合并未加密的和未鉴权的 3xx 响应的 Contact 值。

7. 必要时汇聚所有 Authorization 头字段值

如果所选的响应是一个 401（请求未经授权）或 407（代理服务器需要鉴权）响应，那么代理服务器必须收集响应上下文中收到的所有的 401 和 407 响应中的任何 WWW-Authenticate 和 Proxy-Authenticate 头字段值，并在转发前将它

们原样地加入到所选的响应中。最终发送的 401 或 407 响应中可能会有几个 WWW-Authenticate 和 Proxy-Authenticate 头字段值。

请求消息转发的任意一个或所有的目的地都可能需要鉴权证书(credential),客户端需要接收所有的认证查询并在请求重发时为每一个质询提供鉴权证书。因此上述做法是必须的。具体内容第 8 章。

8. 重写 Record-Route(可选)

如果所选响应包含一个由当前代理服务器提供的原始 Record-Route 头字段值,那么代理服务器可以在转发响应前重写这个头字段值。这样代理服务器能对相邻的上游及下游实体提供不同的 URI。对于多宿主主机就可以使用这种机制。

如果代理服务器是通过 TLS 接收请求,然后用非 TLS 连接将其转发出去,那么代理服务器必须把响应消息中 Record-Route 头字段的 URI 重写为 SIPS URI。如果代理服务器是通过非 TLS 连接收到的请求,然后用 TLS 连接将其转发出去,那么代理服务器必须把响应消息中 Record-Route 头字段的 URI 重写为 SIP URI。

代理服务器加入的新的 URI 与插入到请求消息中 Record-Route 头字段的 URI 除了必须满足同样的要求外,还要作如下修改:新的 URI 不应当包含传输参数,除非代理服务器明确相邻的上游实体支持该参数所表示的传输方式,并且该实体又将出现在后续请求的传输路径上。

当代理服务器决定修改响应消息的 Record-Route 头字段时,它需要找到它所插入的 Record-Route 头字段值。如果请求消息发生了“螺旋路由”,每次重复经过代理服务器时,都会被插入一个 Record-Route 头字段值。本规范建议,当代理服务器要重写 Record-Route 头字段值时,它在请求消息中插入的 URI 应当十分明确,这样重写时才能便于查找。另外一种方法是,代理服务器在 URI 的用户部分附加一个唯一性标识符,以标记此该代理服务器。

代理服务器在响应消息到达时修改标识符与实例相匹配的第一个 Record-Route 头字段值,修改后的 URI 中用户部分不再附加实例标识符数据。下一次重复经过该服务器时,使用相同的算法,查找最上面的与实例标识符参数相匹配的 Record-Route 头字段值,就可以准确地找出上一次所插入的 Record-Route 头字段值。

对于有些请求,虽然代理服务器向其加入了 Record-Route 头字段值,但其对应的响应中可能并不包含 Record-Route 头字段。如果响应中确实包含了 Record-Route 头字段,该字段包含代理服务器以前所加入的值。

9. 转发响应消息

经过上述的处理,代理服务器可以对选定响应进行任何特定操作。代理服务器不可以添加、修改或者删除消息体。除非另有说明,否则代理服务器决不

能删除 Via 头字段值以外的任何其他头字段值，关于 Via 头字段值的删除见 10.5.6 节。代理服务器不可以删除顶端 Via 头字段值（即原始响应消息的第二个 Via 头字段值）中可能出现的任何 received 参数，该参数是在处理该响应的请求时加入的。

代理服务器必须把响应消息发送给与响应上下文相关的服务器端事务进行处理，这将使响应消息被发送到最顶端 Via 头字段值所指的位置。如果服务器端事务不再可用，代理服务器必须用无状态方式转发响应，直接把消息发送到传输层。服务器端事务可能会指示响应消息发送失败，或者发出状态机超时信号。

这些出错信息可能会被记入日志以用于错误诊断，RFC3261 协议并不要求代理服务器必须采取任何补救措施。即使最终响应已经发出，只要相关事务没有全部终止，代理服务器也必须一直维护响应上下文的内容。

10. 生成 CANCEL 请求

当代理服务器转发一个最终响应，或者当它收到 6xx 响应时，它都必须对响应上下文的所有未决客户端事务发出一个 CANCEL 请求。在 RFC3261 标准协议中，未决客户端事务是指已经收到了临时响应，但没有收到最终响应，并且没有收到相关的 CANCEL 请求的事务。

转发最终响应之后取消所有未决客户端事务并不能保证终端不会收到多个 INVITE 请求的 200（OK）响应。在 CANCEL 请求发出并得到处理之前，可能在多个对话分支上就已经产生了 200 响应。因此，RFC3261 协议的扩展协议以后可能会重写这个发送 CANCEL 请求的要求。

10.5.8　异常处理

当代理服务器收到异常消息时（如定时器超时、传输错误和 CANCEL 消息等），它应当有相应的异常处理机制。下面简单介绍代理服务器的异常处理。

1. 处理定时器 C

如果定时器 C 超时，代理服务器必须以新值来重置定时器，或者终止相应的客户端事务。如果客户端事务已收到临时响应，代理服务器必须为之生成一个匹配的 CANCEL 请求；如果客户端事务没有收到临时响应，代理服务器必须按其收到了一个 408（请求超时）响应来处理。代理服务器通过重置超时的定时器 C 来根据当前情况动态地延长事务的生存期。

2. 处理传输错误

如果传输层通知代理服务器在请求消息转发时发生了错误（见 10.8 节），代理服务器必须按收到了一个 503（业务不可用）响应的情况来处理。

如果代理服务器在转发响应消息时被通知有错误发生（见 10.8 节），它必须放弃该响应。代理服务器不应因此取消任何与响应上下文相关的未决客户端

事务。

如果代理服务器取消它的未决客户端事务，某个恶意的或出错的客户端就可以通过其对应的 Via 头字段值使当前代理服务器所有的事务失败。

3. CANCEL 请求的处理

有状态代理服务器在收到了请求消息的临时响应后可以对它产生的任何其他请求发送一个 CANCEL 请求。当收到 CANCEL 请求时，代理服务器必须取消与之匹配的响应上下文的所有未决客户端事务。

由于终端会通知事务终止的事件，所以有状态代理服务器一般不需要基于 INVITE 请求中 Expires 头字段指示的有效期来对 INVITE 请求的未决客户端事务产生 CANCEL 请求。

当 CANCEL 请求在由它自己的服务器端事务处理时，有状态代理服务器不会为它创建新的响应上下文。反之，代理服务器从现有的响应上下文中查找与 CANCEL 请求相关的那个请求所在的服务器端事务。如果找到了匹配的响应上下文，那么代理服务器必须立即对 CANCEL 请求返回一个 200（OK）响应，这种情况下当前实体就作为一个 UAS。另外，代理服务器必须为响应上下文中的所有未决客户端事务产生 CANCEL 请求。如果没有找到相匹配的响应上下文，即当前实体没有任何与 CANCEL 请求对应的请求的信息，那么它必须以无状态方式转发这个 CANCEL 请求。

10.5.9 无状态代理服务器

当作为无状态代理服务器的时候，代理服务器就是一个简单的消息转发者。很多时候无状态代理服务器的处理步骤和有状态的代理服务器的处理步骤很类似。下面讲讲它们不同的地方。

一个无状态的代理服务器并没有事务的概念，或者没有用于描述有状态代理服务器行为的应答上下文。相反的是，无状态代理服务器处理的无论是请求消息还是应答消息，都是直接从通信层处理的（见 10.6 节）。因此，无状态代理服务器自己转发这些消息。但他们转发他们收到的其他任何重发的消息（它们本身并没有能力来分辨那些消息是重发的，那些消息是原始消息）。此外，当无状态的代理服务器处理一个请求的时候，这个节点并不产生它自己的 100 响应或者其他临时性应答消息。

无状态的代理服务器必须用 10.5.3 节描述的那样来验证一个请求，同时无状态的代理服务器必须遵从 10.5.4 至 10.5.6 节定义的步骤来处理请求，但有如下几点例外：

1）无状态的代理服务器必须从目的地集合中，选择一个并且只能选择一个目的地。这个选择必须是根据消息的头域，并且是和服务器时间无关的。特别

是，一个转发请求必须能够每次都转发到相同的目的地。此外，CANCEL 和非路由的 ACK 请求必须和它们相关的 INVITE 请求有相同的转发目的地。

2）对无状态代理服务器的 branch ID 来说，必须要求在时间上和空间上都是唯一的。但是无状态的代理服务器不能简单的使用一个随机数产生器来计算 branch ID 的第一个部分。这是由于重发的请求需要相同的 branch ID 值，并且无状态的代理服务器不能区分重发的请求和原始请求。因此，branch 参数的组成部分要求唯一，这样使得重发的时候能够填写相同的值。对于无状态的代理服务器来说，branch 参数必须作为一个重发无关的消息处理参数存在。

在 RFC3261 中没有规定无状态代理服务器采用何种手段保证 branch ID 的唯一性。不过，RFC3261 推荐下列步骤。无状态代理服务器检查在接收到请求的最上 Via 头域值的 branch ID，如果它是由 magic cookie 打头的，那么 branch ID 的第一个部分就是当作接收到的 branch ID 的 hash 值。否则 branch ID 的第一个部分就当作是 Via 头域的最上值、To 头域的 tag、From 头域的 tag、Call-ID 头域、Cseq 序列号（除了方法部分）、接收到的请求的 Request-URI 的一个 hash 值。同时这些头域值在不同事务中总是不一样的。

1）所有在 10.5.6 节提到的其他消息转换必须保证转发重发的请求的时候能够转发到相同的节点。特别是在代理服务器在 Record-Route 头域中增加了值，或者在 Route 头域中增加了值的时候，代理服务器必须在转发重发的请求的时候增加相同的值。至于 Via 头字段中的 branch 参数，转发必须是基于与时间无关的配置或者请求重发无关的属性。

2）一个无状态代理服务器决定转发的地点是像 10.5.6 节 10 步描述的有状态的代理服务器一样。但是请求是直接交给通信层发送的，而不是交给客户端事务。

由于一个无状态的代理服务器必须转发重发的请求到相同的地方，并且增加标志性的 branch 参数，它只能用消息中本身的信息和与时间无关的配置来计算。如果配置状态不是与时间无关的（比如如果路由表更新了），这些改变了的请求，在改动开始以后，到在事务超时的时间范围内，就不能作为无状态的转发了。这个处理这段时间的请求是实现相关的。通常处理的方法，是把这些请求当作事务有状态地进行转发。

无状态的代理服务器禁止对 CANCEL 做特别的处理，CANCEL 的处理就像对其他请求的处理一样进行，特别是，一个无状态的代理服务器使用相同的 Route 头域来处理 CANCEL 请求，就像处理其他请求一样。

对于 10.5.7 节中定义的应答处理，对于无状态代理服务器来说，并不适用。当一个应答到达一个无状态代理服务器时，代理服务器必须检查最上的 Via 头域值的 sent-by 参数。如果这个地址和这个代理服务器一样（就是和代理服务器插

入的先前请求中的值一样），那么这个代理服务器必须从应答中移除这个头域值，并且转发这个应答到下一个 Via 头域值。同时这个代理服务器禁止增加、修改或者删除消息体。除非有特别的说明，代理服务器禁止移除其他的头域值。如果地址不匹配本代理服务器，消息就必须简单的悄悄扔掉。

10.5.10 代理服务器路由处理小结

如果没有相反的本地策略的情况下，代理服务器对于包含 Route 头字段的请求所执行的处理可被总结为如下几步：

1）检查请求的 Request-URI。如果它指示的是一个由当前代理服务器负责的资源，那么代理服务器将用从定位服务中得到的地址代替它，否则代理服务器将不对它作改动。

2）代理服务器将检查最顶端 Route 头字段中的 URI。如果它指向当前代理服务器，那么将其从 Route 头字段中删除。

3）代理服务器将把请求转发到最顶端的 Route 头字段值中 URI 所指的资源位置，如果没有 Route 头字段出现则转发到 Request-URI 所指的资源位置。在转发请求时代理服务器通过对目标 URI 运用 RFC3263 中所描述的过程来确定所用的 IP 地址、端口和传输方式。

如果在请求消息的传输路径上没有遇到严格路由的实体，Request-URI 将总是指向请求的目的地。下面举例来说明代理服务器路由处理的过程。

1. 基本 SIP 通信模型

该例子是一个基本的 SIP“梯形”结构，U1->P1->P2->U2，U1 和 U2 表示终端，P1 和 P2 表示代理服务器。两个代理服务器都要插入 Record-Route 头字段值。这里给出消息流程：

U1 发送请求给 P1：

INVITE sip：callee@ domain. com SIP/2. 0

Contact：sip：caller@ u1. example. com

P1 是一个外拨代理服务器，它不负责 domain. com，所以它进行 DNS 查询并将请求发往那里。它还在请求中加了一个 Record-Route 头字段值：

INVITE sip：callee@ domain. com SIP/2. 0

Contact：sip：caller@ u1. example. com

Record-Route：<sip：p1. example. com；lr>

P2 收到了这个请求。它负责 domain. com，因此它运行定位服务并重写 Request-URI。它也加入了一个 Record-Route 头字段值。因为请求中没有 Route 头字段，所以它对新的 Request-URI 进行解析以确定向何处转发请求：

INVITE sip：callee@ u2. domain. com SIP/2. 0

Contact：sip：caller@ u1. example. com

Record-Route：＜sip：p2. domain. com；lr＞

Record-Route：＜sip：p1. example. com；lr＞

位于 u2. domain. com 的被叫方收到了请求并以 200（OK）作答：

SIP/2. 0 200 OK

Contact：sip：callee@ u2. domain. com

Record-Route：＜sip：p2. domain. com；lr＞

Record-Route：＜sip：p1. example. com；lr＞

同时，位于 U2 的被叫将其对话状态信息中的远端目标 URI 设为 sip：caller@ u1. example. com 并将其路由集设为：

（＜sip：p2. domain. com；lr＞，＜sip：p1. example. com；lr＞）

这个 200 响应沿着 P2->P1->U1 的路径正常回传。现在，U1 把自己对话状态信息中的远端目标 URI 设为 sip：callee@ u2. domain. com 并将其路由集设为：

（＜sip：p1. example. com；lr＞，＜sip：p2. domain. com；lr＞）

既然路由集中的所有元素都包含 lr 参数，U1 将构造出如下的 BYE 请求：

BYE sip：callee@ u2. domain. com SIP/2. 0

Route：＜sip：p1. example. com；lr＞，＜sip：p2. domain. com；lr＞

就像任何其他的实体（包括代理服务器）一样，U1 运用 DNS 解析最顶端 Route 头字段值中的 URI 以确定向何处发送请求，这时请求被发往 P1。P1 注意到自己不负责 Request-URI 所指的资源位置，因此它不改变其内容，它也确实发现自己是 Route 头字段中的第一个值，所以它将该值删除，并将请求转发到 P2：

BYE sip：callee@ u2. domain. com SIP/2. 0

Route：＜sip：p2. domain. com；lr＞

P2 也注意到它不负责 Request-URI 所指的资源（它负责的是 domain. com，而不是 u2. domain. com），所以它也不改变 Request-URI 的值，P2 也确实发现自己是 Route 头字段的第一个值，所以把这个值删去，并通过对 Request-URI 进行 DNS 查询将如下的请求转发到 u2. domain. com：

BYE sip：callee@ u2. domain. com SIP/2. 0

2. 穿越一个严格路由的代理服务器

该例中，一个对话通过 4 个代理服务器建立，每个代理服务器都加入了 Rocord-Route 头字段值。第三个代理服务器实现了严格路由过程，这些过程在 RFC2543 及 RFC3261 协议先前的很多演进版本中规定。

U1->P1->P2->P3->P4->U2，到达 U2 的 INVITE 请求包含：

INVITE sip：callee@ u2. domain. com SIP/2. 0

Contact：sip：caller@ u1. example. com

Record-Route: <sip: p4. domain. com; lr>

Record-Route: <sip: p3. middle. com>

Record-Route: <sip: p2. example. com; lr>

Record-Route: <sip: p1. example. com; lr>

U2 回应以 200（OK）之后，U2 基于第一个 Route 头字段值发送如下的 BYE 请求给 P4：

BYE sip: caller@ u1. example. com SIP/2. 0

Route: <sip: p4. domain. com; lr>

Route: <sip: p3. middle. com>

Route: <sip: p2. example. com; lr>

Route: <sip: p1. example. com; lr>

P4 不负责 Request-URI 所指的资源，因此保留其值不变。它注意到自己是 Route 头字段的第一个值所指实体，所以将之删除。然后 P4 根据当前的第一个 Route 头字段值 sip: p3. middle. com 准备发送请求，但它注意到这个 URI 中没有 lr 参数，因此在发送之前，它将请求消息变为如下格式：

BYE sip: p3. middle. com SIP/2. 0

Route: <sip: p2. example. com; lr>

Route: <sip: p1. example. com; lr>

Route: <sip: caller@ u1. example. com>

P3 是一个严格路由的代理服务器，它将如下请求转发到 P2：

BYE sip: p2. example. com; lr SIP/2. 0

Route: <sip: p1. example. com; lr>

Route: <sip: caller@ u1. example. com>

P2 看到 Request-URI 是自己以前放到 Record-Route 头字段中的值，所以在进一步处理之前，将请求消息改写为：

BYE sip: caller@ u1. example. com SIP/2. 0

Route: <sip: p1. example. com; lr>

P2 检查不负责 u1. example. com，所以它根据 Route 头字段值的解析结果将请求消息发往 P1。P1 注意到自己被列在最顶端的 Route 头字段值中，因而它将该值删除，消息变为：

BYE sip: caller@ u1. example. com SIP/2. 0

既然 P1 不负责 u1. example. com，并且消息中也没有 Route 头字段，P1 根据 Request-URI 将请求消息转发到 u1. example. com。

3. 重写 Record-Route 头域值

在这里例子中，U1 和 U2 是在不同的私有域空间中，并且他们通过代理服

务器 P1 开始一个对话，这个 P1 作为不同私有域间的一个网关存在。U1->P1->U2

U1 发送：

INVITE sip：callee@ gateway. leftprivatespace. com SIP/2. 0

Contact：<sip：caller@ u1. leftprivatespace. com >

P1 使用自己的定位服务并且发送下边的信息到 U2：

INVITE sip：callee@ rightprivatespace. com SIP/2. 0

Contact：<sip：caller@ u1. leftprivatespace. com >

Record-Route：<sip：gateway. rightprivatespace. com；lr >

U2 发送 200（OK）应答回给 P1：

SIP/2. 0 200 OK

Contact：<sip：callee@ u2. rightprivatespace. com >

Record-Route：<sip：gateway. rightprivatespace. com；lr >

P1 重写它的 Record-Route 头域参数，提供成为 U1 能够使用的参数，并且发送给 P1：

SIP/2. 0 200 OK

Contact：<sip：callee@ u2. rightprivatespace. com >

Record-Route：<sip：gateway. leftprivatespace. com；lr >

稍后，U1 发送接下来的 BYE 到 P1：

BYE sip：callee@ u2. rightprivatespace. com SIP/2. 0

Route：<sip：gateway. leftprivatespace. com；lr >

P1 转发到 U2：

BYE sip：callee@ u2. rightpriatespace. com SIP/2. 0

10.6　SIP 事务

SIP 是基于事务处理的协议，一个 SIP 事务是由请求和其响应，包括临时响应和最终响应组成的。在请求是 INVITE 而响应是非 2xx 的响应时，对应的事务也包括 ACK 消息；如果其响应是 2xx，则相应的事务中不包括 ACK 消息。

事务可分为客户端事务和服务器端事务。客户端事务发送请求，而服务器端事务发送响应。客户端和服务器端事务都是一种逻辑功能，包含在任何实现此功能的实体中。特别是，它们在 UA 中和有状态的代理服务器中存在。如图 10-6 所示，UAC 执行客户端事务，它的外发代理服务器执行服务端事务。外发代理服务器同时也执行客户端事务，把请求发送到一个内发代理服务器的服务端事务。这个代理服务器也同时执行一个客户端事务，把请求发到一个 UAS 的

服务端事务上去。

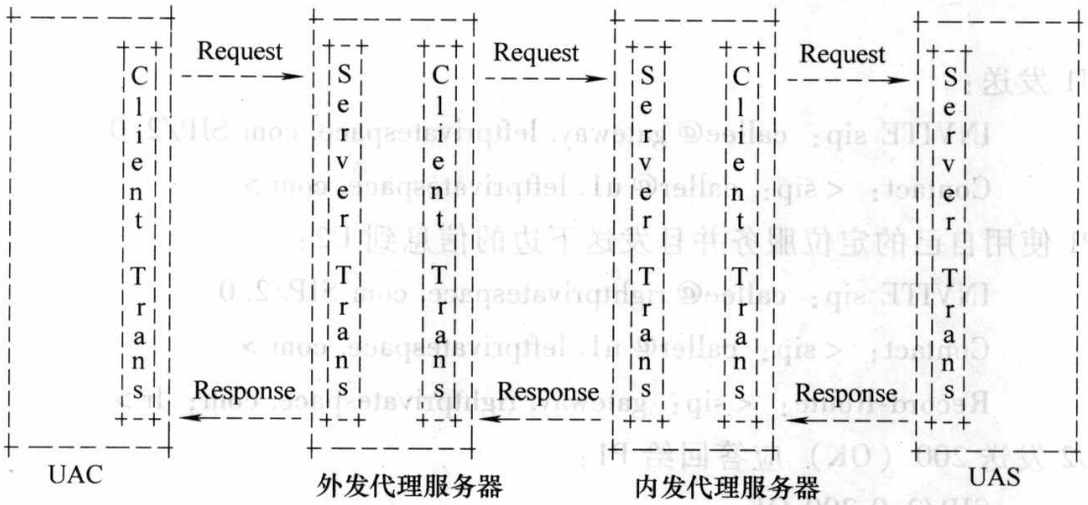


图 10-6 事务关系图

一个无状态的代理服务器没有客户端或服务器端事务功能。客户端事务负责从上层实体（称为事务用户（TU），可以是 UA 或者有状态的代理服务器）接收请求，并将请求可靠地发送到服务器端事务；它也负责接收响应并将响应传送给 TU，同时过滤重复的响应和一些非法的响应，如对 ACK 的响应。另外，对于 INVITE 请求的非 2xx 的最终响应，客户端事务还负责产生相应的 ACK 响应。

服务器端事务负责从传输层接收请求并发送给上层 TU，同时滤掉从网络上收到的重复请求。它也负责从 TU 接收响应并将其发送给传输层。在 INVITE 请求事务中，服务器端事务负责吸收对非 2xx 的最终响应的 ACK。

2xx 的响应的重传和和对应的 ACK 请求的产生是由 UA 核心完成的，由 UAS 重传 2xx 响应，UAC 负责产生对应的 ACK。这种端到端的处理使呼叫者能够了解接受呼叫的所有用户的完整列表。所以在路径上的每个代理服务器都仅仅是转发 INVITE 的 2xx 响应以及对应的 ACK。

10.6.1 客户端事务

客户端事务通过维护一个状态机来实现相应的功能。上层 TU 与客户端事务通过一个简单的接口进行通信，当 TU 初始化一个新的事务时，它产生一个新的客户端事务实例，要发送的 SIP 请求传送给实例，并带上要发往目的地的 IP 地址、端口号和传输协议等参数。客户端事务实例开始执行状态机，同时将有效的响应上传给 TU。

根据 TU 发送的请求的方法类型，有两种客户端事务实例的状态机。一种是处理 INVITE 请求的客户端事务实例，称为 INVITE 客户端事务；另一种是处理除 INVITE 和 ACK 之外的所有请求的客户端事务实例，称为非 INVITE 客户端事务。没有对应于 ACK 的客户端事务存在。如果 TU 要发送一个 ACK，它将直接

将对应的请求发送给传输层。

1. INVITE 客户端事务

（1）INVITE 事务概述

INVITE 事务由三次握手组成：客户端事务发送 INVITE 请求，服务器端事务发送响应，然后客户端事务发送 ACK 消息。对于 UDP 等不可靠传输，客户端事务开始以 T1 的间隔重传请求，并在每次重传后将定时将间隔加倍。T1 是往返时间（RTT）的估计值，默认为 500ms。RFC3261 所涉及的大部分事务定时器都是以 T1 为衡量标准的，可以通过更改 T1 的值来更改其他定时器的值。对于可靠传输，请求不进行重传，在收到 1xx 的临时响应后，停止重传，客户端事务等待进一步的响应。对于不可靠传输服务器端事务可以发送另外的 1xx 响应，最后发送最终响应。对于不可靠传输，响应周期性的进行重传，而对于可靠传输，响应只传输一次。对于每个收到的最终响应，客户端事务都发送 ACK 消息，以便使对端停止重传响应。

（2）状态机模型

INVITE 客户端事务的状态机如图 10-7 所示。在 TU 用 INVITE 请求来生成新事务时，客户端事务首先进入 Calling 状态，同时客户端事务将请求传送给传输层进行网络传输。如果使用不可靠传输协议，客户端事务必须用 T1 启动定时器 A；如果使用可靠传输协议，客户端事务不必启动定时器 A。无论何种情况，事务都必须用 64 × T1 启动定时器 B（定时器 B 控制整个事务的超时）。

如果定时器 A 超时，则客户端事务必须将请求进行重传，然后必须将定时器的间隔设为 2 × T1 进行更新并重新启动。若 A 在 2 × T1 秒后超时，则继续重传请求，同时 A 的间隔时间继续加倍，如此反复。上述重传只在“Calling”状态下进行。T1 的值可根据具体的网络性质进行一定地调整，但上述的重传过程必须遵循。

如果在客户端事务处于“Calling”状态的情况下，定时器 B 超时，则客户端事务应该通知 TU 超时的发生。客户端事务层不能产生 ACK。定时器 B 的值在不可靠传输的情况下刚好等于重传 7 次请求的时间。

如果客户端事务在“Calling”状态接收到一个临时应答，那么就把状态切换到“Proceeding”状态，客户端事务不应当再次重新发送请求了。进一步说，临时应答必须传送给 TU。在“Proceeding”状态的任何临时应答都必须传送给 TU。

当在“Calling”或者“Proceeding”状态的时候，如果接收到一个应答码是 300 ~ 699 的应答，那么就必须把状态切换到“Completed”。客户端事务必须把收到的应答传给 TU，并且客户端事务必须产生 ACK 请求，即使通信层是可靠传输的，并且把 ACK 交给传输层进行传送。ACK 必须和原始请求发送到相同的地

址、端口和用同样的传送器。当客户端事务进入“Completed”状态的时候，应当开始一个定时器 D，默认值是在非可靠通信上是至少 32s，在可靠通信上是 0s。定时器 D 反应了服务端事务在非可靠传输的情况下，在“Completed”状态维持的时间。这个是和 INVITE 请求服务端事务定时器 H 相同的，定时器 H 的默认值是 64 × T1。不过，客户端事务不知道服务端事务使用的 T1 值，所以我们用绝对值 32s 来代替 T1 用作定时器 D 的默认值。在“Completed”状态下，受到的任何终结应答的重传都应当产生一个 ACK 应答到通信层来重新发送，但是新近收到的应答却不能传送给 TU。

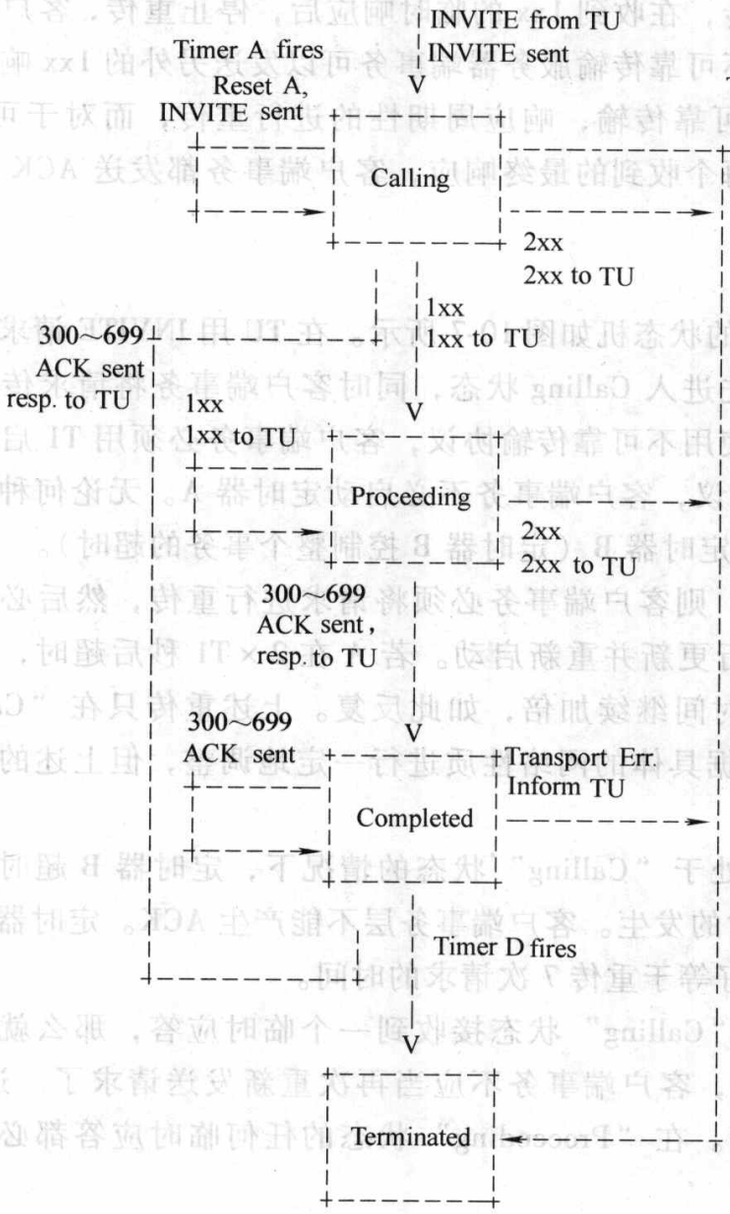


图 10-7 INVITE 客户端事务

如果在客户端事务状态是“Completed”的时候，定时器 D 触发，那么客户端事务必须转到终结状态。当客户端状态是“Calling”或者“Proceeding”状态

的时候，接收到一个 2xx 应答必须导致客户端事务进入“Terminated”状态，并且应答必须交给 TU 处理。处理这个应答的方法依赖于 TU 是否是一个代理服务器核心还是是 UAC 核心。UAC 核心会给应答产生 ACK，代理服务器核心会转发一个 200（OK）应答到上行队列。这个在代理服务器和 UAC 之间，对 200（OK）的不同的处理是导致对应答的处理不在事务层进行的原因。

当客户端事务进入“Terminated”状态以后，客户端事务必须立刻销毁，这样才能保证正确操作。原因是当给一个 INVITE 请求的 2xx 应答的不同处理时，对于代理服务器转发的时候和对 UAC 处理 ACK 的时候是不一样的。因此，每一个 2xx 都需要交给代理服务器核心（只有这样做才能被转发），或者交给 UAC 核心（这样才能被 ACK 确认）。这期间没有事务层的处理。无论应答是否由通信层收到，如果通信层找不到匹配的客户端事务，那么应答就应当交给核心处理。这是由于与之匹配的客户端事务已经被第一个 2xx 应答所销毁，后续的 2xx 应当就匹配不成功了，于是就交给核心来处理。

（3）ACK 请求的生成

事务层生成的 ACK 请求必须与原发送的请求具有同样的 Calll-ID、From 和 Request-URI 头字段。ACK 的 To 头字段必须与对应响应的 To 头字段具有同样的值，一般情况下因为多了 tag 参数而与原请求的 To 头部不同。ACK 请求必须具有单个的 Via 头字段，而且是与原请求的最顶端 Via 头字段具有同样的值。Cseq 头字段必须具有与原请求同样的序列号，且方法必须是“ACK”。

如果原 INVITE 请求具有 Route 头字段，则这些字段也必须出现在 ACK 请求中，这是为了保证 ACK 能被接下来的无状态的代理服务器正确的路由。

RFC3261 建议，在对应非 2xx 最终响应的 ACK 请求中不应包含消息体。如果有消息体的话则分两种情况：对应响应不是 415，ACK 中的消息体类型要严格限制在 INVITE 中包含的消息体类型范围内；如果响应是 415，则 ACK 中包含的消息体类型可以是在响应的 Accept 头字段中所列的消息体类型。

例如对以下的请求：

```
INVITE sip: bob@ biloxi. com SIP/2. 0
Via: SIP/2. 0/UDP pc33. atlanta. com; branch = z9hG4bKkjshdyff
To: Bob <sip: bob@ biloxi. com >
From: Alice <sip: alice@ atlanta. com >; tag = 88sja8x
Max-Forwards: 70
Call-ID: 987asjd97y7atg
CSeq: 986759 INVITE
```

