IT先锋系列丛书

SIP揭密

SIP DEMYSTIFIED

Gonzalo Camarillo 著

白建军 彭晖 田敏 等译

Mc Graw Hill Education

人民邮电出版社
POSTS & TELECOM PRESS

IT先锋系列丛书

SIP揭密

SIP DEMYSTIFIED

McGraw-Hill
全球智慧中文化
http://www.mheducation.com

ISBN 7-115-11038-7

9 787115 110381 >

人民邮电出版社 www.ptpress.com.cn

TN

Mc Graw Hill

ISBN7-115-11038-7/TN·2007
定价:22.00 元

IT 先锋系列丛书

SIP 揭密

Gonzalo Camarillo 著

白建军 彭 晖 田 敏等 译

人民邮电出版社

图书在版编目(CIP)数据

SIP 揭密/（美）卡姆阿洛（Camarillo,G.）著；白建军等译. —北京：人民邮电出版社，2003.6
（IT 先锋系列丛书）
ISBN 7-115-11038-7

I. S… II. ①卡…②白… III. 移动通信—通信协议 IV. TN915.04

中国版本图书馆 CIP 数据核字(2003)第 028861 号

IT 先锋系列丛书

SIP 揭密

◆ 著 Gonzalo Camarillo
译 白建军 彭 晖 田 敏 等
责任编辑 梁 凝

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www. ptpress.com.cn
读者热线 010-67129258
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：800×1000 1/16
印张：12.5
字数：224 千字 2003 年 6 月第 1 版
印数：1-4 000 册 2003 年 6 月北京第 1 次印刷

著作权合同登记 图字：01-2002-2449 号

ISBN 7-115-11038-7/TN • 2007

定价：22.00 元

版 权 声 明

Gonzalo Camarillo
SIP Demystified
ISBN:0-07-137-340-3

内 容 提 要

SIP 是能够在 3G 系统中传输 IP 多媒体业务的信令协议。本书在 SIP 创始人设定的背景下对 SIP 给予了详细介绍，同时解释了怎样才能把它当作一种有创新能力的工具用于电信业务。本书由 IETF 最早的 SIP 发起人之一撰写，深入解释了当今人们谈论最多的关于 SIP 协议是什么以及起草它的标准的原因，评价了 SIP 究竟能够做什么以及传递什么，评估了 SIP 同其他标准和系统的兼容度，设计了新的支持 SIP 的业务。

本书主要读者对象为电信工程师、系统开发人员及 SIP 业务提供商。

献　　辞

谨以本书献给我的家人，他们给了我一贯的支持和鼓励，并使我受到当今最好的教育。

致　谢

在此对在 Ericsson 芬兰公司的管理部门工作的同事致以感谢，他们是 Christian Engblom、Jussi Haapakangas、Rolf Svanbäck、Roger Förström 和 Stefan Von Schantz。他们的鼓励是我写这本书的原动力。他们同来自于 Ericsson 瑞典公司的 Carl Gunnar Perntz 和 Olle Viktorsson 一起，在我完成《SIP 揭密》的写作时让我去纽约 Columbia 大学同 Schulzrinne 教授一起工作。

Schulzrinne 教授的建议和指导使我能顺利地开展 SIP 相关问题的研究，那时 SIP 仅仅是 MMUSIC 工作组中的一个简单的 Internet 草案。

Miguel Angel Garcia 和 Jonathan Rosenberg 对本书进行了指导并提出详细的评语。他们对原稿进行了审阅，使这本书得以顺利出版。

最后，但并非不重要的，感谢 McGraw-Hill 的编辑 Marjorie Spencer，她为本书的手稿进行了大量的编辑工作。她对终稿的影响值得赞扬。她提出了新的想法和不同的观点，帮助本书使各种类型背景的读者能更清楚地理解本书的技术解释并从中获益。

译 者 序

正如 Jonathan Rosenberg 博士在本书的序言中所说的“在电信界以外毫无所知的情况下，一场静悄悄的革命发生了”。确实如此，SIP 的出现打破了传统的电信业务的传输模式，它用基于 Internet 的准则为电信业带来了新的生机。