则其对应非 2xx 最终响应的 ACK 请求将如下所示：

```
ACK sip: bob@ biloxi. com SIP/2. 0
```

Via: SIP/2.0/UDP pc33.atlanta.com; branch = z9hG4bKkjshdyff

To: Bob <sip: bob@biloxi.com>; tag = 99sa0xk

From: Alice <sip: alice@atlanta.com>; tag = 88sja8x

Max-Forwards: 70

Call-ID: 987asjd97y7atg

CSeq: 986759 ACK

2. 非 INVITE 客户端事务

(1) 非 INVITE 事务概述

非 INVITE 事务并不使用 ACK。它们只是简单的请求/应答的交互。对于非可靠的通信来说，请求是间隔倍增 T1 的时间重新传输（直到间隔时间达到 T2）。如果收到了一个临时应答，在非可靠通信上，继续重传直到达到 T2。只有当收到重传的请求的时候，服务端事务会重传其发出的最后一个应答，这个应答既可以是临时的应答也可以是终结应答。这就是为什么请求在收到一个临时应答之后还需要一直重传的原因，它们能够确保收到一个终结应答。不像 INVITE 事务，非 INVITE 事务不需要对 2xx 应答做特别处理。UAC 对一个非 INVITE 请求来说，只会产生一个单个的 2xx 应答。

(2) 状态机模型

非 INVITE 客户端事务的状态机与 INVITE 客户端事务的状态机非常相似。在 TU 生成新事务时事务实例进入“Trying”状态，此时应该设置定时器 F（值为 64 × T1 秒），同时将请求发送到传输层进行传输。如果使用不可靠传输协议，则必须设置定时器 E，时间为 T1，如果仍在该状态且 E 已到期，则复位定时器 E，但时间设置为 min（2 × T1，T2）。如果定时器再次到期，则复位 min（4 × T1，T2），依此类推，直到达到 T2，然后将定时器时间设为 T2。T2 的默认取值为 4s。所以对于 T1 和 T2 的取值，这个间隔为：500ms、1s、2s、4s、4s、4s 等。

如果事务在“Trying”状态定时器 F 到期，则事务实例应该通知 TU 该超时事件，同时状态转到“Terminated”状态。若在此状态期间收到临时响应，则事务实例将临时响应发送给上层 TU，然后将实例状态转到“Proceeding”状态；若在此状态期间收到最终响应（状态码为 200 ~ 699），则事务实例必须将响应发送给上层 TU，然后将实例状态转到“Completed”状态。

如果事务在“Proceeding”状态定时器 E 到期，则必须将请求再次发送给传输层进行重传，同时定时器 E 复位，值为 T2 秒。若在此状态定时器 F 到期，则事务实例必须通知 TU 超时事件，并将状态转到“Terminated”状态；若在此状态收到最终响应（状态码为 200 ~ 699），则事务实例必须将响应发送给上层 TU，然后将实例状态转到“Completed”状态。

一旦客户端事务实例进入“Completed”状态，事务必须启动定时器 K，对

于不可靠传输定时时间为 T4 秒，而对可靠传输时间为 0s。“Completed”状态的存在是为了在不可靠传输时吸收重传的响应。而 T4 则代表了等待的时间，T4 的默认值为 5s。若在此状态定时器 K 到期，则事务实例必须立刻迁到“Terminated”状态。

一旦客户端事务实例进入“Terminated”状态，实例必须立刻销毁。其状态迁移过程如图 10-8 所示。

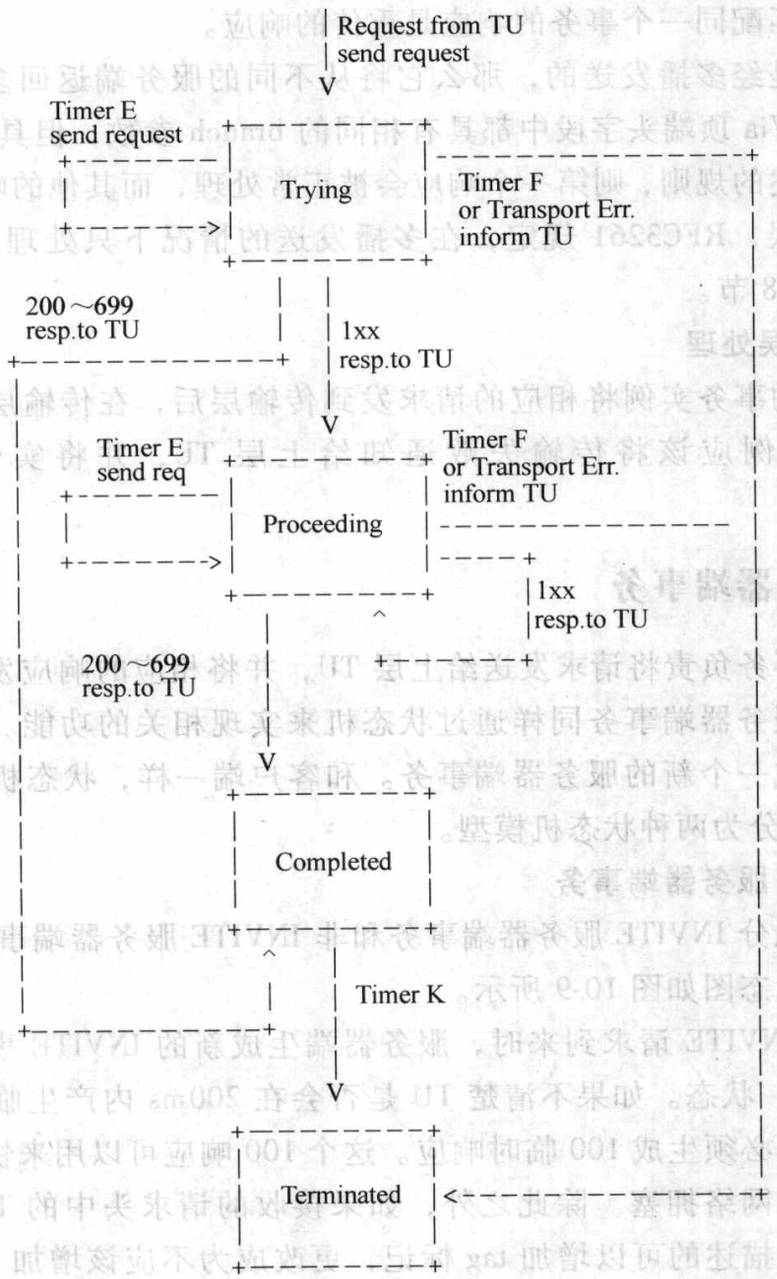


图 10-8　非 INVITE 客户端事务

3. 响应与客户端事务的匹配

当客户端传输层收到一个响应的时候，它必须决定由某个事务来遵循上述过程处理这个响应。在以下两种情况同时满足时，响应和一个客户端事务相匹

配：

1）响应中最顶端 Via 头字段中的 branch 参数与产生此事务的原请求中 Via 顶端头字段中 branch 参数是相同的；

2）响应中 Cseq 头字段中的方法参数与产生此事务的原请求中 Cseq 头字段中的方法参数是相同的。此处需要方法匹配是因为 CANCEL 请求与原请求虽属不同的事务，却具有相同的 branch 参数值。

先后两个匹配同一个事务的响应是重传的响应。

如果请求是经多播发送的，那么它将从不同的服务端返回多个响应。所有的这些响应在 Via 顶端头字段中都具有相同的 branch 参数，但具有不同的 To 标签值。根据上述的规则，则第一个响应会被正常处理，而其他的响应则会视为重传。这并非错误，RFC3261 规定，在多播发送的情况下只处理一个响应消息。具体内容见 10.8 节。

4. 传输错误处理

在客户端的事务实例将相应的请求发到传输层后，在传输层传输失败的情况下事务层实例应该将传输失败通知给上层 TU，并将实例的状态迁为“Terminated”。

10.6.2 服务器端事务

服务器端事务负责将请求发送给上层 TU，并将相应的响应发送给传输层进行网络传输。服务器端事务同样通过状态机来实现相关的功能。在收到一个新的请求时，生成一个新的服务器端事务。和客户端一样，状态机也根据请求是否为 INVITE 而分为两种状态机模型。

1. INVITE 服务器端事务

服务器端也分 INVITE 服务器端事务和非 INVITE 服务器端事务，INVITE 服务器端事务的状态图如图 10-9 所示。

一个新的 INVITE 请求到来时，服务器端生成新的 INVITE 事务实例，并进入“Proceeding”状态。如果不清楚 TU 是否会在 200ms 内产生临时响应或最终响应，事务实例必须生成 100 临时响应。这个 100 响应可以用来快速避免因请求的重传而造成的网络拥塞。除此之外，如果接收的请求头中的 To 头域没有 tag 标志，那么原来描述的可以增加 tag 标记，更改成为不应该增加 tag 标志，这个请求必须交给 TU 处理。

TU 可以发送多个临时响应给服务器端事务实例。在“Proceeding”状态下，事务实例将这些响应发送给传输层进行网络传输，但不影响实例的状态。若在此状态收到重传的请求，则事务实例将最近一次的响应进行重传。若在“Proceeding”状态收到 TU 的 2xx 响应，则事务实例将 2xx 响应发送到传输层进

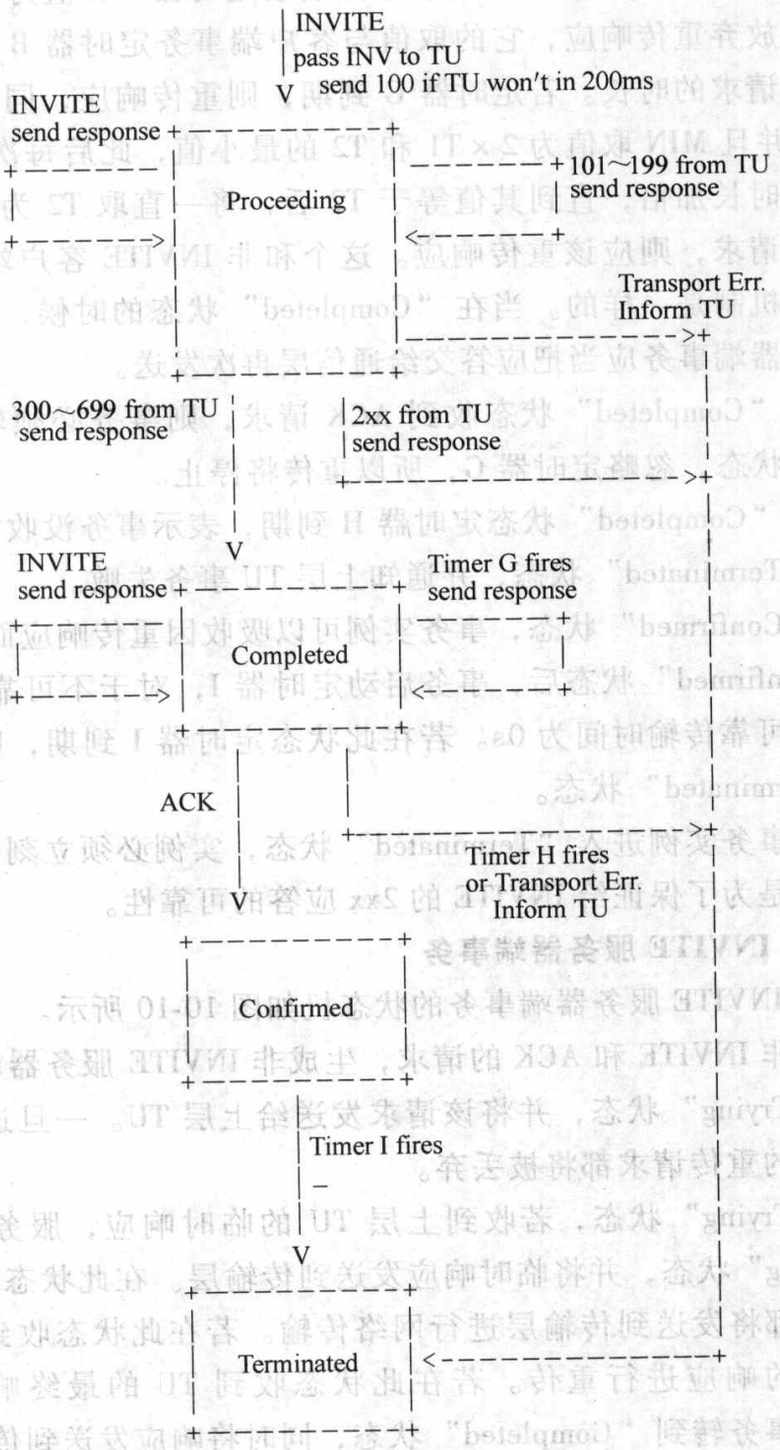


图 10-9　INVITE 服务器端事务

行传输，但并不负责此响应的重传，2xx 响应的重传由 TU 负责。然后事务实例将状态转到“Terminated”状态。若在此状态收到 TU 的 300～699 的响应，则事务实例将响应发送到传输层进行传输，并将状态转到“Completed”状态。对于不可靠的传输，启动定时器 G，值为 T1。对于可靠传输的情况，不设置定时器 G（G=0 的情况就是不设置），这个是和 RFC2543 所不同的，RFC2543 要求应答都要重发，甚至在可靠传输的情况下，也要求应答都要重发。

当进入“Completed”状态，事务启动定时器 H，值为 64 × T1。定时器 H 决定了何时放弃重传响应，它的取值与客户端事务定时器 B 一致，而 B 决定了客户端重传请求的时长。若定时器 G 到期，则重传响应，同时将重新启动 G 设置为 min，并且 MIN 取值为 2 × T1 和 T2 的最小值，此后每次 G 到期都重传响应，并将定时时长加倍，直到其值等于 T2 后，将一直取 T2 为时长。若在此状态收到重传的请求，则应该重传响应。这个和非 INVITE 客户端事务的“trying”请求的重发机制是一样的。当在“Completed”状态的时候，如果接收到重发的请求，服务器端事务应当把应答交给通信层再次发送。

若在“Completed”状态收到 ACK 请求，则事务必须转到“Confirmed”状态。在此状态，忽略定时器 G，所以重传将停止。

若在“Completed”状态定时器 H 到期，表示事务没收到 ACK，则事务将状态转到“Terminated”状态，并通知上层 TU 事务失败。

在“Confirmed”状态，事务实例可以吸收因重传响应而引起的重传的 ACK。进入“Confirmed”状态后，事务启动定时器 I，对于不可靠传输定时时间为 T4 秒，而对可靠传输时间为 0s。若在此状态定时器 I 到期，则事务实例必须立刻转到“Terminated”状态。

一旦事务实例进入“Terminated”状态，实例必须立刻销毁。和客户端事务一样，这是为了保证给 INVITE 的 2xx 应答的可靠性。

2. 非 INVITE 服务器端事务

对非 INVITE 服务器端事务的状态机如图 10-10 所示。

对于非 INVITE 和 ACK 的请求，生成非 INVITE 服务器端事务实例，初始状态机为“Trying”状态，并将该请求发送给上层 TU。一旦进入“Trying”状态，任何收到的重传请求都将被丢弃。

在“Trying”状态，若收到上层 TU 的临时响应，服务器端事务实例进入“Proceeding”状态，并将临时响应发送到传输层。在此状态从 TU 接收到的任何临时响应都将发送到传输层进行网络传输。若在此状态收到重传的请求，则事务将最近的响应进行重传。若在此状态收到 TU 的最终响应（状态码 200 ~ 699），则事务转到“Completed”状态，同时将响应发送到传输层进行发送。

当事务进入到“Completed”状态，必须启动定时器 J，对于不可靠传输值为 64 × T1 秒，对于可靠传输则为 0s。在“Completed”状态，收到重传请求的情况下事务将最后的响应进行重传；并且在“Completed”状态下，任何其他 TU 传递下来给服务器端事务的终结应答都必须被抛弃。服务器端事务保持这个状态直到定时器 J 触发，当定时器 J 触发了以后，服务器端事务必须进入“Terminated”状态。

一旦事务实例进入“Terminated”状态，实例必须立刻销毁。

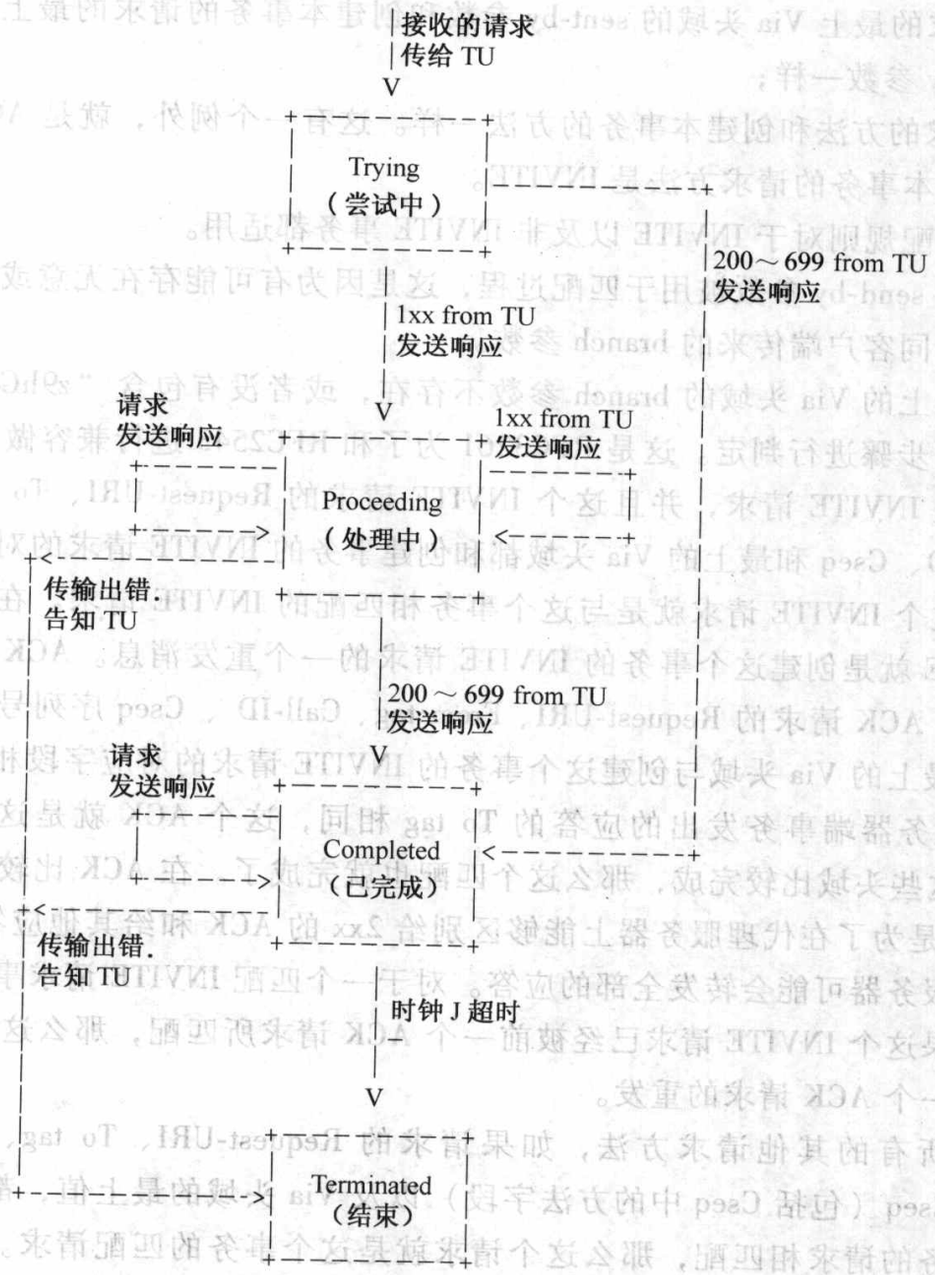


图 10-10　非 INVITE 服务器端事务

3. 请求与服务器端事务的匹配

当服务器端从网络上收到一个请求以后，它必须和现有的事务进行匹配，这个是根据以下的规则来判定的。

首先要检查请求中的 Via 头域的最上一个 branch 参数。如果它以“z9hG4bk”开头，那么这个请求一定是由客户端事务根据 RFC3261 产生的。因此，branch 参数在该客户端发出的所有的事务中都是唯一的。根据下列规则我们可以判定请求是否和事务匹配：

1）请求中的最上 Via 头域的 branch 参数和创建本事务的请求的最上的 Via 头域的 branch 参数一样；

2）请求的最上 Via 头域的 sent-by 参数和创建本事务的请求的最上的 Via 头域的 send-by 参数一样；

3）请求的方法和创建本事务的方法一样。这有一个例外，就是 ACK，ACK 对应的创建本事务的请求方法是 INVITE。

上述匹配规则对于 INVITE 以及非 INVITE 事务都适用。

其中的 send-by 参数被用于匹配过程，这是因为有可能存在无意或恶意的相同的或者不同客户端传来的 branch 参数。

如果最上的 Via 头域的 branch 参数不存在，或者没有包含“z9hG4bk”，那么就用下列步骤进行判定。这是 RFC3261 为了和 RFC2543 进行兼容做的规定。

如果是 INVITE 请求，并且这个 INVITE 请求的 Request-URI、To tag、From tag、Call-ID、Cseq 和最上的 Via 头域都和创建事务的 INVITE 请求的对应字段匹配，那么这个 INVITE 请求就是与这个事务相匹配的 INVITE 请求。在这个情况下，INVITE 就是创建这个事务的 INVITE 请求的一个重发消息。ACK 请求在匹配事务时，ACK 请求的 Request-URI、From tag、Call-ID 、Cseq 序列号（非方法字段）和最上的 Via 头域与创建这个事务的 INVITE 请求的对应字段相同，并且 To tag 和服务器端事务发出的应答的 To tag 相同，这个 ACK 就是这个事务的 ACK。当这些头域比较完成，那么这个匹配也就完成了。在 ACK 比较中包含 To tag 的比较是为了在代理服务器上能够区别给 2xx 的 ACK 和给其他应答的 ACK，这个代理服务器可能会转发全部的应答。对于一个匹配 INVITE 请求事务的 ACK 请求，如果这个 INVITE 请求已经被前一个 ACK 请求所匹配，那么这个 ACK 请求就是上一个 ACK 请求的重发。

对于所有的其他请求方法，如果请求的 Request-URI、To tag、From tag、Call-ID、Cseq（包括 Cseq 中的方法字段）以及 Via 头域的最上值，都和创建服务器端事务的请求相匹配，那么这个请求就是这个事务的匹配请求。匹配是基于针对每一个头域值的判定进行的。当非 INVITE 请求和现有事务匹配了，那么它就是创建这个事务的请求的一个重发。

由于匹配规则中包含了 Request-URI，服务器不能匹配应答对应到事务，所以当 TU 传送了一个应答到服务端事务，它必须为这个应答指定传送到那个服务器端事务。

4. 传输错误处理

在服务器端的事务实例将相应的响应发到传输层后，在传输层传输失败的情况下，可进行如下处理：首先，可以选择将响应发往另一个作为备份的实体（参见 RFC3263）。

如果上述尝试都传输失败，事务层应该将失败通知给 TU，并将相应的事务实例的状态转为 TERMINATED。

10.6.3　定时器

无论是客户端事务，还是服务器端事务，在它们的状态转移中都存在定时器，有些状态的转移是由于定时器超时引起的，下面给出各定时器默认值及意义，如表 10-2 所示。

表 10-2　定时器默认值

定时器	值	意　义
T1	500ms 默认	预估的 RTT 时间
T2	4s	给非 INVITE 请求和 INVITE 应答的最大重传时间间隔
T4	5s	最大网络传送消息时间
定时器 A	初始值 T1	INVITE 请求重传间隔，仅 UDP
定时器 B	64×T1	INVITE 请求超时时间
定时器 C	>3min	proxyINVITE 请求超时时间
定时器 D	>32s UDP 0 TCP/SCTP	应答重发的等待时间
定时器 E	初始值 T1	非 INVITE 请求重传间隔，仅 UDP
定时器 F	64×T1	非 INVITE 请求事务超时时间
定时器 G	初始值 T1	INVITE 应答重传间隔
定时器 H	64×T1	等待 ACK 的时间
定时器 I	T4 UDP 0 TCP/SCTP	ACK 重传的等待时间
定时器 J	64×T1 UDP 0 TCP/SCTP	非 INVITE 请求重传的等待时间
定时器 K	T4 UDP 0 TCP/SCP	应答重传的等待时间

10.7　SIP 在传输层上的通信处理

传输层负责通过网络传送请求和响应，当传输为面向连接时，还要确定请求或者响应可用的连接。传输层可以管理永久连接，这些连接使用的传输协议有 TCP 和 SCTP，或者基于 TCP 和 SCTP 的 TLS，包括那些开放的传输层的协议。这些连接还包括客户机或者服务器传输层的开放连接，客户机和服务器传输层可以共同使用该连接，可以通过该连接的远端地址、端口、传输协议所组成的

三元组来检索该连接。如果一个连接被传输层开放，该索引则设成目的地 IP 地址、端口和传输协议的三元组；如果一个连接被传输层所接受，该索引则设成源 IP 地址、端口和传输协议的三元组。由于源端口通常是暂时的，所以传输层所接受的连接一般不会重新使用，所以通常两个对等代理服务器在使用面向连接传输协议时，会使用两个连接，用于各自方向发起的事务。

RFC3261 建议，在最后一个消息通过之后，该连接仍需要开放一段时间。该时间段至少等于实体将一个事务从实例进行到终止状态所用时间的最大值。这样，事务可以在其发起的连接完成（例如请求、响应、INVITE 、用于非 2xx 响应的 ACK）。该时间段的值至少为 64 × T1。不过，如果当本程序的 TU 使用的是一个比较大的定时器 C 的时候，也可以选取一个比较大的值。所有的 SIP 实体都必须能实现 UDP 和 TCP，SIP 实体也可以使用其他的协议。

与 RFC2543 不同，RFC3261 中强制 UA 使用 TCP，这不是仅因为必须使用 TCP 处理大块的消息，即使一个实体从不发送大块消息，它也必须具备接收这样的消息并处理它的能力。

10.7.1 客户端

1. 发送请求

传输层的客户端负责发送请求并接收响应。传输层的用户将请求、IP 地址、端口、传输协议以及用于组播目的地的 TTL 传递到客户机传输层。

如果一个请求在小于 200KB 或者大于 1300KB 的路径 MTU 中，而该 MTU 值未知，那么该请求必须使用 RFC 2914 中规定的拥塞控制协议，例如 TCP。如果这与顶端 Via 中所指定的传输协议不同，那么顶端 Via 的值需要修改。这有效地防止了使用 UDP 会造成的消息碎片，并且为大块的消息提供了拥塞控制。对于 UDP 来说能够处理的最大数据包就是 65535KB 并包括 IP 和 UDP 字头。

消息大小和 MTU 之间的 200KB 的“缓冲区”适应了 SIP 中响应消息一般比请求大的情况。例如，可以在 INVITE 响应中附加了 Record-Route 头字段值。有了这个额外的缓冲区，响应可以比请求大 170KB 而在 IPv4 中不必被分块（假设没有 IPSec 时，IP/UDP 消耗大约 30KB）。当路径 MTU 未知时，基于以太网的 MTU 是 1500KB 来将 MTU 设为 1300KB。

如果一个实体由于消息大小的缘故才需要通过 TCP 发送某请求，否则该请求可以通过 UDP 传输，如果建立该连接时引起下面两种情况：ICMP 不支持和重新建立 TCP 连接，此时该实体就应该使用 UDP 重新发送请求。这是 RFC3261 为了兼容 RFC2543 中不支持 TCP 的要求。

向组播地址发送请求的客户机必须在 Via 头字段值中附加 maddr 参数指明目的地组播地址，对于 IPv4 来说应该增加一个值为 1 的 ttl 参数，IPv6 的组播在

RFC3261 规范中没有作规定。这个规定对 SIP 组播产生了限制。它的基本功能是“single-hop-discovery-like”业务，也就是向一组同类的服务器发送请求，但是只需要其中一个服务器处理该请求即可。注册时可以使用该功能。在 10.6.1 客户端事务处理的规则中规定，客户机事务层应该接收第一个请求，其余的因为包含同样的 Via 分支标识符而被视为重发。

发送请求之前，客户端事务层必须在 Via 头字段中插入一个“sent-by”字段。该字段包括一个 IP 地址或者是主机名和端口。RFC3261 推荐使用 FQDN 方法描述这个主机名。如果没有规定端口，根据传输协议不同而取不同的端口默认值，UDP 取 5060 而 TCP 和 SCTP 取 5061。

在进行可靠传输时，应在收到请求的连接上发送响应。因此，客户端传输层必须准备在同一个连接上接收响应。发生错误时，服务器可以建立一个新连接发送响应。在这种情况下，传输层也必须在源 IP 地址上引入一个连接，从源 IP 地址和“sent-by”字段中所指的端口发送请求。在不可靠的单播传输时，客户端传输层必须准备在源 IP 地址上接收响应，源 IP 地址就是发送请求的地址，端口号为“sent-by”字段中的值。另外，与可靠传输一样，某些特定情况下，响应也将被传输到别处。客户机还必须准备在服务器依照 RFC3263 的第 5 章中程序选择的地址和端口上接收响应。

在组播的情况下，客户端传输层必须作为发送请求的组播群中的一员在与发送请求相同的组播群和端口上接收响应。

如果一个原有的连接对于某请求的目的 IP 地址、端口和传输协议开放，则 RFC3261 建议这个连接用于发送请求而另外一个连接也可以开放并使用。

如果某请求使用组播发送，它就会被送往一组用户所提供的地址、端口和 TTL；如果某请求使用不可靠的单播传输方式发送，它就会被送往传输用户所提供的 IP 地址和端口。

2. 接收响应

客户机传输层收到响应后需要检查顶端 Via 的值。如果“sent-by”参数中的值与客户端传输层插入到请求中的值不一样，该响应就必须被丢弃。

如果存在一些客户端事务，客户机传输层将在原有的事务中查找与该响应所匹配的事务。如果找到匹配事务，就必须将该响应送到对应的事务中处理，反之该响应就必须被送到核心服务器（无论是无状态代理服务器还是有状态代理服务器还是 UA）做进一步处理。根据核心服务器的不同，对这些响应有不同的处理。

10.7.2　服务器端

1. 接收请求

服务器可以通过 DNS 来搜索对方服务器，搜索结果以 SIP 或者 SIPS URI 形式给出，服务器应该通过搜索结果中的任何 IP 地址、端口和传输协议来接收请求。另外，必须将一个 URI 放到 REGISTER 请求或者重定向响应中的 Contact 字段，或者放在请求或响应中的 Record-Route 头字段中，URI 也可以用放到网页或名片上的形式给出。RFC3261 建议，服务器在公共接口上接收请求是应使用默认的端口值（TCP 和 UDP 是 5060，而 TCP 上的 TLS 是 5061）、私有网络或者同一个主机上运行多个服务器的情况除外。因为如果消息太大则必须使用 TCP 而不能用 UDP 传输，所以任何使用 UDP 端口和接口的服务器必须能够在 TCP 上使用同样的端口和接口。反之，服务器不必因为使用了 TCP 的一个特定的端口和接口就要求在 UDP 情况下使用同一特定端口和接口。服务器传输层通过任何传输协议收到请求后，必须检查 Via 头字段顶端的 sent-by 参数的值。如果 sent-by 参数的主机位置包含一个域名或者包含一个不同于数据包源地址的 IP 地址，服务器必须在 Via 头字段中添加一个 received 参数。该参数必须包括所收数据包的源地址，这样，服务器传输层就可以将响应发送到发出请求的源 IP 地址。所以这个可以用来帮助服务端通信层发送应答。

服务器传输层收到的请求，部分如下：

INVITE sip：bob@ Biloxi. com SIP/2. 0

Via：SIP/2. 0/UDP bobspc. biloxi. com：5060

在源 IP 地址 192. 0. 2. 4 处收到该请求。在传输层继续传输请求之前必须添加一个 received 参数，因此该请求部分如下：

INVITE sip：bob@ Biloxi. com SIP/2. 0

Via：SIP/2. 0/UDP bobspc. biloxi. com：5060；received = 192. 0. 2. 4

然后，服务器传输层将在服务器端事务中寻找与请求相匹配的事务。如果找到了对应的服务器事务，就将请求送到这个事务来处理。如果没有找到相匹配的事务，就将请求送到核心服务器，并建立一个新的服务器端事务来处理它。如果 UAS 核心服务器对 INVITE 请求发出 2xx 响应，服务器事务即终结。这就意味着如果 ACK 到达，将没有相匹配的服务器事务，基于上述原则，ACK 就被送往 UAS 核心进行处理。

2. 发送响应

服务器传输层使用 Via 字段的顶端值来决定将响应发送的目的。具体过程如下：

1）如果“sent-protocol”为可靠的传输协议，如 TCP 或者 SCTP，或者基于 TCP 或 SCTP 的 TLS，且现有的连接仍然可用，则必须使用该连接传输响应到建立该事务的初始请求的源地址。这就要求服务器传输层保持服务器端事务和传输连接的关联。如果该连接不可用，服务器应与 received 参数中的 IP 地址之间

开放一个新连接，并使用“sent-by”中的端口值，如果没有指定端口就使用该传输协议的默认端口值。如果该连接失败，服务器应该根据 RFC3263 中的程序确定所要开放的连接的 IP 地址和端口并向其发送响应消息。

2）否则，如果 Via 字段包含 maddr 参数，就必须将响应转发到参数中所列的地址，所用端口为“sent-by”中指定端口值，如果没有指定，端口值为 5060。如果地址为组播地址，就应该使用 ttl 参数中指定的 TTL 值来发送响应，如果该参数不存在，TTL 的值为 1。

3）不可靠的单播传输的情况下，如果 Via 含有 received 参数，则响应就被送往 received 中指定的地址，所用端口为“sent-by”中指定端口值，如果没有指定，端口值就为 5060。如果传输失败，例如得到 ICMP“端口不可达”的响应，就应该使用 RFC3263 中的 S 程序来确定响应发送的目的。

4）否则，如果没有接收者标签，就应该将响应送到“sent-by”中指定的地址。

10.7.3　数据帧与错误处理

1. 数据帧

面向消息传输的情况下（例如 UDP），如果消息包含 Content-Length 头字段，则消息体则被应为该字段所指示的长度，超长的数据包必须被丢弃，如果传输的数据包在消息体结束之前结束，则产生错误。

如果消息是一个响应，则它必须被丢弃；如果消息为一个请求，实体就应该产生一个 400（Bad Request）响应。如果该消息没有 Content-Length 头字段，那么消息体的长度就认为是所传输的数据包的长度。

面向流传输的情况下（例如 TCP），Content-Length 头字段指示消息体的大小。面向流传输的情况下必须使用 Content-Length 头字段。

2. 错误处理

如果传输的用户通过不可靠传输协议传输消息，并产生 ICMP 错误，则按照 ICMP 错误的类型确定处理方法。如果为主机、网络、端口或者协议不可用的错误或者参数错误，应由传输层通知用户。源终止和 TTL 超时的 ICMP 错误都将忽略。如果传输的用户要求通过可靠传输协议传输消息且连接失败，那么传输层应该通知用户。

第 11 章　SIP 的安全机制

11.1　SIP 遭受的攻击和威胁模式

安全问题在现代通信网中变得越来越重要，而 SIP 本身是一个不容易进行安全保护的协议。其中介的使用（注册服务器、重定向服务器等）、多方信任关系和以后的在网络实体之间交互的互不信任的关系，以及用户与用户的操作使得安全难以保证。SIP 是基于文本的，没有通过加密容易被篡改，容易对 SIP 注册机制进行攻击，以及遭到拒绝服务等攻击。SIP 的终端是运行在网络中的，除了协议自身的安全隐患外，还有承载网的安全问题，在这里我们只讨论针对 SIP 本身安全威胁。

11.1.1　注册攻击

SIP 注册机制就是用户终端向注册服务器登记终端用户在何处可以被访问。注册服务器通过查看注册消息的 From 字段来决定该消息是否有权限修改相应的注册用户的注册地址。由于 From 字段可以被 UA 任意修改，这就为恶意注册开了后门。攻击者通过修改 From 字段就可以成为授权用户，进行恶意注册。攻击者可以取消原先已经有的合法用户的注册，将自己的设备注册成某用户的当前可访问的位置，这样一来，就将被修改者所有的请求都定向到攻击者的设备上了。这类威胁表明需要一类使得 SIP 实体能够认证请求发送者的身份的安全机制。

11.1.2　伪装服务器

攻击者通过伪装服务器而达到攻击目的。例如，用户发往重定向服务器的请求被攻击者截获，并假冒成该重定向服务器向请求者发送一条伪造消息，将用户的请求定向到不正确或者不安全的地方。攻击者只需要将应答的 From 字段改成正确的重定向服务器就可以达到伪造服务器的目的。这类威胁表明需要一类使得 SIP 用户能够认证发送请求所对应的服务器身份的安全机制。

11.1.3　篡改消息

用户终端可以通过代理服务器 SIP 消息正确路由到目的用户，但不能检查 SIP 消息是否被恶意修改。攻击者可以通过修改 SIP 消息体携带的媒体会话的加密密钥而扮演中间人；攻击者还可以尝试修改 SDP 消息体，将 RTP 媒体流指向

窃听器，以达到在后续的语音通信中窃听的目的。除了修改消息体之外，攻击者还可能修改消息的头字段。有些 SIP 消息的头字段对目的用户特别有意义，如主题字段，恶意攻击者可能通过修改 subsequent 将重要的请求改成了次要请求。

所以用户需要一种安全机制来检查消息是否被篡改。但是，中间服务器由于路由的缘故，必须修改某些头字段，所以只能保护 SIP 消息体和一些能够保护的字段。这种安全服务对消息体来讲，包括保密性、完整性和身份认证。

11.1.4　恶意修改以结束会话

SIP 通过 BYE 请求来结束会话。攻击者能够伪造 BYE 请求，一旦伪造的 BYE 请求被接收者收到，会话就被过早地结束。类似的威胁包括发送伪装的 Re-INVITE 请求来改变会话，如改变会话的安全性或者将媒体流重定向为窃听攻击的一部分。攻击者之所以能够伪造这两种请求，是因为捕获了会话建立阶段会话双方的初始消息，从而获得了标志会话的参数（To 字段、From 字段和 Call-ID 字段等）。如果这些标志参数通过加密传输的话，攻击者就不能伪造出 Re-INVITE 请求，但中间的代理服务器在会话建立阶段需要检查这些字段。

11.1.5　拒绝服务与放大

DoS（Denial of Service，拒绝服务）是指特定的网络服务的中断。通常是通过向特定的网络接口发送大量的信息而使系统破坏或暂时不可用。分布式拒绝服务攻击，是指一个网络用户使多台网络机器同时向同一台目的主机发送大量信包，而使得目的主机不能够提供服务。为了接收世界范围内的终端请求，SIP 的代理服务器必须安装在公共的 Internet 上，这就为 SIP 遭受分布式拒绝服务攻击创造了许多潜在的机会。

SIP 主机除了作为直接被攻击的对象之外，还有可能作为 DoS 攻击的帮凶，起到放大 DoS 攻击的作用。攻击者如果想攻击网络中的某台主机，可以通过伪造出一条看似来自这台主机的 SIP 消息，然后将这条消息发往大量的 SIP 网络中的服务器，通过 SIP 服务器向被攻击的目的主机发送大量的 SIP 应答消息，而使得目的主机遭受拒绝服务攻击。攻击者还可以通过伪造 SIP 消息的路由字段，使得要攻击的主机成为 SIP 消息路由路径的下一跳，然后将消息发往代理服务器，从而使得代理服务器放大了对目的主机的攻击。

如果注册服务器没有很好地认证注册请求的话，也会遭受拒绝服务的攻击。攻击者可以通过伪造的注册请求而取消用户的注册，从而使得被取消注册的用户不能被其他用户邀请参与会话，因为这些被取消注册的用户没有了当前位置信息。攻击者还可以通过为某个特定的（Address Of Record）注册了大量的联系地址在被攻击的主机上，而使得注册服务器和相关的代理服务器起到放大拒绝

服务攻击的作用。攻击者还可以尝试通过注册大量的地址绑定来耗尽注册服务器的内存和硬盘资源。多点传送也增加了拒绝服务攻击的可能性。

11.2 SIP 的安全机制

通过上述的威胁模型，可以得出 SIP 安全对策必须提供基本的安全服务：保证消息的机密性和完整性；防止重放攻击或者消息欺骗；提供会话参与者的认证和隐私；防止拒绝服务攻击。

SIP 所需的基本安全服务包括消息的机密性及完整性保留，重发攻击或消息欺骗的预防，对会话参与者提供保密及鉴权服务，以及预防拒绝服务攻击。SIP 消息中的每个部分各自需要其机密性、完整性和鉴权的安全服务。

SIP 除了使用新的专用的安全机制之外还重用了那些来自于 HTTP 和 SMTP 的已有的可用的安全模式。

消息的全加密是为信令传送提供机密性的最好的方法——它也可以保证消息不被恶意的中介修改。但由于在大多数的网络结构中为了正确路由 SIP 请求，消息中如 Request-URI、Route 和 Via 这样的字段对代理应该是可见的，因此 SIP 请求和响应消息不能被简单地进行全部的端到端的加密。代理服务器也需要修改消息的某些特性如增加 Via 头字段的值等，来完成某种功能。因此，UA 必须信任代理服务器。RFC3261 建议使用低层的安全机制，即在传输线上对整个的 SIP 请求或响应进行逐跳的加密，这样就可以让端点校验其请求发向的代理服务器的身份了。

SIP 标准中的安全机制，从体系结构上看，可以分为端到端和点到点的保护。端到端保护是指在一个 SIP 会话的主叫和被叫用户代理之间提供安全机制，SIP 标准中通过认证和加密实现这个安全机制。值得注意的是，在端到端保护中，只有 SIP 消息中的一部分被加密。虽然端到端完全加密 SIP 消息能够提供最大程度的安全保障，但这将使 SIP 服务器无法识别或修改 SIP 消息头中的关键域（Request-URI、Rout 及 Via），最终导致无法建立会话，因此点到点的保护机制也是必须的。SIP 本身没有提供更多特性来保证 SIP 实体之间的点到点安全，而是利用了下层的一些安全协议来实现，包括网络层的 IPsec（Internet Protocol security，因特网协议安全）和传输层的 TLS（Transport Layer Security，传输层安全）。总之，SIP 重用了 HTTP 和 SMTP 的安全模型。

11.2.1 传输层和网络层安全

传输和网络层的安全服务包括加密信令流，并保证消息的机密性和完整性。

通常，证书被用于建立低层的安全。在许多网络结构中，这些证书在也可用

来作为一种鉴权的方式。TLS 和 IPSec 分别用于传输层和网络层提供安全服务。

IPSec 是一套网络层协议工具，它可以全部替换传统 IP 的安全策略。它常用在主机或者管理域已经有相互信任关系的网络结构中。IPSec 通常在主机的操作系统或者在为从一个特定接口收到的所有的流量提供机密性和完整性的安全网关上执行。IPSec 也可以基于逐跳的应用。在许多的网络结构中，IPSec 不需要集成在 SIP 应用中，IPSec 适用于难于直接在主机上加入安全配置的网络结构中。与它们的第一跳代理服务器有着共享密钥环的 UA 很适合用 IPSec 保证安全。任何用于 SIP 的 IPSec 配置都需要一个描述 SIP 安全所需的协议工具。

TLS 提供了基于有连接的协议的传输层的安全；“tls”（TLS over TCP）可以在一个 Via 头字段值或 SIP URI 中被指定为所需的传输协议。TLS 适用于在主机间无现存信任关系并需要逐跳安全的网络结构中。例如，Alice 信任她的本地代理服务器，在证书交换之后，该服务器决定信任 Bob 所信任的本地代理服务器，这样 Bob 和 Alice 可以安全地通信。

TLS 必须和 SIP 应用一起实现。SIP 中的传输机制是逐跳指定的，这样在 TLS 上向代理服务器发送请求的 UA 就不能保证 TLS 是可以用于端到端的安全的。

在具体实现 TLS 时必须至少支持 TLS_RSA_WITH_AES_128_CBC_SHA 密码适配集。为了向后兼容，代理服务器、重定向服务器和注册服务器应该支持 TLS_RSA_WITH_3DES_EDE_CBC_SHA。具体实现时，也可以支持任何其他的密码适配集。

11.2.2　SIPS URI 方案

虽然 SIPS URI 方案附属于 SIP URI，但其语义和 SIP URI 还是有很多不同的。SIPS URI 中，资源可以指定自己是安全的。

SIPS URI 可以作为一个特定用户的记录地址，通过该 URI 可以得知这个用户。SIPS URI 方案当于请求消息中作为 Request-URI 时，即表示在该请求到达负责 Request-URI 中的那部分域的 SIP 实体之前的前向上的每一跳，且必须以 TLS 保证安全；如果它到达一个可疑的域时，它将和本地的安全与路由政策相协调，可能任何最后一跳上使用 TLS。SIPS 在用于请求发起者侧时，如发起者用 SIPS URI 作为目标记录地址，它将指定到达目标域的整个请求路径的安全。

SIPS URI 方案可以用于记录地址、联系地址（包括那些 REGISTER 方法，Contact 头的内容）和 Router 头字段中。在每种情况中，SIPS URI 方案都允许这些现有的字段来指定安全资源。上述任何情况下，任何解除 SIPS URI 的引用的方式都有着自己的安全特性。

RFC3261 协议规定，SIPS 的用法应该使用相互的 TLS 鉴权方式，如密码适

配组 TLS_RSA_WITH_AES_128_CBC_SHA 所要求，在鉴权过程中收到的证书应该与客户所持的根证书一起被证实；证书证实失败应该导致请求的失败。

注意到在 SIPS URI 方案中，传输是独立于 TLS 的，虽然 UDP 不是有效的 SIPS 传输方式这样

" sips：alice@atlanta.com；transport = tcp" 和

" sips：alice@atlanta.com；transport = sctp"

就都是有效的。RFC3261 协议不建议使用" transport = tls"，其中部分原因是它只是针对逐调请求能起到安全作用。把 SIPS URI 作为记录地址分发出去的用户，可以选择那些会拒绝非安全传输请求的设备。

11.2.3 HTTP 鉴权

SIP 中可以使用基于 HTTP 鉴权的质询，该鉴权是依靠 401 和 407 响应代码和携带口令和证书的头字段来进行的。HTTP 分类鉴权方案在 SIP 中无需作大的改变就可以重用，并可完成重发保护和单向鉴权。

SIP 提供一种无状态的基于质询的并基于 HTTP 的鉴权机制。代理服务器或者 UA 在任何时候收到用户发来的请求时都可以向请求者质询鉴权。一旦请求方身份证实，接收方就能够确信该用户是否被授权发出请求。

分类（Digest）鉴权机制只提供了消息的鉴权和重发保护，不涉及到保证消息的完整性和机密性的内容。上文中的保护措施不是分类鉴权所提供的，用来阻止攻击者修改 SIP 请求和响应。

由于基本（Basic）鉴权机制因其安全薄弱性而不再使用。服务器不可以接受使用“Basic”机制的证书，而且服务器也不能使用“Basic”机制发出质询。这 RFC3261 与 RFC2543 不同的地方。

1. 框架

SIP 鉴权的框架类似 HTTP，尤其是 auth-scheme、auth-param、challenge、realm、realm-value，以及 credential 这些的 BNF 语法都是一样的。RFC3261 协议中规定，UAS 使用 401（未鉴权）响应询问 UAC 的身份。注册服务器和重定向服务器也使用该响应要求鉴权。只有代理服务器使用的是 407（代理服务器要求鉴权）响应。在不同的消息中可能分别要求包含 Proxy-Authenticate、Proxy-Authorization、WWW-Authenticate 和 Authorization 头字段，同 RFC2617。

由于 SIP 中没有标准根 URL 的概念，SIP 中保护空间的概念就不同。RFC3261 协议中，realm 字符串定义了保护空间，而在 RFC2543 中则是由 Request-URI 和 realm 共同定义保护域。由于 UAC 发送的 Request-URI 与发出质询的服务器收到的 Request-URI 可能不同，而且实际上 UAC 可能并不知道该 URI 的最终格式，而且在 realm 字符串与 Request-URI 共同定义的保护域的方式中，

依赖于 Request-URI 中的 SIP URI，这样不能使用别的 URI 方案，例如 tel URI。

UA 或者代理服务器要求对所收到的请求进行鉴权，服务器创建一个 realm 字符串的规则如下：

1）realm 字符串必须是惟一的。RFC3261 规范建议，一个 realm 字符串包含一个主机名或一个域名，这部分内容参见 RFC2617。

2）realm 字符串应该包含一个可读的标识符并可以提交给用户。

举例如下：

INVITE sip：bob@ biloxi. com SIP/2. 0

Authorization：Digest realm = " biloxi. com" , <. . . >

一般，SIP 鉴权只对某个特定的保护域有意义。对于分类鉴权，每个这样的保护域都有自己的一组用户名和密码。如果服务器对某请求不要求鉴权，它可以接受一个默认的用户名“anonymous”并且密码为空。同样，UAC 代表许多用户，例如 PSTN 网关，可能有自己的特定设备的用户名和密码，而不是本域内特定的用户账号。

服务器可以合法地质询大多数 SIP 请求，但 ACK 和 CANCEL 需要特殊鉴权。目前的鉴权机制如分类鉴权，要求响应应带有用于计算 nonce 值的数值，这样没有得到响应的请求，包括没有得到 ACK 响应的请求会产生某些问题。因此，收到 ACK 响应的服务器一定要接受 INVITE 请求的证书。发出 ACK 响应的 UAC 将 INVITE 请求中的 Authorization 和 Proxy-Authorization 头字段的值复制到 ACK 响应的对应的字段中。服务器不可以向 ACK 响应发出质询。

尽管 CANCEL 会得到 2xx 响应，但是由于该请求不能重新提交，因此服务器也不能向 CANCEL 提出质询。一般情况下，如果某一个代理服务器发出的请求被取消，那么服务器应该接受该代理服务器发出的 CANCEL 请求。

如果 UAC 收到一个质询，而 UAC 的设备还不知道被质疑的 realm 中的证书，那么该 UAC 应该将询问（WWW-Authenticate 头字段或者 Proxy-Authenticate 字段中）中的“realm”参数的内容提交给用户。当询问一个预配置的设备时，预先给 UA 的 realm 配置证书的业务提供者应该知道该用户是否给该 realm 提供自己的证书。

即使一个 UAC 可以找到与适当的 realm 有关的证书，也可能出现以下情况：证书无效或者发出质询的服务器由于某种原因不再接受这些证书。这种情况下服务器可能会重复质询，或者发出 403（禁止）响应。UAC 不可以使用被拒绝过的证书重新发出请求。

2. 用户间鉴权

在 UAS 处理所收到的 UAC 的请求之前，它可以对请求者进行鉴权。如果请求中没有在 Authorization 头字段中提供证书，UAS 则可以使用 401 状态码拒绝请

求，并请求者发起质询要求其提供证书。

401 响应消息中必须有 WWW-Authenticate 头字段。该字段的至少包含一个质询来指定适用于该 realm 的鉴权机制和参数。

401 质询中的 WWW-Authenticate 头字段示例如下：

WWW-Authenticate: Digest
realm = " biloxi. com",
qop = " auth, auth-int",
nonce = " dcd98b7102dd2f0e8b11d0f600bfb0c093",
opaque = " 5ccc069c403ebaf 9f 0171e9517f40e41"

当请求发起 UAC 收到 401 响应后，它应该使用合适的证书重新发起请求。UAC 也可以在处理请求之前要求用户直接输入证书。如果用户直接提供证书或者 UA 可以从内部关键词字符串中找到证书，UA 就应该把证书以一个给定的值缓存到 To 头字段和“realm”字符串中，并且在下一个请求中重用该值。如果找不到某个域的证书，UAC 可以使用“anonymous”的用户名和空密码重试该请求。

找到证书之后，UA 如果可以在请求中包含一个 Authorization 头字段完成对 UAS 和注册服务器的鉴权。在 Authorization 头字段可以带有一个证书，该证书包含所请求资源的域的鉴权信息以及鉴权和重发保护所需的参数。

Authorization 头字段的例子如下：

Authorization: Digest username = " bob",
realm = " biloxi. com",
nonce = " dcd98b7102dd2f 0e8b11d0f600bf b0c093",
uri = " sip: bob@ biloxi. com",
qop = auth,
nc = 00000001,
cnonce = " 0a4f113b",
response = " 6629fae49393a05397450978507c4ef1",
opaque = " 5ccc069c403ebaf9f0171e9517f40e41"

UAC 收到 401 或者 407（代理服务器要求鉴权）响应之后重新提交含有证书的请求，并且必须将请求中的 Cseq 字段的值加 1。

3. 代理服务器到用户的鉴权

UAC 向代理服务器发送请求的时候，代理服务器在处理请求之前可能要求 UAC 鉴权。如果请求中的 Proxy-Authorization 头字段中没有证书，代理服务器可以使用 407 状态码拒绝请求并要求鉴别用户权限。该 407 响应必须包含一个 Proxy-Authenticate 头字段。

代理服务器不可以向 Proxy-Authorization 头字段中增加值。所有的 407 响应必须按上行方向转发到 UAC。UAC 应该向 Proxy-Authorization 头字段中增加相应的证书，如果代理服务器要重新提交该请求，它需要在新请求中将 Cseq 的值加 1。因为 Cseq 的值的改变，原始请求发起的 UAC 将丢弃 UAS 的响应。

发起方 UAC 收到 407 响应以后，它应该重新发起包含证书的请求。“realm”参数的确定同 401 响应的处理。

如果找不到某个域的证书，UAC 可以使用“anonymous”的用户名和空密码重试该请求。

UAC 应该缓存用于重新发起请求的证书。

RFC3261 建议代理服务器使用下面规则缓存证书：

如果 UA 收到含有某个特定 Call-ID 的请求的 401 或 407 响应中的 Proxy-Authenticate 头字段值，它应该合并随后所有含有相同 Call-ID 的请求的证书。这些证书不能越过对话进行缓存；如果某 UA 被配置了本地对外代理服务器的域，如果有一个域存在，那么该 UA 可以越过对话缓存该域证书。这并不代表对话中以后的请求通过 Route 路径上的每个代理服务器都不需要证书。

任何向代理服务器鉴权自身的 UA 通常在收到 407 响应以后，可以在请求中包含一个 Proxy-Authorization 头字段值来完成鉴权。Proxy-Authorization 头字段允许客户机向要求鉴权的代理服务器证明自己或者其用户的身份。Proxy-Authorization 头字段值是由证书组成，该证书中包含 UA 对于代理服务器或请求的资源域的鉴权信息。

Proxy-Authorization 头字段值只用于某些代理服务器，其标识放在“realm”参数中，这些代理服务器以前可能曾利用 Proxy-Authenticate 头字段要求鉴权。路径中使用多个代理服务器时，Proxy-Authorization 头字段的值不可以用于与“realm”参数匹配的代理服务器。

如果某鉴权方案不支持 Proxy-Authorization 头字段中使用的域，那么代理服务器必须试着解析所有的 Proxy-Authorization 头字段的值来确定是否存在有效证书。但在大型的网络中，为了节省时间，代理服务器应该在 Proxy-Authorization 头字段中使用支持该域的鉴权机制。

如果请求被分叉代理，可能会有不同的代理服务器或 UA 向该 UAC 发出质询。这种情况下，分叉的代理服务器负责将这些质询合并在一个请求中。分叉代理的请求的响应中，每一个 WWW-Authenticate 和 Proxy-Authenticate 头字段值必须放在分叉代理服务器发给 UA 的响应中，其顺序任意。如果某代理服务器在响应中发出质询，它将停止代理该请求，直到 UAC 重新发出含有有效证书的请求。一个分叉代理服务器可能会同时向多个需要鉴权的代理服务器转发某个请求，这些代理服务器都将停止转发该请求，直到 UAC 在各个域中分别鉴权本身。

如果 UAC 没有向每个质询提供证书，发出质询的代理服务器将不再向 UA 转发该请求。

UAC 在收到包含多个质询的 401 或者 407 响应之后重新提交该请求。该请求中的在 WWW-Authenticat 值中应包含一个 Authorization 值，Proxy-Authenticate 值中应包含一个 Proxy-Authorization 值。一个请求中的多个证书应该通过它们的“realm”参数来区分。

多个质询可能只与 401 或者 407 响应中的同一个域相关。当多个代理服务器使用同一个域时，当 UAC 向这些代理服务器发出多个分叉代理的请求，该请求中的可 Authorization 或 Proxy-Authorization 头字段中可能有多个含有相同的“realm”参数的证书。相同的证书应用于同一个域。

4. 分类鉴权

SIP 分类鉴权方案与 HTTP 分类鉴权类似，以下的内容是 HTTP 分类鉴权方案用于 SIP 鉴权时的所作的修改。

RFC2543 鉴权机制是基于 RFC2069 的 HTTP 分类鉴权的定义，所以支持 RFC 2617 的 SIP 服务器必须保证后向兼容 RFC2069。SIP 服务器不可以接受或者发出 Basic 鉴权的请求。

SIP 分类鉴权机制遵从 RFC2617 定义的规则，但需把其中的" HTTP/1.1" 变为" SIP/2.0"。具体变化如下：

1）质询中 URI 的 BNF 如下所示：

URI = SIP-URI /SIPS-URI

2）RFC2617 的 BNF 有点不同，HTTP Digest 鉴权的 Authorization 字段的 uri 参数没有使用引号而 SIP 的 uri 必须使用引号。

3）digest-uri-value 的 BNF 语法为：

digest-uri-value = Request-URI 。

4）基于 Etag 选择 nonce 并不适用于 SIP。

5）RFC2617 中关于缓存的操作不适用于 SIP。

6）RFC2617 要求服务器检查请求中的 URI 和 Authorization 头字段中的 URI 是否指向相同的资源。在 SIP 中，由于请求要在一些代理服务器进行转发，因此这两个 URI 可以指不同的用户。SIP 服务器可以检查 Authorization 头字段中的 Request-URI，如果跟用户匹配，那么服务器就愿意接受转发的或者直接的请求，但是如果这两个域不同也不一定会导致失败。

7）分类鉴权方案中消息完整性保护中，作为 A2 值的计算的净化，具体实现时，应该假设消息体为空，实体的无用信息部分就被解析为 MD5 空字符串，或者是：

H (entity-body) = MD5 ("") = " d41d8cd98f00b204e9800998ecf 8427e"

11.2.4　S/MIME

S/MIME 的最突出的缺点就是它缺少对终端用户很流行的公有密钥结构。如果使用自签名的证书（或者不能被对话中的某个参与方所验证的证书），那么基于 SIP 的密钥交换机制就容易受到中间者攻击，这时攻击者可能检查并修改 S/MIME。攻击者需要中断对话双方的第一次密钥交换，从请求及响应中去掉已有的与 CMS 分开的签名，并插入一个不同的自己的包含证书的与 CMS 分离的签名（但是看来是一个正确的记录地址的证书）。对话双方都认为相互已经交换了密钥，但其实它们只是有了攻击者的公有密钥。

攻击者可以在双方第一次交换密钥时只针对以上该弱点进行攻击，随后密钥的改变对 UA 是显而易见的。以后攻击者也很难留在这个进行以后所有对话的通道里。

SSH 在初次交换密钥时很容易受到同样的中间方的攻击，但是 SSH 能很好地改善连接的安全，并可以像 SSH 那样使用指纹密码。例如，通信双方用 SIP 建立一个语音通信的会话，它们应该读出所收对方的密钥中的指纹并与最初的相比较。当然，同信令相比，中间方更难模仿参与者的语音（用在基于 CLIPPER 芯片的安全的电话）。

使用 S/MIME 机制，UA 可以发送一个加密的但是无同步码的请求，如果该请求在密钥环中含有一个目标记录地址的证书。但是，任何注册为某个目标地址的设备不能持有任何曾经被该设备的当前用户所使用的证书。因此它就不能正确地处理一个加密的请求，这样就不可避免地产生错误的信令。当一个加密的请求被分叉代理的时候这种情况就最可能发生。

S/MIME 有关的密钥适用于与一个特定的用户（记录地址）而不是一个设备（UA）相关联的时候。当用户在设备间移动时，它也许很难在 UA 间安全地传输私有密钥。

在 S/MIME 中，很难解决的问题是它能产生很大的消息，特别是当使用 SIP 隧道机制时。因此，RFC3261 建议在使用 S/MIME 隧道技术时，应当使用 TCP 作为传输协议。

11.3　SIP 安全机制的实现

代理服务器、重定向服务器和注册服务器都必须实现 TLS，并且必须支持双向的和单向的认证关系。RFC3261 中强烈建议 UA 可以初始化 TLS；UA 同样可以作为一个 TLS 服务器。代理服务器、重定向服务器和注册服务器应当有一个站点信任书，这个信任书的主题和它们的规范主机名相关。对于 TLS 的双向认

证，UA 可以有它们自己的信任书，但是在 RFC3261 协议文档中，没有规定它们的具体用法。所有支持 TLS 的 SIP 元素都必须具备在 TLS 协商中，验证信任书的机制，这使得证书机关（可能是类似 Web 浏览器证书发行机构的发行机构）发布的一个或者多个根信任书成为必须。同时所有支持 TLS 的 SIP 元素必须同样支持 SIPS URI 方案。

代理服务器、重定向服务器、注册服务器和 UA 可以实现 IPSec 或者其他底层的安全协议。

当 UA 试图联系一个代理服务器或者重定向服务器或者注册服务器时，UAC 应当初始化一个 TLS 连接，在这个连接上发起 SIP 消息。在某些结构下，UAS 可以同样在这些 TLS 接收请求。

代理服务器、重定向服务器、注册服务器和 UA 必须实现 Digest 身份认证。代理服务器、重定向服务器和注册服务器应当配置成为至少有一个 Digest realm，并且对于给定服务器来说，必须支持至少有一个“realm”字符串和这个服务器的主机名或者域名相关联。UA 可以支持 MIME（Multipurpose Internet Mail Extensions，多用途因特网邮件扩展）包体的加密，并且可以使用 S/MIME 传送信任书。如果 UA 具有一个或者多个根身份认证的信任书，用来鉴定 TLS 或者 IPSec 的信任书，它应当适当的可以用这些来鉴定 S/MIME（Secure/Multipurpose Internet Mail Extensions，安全多用途因特网邮件扩展）的信任书。UA 可以为 S/MIME 身份认证而具有特定的根信任书。随着 S/MIME 实现，将来会有安全扩展，来提高 S/MIME 的强度。

上述安全机制的操作，在某种程度上，可以与现存的 Web 和 Email 安全模式一致。UA 通过 Digest 用户名和口令把它们自己的身份向服务器（代理服务器，重定向服务器和注册服务器）认证；服务器把它们自己向 UA 单节点认证，或者向另外一个服务器进行单节点认证（反之亦然），并且是通过 TLS 来传送服务器节点信任书。

在点到点的级别，UA 一般信任网络来进行对方身份的鉴别，不过，如果网络不能够鉴定对方身份，或者网络本身不被信任的情况下，也可以使用 S/MIME 来提供直接的身份认证。

接下来是一个例子，在这个例子中，不同的 UA 和服务器使用这些安全机制防止 11.1 节描述的攻击威胁。实现者和网络管理员可以以此来防止攻击威胁。

将 SIP 消息在终端系统之间完全加密，能很好地保证消息的机密性，也能保证消息不被第三方恶意修改。但是，由于 SIP 体系结构使用了中间服务器，中间服务器如代理服务器必须通过了解消息中特定的字段（如 Request2URI，Route，Via 等）才能正确完成路由。代理服务器还要修改某些字段，如增加 Via 字段的

值 SIP 消息的完整加密与代理服务器需要了解并修改某些头字段是相互矛盾的，所以必须将两者结合起来，中间服务器能够查看/修改特定的头字段，但不能查看/修改消息体和只对目的终端有意义的头字段。下面从三个方面讨论 SIP 安全解决方案。

11.3.1　用户终端与代理服务器/注册服务器

为方便讨论起见，本文中的代理服务器与注册服务器是合二为一的，否则，方案类似。

用户终端第一步是向代理/注册服务器建立 TLS 连接，注册服务器向用户终端提供证书，接受用户终端的验证。一旦注册服务器的证书通过了用户终端持有的根证书的验证，并且证书标记的主机正是注册请求要发往的服务器，则 TLS 连接成功建立。

当用户终端第一次通过 TLS 连接发送请求，注册服务器可以通过 401（未授权）响应要求用户终端提供合法的认证信息。用户终端向注册服务器提供认证有 3 种方式：基本的 HTTP 认证、摘要 HTTP 认证和提供用户终端自己的证书进行认证。RFC3261 建议采取摘要 HTTP 认证的方式。基本 HTTP 认证在 SIP 中已经被废除，由于用户众多并且有移动性，管理与移动终端用户的证书不方便，所以采用摘要 HTTP 认证。由于注册服务器要求用户终端认证自己，攻击者伪造用户的注册将很困难。既然注册请求是通过加密的 TLS 连接发送过去的，攻击者也就不能够中途截取注册请求，用来做重放攻击。用户终端应当保证 TLS 连接一直是打开的，即所有后续的用户终端与代理服务器之间的请求/应答都通过该 TLS 连接发送/接收。

11.3.2　中间服务器与中间服务器

为确保 SIP 消息在中间服务器之间安全地传送，可以采用底层的 IPSec 或者 TLS 作为安全机制。无论是 IPSec 还是 TLS，双方都需要用证书互相向对方认证自己的身份。中间服务器的数量有限，并且具有稳定性，可以考虑用自签名的证书。下面举例说明中间服务器之间的安全策略。

假设用户 A 通过代理服务器向用户 B 发送 INVITE 请求，A 的代理服务器与 B 的代理服务器通过了彼此的证书验证以后，代理服务器 B 还会检查 INVITE 请求的 From 字段标明的域是否与代理服务器 A 所管理的域一致。如果代理服务器 B 执行严格的安全策略，则会拒绝那些与代理服务器的域不一致的请求；代理服务器 B 还可以执行更严格的安全策略，如果它认为代理服务器 A 认证用户 A 不够严格地话，也可以拒绝请求。

如果代理服务器留在后续消息的路由路径上，则可以继续利用已经建立的

TLS 连接提供安全服务。SIP 通过这种逐跳认证，提供了很好的安全结构，攻击者不能够伪造 BYE 请求并把它插入到终端系统之间的信号流，因为攻击者没有办法探知会话的参数，也因为完整性机制保护使得消息不能被篡改。

11.3.3 终端系统与终端系统

由于 SIP 采用的逐跳解密/加密的方式，使得整个 SIP 消息对中间服务器来说，是完全开放的。如果用户终端认为中间服务器或 SIP 网络不可信任，可以结合采用 S/MIME 制。