不知道读者以前是否听说过 SIP，或许没有，或许听说过一点，但对其协议的细节都不会有所了解。这可以理解，因为 SIP 是一个正在发展和研究中的协议。

作为第三代移动系统的信令协议，SIP 能够提供 IP 多媒体业务，能将蜂窝系统与 Internet 应用领域融合在一起。它借鉴了 Internet 的标准和协议的设计思想，坚持简洁、开放和可扩展的原则，同时也考虑了 Internet 网络环境下的安全问题。同时，它还能够支持传统的各种 PSTN 业务，包括智能网业务和 ISDN。

本书适合于那些想了解新技术、并想通过了解新技术来预测电信市场和技术发展趋势的电信技术人员、市场人员、管理者和网络工作者，也可作为大学通信专业、网络专业学生的参考资料。

本书由白建军、彭晖和田敏等翻译，最后白建军统稿。由于时间紧迫，加之译者水平有限，书中难免有很多翻译不确切的地方，恳请广大读者指正批评。

译者

2002 年 7 月

于长沙

序

在电信界以外毫无所知的情况下，一场静悄悄的革命发生了。这个革命的目标是推翻已有数十年之久、青春不再却仍然是今天的有线和无线电信网络基础的技术。这场革命能将人们从许多电信服务的高投入低增值中解放出来，并把他们带入低投入高增值的服务中去——这是 Internet 的准则。这场革命不用刀剑或者枪炮进行战斗，而是用技术——Internet 技术——用它来重新定义电信网络的体系结构。领导这场静悄悄革命的是会话初始化协议（Session Initiation Protocol，SIP），它是 Internet 工程任务组（Internet Engineering Task Force，IETF）开发的一个 Internet 标准。

以前听说过 SIP 吗？或许没有——这就是问题之所在。发展至今，SIP 知识及其相关技术一直是技术精英们的领地。但是，SIP 在电信工业中引发的变化对于许多人来说都是举足轻重的——从技术管理员到商人，再到企业网络管理员。这些人不需要知道这项技术的细节，但他们必须懂得它的重要性并理解它可能如何影响他们的工作。他们就是本书适合的读者。《SIP 揭密》不仅是一本适合于软件开发者和协议工程师的书，它也适合于更多的读者——需要了解 SIP 背景知识、基本操作和它与其他协议和技术间的相互关系的人。

Gonzalo Camarillo 在技术的完整性和技术概观上表现得非常出色。Gonzalo 是 SIP 革命中一个重要的贡献者。他是几个关键文献（在标准部分中得到了发展）的作者。他还是一位在邮件列表上活跃的投稿者，一个对那些有基本问题的人们的好老师。正是因为技术深度与教学技巧的结合，才产生了这本优秀的专著。我诚恳地向那些想问“什么是 SIP？”和“它为什么那么重要？”的每一个人推荐这本书。

作为 SIP 的合著者之一，我已经为它的发展贡献了几年的时间。我看着这项技术从学术领域开始，发展成为即将在未来几年中改变电信工作方法的推动力。给人真实深刻的印象是：在 SIP 的发展过程中，这项技术所构建的基本目标和规则没有发生改变。发生这种情况的原因是因为有了那些相信 SIP 显现出来的美好前景的人，和那些在团体内努力工作以促进 SIP 发展的人。Gonzalo 在书中就抓住了这个美好前景。因此，我鼓励你——读者——翻开这本书，多了解一些电信发展的未来。

SIP 合著者，DYNAMICSOFT 公司首席科学家

Jonathan Rosenberg 博士

前　言

会话初始化协议（Session Initiation Protocol，SIP）最近的几年在电信界受到极大的关注。近来，将 SIP 作为第三代移动系统的信令协议以提供 IP 多媒体服务的决定，使得希望了解 SIP 的人数急剧增加。SIP 是能将蜂窝系统与 Internet 应用领域融合在一起的协议。它提供了所有使得 Internet 如此成功的服务无所不在的途径，用户将能够把传统的 Internet 服务，比如 E-mail、Web 以及多媒体和即时消息等新服务结合起来。

虽然人们对 SIP 能够提供的服务比较清楚，但相对而言对协议本身却缺乏了解。许多人认为 SIP 是一个能够解决人们所能想到的所有问题的协议，但事实上，SIP 的使用范围是有限的。在我从事 SIP 标准化期间的这几年中，不只一次地听到这种误解和许多类似的误解。这是促使我写这本书的主要原因。本书力图阐明 SIP 的基本原理。

为了能更好地理解一个协议（比如 SIP），需要回答 3 个简单的问题："是什么"、"怎么样"和"为什么"。本书回答了这 3 个问题，但更注重第 3 个问题"为什么"的解释。这样做的原因是来自于我同几个工程师和程序员交谈的体会。我发现他们都了解大量关于 SIP"是什么"和"怎么样"的问题，却不理解这个协议的基本原理。他们不知道为什么 SIP 被设计成这个样子。他们了解协议的局部细节却不知道其总体，这使得他们好像只见树木，不见森林。"为什么"的问题对于商业管理人员也是很有用的，他们不需要深入地了解协议的细节，而只要求知道为什么要在他们的产品中使用某个特定的技术。如果人们使用 SIP 是因为赶时髦，而不是因为它所拥有特别的优点，那么这是很可悲的。

为了理解为什么 SIP 是一个优秀的信令协议，我们需要理解它的体制并了解这个体制同其他体制相比有什么优势。这就是在第 1 章（"电路交换网络中的信令"）介绍传统电话信令的原因。这个简要的介绍能够帮助读者理解为什么需要进行模式转换以及 Internet 体制的优点和缺点。

在第 2 章，一开始介绍了分组交换和 IP。这些内容是为那些在电信界已经有了经验，并准备投身于数据通信技术领域的专业人员准备的。他们将从中发现分组交换网络的优点和缺点，以及在现代网络中，为什么要用 IP 而不是其他网络层协议来实现基于分组的服务。

在第 2 章的其余部分和第 3 章（"Internet 多媒体会议体系结构"）开始介绍 SIP 的内容。这些内容涉及了 SIP 是怎样同其他协议（Internet 多媒体会议体系结构）相互作用的，以及在 Internet 工程任务组（Internet Engineering Task Force，IETF）中 SIP 是如何实现标准化的。这些内容使读者了解各个 SIP 扩展的不同成熟阶段以及它们的含义。弄懂 Internet 多媒体会议体系结构对于理解 SIP 和其他属于这个体系结构协议的作用范围是有用的。所有的这些协议相互作用以向用户提供多媒体服务。

从第 4 章（“会话初始化协议”）到第 6 章（“扩展 SIP：SIP 工具包”），涉及更多的内容是“是什么”和“怎么样”的问题，当然也没有忘记“为什么”的问题。但这两种概念要尽可能地分离。应该将 SIP 提供的功能和怎么样实现这些功能的协议细节区分开来。首先理解协议是干什么的，才能更容易地学习它是怎么样实现的。第 4 章是关于 SIP 是什么，而第 5 章（“SIP：协议操作”）解释了协议的语法。这两章内容的区别在第 6 章解释多个 SIP 扩展时也显现了出来。每种扩展被清晰地分成了两个部分：第一部分解释扩展是什么，第二部分介绍它的实现方法。

最后，第 7 章（“用 SIP 工具包创建应用”）提供了选择 SIP 作为信令协议的体系结构的例子，比如 3G 或 PacketCable。

读完本书后，读者将会对 SIP 的 3 个方面的问题：“是什么”、“怎么样”和“为什么”有较好的理解。你将能理解 SIP 在不同体系结构中的地位和它同其他协议间的交互作用。此外，你将能够决定 SIP 是不是解决你的问题的合适工具，以及如果是这样，和构建这个体系结构所需的其他协议相比，哪个更适合于你的应用。读者要注意，SIP 是 Internet 多媒体会议体系结构（能用来提供多媒体服务的一系列协议的组合）的一部分，而不是一个孤立的协议，这一点是非常重要的。