S/MIME 是用来保证电子邮件端到端的安全技术，S/MIME 用在 SIP 中，提供了消息体的端到端的机密性、完整性和相互认证。利用 SIP 消息隧道的方式，还可以对部分头字段提供完整性和保密性服务。

S/MIME 机制依赖于证书，终端的证书是用来证实终端用户的 AOR 的。当发送 SIP 消息的时候，发送方用自己的私钥签名消息体，然后用对方的公钥加密消息体，所以，终端必须知道对方的公钥。如果终端想使用 S/MIME 机制，在初始化一个会话的请求消息中就要携带自己的证书，以便目的终端用证书上的公钥验证签名。当对方收到消息后，首先验证证书的颁发机构是否可信任，再验证消息的来源 From 字段是否与证书的 Subject 一致。一旦证书通过验证后，将被加入到本地的证书链中。目的终端会在随后的应答消息中携带自己的证书，如果对方验证通过了证书，则证书交换过程完成，双方可以用 S/MIME 机制通信了。

SIP 消息还可以使用隧道方式，隧道方式的 SIP 消息体是没有使用 S/MIME 时的整个 SIP 消息，消息体的类型为“message/sip”，而头字段仅为中间服务器必须要知道的字段，然后按照正常的 S/MIME 方式处理该隧道方式的 SIP 消息。通过 S/MIME 隧道的方式不仅可以保护消息体，还可以提供头字段的保密性和完整性，因为目的终端在接收隧道方式的 SIP 消息后，通过比较“内部”与“外部”的头字段的值就可以知道完整性有没有被破坏，并且由于那些只对目的终端有意义的头字段只放在加密的消息体内部，比较好地实现了保密性。

11.3.4 DoS 防护

基于这些安全解决方案，为了使得拒绝服务（DoS）攻击造成的影响最小，实现者应当注意如下的提议。

当 SIP 代理服务器所在的主机是基于公共因特网做路由的，它应当部署在一个具有防护操作策略的管理域中（比如 block 源路由和过滤 ping 包等）。TLS 和 IPSec 都可以在管理域的边界防护主机中，一起参与安全系统的构架，来提高安全性。这些防护主机可以防护拒绝服务攻击，确保在管理域中的 SIP 主机不会被

大量的消息阻塞。

不管使用什么样的安全措施，对代理服务器的洪水消息攻击可以耗尽代理服务器的资源，并且阻止发送到目的地的正确的请求。对于代理服务器来说，每一个 SIP 事务都会好用一定的资源，对于有状态的服务器来说这个消耗会更大。因此，有状态的代理服务器比无状态的代理服务器更容易受到洪水攻击的影响。

UA 和代理服务器应当用一个 401（Unauthorized）或者 407（Proxy Authentication Required）拒绝有问题的请求，并且对这些有问题的请求不使用正常的应答重发机制，并且对这些未认证的请求把自己当作无状态的服务器使用。如果攻击者使用假的头域值（例如 Via）指向一个第三方的被攻击的服务器，那么对于 401（Unauthorized）或者 407（Proxy Authentication Required）应答的重发可能正中攻击者的下怀。

总得来说，基于 TLS 签名的代理服务器之间的双向认证机制，降低了中间节点潜在的风险，并且减少了可以用作拒绝服务攻击的伪造的请求或者应答。这也同样提高了攻击者利用无知的 SIP 节点进行放大攻击的难度。

11.4　SIP 安全机制带来的限制

虽然有这些安全机制，在正确应用的时候，可以阻止很多攻击，但是必须明白它们也会有很多限制的地方。

11.4.1　HTTP Digest

在 SIP 中，HTTP Digest（分类）对于 SIP 一个限制是 Digest 的完整性机制，在 SIP 下运作得不是很完美。尤其是它们提供了对消息中的 Request-URI 和方法的保护，但是并没有对 UA 希望提供加密的所有头域进行保护。

RFC2617 所提供的回放攻击保护对于 SIP 来说也有一些限制。例如，next-nonce 机制，并不支持通过管道传送的请求。防止回放攻击应当使用 nonce-count 机制。

另一个 HTTP Digest 限制是 realm 的范围。当用户希望把它们自己的身份向一个资源进行认证的时候，Digest 是有用的，它就像一个服务提供者，用户是一个客户端（这是十分常见的场景，因此 Digest 提供了一个十分有用的功能）。与之对应的，TLS 的范围是基于域之间的，或者 multi realm 的，因为信任书通常是全局可验证的，所以 UA 可以在不需要预先存在关系的服务器上进行身份验证。

11.4.2 S/MIME

对于 S/MIME 的一个最大的限制是对终端用户来说，缺少广泛的公共密钥机构。如果使用的是自签名信任书（或者信任书不能被对话的对方所验证），基于 SIP 的密钥交换机制就容易遭受中间人攻击（man-in-the-middle），在这个中间人攻击中，攻击者可以悄悄检查和修改 S/MIME 包体。攻击者需要截取对话中双方的第一个密钥交换，在请求和应答中移去现有的 CMS-detached 签名，并且插入另外的包含攻击者提供的信任书（但是看起来像是正确的 address-of-record 的信任书）的 CMS-detached 签名。通信的双方都会以为它们和对方交换了密钥，实际上它们互相有的只是攻击者的公钥。

必须明确注意到攻击者只能攻击双方的第一次的密钥交换，在后续的情况，密钥的变更对于 UA 来说就是不可见的了，同样使得攻击者难以窃听对话双方以后的对话。

SSH 在第一次密钥交换的时候，也同样容易受到这个中间人攻击，虽然众所周知 SSH 不完美，但是它确实提高了连接的安全性。对于 SIP 来说，使用密钥的指纹可以有些帮助，就像在 SSH 中一样。例如，如果 SIP 的双方建立了一个语音通信会话，每一方都可以读取对方传送的密钥指纹，并且可以根据这个指纹和原始指纹做比较。这使得中间人更加难以在语音中模拟通话双方的密钥指纹。

在 S/MIME 机制下，如果 UA 在它们的密钥组中持有对方 address-of-record 的信任书，允许 UA 不用导言来发送加密的请求。不过，存在这种情况，如果某个设备注册了这个 address-of-record，并且没有持有当前用户先前使用的信任书。并且它将因此不能正常处理加密的请求，于是可能会导致某些原本可以避免的错误信号。这很像加密的请求被分支的情况。

S/MIME 相关的密钥通常用于和特定用户（一个 address-of-record）关联起来使用，而不是和一个设备一起使用。当用户在设备之间移动，很难把私钥安全的在设备之间传递，一个设备如何获得这些密钥不在本文讨论。

另外，使用 S/MIME 更常见的困难是，它可以导致很大的消息，特别是当采用的 SIP 隧道机制的时候。基于这个原因，当使用 S/MIME 隧道机制的时候，一定要使用 TCP 通信协议。

11.4.3 TLS

TLS 最大的问题是，它不能基于 UDP，TLS 要求面向连接的底层通信协议，对于本文来说，就是 TCP。

对于 TLS 来说，在本地外发代理服务器或者注册服务上，是一件费力的事

情。这导致某些容量问题，特别是在使用加强密钥套件的时候更容易出现容量问题。维持冗余的 TLS 长连接，特别是当 UA 独立负责它们的连接，会很耗资源。

TLS 只允许 SIP 实体到它们临近的认证服务器的连接，TLS 提供了严格的节点到节点的安全性。无论 TLS，还是其他本文档规定的安全机制，当不能直接建立 TCP 连接的情况下，都允许客户通过验证自己到代理服务器来到达目的地。

11.4.4　SIPS URI

实际上在请求经过的每一段路径都使用 TLS，要求最终的 UAS 必须能够通过 TLS 到达（可能是通过 SIPS URI 作为一个联系地址注册的）。通常这个目的地是用 SIPS 描述的。在很多结构下，使用 TLS 保护一部分请求路径，最后一步确实依赖于其他的安全机制到 UAS。因此 SIPS 不能保证 TLS 是真正的端到端的使用。注意，这是由于许多 UA 不接受进来的 TLS 连接，甚至那些支持 TLS 的 UA 也可能要求要维持一个永久的 TLS 连接，目的是为了作为 UAS 从 TLS 上接收请求。

位置服务，对于 SIPS Request-URI 来说，是不要求提供 SIPS 绑定的。虽然位置服务通常由用户注册组成，但是对于一个 address-of-record 来说，可以由不同的协议和界面都可以提供联系地址，并且这些工具都可以用来映射 SIPS URI 到适当的 SIP URI。当对绑定信息查询的时候，位置服务返回它的联系地址，而不关心这个是否是从一个 SIPS 的 Request-URI 上收到的请求。如果是转发服务器访问位置服务，那就是说取决于处理转发的 Contact 头域的 SIP 实体来决定联系地址的属性。

如果要求请求经过的全部路径都使用 TLS 通信，那么对目的域来说稍稍有点麻烦。如果实现困难，也允许在请求传送节点中的基于密码认证的代理服务器，不兼容或者选择一个折中方案来略过 SIPS 的转发机制。但是，恶意的中间节点就有可能把一个基于 SIPS URI 的请求重新降级成为 SIP URI。

另外，中间节点也可以正常地把一个基于 SIP 的请求更改成基于 SIPS URI 的请求。请求的接收方发现请求的 Request-URI 是基于 SIPS URI 方案的并不能假设原始请求的 Request-URI 也是基于 SIPS 通过中间节点传送的（从客户端往后）。

为了解决这个问题，我们建议请求的收件人，在请求的 Request-URI 包含一个 SIP 或者 SIPS URI 的情况下，检查 To 头域值，看是否包含一个 SIPS URI（虽然注意到 Request-URI 可以和 To 头域 URI 有相同的 URI 方案，但是它们不相等并不意味者是一个安全隐患）。虽然客户端可以在一个请求中把 Request-URI 和 To 头域做成不一样的，但是当两个字段的 SIPS 不一样的话，那就是可能的安全

隐患，并且请求因此应当被接收方拒绝。接收方也可以检查 Via 头域链来双重检查是否 TLS 在整个请求路径中被使用，一直到最近的本地管理域。源 UAC 也可以用 S/MIME 来帮助确保原始格式的 To 头域被端到端的发送。

如果 UAS 有原因来相信 Request-URI 的 URI 方案在通信中被非法修改，UA 应当提示用户这个潜在的安全隐患。

作为更深远的考虑，为了防止底层的攻击，如果 SIP 实体只接收 SIPS 请求，那么可以拒绝在非安全端口的连接。

终端用户应当完全清楚 SIPS 和 SIP URI（Uniform resource identifier，统一资源标识）的区别，它们可以在应答中手工更改这个 URI 方案，这可以增加或者降低安全性。例如，如果一个攻击者攻击了一个 DNS 的 cache，插入了一个假的记录集，这个记录集有效删去了一个代理服务器的所有 SIPS 记录，接着经过这个代理服务器的 SIPS 请求就会失效。这时候，一个用户，看见这个 SIPS address-of-record 总是失败，它可以手工改掉 URI，从 SIPS 改成 SIP，并且重试。当然，这也有一些安全机制防止这类事情发生，但是这个限制毫无价值。往好了想，用户也可以把 SIP URI 变成 SIPS URI。

第 12 章　SIP 的应用

12.1　SIP 在软交换中的应用

软交换技术是实现新一代语音、通信以及多媒体和数据交换的核心技术。软交换机作为控制实体，完成呼叫接续、业务控制和用户管理等功能，而业务多媒体流的传送由 IP 网络完成。软交换中所支持的呼叫协议中，SIP 是其中重要的一个部分。与存在已久的 ITU-T 的 No.7 信令系统（用于呼叫建立）和 H.323 协议族不同，SIP 独立工作于底层网络传输协议和媒体。它规定一个或多个参与方的终端设备如何能够建立、修改和中断连接。SIP 主要是控制会话的建立、维持和终止，它的主要目的是为了解决软交换网中的信令控制，以及同软交换之间的通信，从而构成下一代的增值业务平台，对电信、银行、金融等行业提供更好的增值服务。其系统结构如图 12-1 所示。

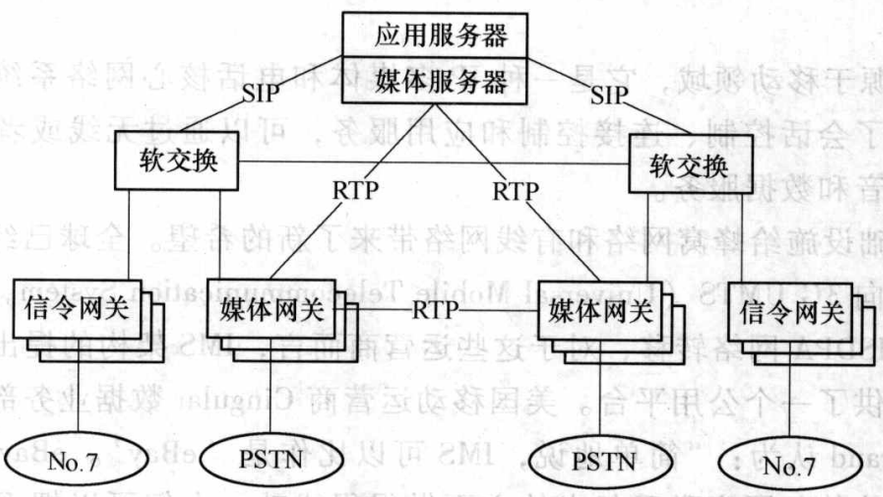


图 12-1　SIP 系统框架结构

图 12-1 中的媒体服务器，它能提供许多语音服务。例如，当某个客户给某个电子商务网站打电话，这个客户会被接到一个 IVR（Interactive Voice Response，自动语音应答）上或者自动总机上。这种应用使用了语音、语音识别和音频等功能，它们就可以在媒体服务器上得到实现。实际上，在下一代网络的实现中，媒体服务器其功能就如同一个 SIP 的终端，并可以享受和用户一样的 SIP 的优点。

总体上讲，媒体服务器提供与呼叫方交互所需的所有语音计算资源。由于嵌入的 DSP 资源允许对语音流进行编程访问，开发者就可以为他们特定的服务

提供所需的语音、音频和语音识别的混合。

12.2 SIP 在 IMS 中的应用

1. IMS 历史与优点

IMS（IP Mutimedia Subsystem，IP 多媒体子系统）是 3GPP（The 3rd Generation Partnership Project）在 Release 5（R5）版本提出的支持 IP 多媒体业务的子系统，它的突出特点是采用了 SIP 以及与接入的无关性。R5 版本主要定义 IMS 的核心结构、网元功能、接口和流程等内容；R6 增加了部分 IMS 业务特性、IMS 与其他网络的互通规范和 WLAN 接入等特性；R7 则加强了对固定、移动融合的标准化制定，要求 IMS 支持 xDSL、cable 等固定接入方式。目前，IMS 支持 2G 和 3G 的移动接入方式，暂还不能支持固定接入方式。在网络融合的发展趋势下，3GPP、ETSI（European Telecommunication Standards Institute，欧洲电信标准化协会）和 ITU-T（国际电信联盟电信标准化部门）都在研究基于 IMS 的网络融合方案，目的是使 IMS 成为基于 SIP 会话的通用平台，同时支持固定和移动的多种接入方式，实现固定网和移动网的融合。目前 IMS 的产品主要基于 3GPP R5 版本。

IMS 发源于移动领域，它是一种 IP 多媒体和电话核心网络系统，在一个集中点上实现了会话控制、连接控制和应用服务，可以通过无线或者有线网络提供融合的语音和数据服务。

IMS 基础设施给蜂窝网络和有线网络带来了新的希望。全球已经有不少移动运营商开始向 3G UMTS（Universal Mobile Telecommunication System，全球移动通信系统）/HSDPA 网络转移，对于这些运营商而言，IMS 架构的提出为他们以后开发服务提供了一个公用平台。美国移动运营商 Cingular 数据业务部门的执行董事 Terry Durand 认为："简单地说，IMS 可以比作是'eBay'。eBay 在把供应商和内容合作伙伴与顾客联系起来的方面做得很成功。人们可以把 IMS 看成是类似于 eBay 的某种集中点，它为我们提供了一种标准方法，可以把合作伙伴与客户联系起来，开展丰富的多媒体服务，还能够进行计费，并且有强验证和服务质量保证"。

虽然 IMS 仍处在发展阶段，但它越来越受到移动运营商乃至固定网运营商的欢迎。一些行业组织已经采用 IMS 作为下一代基础设施的根本起点，比如多服务交换论坛（MSF）、欧洲电信标准化协会（ETSI）和电信行业解决方案联盟（ATIS）。这股 IMS 潮流吸引了那些没有移动网络的固定网运营商们的视线，其中态度比较积极的是美国的 MCI、Qwest 和英国电信（BT）。这些运营商正努力尝试通过成为移动虚拟网络运营商（MVNO），从而切入移动服务领域。

在 IMS 出现之前，每种不同的服务最终都要交给特定的节点和网域，IMS 基础设施的真正价值则在于，可以重复使用部件。要想成为最佳的系统集成商，就要避免在 IP 网络中构建孤立的环境，像当初的公共交换电话网络（PSTN）那样。基于部件的架构是 IMS 的重要理念之一。另外，把视频点播、视频教室和视频会议等先进的宽带服务混合搭配成更好、更快、更便宜的通信产品，这是实现美好的 IP 蓝图的关键之一。

2. 软交换和 IMS

软交换和 IMS 都作为下一代网络呼叫控制的体系架构，它们虽然都采用了下一代网络基于 IP 承载、呼叫控制与承载分离的思想，但是软交换和 IMS 技术还存在较大的差异。软交换技术的主要优势是技术已经成熟，软交换技术标准从 1998 年开始出现已经经历了实验、商用等多个发展阶段，据统计，截止到 2004 年 8 月，全球范围内已经有多家电信运营商开展了软交换技术的实验，部署了 250 多个商用的软交换系统。软交换技术已经具备了替代电路交换机的能力，并具备一定的宽带多媒体业务能力。

IMS 技术的主要优势是 IMS 体系架构可以支持移动性管理，并具有一定的 QoS 保障机制，因此 IMS 在宽带用户的漫游管理和 QoS 保障方面更具有优势。并且，IMS 接入窄带用户提供 PSTN 仿真业务的能力还处于研究的初始阶段，支持 ISDN 业务、V5 接入等传统 PSTN 接入方式还没有开始研究，因此 IMS 距离完全继承 PSTN 网络能力提供 PSTN 业务的目标还有较长的距离。

总体来说，软交换网络体系基于主从控制的特点，使得其与具体的接入手段关系密切，而 IMS 体系由于终端与核心则采用基于 IP 承载的 SIP 通信。IP 技术与承载媒体无关的特性，使得 IMS 体系可支持各类接入方式。IMS 体系虽然最早在移动网中提出，但是其核心侧与接入无关的特性使得 IMS 的应用范围逐步扩大到固定领域，近期软交换网络以解决窄带用户接入为主，宽带用户以企业用户接入为辅，未来随着宽带和融合类业务需求的强烈，再引入 IMS 网络统一解决固定和移动的宽带接入。

12.2.1　IMS 概述

IMS 是基于 UMTS 核心网分组域（PS 域）上的，用 PS 域来传输呼叫控制信令，并负责承载数据业务，是独立于 CS 域的，但保持与 CS 域的互通。IMS 建立了对呼叫/会话进行控制的软交换平台，控制信令在经过 SGSN（Serving GPRS Support Node，服务 GPRS 支持节点）、GGSN（Gateway GPRS Support Node，GPRS 网关支持节点）后，就与媒体平面的消息分开，信令消息将在 CSCF（Call Session Control Funtion，呼叫会话控制功能）间根据用户配置进行传送，媒体消息则在信令选择的路径上传送。3GPP 在定义 IMS 时，并没有对节点标准化，而

更多的是定义了很多功能，这些功能间的通信由标准接口来规范。其在核心网络中位置如图 12-2 所示。

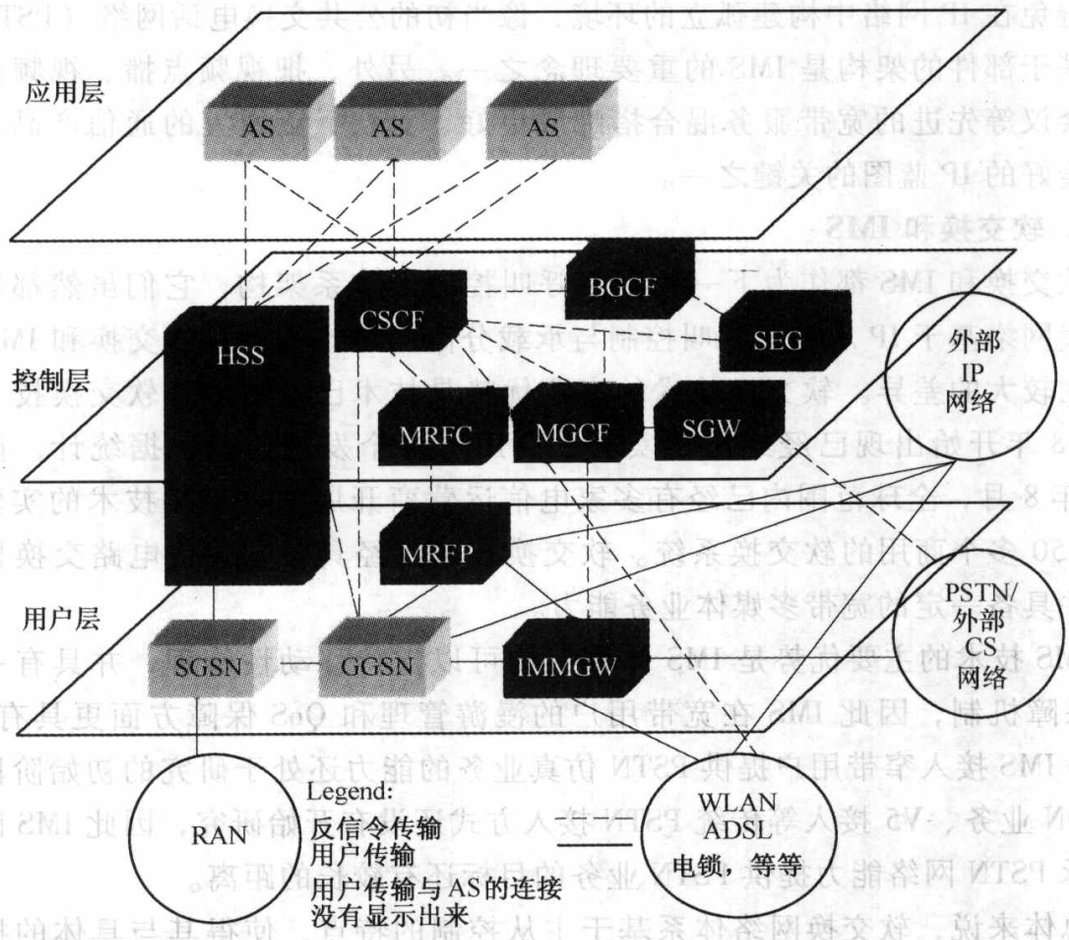


图 12-2 IMS 体系结构图

从架构上看，IMS 是与接入技术无关的网络，不论用户通过使用 ADSL 的计算机接入，还是使用 WCDMA 手机等无线方式接入，都可体验相同的业务。IMS 网络架构分为用户层、会话控制层和应用层。

1. 用户层

用户层（User Plane）完成的主要功能包括各类 SIP 终端 SIP 会话的发起和终结；实现 IP 分组承载各种承载类型之间的转换；根据业务部署和控制层的控制实现各种 QoS 策略；完成与传统 PSTN/PLMN 间的互联互通等功能。接入互联层包括各类 SIP 终端、有线接入、无线接入、互联互通网关等设备。

2. 控制层

控制层（Control Plane）完成基本会话的控制，完成用户注册、SIP 会话路由控制，与应用服务器交互执行应用业务中的会话、维护管理用户数据、管理业务 QoS 策略等功能，与应用层一起为所有用户提供一致的业务环境。控制层包括 CSCF（Call Server Control Function）、MRFC（Multimedia Resource Function Controller，多媒体资源功能控制器）、BGCF（Breakout Gateway Control Function，

出口网关控制功能）等功能实体。其中 CSCF 包括 P-CSCF（Proxy CSCF，代理 CSCF）、I-CSCF（Interrogating CSCF，问询 CSCF）、S-CSCF（Serving CSCF，服务 CSCF）等类型，在物理上可以是合一的，也可以分别设置。在实际组网时，其划分和部署需要综合考虑对 IMS 业务接入方式、IMS 接入点位置、CSCF 的容量、能力及用户业务量需求等因素，另外也与运营商网络拓扑隐藏及互通需求有关。P-CSCF 是 UE 接入 IMS 系统的入口，实现了在 SIP 中的代理服务器和用户代理（User Agent）功能。S-CSCF 在 IMS 核心网中处于核心的控制地位，负责对 UE 的注册鉴权和会话控制，执行针对主叫端及被叫端 IMS 用户的基本会话路由功能，并根据用户签约的 IMS 触发规则，在条件满足时进行到 AS（Application Server，应用服务器）的增值业务触发及业务控制交互。I-CSCF 在 IMS 核心网中起到关口节点的作用，提供本域用户服务节点分配、路由查询以及不同 IMS 域间拓扑隐藏等功能。确定哪个 S-CSCF 为用户提供服务就是由 I-CSCF 通过各种组合条件决定的。

3. 应用层

应用层（Application Plane）向用户提供业务逻辑，包括实现传统的基本电话业务，如呼叫前转、呼叫等待、会议等业务；IMS 通过 IM-SSF 和传统智能业务（CAMEL、INAP 等）的互通，实现 CS（Circuit-switched，电路交换）和 PS（Packet-switched，分组交换）已有的智能业务的继承。IMS 架构除了可以实现 CS、PS 已有的业务外，IMS 通过 AS 可以提供基于 SIP 的非传统电信业务，如 IM、PTT、Presence 等。另外，IMS 通过 OSA-G（Open Services Access-Gateway，开放服务接入网关）提供简单 API（Application program interface，应用编程接口），以便第三方能够通过这个接口安全地使用网络资源和提供业务，实现丰富地娱乐、游戏业务。另外 IMS 充分考虑了实际运营的需求，在 QoS、安全、计费以及和其他网络的互通方面都制定了相关规范。

12.2.2　IMS 网络单元分析

下面详细分析 IMS 系统结构的各个网络单元，可以从以下几个方面来看：由于 IMS 是在 PS 域的基础上架构的，如果网络是基于 IP 的，呼叫控制的实现主要是通过 CSCF、HSS 来实现，但考虑到现存的各种 CS 域网元，便通过一些互通实体如 BGCF、MGCF、MGF（Media Gateway Function，媒体网关功能）、SGW（Signaling GateWay，信令网关）等来实现与 PSTN/CS 域的通信，而为了实现多方通信，便增加了 MRF（媒体资源功能）。各功能模块如图 12-3 所示。

1. IMS 呼叫控制功能

（1）CSCF

IMS 中最重要的功能实体是呼叫会话控制功能（CSCF），负责对用户多媒体

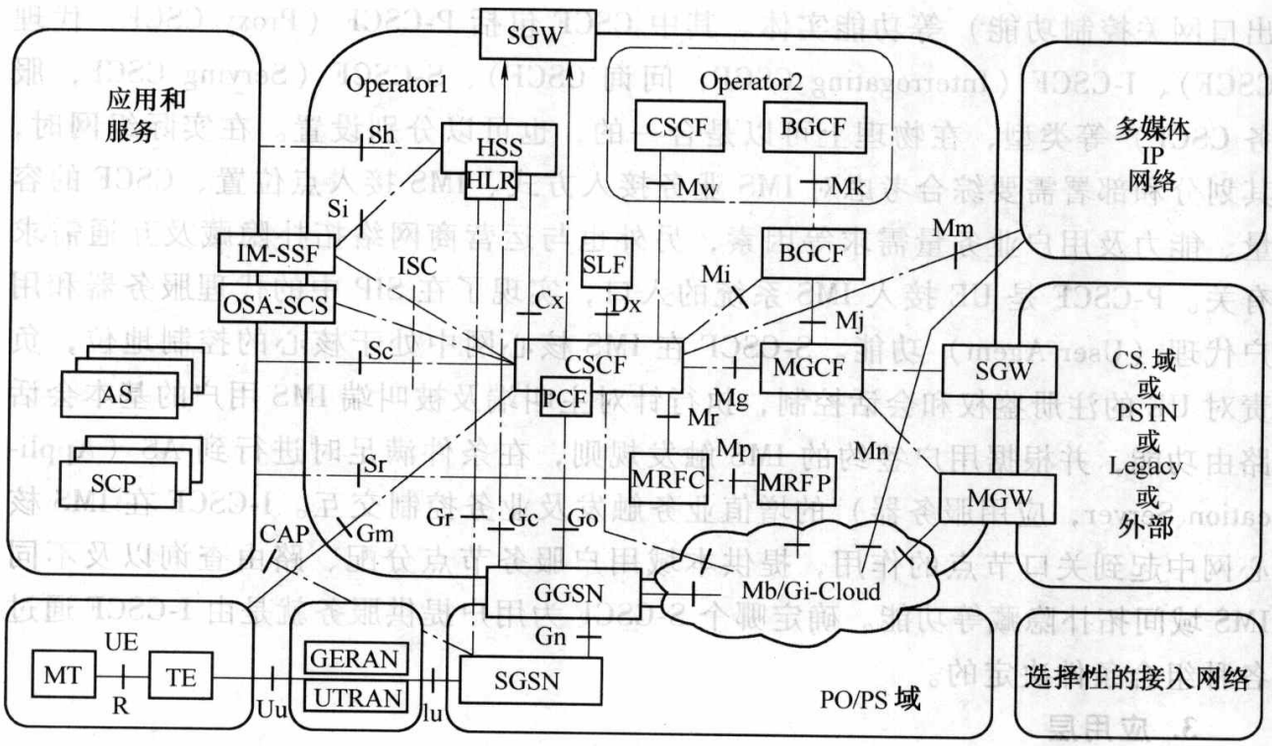


图 12-3 IMS 结构详细图

会话进行处理，其功能包括多媒体会话控制、地址翻译以及对业务协商进行服务转换等。CSCF 在 IMS 中实现了多媒体呼叫中主要的软交换控制功能，可以看作 IETF 架构中的 SIP Server，用于完成呼叫网关功能、呼叫业务触发功能和路由选择功能，根据各自不同的功能分为以下 3 种：

1）P-CSCF（Proxy-CSCF）是接入网到 IMS 的最先连接点，位于访问域中。从 SIP 观点看，P-CSCF 是作为一个出界与入界的 SIP 代理服务器。P-CSCF 从接入网中收到 UA（User Agent）发来的 SIP 注册和会话建立消息，转发到归属域中的 I-CSCF，再发至相应的 S-CSCF，反之也是。P-CSCF 不会对 SIP 的 INVITE 消息中请求的 URL 进行修改，仅将收到的请求消息进行转发。其内部的策略判决功能（Policy Decision Function，PDF）模块对多媒体业务的 QoS 要求进行策略判决。

主要功能：

① 把 UE（User Equipment）发来的 SIP 注册请求转发给 I-CSCF，该 I-CSCF 由 UE 提供的域名决定。

② 把 UE 发来的 SIP 消息转发给 S-CSCF，该 S-CSCF 由 P-CSCF 在 UE 发起注册流程时得到。

2）I-CSCF（Interrogating-CSCF）位于归属域中，是从访问域到归属域的入口点，也是 IMS 与其他 PLMN 的主要连接点，能屏蔽用户信息和网络拓扑，是可选节点。可将 I-CSCF 看作 SIP 代理服务器，用来给每个呼叫灵活选择相应的 S-CSCF，向 HSS 请求，查找 HSS 中的用户属性来确定是由哪个（些）S-CSCF

来为该用户服务；另外若有多个 S-CSCF 来完成负载分担，I-CSCF 就是用来完成这些 S-CSCF 的分配，可能基于轮询或者其他机制。主要功能：

①　注册功能：为用户指定一个 S-CSCF 来执行 SIP 注册。

②　对消息流的处理功能：从 HSS 中获取 S-CSCF 的地址，转发 SIP 请求；将其他网络传来的 SIP 请求路由到 S-CSCF。

3）S-CSCF（Serving-CSCF）与 I-CSCF 一起位于归属域中，是整个 IMS 的控制核心、IMS 会话管理的执行节点，控制呼叫和业务的相关状态，与 SCP 中的应用服务器互通。

可看作是 SIP 注册服务器和 SIP 代理服务器，用来接受用户注册，进行 URI 分析和重定向路由、触发应用服务器以及完成呼叫的控制和接续。维持了用户位置和用户 SIP 地址的绑定，S-CSCF 包含有网络运营上所需的状态信息，以协助进行 IMS 网络的注册、会话控制和业务支持。

主要功能：

①　注册功能：作为一个注册机，接收注册请求后，通过 HSS 使注册请求生效。

②　对消息流的处理：对已注册的会话终端进行会话控制，可作为代理服务器，接收请求后，进行内部的处理或转发，也可作为 UA，中断或发起 SIP 事务。

③　与业务平台进行交互，提供多媒体业务。

呼叫控制流程：

④　当代表主叫终端时，从数据库中取出为被叫用户提供服务的 I-CSCF 地址，判断若在另一网络，把 SIP 请求转发给该网络的 I-CSCF；若呼叫要路由到 PSTN 电路域，则转发给 BGCF。

⑤　当代表被叫终端时，若用户在归属网中，把 SIP 请求转发到 P-CSCF；若用户在访问域中，就将 SIP 请求发给 I-CSCF；当呼叫要路由到 PSTN/CS 域，就将 SIP 呼叫转给 BGCF。

4）基于 SIP 的各单元接口

IMS 中呼叫控制的具体接口如图 12-3 所示，其中各 CSCF 间接口为 Mw，CSCF 与其他网络接口为 Mm，CSCF 与 MGCF 接口为 Mg。各接口采用的协议均是基于对话的 SIP。

（2）HSS

HSS（Home Subscriber Server，归属签约用户服务器）是集中的用户数据库系统，存放着用户的认证信息、用户的业务信息、用户的漫游信息等。HSS 也用来记录用户的原始计费数据，并提供给分拣系统出账单。HSS 是由归属位置寄存器（HLR）和鉴权中心（AUC）演进而来，保存了用户文件和现在正由哪

个核心网络对这个用户服务的有关信息。它也支持用户认证和授权功能(AAA)。HSS 使用的 Cx 接口是惟一一个不基于 IETF 的协议，而是基于 IP 的(类似于 DNS)。

网络可能配有多个 HSS，这时需要配置 SLF (Subscriber Location Function，签约用户位置功能)，SLF 是一个简单的数据库，将用户地址映射到 HSS，以用户地址作为 SLF 的输入，请求获得包含了该用户配置信息的 HSS 作为输出。HSS 和 SLF 都执行 Diameter 协议，作为认证和鉴权的应用。

(3) 基于 IP 的网络单元间的通信流程

如果所处的环境是所有会话均在支持 IP 的终端上进行的，那么仅需要 CSCF 和 HSS 就够了。

2. IMS 与现存网络互通节点

实际上所处的环境中，并非所有会话均在支持 IP 的终端上进行，还有很长的与现存 PSTN 和移动网络的过渡时期，所以 IMS 提供了与现存网络互通的节点，主要有：

(1) MGCF

MGCF (Media Gateway Control Function，媒体网关控制功能) 控制一个或多个 IM-MGW 中媒体信道的连接，管理 PSTN 承载和 IP 流间的连接，从 CSCF 那里收到 SIP 消息，确定 IM-MGW 要建立什么内容，并创建适当的 ISUP 消息，送到 T-SGW。MGCF 根据被叫号码和来话情况选择 CSCF，并完成 PSTN 与 IMS 之间的呼叫控制协议转换。主要是将 IP 消息转换成 ISUP 或 Mega co 消息。

(2) BGCF

BGCF (Breakout Gateway Call Function，出口网关控制功能) 主要实现呼叫路由功能，用来选择与 PSTN/CS 域切入点相连的网络，收到 S-CSCF 请求，为呼叫选择适当的 PSTN 接口点。

1) 若发现该接口点与自己在同一网络，则选一个 MGCF，由 MGCF 来负责与 PSTN 交互。

2) 若发现接口点在另一网络时，BGCF 会把会话信令转发给另一网络相应的 BGCF。

(3) IM-MGW

IM-MGW (IP Multimedia-MGW，IP 多媒体媒体网关控制器) 在 MGCF 控制下完成资源控制，通过回波消除器和码转换器，实现媒体转换和帧协议转换功能，将媒体从一个形式转换到另一个形式，UMTS 中一般是 PSTN 的 PCM 和基于 IP 的编解码格式。

(4) T-SGW

T-SGW 完成传输层的信令转换，不对应用层消息解释，仅对底层的 SCTP 进

行解释，保证信令正确路由。IMS 中多数通信都是基于 IP 的，仅在与 PSTN 和现存的移动网互通的信令路径上不是，PSTN 目前只能理解 SS7，T-SGW 就是将 SS7 向 IP 转换（即 SCTP/IP 和 SS7 MTP 间），如果将来 MGCF 能支持 SS7，就不再需要 SGW 了。

(5) 互通时的通信流程

移动终端发送一个请求消息，来建立一个业务，从 SGSN 到 GGSN，该请求被发送给 S-CSCF（经 P-CSCF 和 I-CSCF），由 S-CSCF 来确定如何建立通信链路。

1）与 PSTN 互通。若 S-CSCF 判定呼叫要传往 PSTN，则将 INVITE 消息转发给网内的 BGCF，BGCF 依据 Local Policy 来选择互通发生的网络，若判断互通发生在同一网络内，BGCF 会选择 MGCF 来执行互通，若不在同一网络内，传给该网中的 BGCF，由 MGCF 来控制 MGW 转换媒体，并执行互通。

2）与承载互通。IMS 上的语音承载需与其他网络互通，由 MGW 来支持，该 MGW 需能支持双方的编解码转换。与 CS 网（PSTN、ISDN、PLMN 的 CS 域）互通，是支持 AMR 到 G.711 实现的。

3. IMS 多方会话功能单元

如果遇到多方会话时，加入了 MRF 模块。MRF 分为 MRFC（Media Resource Function Controller Part，媒体资源功能控制部分）和 MRFP（Media Resource Function Process Part，媒体资源功能处理部分），分别完成媒体流的控制和承载功能，根据 S-CSCF 和/或应用服务器的调用，MRFC 通过 H.248 和 RTSP 控制 MRFP 完成相应的媒体流编解码、转换、混合和播放。

1）MRFC 控制 MRFP 中的媒体流资源，翻译来自应用服务器和 S-CSCF 的信息。

2）MRFP 控制 Mb 接口点的承载，处理多媒体流（如语音编码的转换、媒体流分析等），在多方会议时，混合输入的媒体流。

12.2.3　IMS 组网结构分析

为了表示其具有自组网能力，可以将其组件分处在 3 个逻辑区域：归属域（Home Network）、访问域（Visit Network）和网关域（Gateway Network）。

1. IMS 归属域与访问域

网关域的功能非常简单，就是完成 IMS 和语音交换网（PSTN、PLMN 以及 MSC-S 域等）的互通。媒体网关（MGW）完成媒体流的转换，即 PCM 媒体流到 RTP 媒体流的编解码转换，并在媒体网关控制器（MGC）的控制下完成呼叫的接续，控制协议采用的是 H.248 或者 MGCP，它也要同时处理来自语音交换网的 ISUP 信息和来自 IMS 的 SIP 信息，并完成被叫定位、路由分析等相关事件；信令网关（SGW）使基于 MTP 的 ISUP（SCCP）信息能在 IP 网上转送到 MGC，

通常采用的是 ISUP（SCCP）/M3UA/SCTP/IP 方式。

访问域包含 SIP 客户端和 P-CSCF（代理 CSCF），P-CSCF 其本质是一个 SIP 代理服务器，其作用是受理所有 SIP 客户端的注册和呼叫，根据主叫/被叫的 SIP URI 去找到其相应的归属域，完成注册过程和呼叫接续。多个 P-CSCF 可以共同存在用来完成负载分担，在规划网络的时候，可以根据该区域内活动用户数目（包括本地用户和漫游用户）来配置 P-CSCF 的数量。

归属域对应于 SIP URI（SIP：user@ domain）中的 domain，一般来说，每个 domain 都应该有自己的归属域，但是对于用户数目不多的 domain 来说，其实几个 domain 可以共享一套归属域设备，这也意味着运营商可以将归属域设备租用给企业用户，因此企业用户不需要购置相对昂贵的归属域设备。

采用这种分为访问域和归属域的结构，优点在于，由归属域来提供业务特征，这就意味着移动终端不受限于访问网络的性能，避免了像当前网络中业务的提供非常受 MSC 配置的影响。

2. 从访问域和归属域来看的数据流程

从移动终端到 SGSN，经 GGSN 发到目标网络，可能是另一个 R5 网的 GGSN，或者是到 PSTN 的 MGW，或者是基于 IP 的 Internet，而将 CSCF 全部旁路掉。可见，在 IMS 中还是采用信令与用户数据不同路的方式来进行通信的。

12.2.4 IMS 业务体系介绍

3GPP 标准为业务的开发提供了 3 种机制，即针对 IP 业务的 CAMEL 功能、开放服务体系（OSA）和会话初始协议（SIP），并在不同的版本中给出了相应的定义。在 UMTS R5 中，开放、融合的业务架构改变了业务提供和业务开发模式，这种改变可快速提供新业务。在以往的业务开发模式中，业务的开发是一种“垂直型”模型，与下层网络紧密结合：

1）业务的发布与网络结构有关，不同的网络可能需要发布不同的版本；

2）不同的网络之间，业务交互困难，业务融合难度大；

3）业务实现复杂，开发周期长，投资收益低，不利于第三方开发。

IMS 引入了新型业务开发模式，业务的开发呈现“水平型”结构，业务与下层网络松耦合：

1）符合协议标准的业务，可发布于任何网络，只需一个版本；

2）具有快速、易用、灵活、动态的业务开发环境；

3）通过 SIP、API（如 Parlay），使业务间具有良好的集成关系；

4）开发难度和周期大大减少，投资收益高，第三方开发者只需要了解 SIP、API，而不需深入了解电信网络知识。

1. IMS 支持的业务架构

为了适应下一代网络业务与控制分离的原则，IMS 必须提供开放的接口来接入各种业务服务器，允许各种业务提供商通过标准的接口向网络提供服务。

IMS 的业务框架是 IMS 系统比较成功的一个方面，通过基于 SIP 的 ISC（Internal Service Control）接口，由 S-CSCF 与各种应用服务器组成。由于 IMS 融合了语音和数据业务，又结合了 IP 网的灵活性和 SIP 的可扩展性，使得 IMS 支持的业务更加丰富。R5 在业务方面除原有的 CAMEL、OSA 之外，又新支持 SIP AS。IMS 的业务架构要能适应不同业务的接入要求，IMS 的业务架构由 S-CSCF 以及各种应用服务器组成，与当前普遍采用的开放性业务架构一致，都是三层结构：第一层是应用服务器（AS）；第二层是业务能力服务器（SCS）；第三层是 S-CSCF。业务可以在分布式架构中开发。

（1）应用服务器层

应用服务器层：AS（Application Server）提供业务逻辑，用来支持用户的各种业务属性。根据所支持业务类型的不同分为以下三种：

1）SIP 应用服务器：完成 Internet 业务，由于 ISC 采用了 SIP，故可直接与 S-CSCF 相连，CSCF 能以 SIP 服务器的方式直接调用这些业务，减少了信令转换。SIP 基于 C/S 模式，是一种多媒体会话的应用层控制信令，完成双方或多方多媒体会话的创建、修改和终止等工作，以提供 IP 电话和基于 IP 的多媒体会话业务。其业务逻辑的提供可通过脚本语言、SIP-CGI 或 API（如 Servlet 等）编程实现。

2）OSA 应用服务器：完成基于 OSA API 开发的第三方业务。UMTS 定义了 OSA 来允许第三方业务提供商进行新业务的开发，由 OSA 提供的安全 API 来接入 UMTS，使用网络的性能，而不再受限于运营商提供的业务。不同的应用提供商就可以通过开放的网络访问接口获得底层网络的承载能力，并进行应用开发。

3）CAMEL SE：完成传统的移动智能网业务。

（2）业务能力服务器层

由于 OSA API 和 CAMEL 环境不支持基于 SIP 和扩展 SIP 的 ISC 接口，所以出现了中间层即业务能力服务器层。

1）OSA SCS：完成 OSA API 与 ISC 接口的信令进行映射。通过 SIP 方式执行一个或多个 OSA 业务能力特征（OSA SCF）

2）IM-SSF：其行为类似于传统智能网的呼叫控制功能和业务转换功能，使现有的基于 CAMEL 业务环境的增值业务能继续提供，并生成新的增值业务。SSF 完成了 CAP 与 SIP 的转换，是 SIP 和 CAMEL 间互通模块。

（3）业务控制层（S-CSCF）

S-CSCF 通过 ISC 接口与业务平台相连，在 S-CSCF 看来，SIP AS、OSA SCS 和 IM-SSF 都执行相同的接口行为，ISC 使得各种服务器都能接入 IMS。这里 S-

CSCF 检测是否并如何包含业务逻辑来提供增值业务。同时，S-CSCF 使用 Cx 接口从 HSS 中提取用户配置信息，再由 S-CSCF 使用 ISC 接口与不同业务平台互相通信。

AS 和 CSCF 之间采用 ISC 通信，选择采用单一的业务控制接口是为了避免增加 S-CSCF 的复杂性。现在的 AS 一般基于 J2EE 规范，应用可以是基于 EJB 的，通过 SIP Servlet 和 SIP 实体对话，通过 HTTP Servlet 和 Web 实体对话。

2. IMS 业务提供框架

IMS 中的业务提供是基于在归属域中的业务控制，如所有消息都是通过在归属域中的 S-CSCF 路由的。在其业务提供框架中，S-CSCF 根据从 HSS 下载的触发信息，直接将 SIP 消息转发给 AS，完成将业务逻辑转换成业务配置的应用。

3. IMS 3 种支持业务分析

IMS 提供的连在 CSCF 上的 3 种业务平台，是分别针对不同种类的业务提出的。

（1）基于 SIP 的业务应用

SIP 是由 IETF 提出的 IP 电话信令协议。主要目的是为了解决 IP 网中的信令控制，以及同软交换的通信，从而构成下一代的增值业务平台，对电信、银行、金融等行业提供更好的增值业务。

在 IMS 业务体系中，S-CSCF 相当于是软交换的呼叫控制平台，IMS 基于 SIP 的业务平台是通过专用的 SIP AS 实现的，实现 IM（即时通信）、点击呼叫、Web800、Presence、会议电话、数字传真、多媒体消息、VOD、彩铃、彩名、可视电话等丰富的实时和非实时的多媒体业务。

（2）基于 OSA 的业务应用

IMS 支持 PARLAY/OSA 标准业务接口，不仅可以完美地实现传统的基本电信业务，如：呼叫前转、呼叫等待、多方通话、主叫显示、主叫显示限制、呼叫限制等，而且还可以为第三方业务供应商提供标准接口，大大提高运营商提供新业务的能力。

（3）基于 CAMEL 的业务应用

CAMEL 业务是一种网络特性而不是补充业务，它采用智能网的原理，通过增加智能网的功能模块，即使当用户漫游出归属 PLMN，网络运营者也可以为用户提供运营者特定的业务。为保护运营商已有的投资，IMS 提供了对传统的智能网络的接口，可以通过 SIP/INAP/CAMEL/WIN 接口接入 SCP，支持传统的智能业务，如预付费、虚拟专网、卡类业务等，目前移动网络中应用的预付费业务、VPN 等业务都是采用这种方式实现的，从而实现了对已有智能业务的完美继承。

12.2.5　SIP 与 IMS

IMS 是目前国际认可的 NGN 体系架构的子系统之一，它还处在不断的发展和完善中。3GPP 采用了 SIP 作为 IMS 网络的核心信令协议，用于会话的建立、更改和拆除，并且独立于媒体的传输。同时 3GPP 对 IETF 定义的 SIP 做了扩展，通过开发新的方法、头部字段、消息体类型来实现强大的可扩展性，支持了未来潜在的应用。例如，由于无线传输资源的带宽有限，要求支持 SIP 消息的压缩；由于无线接入的引入使呼叫建立时间增加，要求 SIP 的基础定时器由 500ms 改成 2s。如果 IMS 同时支持固定和移动用户，需要支持固定和移动不同能力的 SIP，并且要求 IMS 网络能够识别终端的接入方式，根据不同的接入方式选用不同的协议参数。从 IMS 的体系结构中我们可以看出，SIP 在 IMS 中起着核心作用，如图 12-4 所示。

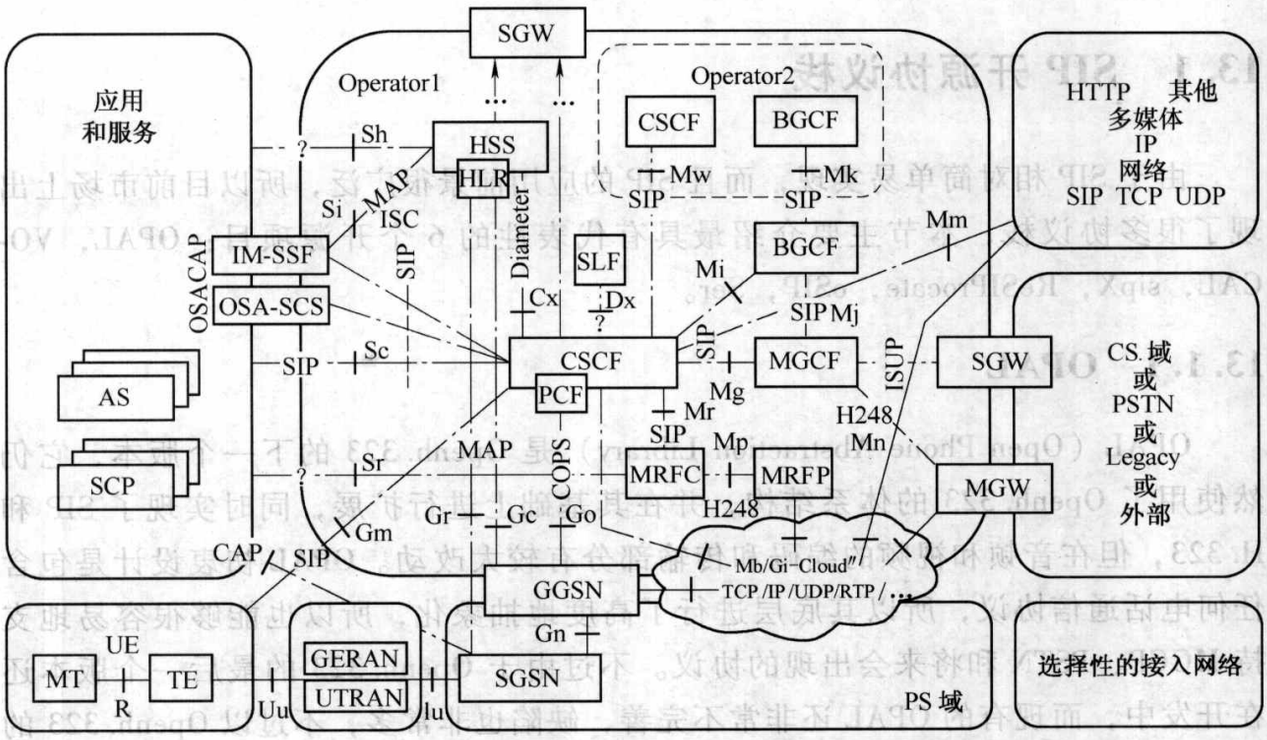


图 12-4　IMS 中功能间使用的协议图

从图 12-4 可以看出，SIP 在 IMS 中处于非常重要的地位，它在 IMS 的核心网中起到联系和控制各功能单元的作用。

第 13 章　SIP 应用开发

随着 VoIP、IMS 和 NGN 技术的发展，H.323 时代即将过渡到 SIP 时代，在 H.323 的开源协议栈中，Open h.323 占统治地位，它把一个复杂而又先进的 H.323 协议栈展现在普通程序员的眼前，为 H.323 的普及立下了汗马功劳。然而当在 SIP 时代，则出现了群雄割据的状况，SIP 相对于 H.323 简单，灵活，于是各种协议栈层出不穷，使得开发者不知如何选择相应的协议栈来开发自己的协议栈，本章的主要是介绍一些目前流行的 SIP 协议栈，以及如何进行应用开发。

13.1　SIP 开源协议栈

由于 SIP 相对简单易实现，而且 SIP 的应用前景很广泛，所以目前市场上出现了很多协议栈，本节主要介绍最具有代表性的 6 个开源项目：OPAL，VOCAL，sipX，ReSIProcate，oSIP，Ser。

13.1.1　OPAL

OPAL（Open Phone Abstraction Library）是 Openh.323 的下一个版本，它仍然使用了 Openh.323 的体系结构，并在其基础上进行扩展，同时实现了 SIP 和 H.323，但在音频和视频的编码和传输部分有较大改动。OPAL 初衷设计是包含任何电话通信协议，所以其底层进行了高度地抽象化，所以也能够很容易地支持 MGCP、PSTN 和将来会出现的协议。不过由于 Openh.323 的最后一个版本还在开发中，而现有的 OPAL 还非常不完善，缺陷也非常多，不过以 Openh.323 的开发班底，一定能让 OPAL 做得完美。

可以到下面的 CVS 中下载当前最新的开发版本：

CVS:: pserver: anonymous@ cvs. sourceforge. net: /cvsroot/openh323/opal

Language: C + +

VxWorks port: Yes

Win32 port: Yes

Linux port: Yes

Supports RFC 3261: Yes

Supports RFC 2327: Yes

Supports RFC 3264：Yes
Supports RFC 3263：No
Supports RFC 3515：Yes
Supports RFC 3262：No
Supports RFC 3311：No
TCP：Yes
UDP：Yes
SIZE：8MB
License：MPL
Document：None
Samples：UA，GK

13.1.2　VOCAL

VOCAL 是由 vovida. org 开发的 SIP 系统，VOCAL 应该是目前功能最完善、使用者最多的开源 SIP 协议栈了。它不只包括了协议栈，还包括了 h. 323 与 SIP 转换网关，对 SIP 的各种服务器的功能支持也非常完善。不过很可惜，不支持 windows 平台，而且自从 vovida 被 CISCO 收购以后就停止了开发，最后的版本是 2003 年 4 月的 1. 5. 0。可以到下面的 CVS 中下载当前最新的开发版本：

CVS:: pserver：anonymous@ cvs. vovida. org：/cvsroot/vocal
Language：C + +
VxWorks port：No
Win32 port：Partial
Linux port：Yes
Supports RFC 3261：Partial
Supports RFC 2327：Yes
Supports RFC 3264 ：
Supports RFC 3263：
Supports RFC 3515：Yes
Supports RFC 3262：
Supports RFC 3311：
TCP：Yes
UDP：Yes
SIZE：6MB
License：Vovida software licencse
Document：Few

Samples: UA, GK, GW

13.1.3 sipX

sipX 是一个 SIP 系统，由 SIPFoundry 开发。sipX 是从 reSIProcate 分离出来的，sipX 除了包括 SIP 协议栈外，还包括了 sipXphone, sipXproxy, sipXregistry 等等，由它们构成了完整的 SIP 系统，而且 sipX 还支持嵌入式系统，各个模块可以按需取舍。不过可惜是几乎没有任何开发文档。可以到下面的 SVN 中下载当前最新的开发版本：

SVN: http: //scm. sipfoundry. org/viewsvn/

Language: C + +

VxWorks port: Yes

Win32 port: Yes

Linux port: Yes

Supports RFC 3261: Yes

Supports RFC 2327: Yes

Supports RFC 3264: Yes

Supports RFC 3263: Yes

Supports RFC 3515: Yes

Supports RFC 3262: No

Supports RFC 3311: No

TCP: Yes

UDP: Yes

SIZE: <4 MB

License: LGPL

Document: None

Samples: UA, GK, GW

13.1.4 ReSIProcate

ReSIProcate 同样也是由 SIPFoundry 开发，ReSIProcate 最开始起源于 Vocal，由于 Vocal 开始只支持 RFC3254，为了支持最新的 RFC3261，ReSIProcate 诞生了，但现在，ReSIProcate 已经成为一个独立 SIP 协议栈了，它十分稳定，并且很多商业程序都在使用。可以到下面的 SVN 中下载当前最新的开发版本：

SVN: http: //scm. sipfoundry. org/viewsvn/resiprocate/main/sip/

Language: C + +

VxWorks port: No

Win32 port: Yes
Linux port: Yes
Supports RFC 3261: Yes
Supports RFC 2327: Yes
Supports RFC 3264: Yes
Supports RFC 3263: Partial
Supports RFC 3515: Yes
Supports RFC 3262: No
Supports RFC 3311: No
TCP: Yes
UDP: Yes
SIZE: < 2.5 MB
License: Vovida
Document: Few
Samples: None

13.1.5　oSIP

oSIP 的开发开始于 2000 年 7 月，第一个版本在 2001 年 5 月发布，到现在已经发展到 2.0.9 了。它采用 ANSI C 编写，而且结构简单小巧，所以速度特别快，它并不提供高层的 SIP 会话控制 API，它主要提供一些解析 SIP/SDP 消息的 API 和事务处理的状态机，oSIP 的作者还开发了基于 oSIP 的 UA lib：exosip 和 proxy server lib：partysip.

CVS:: ext: anoncvs@ savannah. gnu. org: /cvsroot/osip
Language: C
VxWorks port: Yes
Win32 port: Yes
Linux port: Yes
Supports RFC 3261: Yes
Supports RFC 2327: Yes
Supports RFC 3264: Yes
Supports RFC 3263: Yes
Supports RFC 3515: No
Supports RFC 3262: No
Supports RFC 3311: Yes
TCP: Yes

UDP：Yes

SIZE：400kB

License：LGPL

Samples：UA，GK

综合上述评测，可以看出 5 种 SIP 栈各有千秋，OPAL 有发展潜力，VOCAL 比较完善，sipX 兼容性好，ReSIProcate 较稳定，oSIP 小巧而快速。所以要根据应用的不同选择恰当的协议栈进行研究开发。

13.2 oSIP 简介

oSIP 是按照 RFC3261（SIP）和 RFC2327（SDP）标准，并使用标准 C 语言编写的一个 SIP 协议栈。oSIP 协议栈结构简单而小巧，它并不提供高层的 SIP 会话控制的 API，它主要提供一些解析 SIP/SDP 消息的 API 和事务处理的状态机。oSIP 开发库是第一个自由软件项目。在第三代网络体系中，越来越多的电信运营商将要使用软交换。发展的一个侧面是在不久的将来，Linux 将更多支持多媒体工具。oSIP 作为 SIP 开发库，将允许建造互操作的注册服务器、用户代理（软电话）和代理服务器。

oSIP 支持线程安全，既可以用于多线程的编程模式，也可以用于单线程的编程模式；支持多种操作系统，如 Solaris、UNIX 、Windows、Linux 系列，oSIP 也可用于实时操作系统 VxWorks 当中；oSIP 可以用来开发 User Agent、IP soft-phone 和 SIP 代理服务器等。

oSIP 协议栈的在于：

1）oSIP 没有给开发者限定在特定的某个执行模式下，能够使开发者选定一个比较适合自己的模式。

2）oSIP 的各个模块是相对清晰、独立的，因而去掉某个模块时也比较容易。

3）oSIP 的解析器提供了较为完善的 API，包含了消息的构造、修改和产生等。

13.2.1 oSIP 的结构

oSIP 主要包括三大部分的内容：解析器模块（也叫语法分析器）、状态机模块（有限状态机和无限状态机）和工具模块。

解析器模块的功能：该模块主要完成对 SIP 消息结构剖析、SDP 消息的结构剖析以及 URI 结构的剖析；oSIP 现在支持 utf8 和完整的 SIP 语法。这个语法分析器，其能够读写任何在 RFC 中描述的 SIP 消息，如 Via、Call-ID、To、From、

Contact、Cseq、Route、Record-Route、MIME-version、Content-Type 和 Content-length 等。

状态机模块的功能：完成对某个事务（注册过程和呼叫过程等）状态记录，并在特定状态下触发相应的事件或回调函数。

工具模块的功能：该模块提供一些 SDP 等处理的一些工具。

oSIP 的模块结构图如图 13-1 所示。

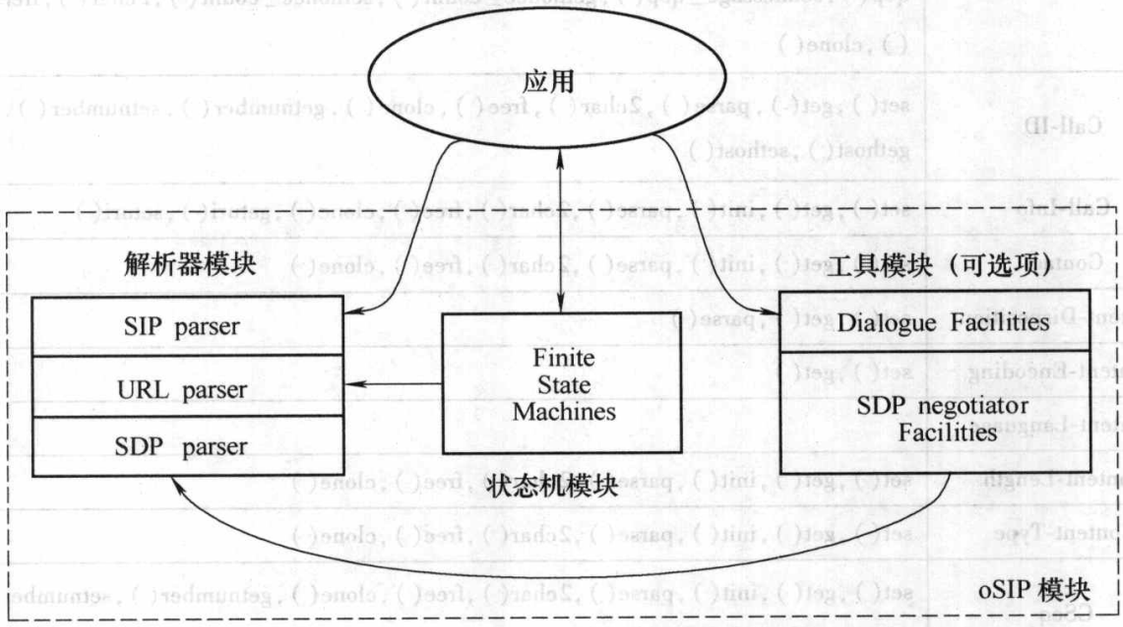


图 13-1　oSIP 模块结构

13.2.2　解析器模块

从图 13-1 中可以看出，解析器（Parsers）模块又分成三部分：SIP parser、URL parser、SDP parser。下面分别介绍。

1. SIP Parser

oSIP 的 SIP Parser 处理 SIP 头域（SIP Header fields）及其相应的操作功能列表 13-1 如下。

表 13-1　SIP 头域功能函数

SIP Header(头域)	Functions(函数名称简写)	
Accept	set( ),get( )	☆
Accept-Encoding	set( ),get( ),init( ),parse( ),2char( ),free( ),clone( ) Getelement( ),setelement( )	☆
Accept-Language	set( ),get( )	☆
Alert-Info	set( ),get( )	☆
Allow	set( ),get( )	☆
Authentication-Info		

（续）

SIP Header（头域）	Functions（函数名称简写）	
Authorization	Init( ), set( ), parse( ), get( ), getauth_type( ), setauth_type( ), Getusername( ), setusername( ), getrealm( ), setrealm( ), Getnonce( ), setnonce( ), geturi( ), seturi( ), getresponse( ), Setresponse( ), getdigest( ), setdigest( ), getalgorithm( ), Setalgorithm( ), getcnonce( ), setcnonce( ), getopaque( ), Setopaque( ), getmessage_qop( ), setmessage_qop( ), getnonce_count( ), setnonce_count( ), 2char( ), free( ), clone( )	☆
Call-ID	set( ), get( ), parse( ), 2char( ), free( ), clone( ), getnumber( ), setnumber( ), gethost( ), sethost( )	☆
Call-Info	set( ), get( ), init( ), parse( ), 2char( ), free( ), clone( ), geturi( ), seturi( )	☆
Contact	set( ), get( ), init( ), parse( ), 2char( ), free( ), clone( )	☆
Content-Disposition	set( ), get( ), parse( )	☆
Content-Encoding	set( ), get( )	☆
Content-Language		
Content-Length	set( ), get( ), init( ), parse( ), 2char( ), free( ), clone( )	☆
Content-Type	set( ), get( ), init( ), parse( ), 2char( ), free( ), clone( )	☆
CSeq	set( ), get( ), init( ), parse( ), 2char( ), free( ), clone( ), getnumber( ), setnumber( ), getmethod( ), setmethod( )	☆
Date		
Error-Info	set( ), get( )	☆
Expires		
From	set( ), get( ), init( ), parse( ), 2char( ), free( ), clone( ), getdisplayname( ), setdisplayname( ), geturl( ), seturl( ), param_get( ), param_parseall( ), param_setvalue( ), param_getvalue( ), param_getname( ), param_setname( ), compare( )	☆
In-Reply-To		
Max-Forwards		
Min-Expires		
MIME-Version	set( ), get( )	☆
Organization		
Priority		
Proxy-Authenticate	set( ), get( )	☆
Proxy-Authorization	set( ), get( )	☆
Proxy-Require		

（续）

SIP Header(头域)	Functions(函数名称简写)	
Record-Route	set( ) ,get( ) ,init( ) ,parse( ) ,2char( ) ,free( )	☆
Reply-To		
Require		
Retry-After		
Route	set( ) ,get( ) ,init( ) ,parse( ) ,2char( ) ,free( )	☆
Server		
Subject		
Supported		
Timestamp		
To	set( ) ,get( ) ,init( ) ,parse( ) ,2char( ) ,free( ) ,clone( )	☆
Unsupported		
User-Agent		
Via	set( ) ,append( ) ,get( ) ,init( ) ,free( ) ,parse( ) ,2char( ) ,setversion( ) ,getversion( ) ,setprotocol( ) ,getprotocol( ) ,sethost( ) ,gethost( ) ,setport( ) ,getport( ) ,setcomment( ) ,getcomment( ) ,clone( )	☆
Warning		
WWW-Authenticate	Init( ) ,set( ) ,quoted_string_set( ) ,token_set( ) ,parse( ) ,get( ) ,getauth_type( ) ,setauth_type( ) ,getrealm( ) ,setrealm( ) ,getdomain( ) ,setdomain( ) ,getnonce( ) ,setnonce( ) ,getstale( ) ,setstale( ) ,getopaque( ) ,setopaque( ) ,getalgorithm( ) ,setalgorithm( ) ,getqop_options( ) ,setqop_options( ) ,2char( ) ,free( ) ,clone( )	☆