目　录

第 1 章　电路交换网络中的信令

通过电话网，也就是公共交换电话网（PSTN，Public Switched Telephone Network），可以将电话打到世界上的每一个国家。我们几乎可以在任何一个房间找到电话设备，包括简单的模拟电话、更高级一些的综合业务数字网（ISDN，Integrated Services Digital Network）电话、无绳电话、蜂窝电话甚至卫星电话。在所有的应用技术中，电话系统分布的广泛性真是令人吃惊。

所有这些电话都有一个共同点，那就是它们都使用电路交换网络来彼此通信。PSTN 已经存在了很长的时间了，并且受到了使用者的欢迎。大洋两岸的人们通过它可以相互交谈，就像近在咫尺一样。甚至在两个移动电话之间进行远距离通话，也能很好地理解对方所讲的内容。

另外，PSTN 是一个高度可靠的网络。当拿起电话拨号时，一般都能成功地拨出。电话交换机很少会崩溃。即使出现了这种情况，后备系统也会立即接管，继续为用户提供服务。

总而言之，人们信赖 PSTN。人们对每天无故障地使用 PSTN 越来越有信心，并且在紧急的情况下总是依靠 PSTN，因为通过 PSTN 可以和医院、警察局、消防局相连。

考虑到 PSTN 的所有这些特性，一种极自然的想法就是：其他提供类似服务的网络也都会效仿 PSTN 的过程和机制。因为 PSTN 工作得如此之好，所以后面我们会模仿语音网络。特别是，我们会使用类 PSTN 的信令协议——因为它们工作得非常好。

但是，这个想法是错误的。我们将会看到对于不同的环境，比如 Internet，为什么类 PSTN 协议就不适用。PSTN 体系结构和互联网体系结构由于使用的介质不同，需要用一种全新的信令协议，而仅仅对可信赖的、老的协议进行改进是行不通的。

这一章简要介绍 PSTN 信令的历史，解释信令协议如何从模拟机制发展到数字机制。还将会看到 PSTN 与 Internet 除了传输技术的不同以外（前者是电路交换的，而后者是分组交换的）还有哪些差异，由于它们各自管理和操作的模式不同，导致了它们必须遵循哪些不同的信令设计。

1.1　电路交换的起源

电话网络的目标就是为它的使用者提供特定的服务。大部分的服务可归结为电话服务的范畴。但是，首要的和最重要的服务就是用户之间的语音传输。网络中的任何用户都能呼叫网络中的其他用户，这是所有的电话网络都必须满足的基本要求。

最早的电话网允许两个用户通过两个电话设备和一根相连的电缆线进行通信。这个系统

可向网络中的两个用户提供足够的语音服务。但是当更多的用户希望使用这个网络时，电话设备的数量也会相应增加。向这个网络中增加一个新用户的最好方法是：从这个新用户的电话设备上安装电缆连接到其他所有已经安装好的电话设备上，从而产生一个全连接的网状拓扑，使每一个电话设备都可以直接连接到其他的电话设备上，如图 1-1 所示。

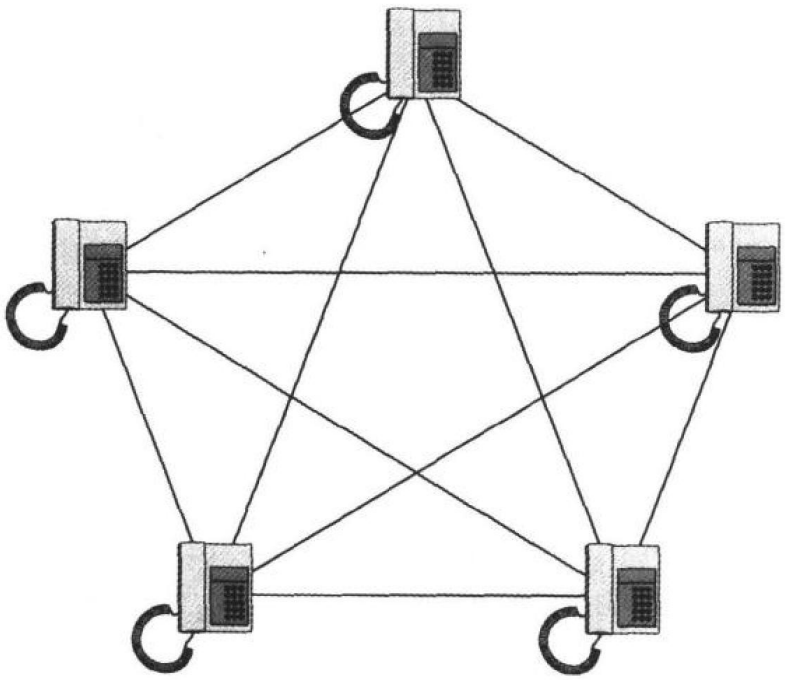


图 1-1　一个全连接的网状拓扑

然而，全连接的网状拓扑在电话环境中存在许多缺点。最大的问题是它不能扩展。在全连接的网状拓扑中，每一个用户都要和其他潜在的被呼叫的用户建立单独的连接。在系统很小的时候，和每一个用户建立物理连接是可以承担得起的；但是当用户数量增加了，连接的数量以指数式增长。电缆既贵又难于安装，因此向网络中增加新用户的成本会变得非常昂贵，并且安装速度非常慢（译者注：原文中此处误为 quickly）。

除了成本因素以外，如果一个电话设备有许多电缆线连接到网络中的其他用户端，路由就将变得十分复杂并难以管理。每个电话设备为了路由呼叫不得不维护一个巨大的路由表，用来指示哪一条电缆连接到路由目的地。

由于全连接网状拓扑存在可伸缩性和可管理性的局限性，故早期的系统从全连接网状拓扑发展为星形拓扑。在星形拓扑中，所有电话设备都与一个叫做交换机的中央单元相连接，如图 1-2 所示，并且所有的呼叫都通过它从源端到目的端进行路由。交换机将呼叫始发者的电缆连接到呼叫接收者的电缆。最初这个操作是手工进行的，它通过一个叫做交换台话务员（Switchboard Operator）的人把线缆插入到插座中来执行这种连接，现在看起来有点离奇。

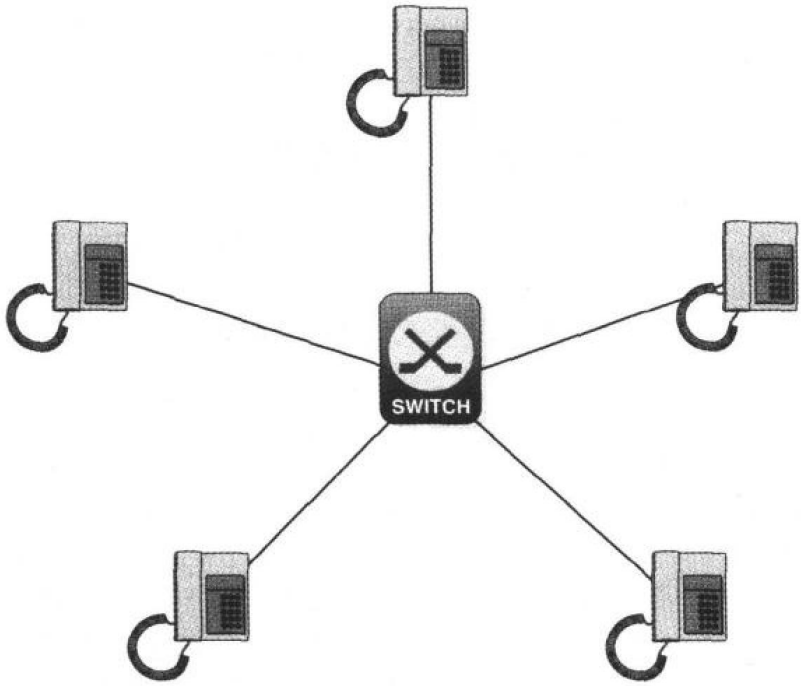


图 1-2　星形拓扑

原来的想法可能是：这项技术既然还在使用，就仍把它称作电路交换。它的明显的优点是，电路交换易于向系统中增加新的用户，因为在交换机和新的电话设备之间只需要添加一根电缆，它降低了“成本”。系统的管理变得简化了。虽然交换机仍然要处理大量的电路，但是用户设备——电话机——依旧简单。