注：标示“☆”表示 oSIP 支持该头域（SIP Header fields）解析处理，未注的表示目前还没有解析处理（这些被保存在字符串中，可自行处理分析），会在后续版本中逐步补充。

2. SDP Parser

SDP 的格式一般为：

< type > = < value >

type 通常为一个英文字母，其取值如下（按照 RFC2327，带“*”的表示为可选条目）：

Session description

v = (protocol version)

o = (owner/creator and session identifier).

s = (session name)

i = * (session information)

u = * (URI of description)
e = * (email address)
p = * (phone number)
c = * (connection information-not required if included in all media)
b = * (bandwidth information)
One or more time descriptions (see below)
z = * (time zone adjustments)
k = * (encryption key)
a = * (zero or more session attribute lines)
Zero or more media descriptions (see below)

Time description
t = (time the session is active)
r = * (zero or more repeat times)

Media description
m = (media name and transport address)
i = * (media title)
c = * (connection information - optional if included at session-level)
b = * (bandwidth information)
k = * (encryption key)
a = * (zero or more media attribute lines)

在 oSIP 中处理的 type 和相应操作功能如表 13-2 所示。

表 13-2 SDP Parser 功能函数表

type(类型)	Functions(函数名称简写)
v	version_set( ), version_get( )
o	origin_set( ), username_get( ), sess_id_get( ), sess_version_get( ), nettype_get( ), addrtype_get( ), addr_get( )
s	name_set( ), name_get( )
i	info_set( ), info_get( )
u	uri_set( ), uri_get( )
e	email_add( ), email_get( )
p	phone_add( ), phone_get( )
c	connection_add( ), connection_get( ), nettype_get( ), addrtype_get( ), addr_get( ), addr_multicast_ttl_get( ), addr_multicast_int_get( )

（续）

type（类型）	Functions（函数名称简写）
b	bandwidth_add( ), bwtype_get( ), bandwidth_get( )
t	time_descr_add( ), start_time_get( ), stop_time_get( )
r	repeat_add( ), repeat_get( )
z	adjustments_set( ), adjustments_get( )
k	key_set( ), keytype_get( ), keydata_get( )
a	attribute_add( ), att_field_get( ), att_value_get( )
m	media_add( ), media_get( ), port_get( ), number_of_port_get( ), proto_get( ), payload_add( ), payload_get( )

另外，oSIP 还包含对 SDP 包的一些基本操作［set（），get（），init（），parse（），2char（），free（），clone（）］，及对各类 type 的 init（）和 free（）操作。

3. URL Parser

这里的 URL 是指 SIP 中的 URI，URI 有很多参数格式，在 RFC3261 中列举了一些比较全面的例子：

The URIs within each of the following sets are equivalent:

sip:alice@ atlanta. com;transport = TCP
sip:alice@ AtLanTa. CoM;Transport = tcp

sip:carol@ chicago. com
sip:carol@ chicago. com;newparam = 5
sip:carol@ chicago. com;security = on

sip: biloxi. com; transport = tcp; method = REGISTER? to = sip: bob @ biloxi. com
sip: biloxi. com; method = REGISTER; transport = tcp? to = sip: bob @ biloxi. com

sip:alice@ atlanta. com? subject = project x&priority = urgent
sip:alice@ atlanta. com? priority = urgent&subject = project x

The URIs within each of the following sets are not equivalent:

SIP:ALICE@ AtLanTa. CoM;Transport = udp　　（different usernames）
sip:alice@ AtLanTa. CoM;Transport = UDP

```
sip:bob@ biloxi. com                                    (can resolve to different ports)
sip:bob@ biloxi. com:5060

sip:bob@ biloxi. com                                    (can resolve to different transports)
sip:bob@ biloxi. com;transport = udp

sip:bob@ biloxi. com                    (can resolve to different port and transports)
sip:bob@ biloxi. com:6000;transport = tcp

sip:carol@ chicago. com                 (different header component)
sip:carol@ chicago. com? Subject = next%20meeting

sip:bob@ phone21. boxesbybob. com       (even though that's what
sip:bob@ 192. 0. 2. 4                  phone21. boxesbybob. com resolves to)
```

在 oSIP 中处理 SIP URI 提供有比较多的操作函数，主要有对 host、port、username、password、scheme 的 get（）和 set（），以及对参数的初始化设置和剖析处理。详细函数名称请参考源代码中的 url. h。

13.2.3 状态机模块

oSIP 状态机（Finite State Machines）主要分为 4 类，分别为：

1）ICT——Invite Client（outgoing）Transaction

2）NICT——Non-Invite Client（outgoing）Transaction

3）IST——Invite Server（incoming）Transaction

4）NIST——Non-Invite Server（incoming）Transaction

下面分别介绍 oSIP 的各种状态机的处理过程：

1. ICT 状态机（见图 13-2）

注：——cb_ict_xxx_xxxx——→表示从一种状态转换到另一种状态时将调用该回调函数；

————→表示从一种状态转换到另一种状态时不调用任何回调函数。

cb_ict_Nxx_received 中的 N 表示一下几个值：

3——cb_ict_3xx_received

4——cb_ict_4xx_received

5——cb_ict_5xx_received

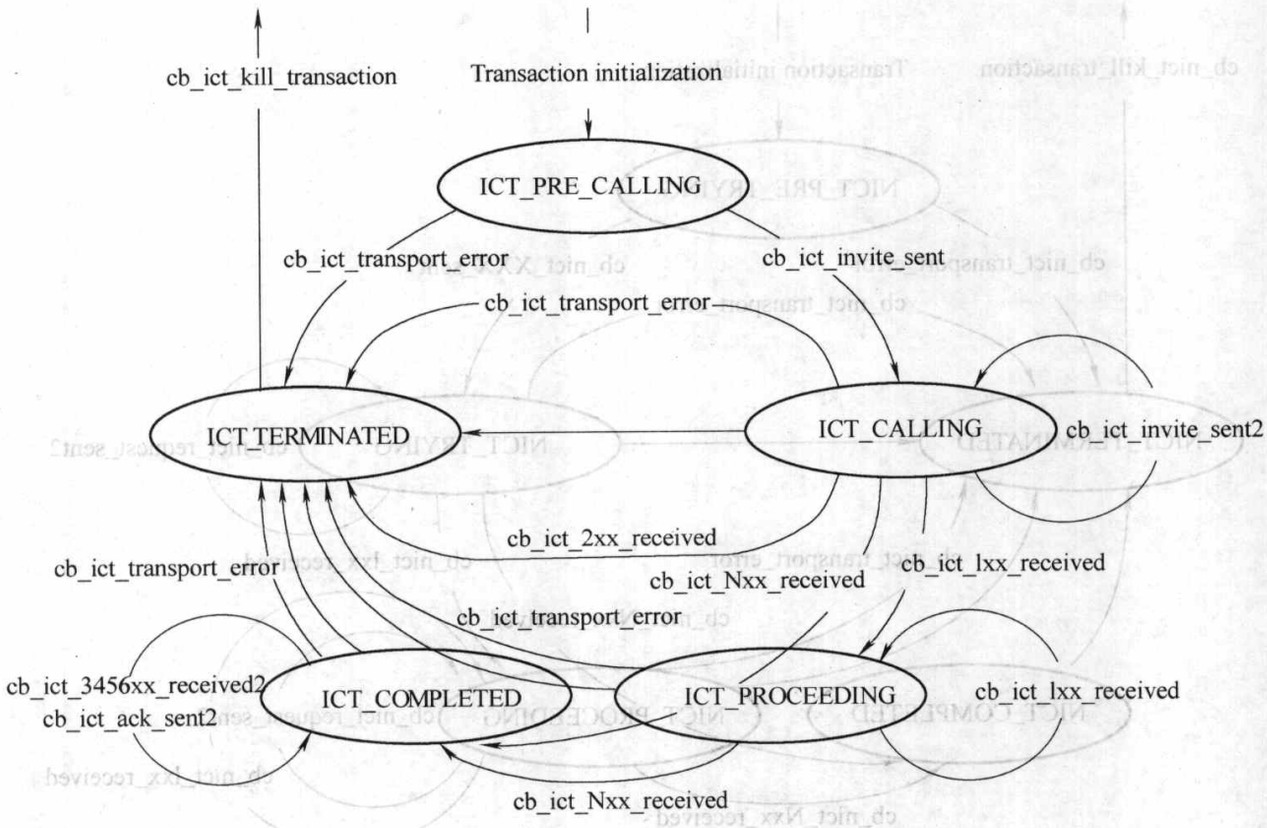


图 13-2　ICT 状态机

6——cb_ict_6xx_received

2. NICT 状态机（见图 13-3）

注：——cb_nict_xxx_xxxx ——→表示从一种状态转换到另一种状态时将调用该回调函数；

————→表示从一种状态转换到另一种状态时不调用任何回调函数。

cb_nict_XXX_sent 中的 XXX 表示一下几种消息类型：

register——cb_nict_register_sent

bye——cb_nict_bye_sent

options——cb_nict_options_sent

info——cb_nict_info_sent

cancel——cb_nict_cancel_sent

notify——cb_nict_notify_sent

subscribe——cb_nict_subscribe_sent

unknown——cb_nict_unknown_sent

cb_nict_Nxx_received 中的 *N* 表示一下几个值：

2——cb_nict_2xx_received

3——cb_nict_3xx_received

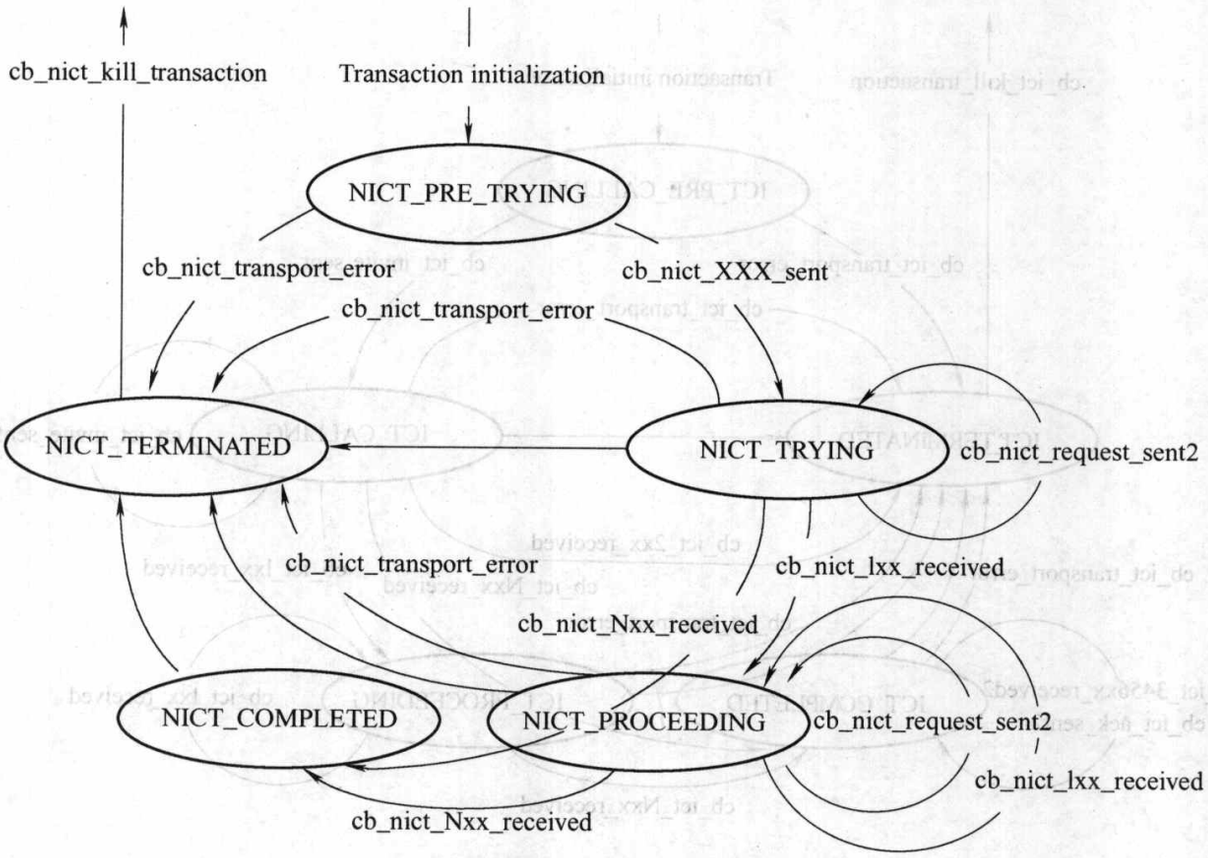


图 13-3 NICT 状态机

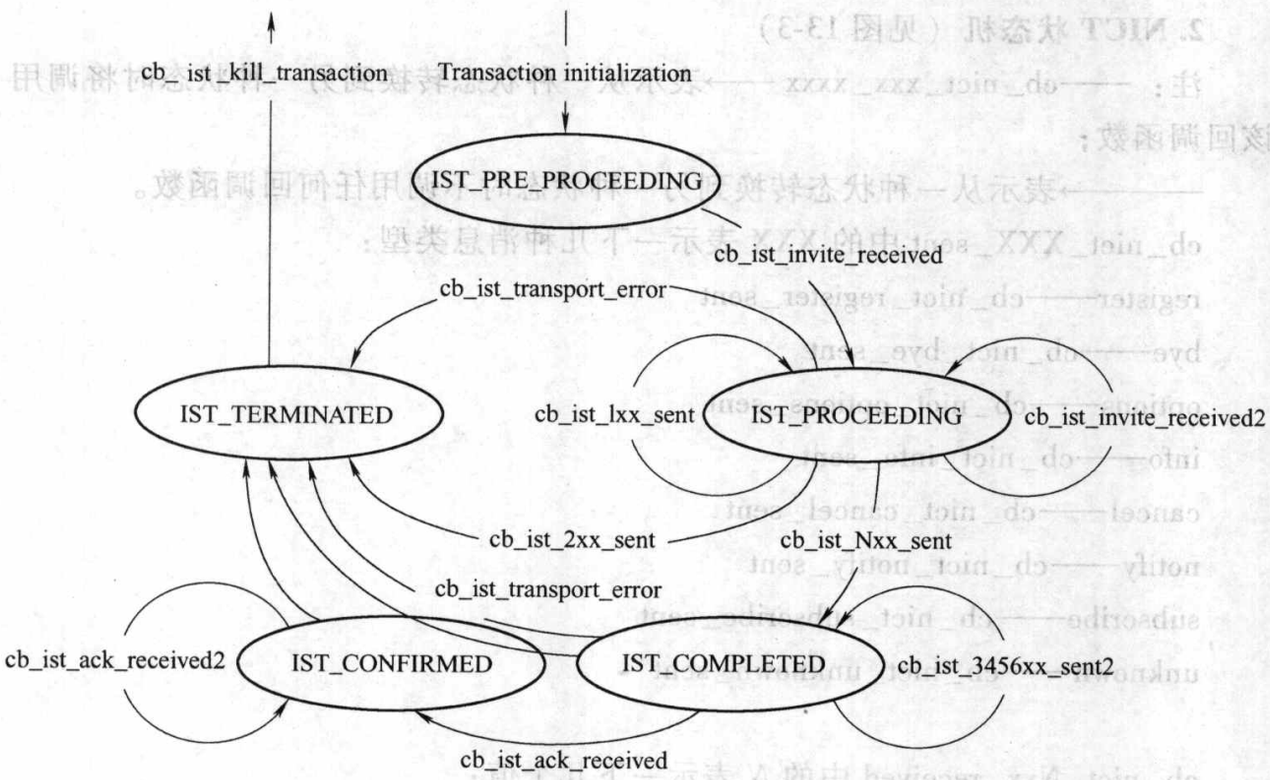


图 13-4 IST 状态机

4——cb_nict_4xx_received

5——cb_nict_5xx_received

6——cb_nict_6xx_received

3. IST 状态机（见图 13-4）

注：——cb_ist_xxx_xxxx ——→表示从一种状态转换到另一种状态时将调用该回调函数；

————→表示从一种状态转换到另一种状态时不调用任何回调函数。

cb_ist_Nxx_sent 中的 *N* 表示一下几个值：

3——cb_ist_3xx_sent

4——cb_ist_4xx_sent

5——cb_ist_5xx_sent

6——cb_ist_6xx_sent

4. NIST 状态机（见图 13-5）

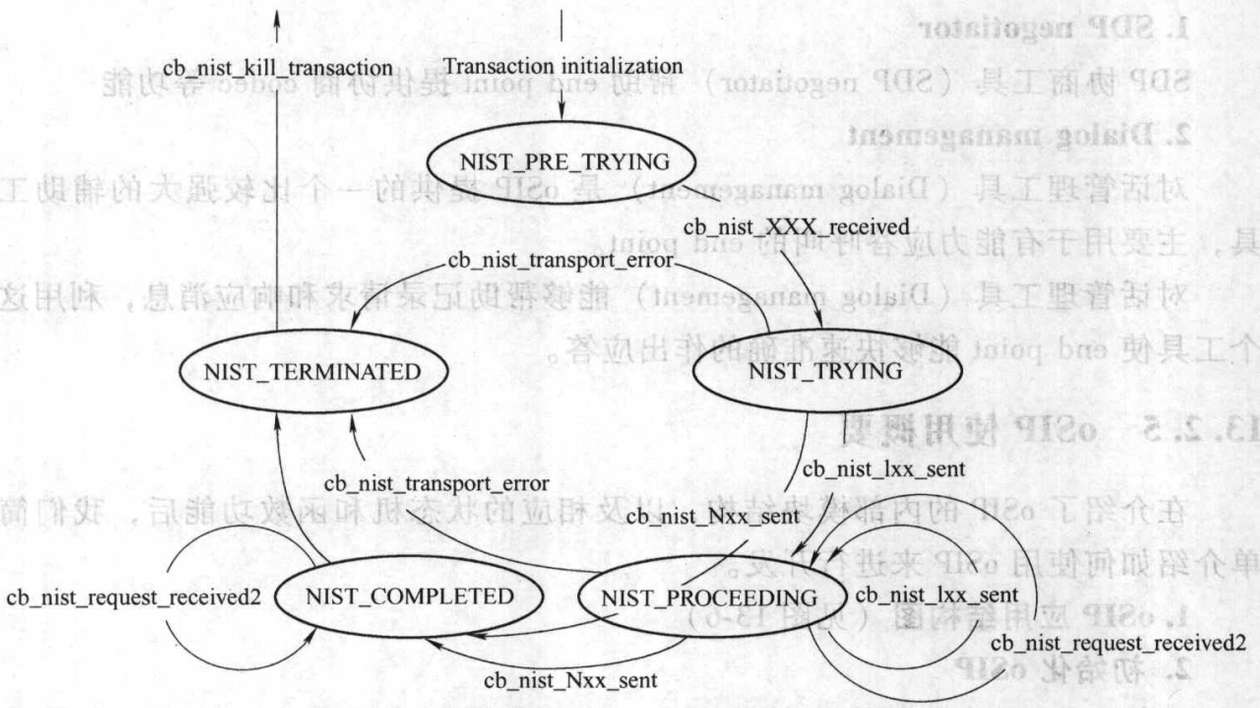


图 13-5　NIST 状态机

注：——cb_nist_xxx_xxxx ——→表示从一种状态转换到另一种状态时将调用该回调函数；

————→表示从一种状态转换到另一种状态时不调用任何回调函数。

cb_nist_XXX_ received 中的 XXX 表示一下几种消息类型：

register——cb_nist_register_received

bye——cb_nist_bye_received

options——cb_nist_options_received

info——cb_nist_info_received

cancel——cb_nist_cancel_received

notify——cb_nist_notify_received

subscribe——cb_nist_subscribe_received

unknown——cb_nist_unknown_received

cb_nist_Nxx_ sent 中的 *N* 表示一下几个值：

2——cb_nist_2xx_sent

3——cb_nist_3xx_sent

4——cb_nist_4xx_sent

5——cb_nist_5xx_sent

6——cb_nist_6xx_sent

13.2.4 工具模块

1. SDP negotiator

SDP 协商工具（SDP negotiator）帮助 end point 提供协商 codec 等功能

2. Dialog management

对话管理工具（Dialog management）是 oSIP 提供的一个比较强大的辅助工具，主要用于有能力应答呼叫的 end point。

对话管理工具（Dialog management）能够帮助记录请求和响应消息，利用这个工具使 end point 能够快速准确的作出应答。

13.2.5 oSIP 使用概要

在介绍了 oSIP 的内部模块结构，以及相应的状态机和函数功能后，我们简单介绍如何使用 oSIP 来进行开发。

1. oSIP 应用结构图（见图 13-6）

2. 初始化 oSIP

在使用 oSIP 前必须先初始化 oSIP，主要调用函数 osip_global_init（）和 osip_init（），具体操作代码如下：

```
osip_t * osip;
// initialise internal element first
if (0! = osip_global_init( ) )
  return -1;
// allocate a global osip element.
if (0! = osip_init( &osip) )
  return -1;
```

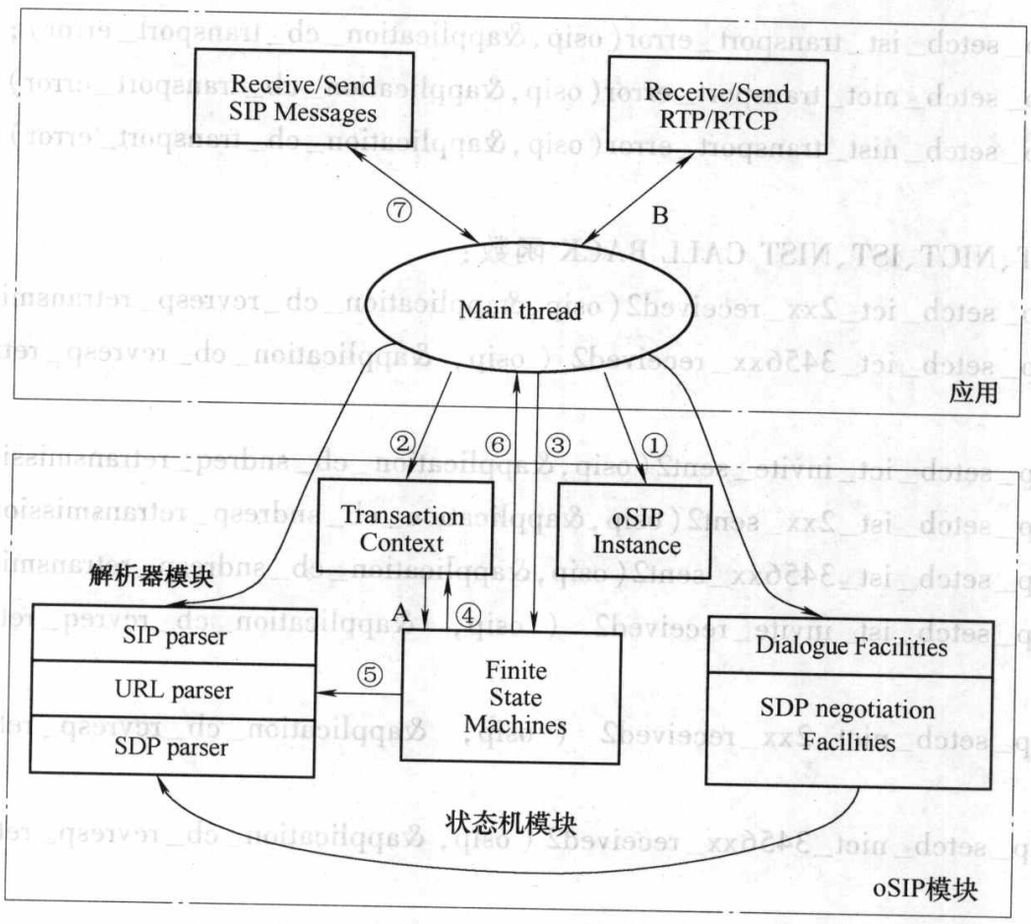


图 13-6　oSIP 应用

①—初始化 oSIP 和注册 CALL BACK 函数　②—添加事件 A　③—执行事务　④—取消事件 A
⑤—解析消息　⑥—触发 CALL BACK 函数　⑦—接收/发送消息
A—保存状态　B—接收/发送语音包

3. 注册 CALL BACK 函数

需要注册的 CALL BACK 函数主要包含发送消息、结束事务、发送失败、4 个状态机（ICT、NICT、IST、NIST）相关函数。

注册发送消息的 CALL BACK 函数：

```
osip_setcb_send_message (osip, &application_cb_snd_message);
```

注册结束事务的 CALL BACK 函数：

```
osip_setcb_ict_kill_transaction(osip,&application_cb_ict_kill_transaction);
osip_setcb_ist_kill_transaction(osip,&application_cb_ist_kill_transaction);
osip_setcb_nict_kill_transaction(osip,&application_cb_nict_kill_transaction);
osip_setcb_nist_kill_transaction(osip,&application_cb_nist_kill_transaction);
```

注册发送失败的 CALL BACK 函数：

```
osip_setcb_ict_transport_error(osip,&application_cb_transport_error);
```

```
osip_setcb_ist_transport_error(osip,&application_cb_transport_error);
osip_setcb_nict_transport_error(osip,&application_cb_transport_error);
osip_setcb_nist_transport_error(osip,&application_cb_transport_error);
```

注册 ICT、NICT、IST、NIST CALL BACK 函数:

```
osip_setcb_ict_2xx_received2(osip,&application_cb_rcvresp_retransmission);
osip_setcb_ict_3456xx_received2 (osip, &application_cb_rcvresp_retransmission);
osip_setcb_ict_invite_sent2(osip,&application_cb_sndreq_retransmission);
osip_setcb_ist_2xx_sent2(osip,&application_cb_sndresp_retransmission);
osip_setcb_ist_3456xx_sent2(osip,&application_cb_sndresp_retransmission);
osip_setcb_ist_invite_received2 (osip, &application_cb_rcvreq_retransmission);
osip_setcb_nict_2xx_received2 (osip, &application_cb_rcvresp_retransmission);
osip_setcb_nict_3456xx_received2 (osip, &application_cb_rcvresp_retransmission);
osip_setcb_nict_request_sent2(osip,&application_cb_sndreq_retransmission);
osip_setcb_nist_2xx_sent2(osip,&application_cb_sndresp_retransmission);
osip_setcb_nist_3456xx_sent2 (osip, &application_cb_sndresp_retransmission);
osip_setcb_nist_request_received2 (osip, &application_cb_rcvreq_retransmission);
osip_setcb_ict_invite_sent (osip,&application_cb_sndinvite);
osip_setcb_ict_ack_sent (osip,&application_cb_sndack);
osip_setcb_nict_register_sent(osip,&application_cb_sndregister);
osip_setcb_nict_bye_sent (osip,&application_cb_sndbye);
osip_setcb_nict_cancel_sent (osip,&application_cb_sndcancel);
osip_setcb_nict_info_sent (osip,&application_cb_sndinfo);
osip_setcb_nict_options_sent (osip,&application_cb_sndoptions);
osip_setcb_nict_subscribe_sent (osip,&application_cb_sndoptions);
osip_setcb_nict_notify_sent (osip,&application_cb_sndoptions);
osip_setcb_nict_unknown_sent(osip,&application_cb_sndunkrequest);
osip_setcb_ict_1xx_received(osip,&application_cb_rcv1xx);
osip_setcb_ict_2xx_received(osip,&application_cb_rcv2xx);
```

```
osip_setcb_ict_3xx_received(osip,&application_cb_rcv3xx);
osip_setcb_ict_4xx_received(osip,&application_cb_rcv4xx);
osip_setcb_ict_5xx_received(osip,&application_cb_rcv5xx);
osip_setcb_ict_6xx_received(osip,&application_cb_rcv6xx);
osip_setcb_ist_1xx_sent(osip,&application_cb_snd1xx);
osip_setcb_ist_2xx_sent(osip,&application_cb_snd2xx);
osip_setcb_ist_3xx_sent(osip,&application_cb_snd3xx);
osip_setcb_ist_4xx_sent(osip,&application_cb_snd4xx);
osip_setcb_ist_5xx_sent(osip,&application_cb_snd5xx);
osip_setcb_ist_6xx_sent(osip,&application_cb_snd6xx);
osip_setcb_nict_1xx_received(osip,&application_cb_rcv1xx);
osip_setcb_nict_2xx_received(osip,&application_cb_rcv2xx);
osip_setcb_nict_3xx_received(osip,&application_cb_rcv3xx);
osip_setcb_nict_4xx_received(osip,&application_cb_rcv4xx);
osip_setcb_nict_5xx_received(osip,&application_cb_rcv5xx);
osip_setcb_nict_6xx_received(osip,&application_cb_rcv6xx);
osip_setcb_nist_1xx_sent(osip,&application_cb_snd1xx);
osip_setcb_nist_2xx_sent(osip,&application_cb_snd2xx);
osip_setcb_nist_3xx_sent(osip,&application_cb_snd3xx);
osip_setcb_nist_4xx_sent(osip,&application_cb_snd4xx);
osip_setcb_nist_5xx_sent(osip,&application_cb_snd5xx);
osip_setcb_nist_6xx_sent(osip,&application_cb_snd6xx);
osip_setcb_ist_invite_received(osip,&application_cb_rcvinvite);
osip_setcb_ist_ack_received(osip,&application_cb_rcvack);
osip_setcb_ist_ack_received2 (osip,&application_cb_rcvack2);
osip_setcb_nist_register_received(osip,&application_cb_rcvregister);
osip_setcb_nist_bye_received (osip,&application_cb_rcvbye);
osip_setcb_nist_cancel_received (osip,&application_cb_rcvcancel);
osip_setcb_nist_info_received (osip,&application_cb_rcvinfo);
osip_setcb_nist_options_received (osip,&application_cb_rcvoptions);
osip_setcb_nist_subscribe_received(osip,&application_cb_rcvoptions);
osip_setcb_nist_notify_received (osip,&application_cb_rcvoptions);
osip_setcb_nist_unknown_received (osip,&application_cb_rcvunkrequest);
```

4. Transaction 操作

在注册完 CALL BACK 函数后，应用程序可以建立 Transaction 来调用 oSIP 的

解析器和状态机模块的操作，来实现不同应用程序的需求。

13.3 基于 oSIP 的 PartySIP 开发举例

PartySIP 是基于 oSIP 协议栈开发的一个包括注册服务器、SIP 代理服务器于一体的 SIP 服务器，下面对 PartySIP 进行详细介绍。

13.3.1 PartySIP 的主要数据结构

1. psp_core_t

psp_core_t 是 PartySIP 的核心数据结构，PartySIP 代理服务器状态迁移数据结构、oSIP 数据结构（它主要是跟 SIP 消息相关的数据结构，包括各种方法、应答和头域等）、传输层数据结构、代理服务器的 IP 可端口号等数据结构体。下面是 psp_core_t 的结构体：

```
typedef struct _psp_core_t
{
psp_osip_t * psp_osip;
pspm_resolv_t * resolv;
pspm_tlp_t * tlp;
pspm_sfp_t * sfp;
int iptables_dynamic_natrule;
char * ipt_server;
int ipt_port;
struct addrinfo * ipt_addr;
int masquerade_sdp;
int masquerade_via;
char * remote_natip;
char * lan_ip;
char * lan_fqdn;
char * lan_mask;
char * ext_ip;
char * ext_fqdn;
char * ext_mask;
int recovery_delay;
char * serverip[MAX_SERVER_IDENTIFIERS];
char * servername[MAX_SERVER_IDENTIFIERS];
```

```
    int   ipv6_enable;
    char * serverip6[MAX_SERVER_IDENTIFIERS];
    int port;
    struct osip_mutex * gl_lock;
    firewall_entry_t fw_entries[PSP_NAT_MAX_CALL];
    int disable_check_for_osip_to_tag_in_cancel;
    char banner[128];
} psp_core_t;
```

2. psp_osip_t

psp_osip_t 数据结构是跟用户相关的一些配置信息的结构体,具体结构如下:

```
typedef struct _psp_osip_t
{
  osip_t * osip;
  osip_fifo_t * osip_tr_terminated; /* osip transactions in the terminated state */
  int delay;          /* delay between update (should be 500ms) */
  int exit_flag;      /* quick hack to shutdown the timers thread */
  struct osip_thread * timers;     /* internal osip timer handler */
  ppl_pipe_t * wakeup;
  /* basic statistics */
  int total_ict;
  int successfull_ict;
  int total_nict;
  int successfull_nict;
  int total_ist;
  int successfull_ist;
  int total_nist;
  int successfull_nist;
} psp_osip_t;
```