电路交换的下一步就是连接几个交换机构建成电路交换网络，如图 1-3 所示。在这种网络中，每个用户的电话设备连接到与它最近的交换机上，这个交换机再连接到其他的交换机上，依此类推，连接整个系统。

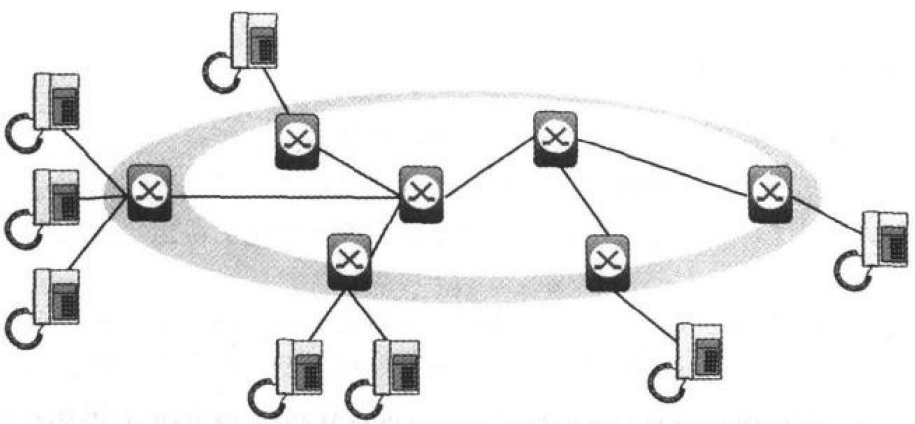


图 1-3　分层拓扑

在电话技术中，交换机也被称为 Exchange。因此，距离电话设备最近的交换机叫做本地交换机（Local Exchange）。在两个电话机间的呼叫，首先要路由到呼叫者的本地交换机上，然后穿过其他的交换机，直至到达被呼叫者的本地交换机上，从而可以通过振铃来通知被呼叫者。这种电路交换技术克服了前面介绍的电话系统的缺点，甚至现在仍然在语音传输中占统治地位。但是电路交换技术从最初的单独连接经过不断发展，目前若干分层的不同级别的交换机已经实现了相互连接，组成一个世界范围的通信网络——PSTN。

1.2 电路交换的特性

除了语音传输以外，电路交换经常用来传输不同类型的信息流（Traffic）。例如，电路交换网络可以在两台计算机间传输数据，在两个终端间传输控制信号。但是，无论传输哪种类型的信息流，用户设备都被叫做终端（Terminal），而交换机设备则叫做网络（Network）。

当网络由服务提供商管理时，通常终端用户可以熟练地操作终端（通常终端由端用户操作而网络由服务提供商管理）。网络建立了终端之间的通信路径，终端的任务就是从用户那里接收信息并把信息以适当的格式发送到网络中去。同时，终端也执行相反的任务：从网络中接收信息并把信息发送给用户。在这两种情况下，检查将要发送信息的内容并且执行必要的操作都是由终端独立完成的，不需要用户的干预。相反，交换机在两个或更多个终端之间建立了专用路径，但它并不关心在终端之间传输的是什么内容。

我们已经看到，这个专用路径基本上可以看作是在终端和交换机之间连成一排的一组电缆线。在更高级的系统中，路径由电缆中的频率、时隙（Time Slot）或者是光纤（Pptical Fiber）中的波长组成。无论这个路径多么复杂，电路交换的关键点还是一样的：交换机不考虑它传输信息的内容。连接的方式只取决于电缆的位置（也就是说，1 号电缆必须和 4 号电缆连接）、频率（也就是说，从 1 号电缆中以 400Hz 输出后必须在 4 号电缆中以 600Hz 传输）和帧中的时隙。我们将会在第 2 章“分组交换、IP 和 IETF”中看到，这就是电路交换和分组交换的主要不同之处。

1.2.1 电路交换的优势

电路交换具有一些非常好的特性。这些系统的传输速度非常快。因为交换机不必检查传输的内容，决定在某一接口上接收到的信息发送到什么地方，只需要在初始建立连接时执行一次路由决定就可以了，在该连接的持续时间里都会使用相同的决定。因此交换引起的延迟几乎可以忽略不计。

电路交换网络在终端之间提供的专用路径特别适合于语音的模拟传输。当一个电路建立后，传输延迟非常小并且在这个连接的持续时间里保持不变。适合于模拟传输是电路交换技术在前数字时代广泛传播和发展的主要原因。我们将会看到，当新的信息流模式出现以后（包括计算机间的数据传输），电路交换开始显现出了它的局限性。

1.2.2　电路交换的弱点

在终端间进行信息交换之前，网络中必须建立一条路径。建立一条路径要花费时间，实际的传输要产生延迟，直到路径建立阶段完成，如图 1-4 所示。

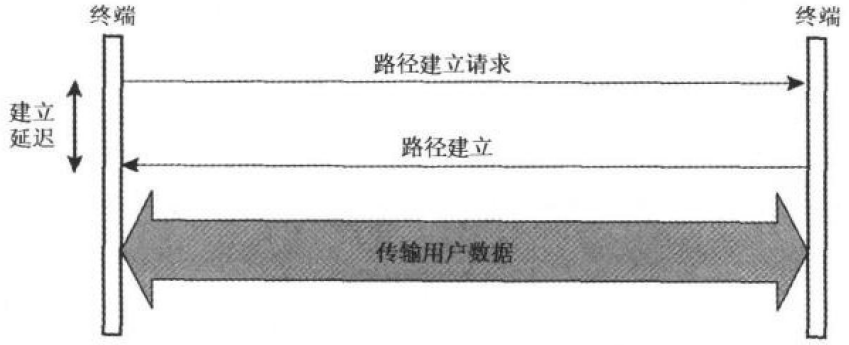


图 1-4　电路交换网络中的建立延迟

注意，路径建立延迟不同于交换延迟。就其本质而言，建立延迟产生于任何传输之前；而由交换引入的在用户数据传输期间的延迟发生在路径建立以后。

一旦两个终端间的专用路径建立起来了，与之相关的资源就不能被其他的连接所使用，直到这个路径拆卸。因此，即使是在某些时候两个终端都停止传输了，路径仍保持开放状态，并且沿着这条路径的所有交换机中分配给这个连接的资源仍然被占用。退一步说，这就是可利用资源的无效使用。这个弱点，对于模拟语音传输不是一个很大的问题，但对于数字传输，特别是在计算机间数据传输中，就变得相当严重了。

执行交换数据的两个终端事实上大部分时间是空闲的，同样，分配给这个交换的资源大部分时间也是空闲的。无论什么情况下，信息流交换是突发的和非均匀的，电路交换会无效地耗费着资源。在后面将会看到，分组交换克服了这个弱点（但以降低交换速度为代价）。

在整个网络中连接终端，交换机需要知道什么时候建立路径、什么时候拆卸这个路径、需要控制什么资源（也就是说，使用电路的数量）以及怎样路由呼叫。所有的这些信息都在整个网络中的交换机之间交换，对于每一个呼叫，这些信息同时也传达到对应的两个终端和整个网络中。总起来说，这些与控制连接相关的信息被称为信令（Signalling）。

1.3　信 令 介 绍

信令的作用是在任何类型的系统之间交换控制信息。信令在其他系统中常见的应用包括：铁路交通和空中交通的控制。当一个飞机驾驶员提出在一个机场降落的要求时，他就要和机场指挥塔台（Control Tower）交换信令信息。机场指挥塔台然后给这架飞机提供一个时

间间隙和一条跑道以供其降落。在电话技术中，信令的存在隐含表明在一个电话呼叫中有两种类型的信息流（traffic）要进行交换：信令流（控制建立和释放语音路径）和语音流。我们通常把这些不同类型的信息流称为平面（Plane）。因而，在一个电话呼叫中能发现两个平面：控制平面（Control Plane）和用户平面（User Plane）。控制平面处理控制用户平面的进程，而用户平面处理实际的数据或语音传输，如图 1-5 所示。