3. pspm_resolv_t

pspm_resolv_t 主要是一个模块结构体,其中又包括 oSIP 的线程结构体和排队队列结构体。它主要作用是记录每个模块(如事件处理模块、传送模块等)的处理队列和处理线程。它的结构如下:

```
typedef struct _pspm_resolv_t
{
```

```
    module_t * module;
    /* no plugin */
} pspm_resolv_t;

typedef struct _module_t
{
#define MOD_THREAD 1/* start module_execute() once */
    int flag;      /* MOD_THREAD */
    struct osip_thread * thread; /* modules are initiated in a thread
(optionnaly) */
    ppl_pipe_t * wakeup;      /* descriptor used to wake up the thread
*/
    char * filename;
} module_t;
```

4. pspm_tlp_t

pspm_tlp_t 主要是 tlp_plugin_t,主要是跟模块相关的数据结构。PartySIP 是分模块设计的,这便于在后续的开发中加入其他功能。比如 PartySIP 目前只支持 UDP 传送,如果要在后续的版本加入对 TCP 的支持,就应该用到这个数据结构,可以把 TCP 作为一个插件来开发,而对整体不用修改。这样做的原因很简单,就是为了系统的扩展方便。它的结构体如下:

```
typedef struct _pspm_tlp_t
{
    module_t * module;
    tlp_plugin_t * tlp_plugins; /* list of tlp_plugins */
} pspm_tlp_t;
typedef struct tlp_plugin_t tlp_plugin_t;
    struct tlp_plugin_t
    {
        psp_plugin_t * psp_plugin;
        int plug_id;
        int in_socket;
        int out_socket;
        int mcast_socket;
        #define TLP_UDP                    0x01
```

```
#define TLP_TCP                0x02
#define TLP_RELIABLE           0x04
#define TLP_UNRELIABLE         0x08
#define TLP_STCP               0x10
int flag;      /* PLUG_{TCP|UDP|RELIABLE|UNRELIABLE}
*/
  tlp_rcv_func_t * rcv_func; /* methods to receive SIP MES-
SAGE (optionnal) */
    tlp_snd_func_t * snd_func; /* methods to send SIP
MESSAGE (mandatory) */
    tlp_plugin_t * next;
    tlp_plugin_t * parent;
};
tlp_rcv_func_t
typedef struct tlp_rcv_func_t
{int plug_id; /* plug_id of the plugin */
  int ( * cb_rcv_func) (int); /* callback */
}
tlp_rcv_func_t;
tlp_snd_func_t
typedef struct tlp_snd_func_t
{int plug_id;      /* plug_id of the plugin */
  int ( * cb_snd_func) (osip_transaction_t * , osip_message_t
* , char * , int, int);
}
```

5. pspm_sfp_t

pspm_sfp_t 结构体是跟 PartySIP 的状态机相关的，包括各种方法体（如：invite、ack、bye）、响应消息（如 1xxs、2xx、3xx、4xx 等）和需要转发消息的回调函数表，主要是在代理服务器收到不同的 SIP 消息，进行不同的处理以及对应状态的迁移。其结构体如下：

```
typedef struct _pspm_sfp_t
{module_t * module;
sfp_plugin_t * sfp_plugins;
sfp_inc_func_tab_t * rcv_invites;    /* reception of INVITE */
sfp_inc_func_tab_t * rcv_acks; /* reception of ACK */
```

```
  sfp_inc_func_tab_t * rcv_registers; /*      REGISTER */
  sfp_inc_func_tab_t * rcv_byes;/*            BYE */
  sfp_inc_func_tab_t * rcv_optionss; /*       OPTIONS */
  sfp_inc_func_tab_t * rcv_infos; /*          INFO */
  sfp_inc_func_tab_t * rcv_cancels;/*         CANCEL */
  sfp_inc_func_tab_t * rcv_notifys;/*         NOTIFY */
  sfp_inc_func_tab_t * rcv_subscribes;/*      SUBSCRIBE */
  sfp_inc_func_tab_t * rcv_unknowns; /*       UNKNOWN */
  osip_fifo_t * sip_ack;        /* osip_message_t ACK messages */
  osip_fifo_t * osip_message_traffic; /* New osip_transaction_t element. */
  sfp_fwd_func_tab_t * fwd_invites;/* forward   INVITE */
  sfp_fwd_func_tab_t * fwd_acks; /* forward     ACK */
  sfp_fwd_func_tab_t * fwd_registers;/*        REGISTER */
  sfp_fwd_func_tab_t * fwd_byes; /*           BYE */
  sfp_fwd_func_tab_t * fwd_optionss; /*       OPTIONS */
  sfp_fwd_func_tab_t * fwd_infos; /*          INFO */
  sfp_fwd_func_tab_t * fwd_cancels; /*        CANCEL */
  sfp_fwd_func_tab_t * fwd_notifys; /*        NOTIFY */
  sfp_fwd_func_tab_t * fwd_subscribes; /*     SUBSCRIBE */
  sfp_fwd_func_tab_t * fwd_unknowns; /*       UNKNOWN */
  sfp_rcv_func_tab_t * rcv_1xxs;/* receive    1xx */
  sfp_rcv_func_tab_t * rcv_2xxs; /*           2xx */
  sfp_rcv_func_tab_t * rcv_3xxs; /*           3xx */
  sfp_rcv_func_tab_t * rcv_4xxs; /*           4xx */
  sfp_rcv_func_tab_t * rcv_5xxs; /*           5xx */
  sfp_rcv_func_tab_t * rcv_6xxs; /*           6xx */
  sfp_snd_func_tab_t * snd_1xxs; /* forward    1xx */
  sfp_snd_func_tab_t * snd_2xxs; /*           2xx */
  sfp_snd_func_tab_t * snd_3xxs; /*           3xx */
  sfp_snd_func_tab_t * snd_4xxs; /*           4xx */
  sfp_snd_func_tab_t * snd_5xxs; /*           5xx */
  sfp_snd_func_tab_t * snd_6xxs; /*           6xx */
  osip_fifo_t * sfull_request; /* psp_request_t ELEMENT in STATEFULL_MODE */
```

```
osip_fifo_t * sfull_cancels; /* osip_message_t ELEMENT */
osip_list_t * broken_transactions;
/* struct osip_mutex * mut_psp_requests; */
psp_request_t * psp_requests;
} pspm_sfp_t;
```

13.3.2　PartySIP 的状态机图

我们在第 10 章介绍了 SIP 的 4 个状态机的迁移过程，这 4 个状态机是 SIP 的核心，它们分别在 oSIP 中实现了。这里的状态机是作为 PartySIP（即代理服务器）的状态机，它是建立在 SIP 的 4 个状态机之上的。PartySIP 状态机主要包括 4 个状态：PROXY_CALLING、PROXY_PRE_CALLING、PROXY_PROCEEDING、PROXY_CLOSING。图 13-7 为 PartySIP 状态机迁移图。

13.3.3　PartySIP 的初始化过程

了解 PartySIP 的初始化过程对基于 PartySIP 进行二次开发具有重要作用，了解了 PartySIP 的初始化过程可以深入理解 PartySIP 内核处理流程，对加入后续功能有指导意义。内核处理主要由 psp_core_init ( )函数来完成对 psp_core_t 结构体（见 11.3.1）的初始化，下面是内核初始化过程：

1）首先给一个内核结构体分配内存地址，然后进行初始化：

```
core = (psp_core_t * ) osip_malloc (sizeof (psp_core_t));
```

调用 psp_osip_init(psp_osip_t * * psp_osip, int delay)函数对 psp_osip_t 结构体进行初始化。其中 psp_osip_t 结构体如下：

```
typedef struct _psp_osip_t
{
  osip_t * osip;/* osip main element */
  osip_fifo_t * osip_tr_terminated; /* osip transactions in the termi-
                                       nated state */
  int delay; /* delay between update (should be 500ms) */
  int exit_flag; /* quick hack to shutdown the timers thread */
  struct osip_thread * timers; /* internal osip timer handler */
  ppl_pipe_t * wakeup;
  /* basic statistics */
  int total_ict;
  int successfull_ict;
  int total_nict;
```

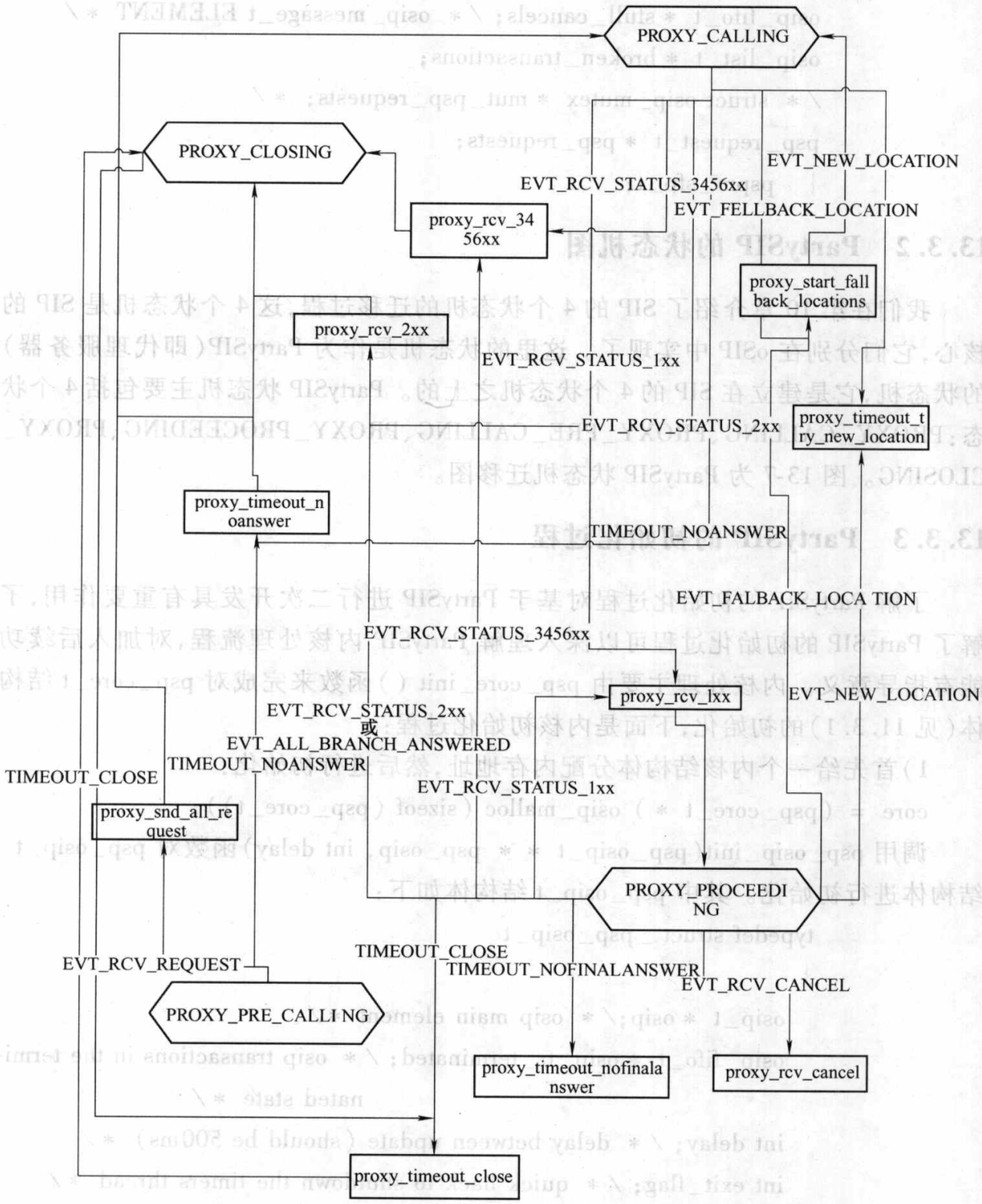


图 13-7 PartySIP 状态机转移图

```
int successfull_nict;
int total_ist;
int successfull_ist;
int total_nist;
int successfull_nist;
```

}psp_osip_t;

由 osip_init (&osip)函数对 osip 结构体初始化(见 oSIP),然后调用 osip_set_cb_send_message (osip, &psp_core_cb_snd_message)函数设置一些与 SIP 中的 4 种状态机相关的回调函数,也是对上面 osip 结构体的初始化。

//@@@为 ict,ist,nict,nist 结束事件的回调函数。

//@@ ict

osip_set_kill_transaction_callback (osip, OSIP_ICT_KILL_TRANSACTION, &psp_core_cb_ict_kill_transaction);

//@@@ ist

osip_set_kill_transaction_callback (osip, OSIP_IST_KILL_TRANSACTION, &psp_core_cb_ist_kill_transaction);

//@@@ Nict

osip_set_kill_transaction_callback (osip, OSIP_NICT_KILL_TRANSACTION, &psp_core_cb_nict_kill_transaction);

//@@@ nist

osip_set_kill_transaction_callback (osip, OSIP_NIST_KILL_TRANSACTION, &psp_cor_cb_nist_kill_transaction);

//@@@以下部分是设置 ict,ist,nict,nist 的接收、重新接收、发送和重发的回调函数。

//当然是根据不同的状态来定。

//@@@ ICT

osip_set_message_callback (osip, OSIP_ICT_STATUS_2XX_RECEIVED_AGAIN,&psp_core_cb_rcvresp_retransmission);

osip_set_message_callback (osip, OSIP_ICT_STATUS_3456XX_RECEIVED_AGAIN, psp_core_cb_rcvresp_retransmission);

osip_set_message_callback (osip, OSIP_ICT_INVITE_SENT_AGAIN, &psp_core_cb_sndreq_retransmission);

//@@@ IST

osip_set_message_callback (osip, OSIP_IST_STATUS_2XX_SENT_AGAIN, &psp_core_cb_sndresp_retransmission);

osip_set_message_callback (osip, OSIP_IST_STATUS_3456XX_SENT_AGAIN, &psp_core_cb_sndresp_retransmission);

osip_set_message_callback (osip, OSIP_IST_INVITE_RECEIVED_AGAIN, &psp_core_cb_rcvreq_retransmission);

//@@@ NICT

```
    osip_set_message_callback (osip, OSIP_NICT_STATUS_2XX_RECEIVED_
AGAIN, &psp_core_cb_rcvresp_retransmission);
    osip_set_message_callback (osip, OSIP_NICT_STATUS_3456XX_RE-
CEIVED_AGAIN,&psp_core_cb_revresp_retransmission);
    osip_set_message_callback (osip, OSIP_NICT_REQUEST_SENT_AGAIN,
&psp_core_cb_sndreq_retransmission);
    //@@@ NIST
    osip_set_message_callback (osip, OSIP_NIST_STATUS_2XX_SENT_AGAIN,
&psp_core_cb_sndresp_retransmission);
    osip_set_message_callback (osip, OSIP_NIST_STATUS_3456XX_SENT_
AGAIN, &psp_core_cb_sndresp_retransmission);
    osip_set_message_callback (osip, OSIP_NIST_REQUEST_RECEIVED_
AGAIN, &psp_core_cb_rcvreq_retransmission);
    //@@@为 ict,ist,nict,nist 传输错误事件的回调函数。
    osip_set_transport_error_callback (osip, OSIP_ICT_TRANSPORT_ERROR,
&psp_core_cb_transport_error);
    osip_set_transport_error_callback (osip, OSIP_IST_TRANSPORT_ERROR,
&psp_core_cb_transport_error);
    osip_set_transport_error_callback (osip, OSIP_NICT_TRANSPORT_ERROR,
&psp_core_cb_transport_error);
    osip_set_transport_error_callback (osip, OSIP_NIST_TRANSPORT_ERROR,
&psp_core_cb_transport_error);
    //@@@为 ict 设置发生 INVITE,ACK 的回调函数
    osip_set_message_callback (osip, OSIP_ICT_INVITE_SENT, &psp_core_
cb_sndinvite);
    osip_set_message_callback (osip, OSIP_ICT_ACK_SENT, &psp_core_cb_
sndack);
    //@@@ 为 NICT 设置发送 REGISTER,BYE,CANCEL,INFO,OPTIONS,SUB-
SCRIBE,UNKNOWN_REQUEST
    //@@@ 消息的发送回调函数
    osip_set_message_callback (osip, OSIP_NICT_REGISTER_SENT, &psp_
core_cb_sndregister);
    osip_set_message_callback (osip, OSIP_NICT_BYE_SENT, &psp_core_cb_
sndbye);
    osip_set_message_callback (osip, OSIP_NICT_CANCEL_SENT, &psp_core_
```

```
cb_sndcancel);
    osip_set_message_callback (osip, OSIP_NICT_INFO_SENT, &psp_core_cb_
sndinfo);
    osip_set_message_callback (osip, OSIP_NICT_OPTIONS_SENT, &psp_core_
cb_sndoptions);
    osip_set_message_callback (osip, OSIP_NICT_SUBSCRIBE_SENT, &psp_
core_cb_sndoptions);
    osip_set_message_callback (osip, OSIP_NICT_NOTIFY_SENT, &psp_core_
cb_sndoptions);
    /* osip_set_cb_nict_sndprack(osip,&psp_core_cb_sndprack); */
    osip_set_message_callback (osip, OSIP_NICT_UNKNOWN_REQUEST_
SENT, &psp_core_cb_sndunkrequest);
    //@@@为 ict 设置收到 1XX-6XX 消息的处理的回调函数
    osip_set_message_callback (osip, OSIP_ICT_STATUS_1XX_RECEIVED,
&psp_core_cb_rcv1xx);
    osip_set_message_callback (osip, OSIP_ICT_STATUS_2XX_RECEIVED,
&psp_core_cb_rcv2xx);
    osip_set_message_callback (osip, OSIP_ICT_STATUS_3XX_RECEIVED,
&psp_core_cb_rcv3xx);
    osip_set_message_callback (osip, OSIP_ICT_STATUS_4XX_RECEIVED,
&psp_core_cb_rcv4xx);
    osip_set_message_callback (osip, OSIP_ICT_STATUS_5XX_RECEIVED,
&psp_core_cb_rcv5xx);
    osip_set_message_callback (osip, OSIP_ICT_STATUS_6XX_RECEIVED,
&psp_core_cb_rcv6xx);
    //@@@为 iSt 设置发生 1XX-6XX 消息的处理的回调函数
    osip_set_message_callback (osip, OSIP_IST_STATUS_1XX_SENT, &psp_
core_cb_snd1xx);
    osip_set_message_callback (osip, OSIP_IST_STATUS_2XX_SENT, &psp_
core_cb_snd2xx);
    osip_set_message_callback (osip, OSIP_IST_STATUS_3XX_SENT, &psp_
core_cb_snd3xx);
    osip_set_message_callback (osip, OSIP_IST_STATUS_4XX_SENT, &psp_
core_cb_snd4xx);
    osip_set_message_callback (osip, OSIP_IST_STATUS_5XX_SENT, &psp_
```

```
core_cb_snd5xx);
    osip_set_message_callback (osip, OSIP_IST_STATUS_6XX_SENT, &psp_
core_cb_snd6xx);
    //@@@为 nict 设置收到 1XX-6XX 消息的处理的回调函数
    osip_set_message_callback (osip, OSIP_NICT_STATUS_1XX_RECEIVED,
&psp_core_cb_rcv1xx);
    osip_set_message_callback (osip, OSIP_NICT_STATUS_2XX_RECEIVED,
&psp_core_cb_rcv2xx);
    osip_set_message_callback (osip, OSIP_NICT_STATUS_3XX_RECEIVED,
&psp_core_cb_rcv3xx);
    osip_set_message_callback (osip, OSIP_NICT_STATUS_4XX_RECEIVED,
&psp_core_cb_rcv4xx);
    osip_set_message_callback (osip, OSIP_NICT_STATUS_5XX_RECEIVED,
&psp_core_cb_rcv5xx);
    osip_set_message_callback (osip, OSIP_NICT_STATUS_6XX_RECEIVED,
&psp_core_cb_rcv6xx);
    //@@@为 iSt 设置发生 1XX-6XX 消息的处理的回调函数
    osip_set_message_callback (osip, OSIP_NIST_STATUS_1XX_SENT, &psp_
core_cb_snd1xx);
    osip_set_message_callback (osip, OSIP_NIST_STATUS_2XX_SENT, &psp_
core_cb_snd2xx);
    osip_set_message_callback (osip, OSIP_NIST_STATUS_3XX_SENT, &psp_
core_cb_snd3xx);
    osip_set_message_callback (osip, OSIP_NIST_STATUS_4XX_SENT, &psp_
core_cb_snd4xx);
    osip_set_message_callback (osip, OSIP_NIST_STATUS_5XX_SENT, &psp_
core_cb_snd5xx);
    osip_set_message_callback (osip, OSIP_NIST_STATUS_6XX_SENT, &psp_
core_cb_snd6xx);
    //@@@为 iSt 设置收到 INVITE,ACK,ACK_RECEIVED_AGAIN 消息的处
理的回调函数
    osip_set_message_callback (osip, OSIP_IST_INVITE_RECEIVED, &psp_
core_cb_rcvinvite);
    osip_set_message_callback (osip, OSIP_IST_ACK_RECEIVED, &psp_core_
cb_rcvack);
```

osip_set_message_callback　(osip,　OSIP_IST_ACK_RECEIVED_AGAIN, &psp_core_cb_rcvack2);

//@@@ 为 nist 收到 REGISTER, BYE, CANCEL, INFO, OPTIONS, SUBSCRIBE, UNKNOWN_REQUEST

//@@@消息后的处理函数

osip_set_message_callback　(osip,　OSIP_NIST_REGISTER_RECEIVED, &psp_core_cb_rcvregister);

osip_set_message_callback　(osip,　OSIP_NIST_BYE_RECEIVED,　&psp_core_cb_rcvbye);

osip_set_message_callback (osip, OSIP_NIST_CANCEL_RECEIVED, &psp_core_cb_rcvcancel);

osip_set_message_callback　(osip,　OSIP_NIST_INFO_RECEIVED,　&psp_core_cb_rcvinfo);

osip_set_message_callback (osip, OSIP_NIST_OPTIONS_RECEIVED, &psp_core_cb_rcvoptions);

osip_set_message_callback　(osip,　OSIP_NIST_SUBSCRIBE_RECEIVED, &psp_core_cb_rcvoptions);

osip_set_message_callback (osip, OSIP_NIST_NOTIFY_RECEIVED, &psp_core_cb_rcvoptions);

osip_set_message_callback　(osip,　OSIP_NIST_UNKNOWN_REQUEST_RECEIVED, &psp_core_cb_rcvunkrequest);

在 osip_set_message_callback 中主要是进行下面一个赋值操作 osip->msg_callbacks[type] = cb;

(*psp_osip)->timers = osip_thread_create (20000, (void *(*)(void *)) osip_timers_thread,(void *)(*psp_osip));

主要是初始化一个线程,在 osip_thread_create 通过调用 pthread_create 创建 osip_timers_thread 线程,它是一个死循环体一旦有消息就可以触发他运行。其中的主要的部分是:

```
psp_core_lock ();
//@@@ 为 ict 设置超时事件
osip_timers_ict_execute (psp_osip->osip);
//@@@ 为 ist 设置超时事件
osip_timers_ist_execute (psp_osip->osip);
//@@@ 为 nict 设置超时事件
osip_timers_nict_execute (psp_osip->osip);
```

```
//@@@ 为 nist 设置超时事件
osip_timers_nist_execute (psp_osip->osip);

//@@@ 执行 ict 事件队列中的各个事件。
osip_ict_execute (psp_osip->osip);
//@@@ 执行 nict 事件队列中的各个事件。
osip_nict_execute (psp_osip->osip);
//@@@ 执行 ist 事件队列中的各个事件。
osip_ist_execute (psp_osip->osip);
//@@@ 执行 nist 事件队列中的各个事件。
osip_nist_execute (psp_osip->osip);
psp_core_unlock ();
if (psp_osip->exit_flag == 1)
    return NULL;
```

(详细的处理过程参见 13.3.4 osip_timers_thread 简单处理过程)

只有当 psp_osip->exit_flag == 1 为真时才停止

2)然后通过 pspm_resolv_init (&(core->resolv))函数和 module_init (&((*resolv)->module), MOD_THREAD, "resolv")函数分别对 pspm_resolv_t 和 module_t(见 10.3.1)结构体进行初始化。其中(*module)->wakeup = ppl_pipe ()很重要以后该模块在什么时候唤醒就是由它来标识。

3)再通过 pspm_tlp_init (&(core->tlp))函数和 module_init (&((*pspm)->module), MOD_THREAD, "tlp")函数分别对 pspm_tlp_t 和 module_t(见 11.3.1)结构体进行初始化。

4)pspm_sfp_init (&(core->sfp))函数主要是完成 pspm_sfp_t 结构体(见 11.3.1)的初始化,它主要是初始化 PartySIP 代理服务器的状态机。

```
(*pspm) = (pspm_sfp_t *) osip_malloc (sizeof (pspm_sfp_t));
module_init (&((*pspm)->module), MOD_THREAD, "sfp")
(*pspm)->sfull_request = (osip_fifo_t *) osip_malloc (sizeof (osip_fifo_t));
osip_fifo_init ((*pspm)->sfull_request);
//@@@ 初始化一个先进先出的请求队列
(*pspm)->sfull_cancels = (osip_fifo_t *) osip_malloc (sizeof (osip_fifo_t));
osip_fifo_init ((*pspm)->sfull_cancels);
//@@@ 初始化一个先进先出的取消队列
```

```
(*pspm)->broken_transactions = (osip_list_t *) osip_malloc (sizeof (osip_list_t))
    osip_list_init ((*pspm)->broken_transactions);
    //@@@初始化一个 broken_transactions 链表
(*pspm)->osip_message_traffic = (osip_fifo_t *) osip_malloc (sizeof (osip_fifo_t));
osip_fifo_init ((*pspm)->osip_message_traffic);
//@@@初始化一个先进先出的消息信号量的队列
(*pspm)->sip_ack = (osip_fifo_t *) osip_malloc (sizeof (osip_fifo_t));
osip_fifo_init ((*pspm)->sip_ack);
//@@@初始化一个先进先出的 ACK 队列
sfp_inc_func_tab_init (sfp_inc_func_tab_t * * tab)
//@@@ 初始化接收回调函数表
.....
```

根据对 sfp_inc_func_tab_init 传递的参数不同，初始化不同的回调函数表。分别处理了(*pspm)->rcv_invites、(*pspm)->rcv_acks、(*pspm)->rcv_registers、(*pspm)->rcv_byes、(*pspm)->rcv_optionss、(*pspm)->rcv_infos、(*pspm)->rcv_cancels、(*pspm)->rcv_notifys、(*pspm)->rcv_subscribes、(*pspm)->rcv_unknowns、(*pspm)->fwd_invites、(*pspm)->fwd_acks、(*pspm)->fwd_registers、(*pspm)->fwd_byes、(*pspm)->fwd_optionss、(*pspm)->fwd_infos、(*pspm)->fwd_cancels、(*pspm)->fwd_notifys、(*pspm)->rcv_1xxs、(*pspm)->rcv_2xxs、(*pspm)->rcv_3xxs、(*pspm)->rcv_4xxs、(*pspm)->rcv_5xxs、(*pspm)->rcv_6xxs、(*pspm)->snd_1xxs、(*pspm)->snd_2xxs、(*pspm)->snd_3xxs、(*pspm)->snd_4xxs、(*pspm)->snd_5xxs、(*pspm)->snd_6xxs、__proxy_load_fsm ();

```
//@@@ 初始化 partysip 的代理状态机
typedef struct psp_statemachine psp_statemachine_t;
struct psp_statemachine
{
  osip_list_t *transitions;
};
typedef struct _transition_t
{
  state_t state;
  type_t type;
```

```
    void (*method)(void *, void *);
    }
    transition_t;
```

主要是对上述两数据结构的操作，基本上全部是初始化一个 transition_t 结构体，然后加到 psp_statemachine_t 结构体的链中。

5) core->gl_lock = osip_mutex_init();

osip_mutex_init()系统调用 pthread_mutex_init 来生成一个互斥锁，即给核心创建一个互斥锁。

6) 除上述过程还有对 core 的其他成员变量的赋值，到此 psp_core_init()基本完成。

13.3.4 osip_timers_thread 的简单处理过程

osip_timers_thread 其中有一个死循环体，它的结束是以 psp_osip->exit_flag == 1 为条件的，如果这个条件不满足，则一直在运行。其原型为：

```
static void * osip_timers_thread (psp_osip_t * psp_osip)
```

这里先说明一下有关的数据结构：

结构体 psp_osip_t：它是 psp_core_t 结构体中的一部分，因为 psp_core_t 是全局变量，所有 psp_osip 也是全局变量，所以在内核初始化的时候已经初始化了，具体形式：

```
typedef struct _psp_osip_t
{
    osip_t *osip;                    /* osip main element */
    osip_fifo_t *osip_tr_terminated; /* osip transactions in the terminated state */
    int delay;                       /* delay between update (should be 500ms) */
    int exit_flag;                   /* quick hack to shutdown the timers thread */
    struct osip_thread *timers;      /* internal osip timer handler */
    ppl_pipe_t *wakeup;
    /* basic statistics */
    int total_ict;
    int successfull_ict;
    int total_nict;
    int successfull_nict;
    int total_ist;
    int successfull_ist;
```

```
    int total_nist;
    int successfull_nist;
} psp_osip_t;
```

线程的唤醒只要是由结构体中的 wakeup 这个变量来传递的。

程序一进入循环体就设置一个时间长度,通过下面一条语句来阻塞这段时间长度:

```
i = select (s + 1, &osip_fdset, NULL, NULL, &tv);
```

这条语句的作用是:当在 tv 时间内有信号的到了,则立即继续这行此后的操作;当 tv 时间内没有信号的到来,那么就等待 tv 时间后,再继续这行此后的操作。相当与在没有信号的时候冬眠一段时间,然后醒来继续这行此后的操作,再进入下一轮循环。

在这句后有以下几个主要的部分:

```
    psp_core_lock ();
    //@@@ 为 ict 设置超时事件
    osip_timers_ict_execute (psp_osip->osip);
    //@@@ 为 ist 设置超时事件
    osip_timers_ist_execute (psp_osip->osip);
    //@@@ 为 nict 设置超时事件
    osip_timers_nict_execute (psp_osip->osip);
    //@@@ 为 nist 设置超时事件
    osip_timers_nist_execute (psp_osip->osip);
    //@@@ 执行 ict 事件队列中的各个事件。
    osip_ict_execute (psp_osip->osip);
    //@@@ 执行 nict 事件队列中的各个事件。
    osip_nict_execute (psp_osip->osip);
    //@@@ 执行 ist 事件队列中的各个事件。
    osip_ist_execute (psp_osip->osip);
    //@@@ 执行 nist 事件队列中的各个事件。
    osip_nist_execute (psp_osip->osip);
    psp_core_unlock ();
```

下面将一一介绍:

1. osip_timers_ict_execute(psp_osip->osip)

本函数的功能就是:取出 psp_osip->osip->osip_ict_transactions 中的所有时间进行超时设置,时根据 ict 状态机来完成的,有 A,B,D 超时,设置完后又把它插入到对应的事件队列中。

2. 其他几个函数 osip_timers_ist_execute(psp_osip->osip);

osip_timers_nict_execute(psp_osip->osip);osip_timers_nist_execute (psp_osip->osip);这几个和 osip_timers_ict_execute 中的处理基本上是一样的,只是状态机和超时类型不同而已。

3. osip_ict_execute(psp_osip->osip)

psp_osip->osip 数据结构和第 1 部分中的一样,主要操作的是 psp_osip->osip->osip_ict_transactions 中的事件。由一个循环体来完成,主要由 transaction = osip_list_get (osip->osip_ict_transactions, tr),se = (osip_event_t *) osip_fifo_tryget (transaction->transactionff)和 osip_transaction_execute (transaction, se)来完成。用到两个数据结构

```
struct osip_transaction
  {
  void * your_instance;      /* * < User Defined Pointer. */
  int transactionid;      /* * < Internal Transaction Identifier. */
  osip_fifo_t * transactionff;    /* * < events must be added in this fifo */
  osip_via_t * topvia;      /* * < CALL-LEG definition (Top Via) */
  osip_from_t * from;      /* * < CALL-LEG definition (From) */
  osip_to_t * to;      /* * < CALL-LEG definition (To) */
  osip_call_id_t * callid;      /* * < CALL-LEG definition (Call-ID) */
  osip_cseq_t * cseq;      /* * < CALL-LEG definition (CSeq) */
  osip_message_t * orig_request;      /* * < Initial request */
  osip_message_t * last_response;      /* * < Last response */
  osip_message_t * ack;      /* * < ack request sent */
  state_t state;/      * * < Current state of the transaction */
  time_t birth_time;      /* * < birth date of transaction */
  time_t completed_time;      /* * < enddate of transaction */
  int in_socket;      /* * < Optional socket for incoming message */
  int out_socket;      /* * < Optional place for outgoing message */
  void * config;      /* * @ internal transaction is managed by osip_t */
  osip_fsm_type_t ctx_type;      /* * < Type of the transaction */
  osip_ict_t * ict_context;      /* * @ internal */
  osip_ist_t * ist_context;      /* * @ internal */
  osip_nict_t * nict_context;      /* * @ internal */
  osip_nist_t * nist_context;      /* * @ internal */
  };
```

```
typedef struct osip_event osip_event_t;
  struct osip_event
{
  type_t type; /* * < Event Type */
  int transactionid;    /* * < identifier of the related osip transaction */
  osip_message_t * sip;    /* * < SIP message (optional) */
};
```

其中 osip_transaction_execute (transaction, se)这个函数的主要功能是处理这个事件。函数原型:

int osip_transaction_execute (osip_transaction_t * transaction, osip_event_t * evt)

首先判断 evt 是不是已经是 kill 的事件,如果是则还回。否则继续进行。

然后根据 transaction->ctx_type 的类型得到状态机类型赋给 statemachine。

再通过 fsm_callmethod (evt->type, transaction->state, statemachine, evt, transaction),调用对应状态机的相应的回调函数处理该事件。原型:

Int fsm_callmethod (type_t type, state_t state, osip_statemachine_t * statemachine, void * sipevent, void * transaction)

本函数是找到对应的状态机的处理函数,transition 是一个 transition_t 结构体

4. osip_nict_execute(psp_osip->osip); **osip_ist_execute** (psp_osip->osip)

osip_nist_execute (psp_osip->osip) 和第 3 部分的处理过程是一样的。

13.3.5　向 PartySIP 中加入模块

PartySIP 只实现了一个代理服务器最基本的功能,由于实际应用的需要,可能要加入其他功能时(如对 TCP 的支持),那么就需要对 Party SIP 进行修改。PartySIP 设计的时候就到了功能扩展问题,新加入的功能模块可以以一个插件的方式加入到 PartySIP 中。下面给出一个加入一个 UDP 模块的过程,其他插件功能的加入可以参照下面的例子:

```
/*
The udp plugin is a GPL plugin for partysip.
Copyright (C) 2002,2003 WellX Telecom- <partysip@ wellx. com >
Copyright (C) 2002,2003 Aymeric MOIZARD - <jack@ atosc. org >

The udp plugin is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
```

```
   (at your option) any later version.

   The udp plugin is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with Foobar; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 */

#include <partysip/partysip.h>

//#include "sample.h"
/* this structure is retreived by the core application with dlsym */
psp_plugin_t PSP_PLUGIN_DECLARE_DATA sample_plugin;

/* element shared with the core application layer */
/* all xxx_plugin elements MUST be DYNAMICLY ALLOCATED
   as the core application will assume they are */
tlp_plugin_t *sample_plug;

/* this is called by the core application when the module
   is loaded (the address of this method is located in
   the structure upd_plugin->plugin_init() */
//初始化插件功能模块
   PSP_PLUGIN_DECLARE (int) plugin_init (char *name_config)
   {
   printf("I am sample plugin and now I am started! \n");
   return 0;

   }
//启动插件功能模块
PSP_PLUGIN_DECLARE (int) plugin_start ()
```

```
{
  /* OSIP_TRACE(osip_trace(__FILE__,__LINE__,OSIP_INFO3,NULL,
    "udp plugin: plugin_start()! \n")); */
    return -1;
}
//释放插件功能模块
PSP_PLUGIN_DECLARE (int) plugin_release ()
{
  OSIP_TRACE (osip_trace (__FILE__, __LINE__, OSIP_INFO3, NULL,
              "udp plugin: plugin_release()! \n"));
  return 0;
}

psp_plugin_t PSP_PLUGIN_DECLARE_DATA sample_plugin = {
  0,                    /* uninitialized */
  "Udp plugin",
  "0.5.1",
  "plugin for receiving and sending UDP message",
  PLUGIN_TLP,
  0, /* number of owners is always 0 at the begining */
  NULL, /* future place for the dso_handle */
  &plugin_init,
  &plugin_start,
  &plugin_release
};
```

附　录

附录 A　Megaco/H.248 协议基本包

以下是部分 Megaco/H.248 协议基本包，其中列举了每个包所包含的特性、事件、信号和统计。

表　A-1

包名	Generic	
包 ID	g（0x0001）	
属性		
事件	Event Name	EventID
	Cause	Cause（0x0001）
	Signal Completion	sc（0x0002）
信号		
统计		

表　A-2

包名	Base Root Package	
包 ID	root（0x0002）	
属性	Property Name	PropertyID
	MaxNrOfContexts	MaxNumberOfContexts（0x0001）
	MaxTerminationsPerContext	MaxTerminationsPerContext（0x0002）
	normalMGExecutionTime	normalMGExecutionTime（0x0003）
	normalMGCExecutionTime	normalMGCExecutionTime（0x0004）
	ProvisionalResponseTimerValue	ProvisionalResponseTimerValue（0x0005）
事件		
信号		
统计		

表　A-3

包名	Tone Generator Package	
包 ID	tonegen（0x0003）	
属性		
事件		
信号	Signal Name	SignalID
	play tone	pt（0x0001）
统计		

表 A-4

包名	Tone Detection Package	
包 ID	tonedet (0x0003)	
属性		
事件	Event Name	EventID
	Start tone detected	std (0x0001)
	End tone detected	etd (0x0002)
	Long tone detected	ltd (0x0003)
信号		
统计		

表 A-5

包名	Basic DTMF Generator Package (Extends tonegen version 1)	
包 ID	dg (0x0005)	
属性		
事件		
信号	Signal Name	SignalID/tone id
	dtmf character 0	d0 (0x0010)
	○○○○○○	○○○○○○○
	dtmf character 9	d9 (0x0019)
	dtmf character *	ds (0x0020)
	dtmf character #	do (0x0021)
	dtmf character A	da (0x001a)
	dtmf character B	db (0x001b)
	dtmf character C	dc (0x001c)
	dtmf character D	dd (0x001d)
统计		

表 A-6

包名	DTMF detection Package (Extends tonedet version 1)	
包 ID	dd (0x0006)	
属性		
事件	Event Name	EventID
	DTMF digits	与上表中的 SignalID 的名字相同
	Digit Completion Event	ce (0x0001)
信号		
统计		

表 A-7

包名	Call Progress Tones Generator Package（Extends tonegen version 1）	
包 ID	cg（0x0007）	
属性		
事件		
信号 具体含义看 E. 180/Q. 35	Signal Name	SignalID/tone id
	Dial Tone	dt（0x0030）
	Ringing Tone	rt（0x0031）
	Busy Tone	bt（0x0032）
	Congestion Tone	ct（0x0033）
	Special Information Tone	sit（0x0034）
	Warning Tone	wt（0x0035）
	Payphone Recognition Tone	pt（0x0036）
	Call Waiting Tone	cw（0x0037）
	Caller Waiting Tone	cr（0x0038）
统计		

表 A-8

包名	Call Process Tones Detection Package（Extends tonedet version 1）
包 ID	cd（0x008）
属性	
事件	与 cg 包中定义的语音相同
信号	
统计	

表 A-9

包名	Analog Line Supervision Package	
包 ID	al（0x0009）	
属性		
事件	Event Name	EventID
	onhook	on（0x0004）
	offhook	of（0x0005）
	flashhook	fl（0x0006）
信号	Signal Name	SignalID
	ring	ri（0x0002）
统计		

表 A-10

包名	Basic Continuity Package	
包 ID	ct (0x000a)	
属性		
事件	Event Name	EventID
	Completion	cmp (0x0005)
信号	Signal Name	SignalID
	Continuity test	ct (0x0003)
	Respond	rsp (0x0004)
统计		

表 A-11

包名	Network Package	
包 ID	nt (0x000b)	
属性	Property Name	PropertyID
	Manimum Jitter Buffer	jit (0x0007)
事件	Event Name	EventID
	network failure	netfail (0x0005)
	quality alert	qualert (0x0006)
信号		
统计	Statistics Name	StatisticsID
	Duration	dur (0x0001)
	Octects Sent	os (0x0002)
	Octects Receive	or (0x0003)

表 A-12

包名	RTP Package (Extends Network Package version 1)	
包 ID	rtp (0x000c)	
属性		
事件	Event Name	EventID
	Payload Transition	pltrans (0x0001)
信号		
统计	Statistics Name	StatisticsID
	Packets Sent	ps (0x0004)
	Packets Received	pr (0x0005)
	Packet Loss	pl (0x0006)
	Jitter	jit (0x0007)
	Delay	delay (0x0008)

表 A-13

包名	TDM Circuit Package（Extends Network Package version 1）	
包 ID	tdmc（0x000d）	
属性	Property Name	PropertyID
	Echo Cancellation	ec（0x0008）
	Gain Control	gain（0x000a）
事件		
信号		
统计		

附录B 缩 略 语

缩写	全 称	中 文
3GPP	3rd Generation Partnership Project	第三代合作伙伴计划
3GPP2	3rd Generation Partnership Project 2	第三代合作伙伴计划 2
AAA	Authentication、Authorization and Accounting	用户身份认证、授权和计费协议
ACELP	Algebraic Code-Excited Linear Prencdiction	代数码激励线性预测
ACF	Admission Confirm	接入确认
ACK	ACKnowledgement message	认可消息
ACM	Address Complete Message	地址全消息
ADPCM	Adaptive Differential ulse Code Modulation	自适应差分脉冲编码调制
ADSL	Asymetric Digital Subscriber Line	非对称数字用户线
AEC	Acoustic Echo Chancellor	回声消除器
AEG	Application Expert Group	应用专家组
ANC	Answer Signal Charge	应答信号，计费
API	Application Program Interface	应用编程接口
APON	ATM PON	ATM 无源光纤网络
ARJ	Admission Reject	接入拒绝
ARM	Admin Runtime Module	管理运行模块
ARPANET	Advanced Research Projects Agency Network	ARPA 网
ARQ	Admission Request	接入请求
ATM	Asynchronous Transfer Mode	异步转移模式
AVT	Audio/Video Transport	音频视频传输
BGCF	Breakout Gateway Control Function	出口网关控制功能

（续）

缩写	全 称	中 文
BICC	Bearer Independent Call Control	与承载无关的呼叫控制
BOSS	Business Operations Support System	业务运营支撑系统
BSS	Business Support System	业务支撑系统
BSSAP	Base Station Subsystem Application Part	基站子系统应用部分
CAP	CAMEL Application Part	CAMEL 应用部分
CCF	Call Control Function	呼叫控制功能
CDR	Call Detail Record	呼叫详细记录
CGI	Common Gateway Interface	通用网关接口
CIC	Circuit Identification Code	电路识别码
CIF	Common Intermediate Format	通用中间格式
CMIP	Common Management Information Protocol	公共管理信息协议
CMIS	Common Management Information Service	公共管理信息服务
CMISE	Common Management Information Service Element	公共管理信息服务元素
COPS	Common Open Policy Service	公共开放策略服务
CORBA	Common Object Request Broker Architecture	公共对象请求代理体系结构
C/S	Client/Server	客户端/服务器端
CS	Circuit-Switched	电路交换
CSCF	Call Session Control Funtion	呼叫会话控制功能
CSMA/CD	Carrier Sense Multiple Access with Collision Detect	载波侦听多路访问/冲突检测技术
CWM	Common Warehouse Meta Model	公共仓库元模型
DCE	Distributed Computing Environment	分布式计算环境
DCF	Disengage Confirm	退出确认
DoS	Denial of Service	拒绝服务
DP	Detection Point	监测点
DPC	Destination Point Code	目的地信令点编码
DRJ	Disengage Reject	退出拒绝
DRQ	Disengage Request	退出请求
DSI	Dynamic Skeleton Interface	动态骨架接口
DSVD	Digital Simultan Eous Voice and Data	数字语音数据同步传送
DTMF	Dual Tone Multi-Frequency	双音多频
EPON	Ethernet Passive Optical Network	以太网无源光网
ETOM	Enhanced Telecom Operations Map	增强型电信运营图

（续）

缩写	全　称	中　文
ETSI	European Telecommunication Standards Institute	欧洲电信标准化协会
FTAM	File Transfer Access and Management	文件传输访问和管理
FTP	File Transfer Protocol	文件传输协议
FTTH	Fiber To The Home	光纤入户
GCCS	General Call Control Service	普通呼叫控制服务
GGSN	Gateway GPRS Support Node	GPRS 网关支持节点
GK	Gate Keeper	网守
GPON	Gigabit PON	吉比特无源光纤网络
GSTN	General Switched Telephone Network	普通电话交换网
GT	Global Title	全局码
HEMS	High -level Entity Management System	高层实体管理系统
HMP	Host Monitoring Protocol	主机监控协议
HSS	Home Subscriber Server	归属签约用户服务器
IAB	Internet Architecture Board	因特网架构委员会
IANA	Internet Assigned Numbers Authority	因特网分配号码管理机构
ICP	Internet Content Providers	网络内容提供商
ICV	Integerity Check Value	集成校验值
I-CSCF	Interrogating CSCF	问询 CSCF
IETF	Internet Engineering Task Force	互联网工程任务组
IMTC	International Multimedia Teleconferencing Consortium	国际多媒体电信会议协会
IIOP	Internet Inter-ORB Protocol	网络 ORB 交换协议
IM	Instant Message	实时消息
IMS	IP Mutimedia Subsystem	IP 多媒体子系统
IM-SSF	IP Mutimedia Service Switching Function	IP 多媒体服务交换功能
INAP	Intelligent Network Application Part	智能网应用部分
IPDC	IP Device Control	IP 设备控制
IPSec	Internet Protocol Security	因特网协议安全
ISC	International Softswitch Consortium	国际软交换协会
ISO	International Organization for Standardization	国际标准组织
ISP	Internet Service providers	网络服务提供商
ISUP	ISDN User Part	ISDN 用户部分
ISV	Independent Software Vendor	独立软件供应商

（续）

缩写	全　称	中　文
ITU	International Telecommunication Union	国际电信联盟
IVR	Interactive Voice Response	自动语音应答
JCAT	Java Coordination and Transactions	Java 调度及事务
JCC	Java Call Control	Java 呼叫控制
JCP	Java Communlty Process	Java 标准制定组织
JDMK	Java Dynamic Management Kit	Java 动态管理工具集
JMIBC	Java-based Management Information Base	Java 为基础的管理信息库
JMX	Java Management Extensions	Java 管理扩展
JVM	Java Virtual Machine	Java 虚拟机
LD-CELP	Low-Delay Code-Excited Linear Prencdiction	低延迟激励线性预测
LPC	Linear Predicative Coding	线性预测编码
LSSU	Link Status Singnal Unit	链路状态信号单元
MAP	Mobile Application Part	移动应用部分
MC	Multipoint Controller	多点控制器
MCU	Multipoint Control Unit	多点控制单元
MDCP	Media Device Control Protocol	媒体设备控制协议
MG	Media Gateway	媒体网关
MGC	Media Gateway Controller	媒体网关控制器
MGCP	Media Gateway Control Protocol	媒体网关控制协议
MGW	Media gateway Function	媒体网关功能
MIB	Manage Information Base	管理信息库
M2PA	MTP2-User Peer-to-Peer AdaptationLayer	MTP2 用户对等层间适配层
M3UA	MTP3-User Adaptation Layer	MTP3 用户适配层
MP	Multipoint Processor	多点处理器
MPLS	Multiple Protocol Label Switch	多协议标签交换
MP-MLQ	MultiPulse Maximum Likelihood Quantization	多脉冲最大似能量化
MRFC	Multimedia Resource Function Controller	多媒体资源功能控制器
MTP	Multimedia Trasmision Protocol	多媒体传送协议
NAS	Network Access Servers	网络接入服务
NAT	Network Address Translation	网络地址转换
NE	Network Element	网元
NGN	Next Generation Network	下一代网络

（续）

缩写	全　称	中　文
NSC	Network Service Certer	网络服务中心
NSP	Network Service Part	网络服务部分
NTP	Network Time Protocol	网络时间协议
OID	Object Identifiers	对象标识
OMG	Object Management Group	对象管理组织
OMG IDL	OMG Interface Definition Language	OMG 接口描述语言
OPC	Originating Point Code	源信令点编码
ORB	Object Request Braker	对象请求代理
OSA	Open Services Architecture	开放服务体系
OSA-W	Open Services Access-Gateway	开放服务接入网关
OSS	Operations Support System	运营支撑系统
P-CSCF	Proxy CSCF	代理 CSCF
PBN	Packet Based Networks	分组网络
PCM	Pulse Code Modulation	脉冲编码调制
PDF	Policy Decision Function	策略判决功能
PDR	Package Detail Record	包详细记录
PEG	Protocol Expert Group	协议专家组
POTS	Plain Old Telephone Service	普通老式电话业务
PS	Packet-Switched	分组交换
PSTN	Public Switched Telephone Network	公用电话交换网
QCIF	Quarter Common Intermediate Format	四分之一公共中间格式
QoS	Quality of Service	服务质量
RADIUS	Remote authentication dial-in user service	远程鉴别拨入用户服务
RAS	Registration, Admissin and Status	注册、接入和状态
RCF	Registration Confirm	注册确认
REL	Release	释放消息
RLC	Release Complete	释放完成
RMI	Remote Method Invocation	远程方法调用
RMON	Remote Monitoring	远程监控
RR	Receiver Report	接收者报告
RRJ	Registration Reject	拒绝注册
RRQ	Registration Request	注册请求

（续）

缩写	全 称	中 文
RTSP	Real-Time Streaming Protocol	实时流协议
RTP	Real-time Transport Protocol	实时传输协议
S-CSCF	Serving CSCF	服务 CSCF
S/MIME	Secure/Multipurpose Internet Mail Extensions	安全多用途因特网邮件扩展
SCCP	Signaling Connection Control Part	信令连接控制部分
SCE	Service Creation Environment	业务生成环境
SCF	Service Control Function	业务控制功能
SCN	Switched Circuit Network	电路交换网
SCP	Service Control Point	业务控制点
SCTP	Stream Control Transmission Protocol	流控制传输协议
SDF	Service Data Function	业务数据功能
SDH	Synchronous Digital Hierarchy	同步数字体系
SDP	Session Description Protocol	会话描述协议
SG	Signalling Gateway	信令网关
SGCP	Simple Gateway Control Protocol	简单网关控制协议
SGMP	Simple Gateway Monitoring Protocol	简单网关监控协议
SGP	Signaling Gateway Process	信令网关进程
SGSN	Serving GPRS Support Node	服务 GPRS 支持节点
SIGTRAN	Signaling Transport	信令传输
SIO	Service Information	业务信息字段
SIP	Session Initiation Protocol	会话起始协议
SLA	Service Level Agreement	服务级别协议
SLF	Subscriber Location Function	签约用户位置功能
SLEE	Service Logic Execution Environment	业务逻辑执行环境
SMI	Structure of Management Information	管理信息结构
SMP	Simple Management Protocol	简单管理协议
SNMP	Simple Network Management Protocol	简单网络管理协议
SP	Signalling Point	信令点
SPC	Signalling Point Code	信令点编码
SR	Sender Report	发送者报告
SRF	Specialized Resource Function	专用资源功能
SUERM	Signal Unit Error Rate Monitor	信号单元错误率监视器

（续）

缩写	全　称	中　文
SS7	Signalling System No. 7	No. 7 信令系统
SSF	Service Switching Function	业务交换功能
SSN	Stream Sequence Number	流序列号
SSP	Service Switch Point	业务交换点
STP	Signalling Transfer Point	信令传输点
SUA	Signalling Connection Control Part User Adaptation Layer	信令连接控制部分用户消息适配层
TACACS	Terminal Access Controller Access Control System	终端访问控制器访问控制系统
TAPA	Trillium Advanced Portability Architecture	Trillium 高级可移植结构
TCAP	Transaction Capabilities Application Part	事务处理应用部分
TCP	Transmission Control Protocol	传输控制协议
TDM	Time Division Multiplexing	时分多路复用
TLS	Transport Layer Security	传输层安全
TOM	Telecom Operations Map	电信运营图
UA	User Agent	用户代理
UAC	User Agent Client	用户代理客户机
UAS	User Agent Server	用户代理服务器
UCF	Unregistration Confirm	注销确认
UDP	User Datagram Protocol	用户数据报协议
URJ	Unregistration Reject	注销拒绝
URQ	Unregistration Request	注销请求
UML	United Modeling Language	统一建模语言
UMTS	Universal Mobile Telecommunication System	全球移动通信系统
URI	Uniform Resource Identifier	统一资源标识
US	User Server	用户服务器
VDSL	Very-High-Bit-Rate Digital Subscriber Line	高速数字用户线
VoIP	Voice over IP	在 IP 网络上的语音传送
VQ	Vector Quantization	矢量量化
WBM	Web-Based Management	基于 Web 的管理
WLAN	Wireless LAN	无线局域网

参考文献

行业标准：

[1] 中华人民共和国信息产业部．YD/T1176—2002 公用电信网计费的基本技术要求［S］．

[2] 中华人民共和国信息产业部．YD/T1434—2006 软交换设备总体技术要求［S］．

[3] 中华人民共和国信息产业部．YD/T5987—2004 智能网设备 SSP［S］．

[4] 中华人民共和国通信行业标准．YD/T 1127—2001 No.7 信令与 IP 互通的技术要求［S］．

[5] 中华人民共和国邮电部．YDN 035—1997 国内 No.7 信令方式技术规范——综合业务数字网用户部分（ISUP）［S］，1997.

[6] 中华人民共和国信息产业部．YD/T 1481—2006 会话初始协议服务器设备技术要求［S］．

著作：

[7] 徐培文，王鹰，等．软交换及其管理技术［M］．北京：机械工业出版社，2006.

[8] 石昌林，丁炜，等．MPLS 宽带网络互连技术［M］．北京：人民邮电出版社，2001.

[9] OMG，CORBA 系统结构、原理与规范［M］．韦乐平，编译．北京：电子工业出版社，2000.

[10] 赵学军，陆立，林俐，等．软交换技术与应用［M］．北京：人民邮电出版社，2004.

[11] 赵慧玲，叶华，等．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.

[12] 糜正琨，王文鼐．软交换技术与协议［M］．北京：人民邮电出版社，2002.

[13] 孟洛明，等．现代网络管理技术［M］．北京：北京邮电大学出版社，1999.

[14] 郭军．网络管理［M］．北京：北京邮电大学出版社，2001.

[15] Heinz-Gerd Hegering，Sebastian Abeck，Bernhard Neumair. 网络系统的集成管理：概念、体系及其应用［M］．曹阳，尹建华，凌军，等译．北京：清华大学出版社，2001.

[16] 杨家海，任宪坤，王沛瑜，等．网络管理—原理与实现技术［M］．北京：清华大学出版社，2000.

[17] 陈建亚，余浩．软交换与下一代网络［M］．北京：北京邮电大学出版社，2003.

[18] 杨家海，任宪坤，王沛瑜．网络管理原理与实现技术［M］．北京：清华大学出版社，1998.

[19] 岑贤道，安常青．网络管理协议及应用开发［M］．北京：清华大学出版社，1998.

[20] 糜正琨．IP 网络电话技术［M］．北京：人民邮电出版社，2000.

[21] 杨放春，廖建新．智能化现代通信网［M］．北京：北京邮电大学出版社，1999.

[22] Alan B Johnston. SIP：Understanding the Session Initiation Protocol［M］．2nd. Artech House，2004.

[23] 谢希仁．计算机网络［M］．4 版．北京：电子工业出版社，2005.

[24] Miikka Poikselkă，Georg Mayer，Hisham Khartabil，et al. The 3G IP Multimedia Subsystem（IMS）：Merging the Internet and the Cellular Worlds［M］．John Wiley & Sons，2004.

[25] Miikka Poikselka，Georg Mayer et al. The IMS IP Multimedia Concepts and Services in the

Mobile Domain [M]. John Wiley Sons, 2004.

IETF RFC:

[26] J Mogul, S Deering. RFC1191 Path MTU Discovery. 1990.

[27] R Braden, D Clark. RFC 1633 Integrated Services in the Internet Architecture: An Overview. 1994.

[28] H Schulzrinne, S Casner. RFC 1889 RTP: A Transport Protocol for Real-Time Applications. 1996.

[29] H Schulzrinne. RFC 1890 RTP Profile for Audio and Video Conferences with Minimal Control. 1996.

[30] R Braden, Ed, L Zhang. RFC 2205 Resource ReSerVation Protocol (RSVP) -Version 1 Functional Specification. 1997.

[31] S Kent, R Atkinson. RFC2402 IP Authentication Header. 1998.

[32] S Kent, R Atkinson. RFC2406 IP Encapsulating Security Payload (ESP). 2002.

[33] S Blake, D Black. RFC2475 An Architecture for Differentiated Services. 1998.

[34] M Arango, A Dugan. RFC 2705 Media Gateway Control Protocol (MGCP) Version 1.0. 1999.

[35] L Ong, I Rytina. RFC2719 Framework Architecture for Signaling Transport. 1999.

[36] N Greene, M Ramalho. RFC2805 Media Gateway Control Protocol Architecture and Requirements. 2000.

[37] C Rigney, S Willens. RFC2865 - Remote Authentication Dial In User Service (RADIUS). 2000.

[38] C Rigney. RFC2866 RADIUS Accounting. 2000.

[39] R Stewart, Q Xie. RFC2960 Stream Control Transmission Protocol. 2000.

[40] F Cuervo, N Greene. RFC3015 Megaco Protocol Version 1.0. 2000.

[41] P Srisuresh, K Egevang. RFC 3022. Traditional IP Network Address Translator (Traditional NAT). 2001.

[42] B Carpenter, S Brim. RFC3234 Middleboxes: Taxonomy and Issues. 2002.

[43] J Stone, R Stewart. RFC3309 Stream Control Transmission Protocol (SCTP) Checksum Change. 2002.

[44] K Morneault, R Dantu. RFC3331 Signaling System 7 (SS7) Message Transfer Part 2 (MTP2) - User Adaptation Layer. 2002.

[45] G Sidebottom, K Morneault. RFC3332 Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) -User Adaptation Layer (M3UA). 2002.

[46] A Jungmaier, E Rescorla. RFC3436 Transport Layer Security over Stream Control Transmission Protocol. 2002.

[47] J Rosenberg, J Weinberger. RFC3489 STUN-Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). 2003.

[48] P Calhoun, J Loughney. RFC3588 Diameter Base Protocol. 2003.

[49] Postel J. RFC0791 Internet Protocol (IP). STD 5, September 1981.

[50] Postel J. RFC0793 Transmission Control Protocol (TCP). STD 7, September 1981.

[51] Dierks T, Allen C, Treese W, et al. RFC2246 The TLS Protocol Version 1.0 (TLS). January 1999.

[52] Handley M, V Jacobson. RFC2327 Session Description Protocol (SDP). April 1998.

[53] Berners-Lee T, Fielding, R, Irvine U C, et al. RFC2396 Uniform Resource Identifiers (URI): Generic Syntax. August 1998.

[54] Kent S R Atkinson. RFC2401 Security Architecture for the Internet Protocol. November 1998.

[55] Handley M, Schulzrinne H, Schooler E, et al. RFC2543 SIP: Session Initiation Protocol. March 1999.

[56] Fielding R, Gettys J, Mogul, J, et al. RFC2616 Hypertext Transfer Protocol-HTTP/1.1. 1999.

[57] Ramsdell B. RFC2633 S/MIME Version 3 Message Specification. June 1999.

[58] Vaha-Sipila A. RFC2806 URLs for Telephone Calls. April 2000.

[59] Schulzrinne H, S Petrack. RFC2833 RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals. May 2000.

[60] Beadles M, D Mitton. RFC3169Criteria for Evaluating Network Access Server Protocols. September 2001.

[61] Rosenberg J, Schulzrinne H, Camarillo G, et al. RFC3261 SIP: Session Initiation Protocol. June 2002.

[62] Rosenberg J, H Schulzrinne. RFC3262 Reliability of Provisional Responses in the Session Initiation Protocol (SIP). June 2002.

[63] Rosenberg J, H Schulzrinne. RFC3263 SIP: Locating SIP Servers. June 2002.

[64] Rosenberg J, H Schulzrinne. RFC3264 An Offer/Answer Model with SDP. June 2002.

[65] A B Roach. RFC3265 Session Initiation Protocol (SIP) -specific Event Notification. June 2002.

[66] Niemi A, Arkko, et al. RFC3310 Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA). September 2002.

[67] Rosenberg J. RFC3311 The Session Initiation Protocol (SIP) UPDATE Method. October 2002.

[68] Camarillo G, Marshall W, Rosenberg J. RFC3312 Integration of Resource Management and Session Initiation Protocol. October 2002.

[69] Marshal W. RFC3313 Private Session Initiation Protocol (SIP) Extensions for Media Authorization. January 2003.

[70] Price R, Bormann C, Christoffersson J, et al RFC3320 Signaling Compression (SigComp). January 2003.

[71] Peterson J, RFC3323 A Privacy Mechanism for the Session Initiation Protocol (SIP). November 2002.

[72] Jenning C, Peterson J, M Watson. RFC3325 Private Extensions to the Session Initiation Pro-

tocol (SIP) for Asserted Identity within Trusted Networks. November 2002.

[73] Willis D, B Hoeneisen. RFC3327 Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts. December 2002.

[74] Arkko J, Torvinen V, Camarillo G, et al. RFC3329 Security Mechanism Agreement for the Session Initiation Protocol (SIP). January 2003.

[75] Garcia-Martin M, Bormann C, Ott J, et al. RFC3485 The Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Static Dictionary for Signaling Compression (SigComp). February 2003.

[76] G Camarillo. RFC3486 Compressing the Session Initiation Protocol (SIP). February 2003.

[77] R Sparks. RFC3515 The Session Initiation Protocol (SIP) REFER Method. April 2003.

[78] Hamer L-N, Gage B, Kosinski B, et al. RFC3520 Session Authorization Policy Element. April 2003.

[79] Camarillo G, A Monrad. RFC3524 Mapping of Media Streams to Resource Reservation Flows. April 2003.

[80] Schulzrinne H, Casner S, Frederick R, et al. RFC3550 RTP: Transport Protocol for Real-Time Applications. July 2003.

[81] Schulzrinne H, S Casner. RFC3551 RTP Profile for Audio and Video Conferences with Minimal Control. July 2003.

[82] Willis D, B Hoeneisen. RFC3608 Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration. December 2002.

国际电信联盟：

[83] ITU-T Recommendations H. 323, Visual telephone system and equipments for local Area networks which provides an non-guaranteed quality of service.

[84] ITU-T H. 323 Packet-based multimedia communications systems. 2000.

ITU-T Recommendations H. 245, Control protocol for multimedia communication.

ITU-T Recommendations H. 225, Call signalling protocols and media stream packetization for packet-based multimedia communication systems.

[85] ITU-T M. 3100, Principles for a telecommunication management network.

电信论坛：

[86] Telemanagement Forum, enhanced Telecom Operations Map TM (e TOM): The Business Process Framework for the Information and Communications Services Industry (V3. 0), GB921.

[87] Telemanagement Forum, Telecom Operations Map, GB910, Evaluation Version 2. 1.

[88] TeleManagement Forum, NGOSS Development and Integration methodology, TR127.

[89] TeleManagement Forum, New Generation Gerational Support Systems (NGOSS) Architecture Overview, GB920, Version 1. 5.

[90] TeleManagement Forum, SLA Management Handbook, GB917, Public Evaluation/Version 1. 5, June 2001.

[91] http: //www. citiy. com/mainfrm/forumdisplay. php? fid = 8 [OL]. 汇思通讯技术论坛.

[92] http://www.sinosip.com/phpBB2/［OL］. SinoSIP 中国思拓 SIP 通信技术网.

论坛及网站：

[93] http://www.sun.com/.

[94] http://www.parlay.org/.

[95] http://java.sun.com/products/jai.

[96] http://www.pudn.com/downloads34/sourcecode/multimedia/ip_phone/detail107436.html.

[97] http://17519628.pudn.com/program/program-65_100.htm.

[98] www.opal-furniture.com.

[99] http://www.1shang.com/tech/tech9/techList26840.htm.

[100] http://www.opal-furniture.com/.

[101] http://cvs.sourceforge.net/viewcvs.py/openh323/opal.

[102] www.vovida.org/pipermail/vocal/.

[103] http://scm.sipfoundry.org/viewsvn/resiprocate/main/sip/.

[104] http://tag.csdn.net/Article/dbd0da39-5453-4318-a0cc-fed68e085251.html.

[105] http://osip.atosc.org.

[106] http://www.partysip.org.

[107] http://www.nongnu.org/partysip/.

[108] www.sipforum.org/.

Document generated by Anna’s Archive around 2023-2024 as part of the DuXiu collection (https://annas-blog.org/duxiu-exclusive.html).

Images have been losslessly embedded. Information about the original file can be found in PDF attachments. Some stats (more in the PDF attachments):

{
  "filename": "6L2v5Lqk5o2i5LiOU0lQ5a6e55So5oqA5pyvXzEyMDAwMDM5LnppcA==",
  "filename_decoded": "\u8f6f\u4ea4\u6362\u4e0eSIP\u5b9e\u7528\u6280\u672f_12000039.zip",
  "filesize": 142545713,
  "md5": "c9200bec68ad7c2def103750056613dd",
  "header_md5": "4dac45401ee0a564379eae4e128b1c65",
  "sha1": "fc40b3718ecc022822d2a4214e337ab109810d35",
  "sha256": "eee2c40cf91231d6f838a9e6f44cc635394ddff31801b38a6344de45fad0126c",
  "crc32": 1390727565,
  "zip_password": "",
  "uncompressed_size": 149586565,
  "pdg_dir_name":
"\u255a\u03c6\u255c\u2557\u2557\u2557\u2559\u03b4SIP\u2569\u2561\u2559\u251c\u255d\u255d\u2569\u2321_12000039",
  "pdg_main_pages_found": 391,
  "pdg_main_pages_max": 391,
  "total_pages": 405,
  "total_pixels": 2149403040,
  "pdf_generation_missing_pages": false
}