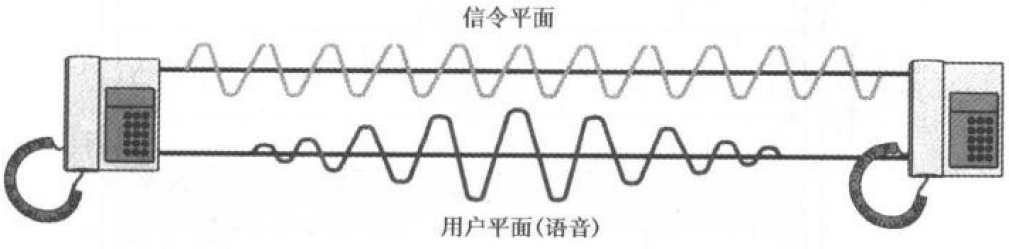


图 1-5　一个电话呼叫中的两个不同的平面

信令平面的发展同用户平面的发展紧密相连，如果信令系统没有利用新特性的能力，那么交换机中的新特性也就不能够被使用。设计新的信令系统时，在用户平面中应能够最有效地利用最新的进展。（新的信令系统被设计来最好地利用用户平面中最新的进展。）但是，仍有新的信令系统被引入而同时用户平面中没有取得任何进步的例子，在这种情况下，其他的收益——简洁性、有效性、耐用性（健壮性）——就成为生死攸关的关键性问题。要理解信令传输协议是如何从 19 世纪后期的第一个电话系统发展到现在的，就要理解获得了什么收益和在某一特定时期是什么触发了协议的设计。SIP 就是这样的一个协议，并且它也是这个发展过程中的一个阶段。

1．本地和中央电池系统

1876 年 3 月 10 日，在电报被使用了几年后，Alexander Graham Bell 获得了使用连续的电流来进行电子语音传输的专利，从此电话诞生了。

在早期的电话系统中，信令非常简单。当用户拿起听筒时，一个电路就接通了，终端然后就向这个电路提供电流。接通电路就意味着占线（seizure）。如果线路占线，电话交换机房中的接线员就会响应，用户只要告诉接线员被呼叫者的身份就完成了路由过程，接线员然后人工交换这个呼叫。这就是本地电池系统（Local Battery System）。

本地电池系统存在一些问题。把电池放在终端，使得维护工作变得更加困难，因为把它留在了用户的手中。因为电池技术不像现代终端那样先进，用户发现电池的使用很复杂，就不热心接受它。中央电池系统（Central Battery System）的设计就是为了吸引用户，它从交换机中向终端提供电流。系统自动化是使电话吸引用户的下一步。从 1878 年出现第一个真正的交换台开始，电路交换一直是由人工进行的。在 1889 年，Almon B.Strowger 申请了自动电话机电交换机的专利，从此不再需要人工干预交换。用户在线路占线时获得了一个拨号音（Dial Tone）。当本地交换机提供了一个拨号音后，终端就可以发送被呼叫者的电话号码。得到了这

些信息后，本地交换机就可以将该呼叫路由到正确的目的地。

在第一个自动化的系统中，信令的方向只能是前向的，比如从呼叫者的终端到交换机。任何向后传输给用户的信息则要通过用户平面进行发送。因此，如果被呼叫者忙，一个忙音（Busy Tone）会通过用户平面发送过来，这个忙音会一直持续到用户挂上电话。如果在被呼叫者回答之前，或者是由于被呼叫者忙，用户决定放弃呼叫而挂上了电话，控制平面却不会知道，这已被证明是非常严重的缺点。

2．直流和交流模拟系统

接下来发展的信令系统是直流和交流模拟系统（DC and AC analog system），之所以这么称呼是依据它们使用的电流。它的优势是允许信令向后传输。允许信令以“被呼叫者”→“呼叫者”的方向进行（如图 1-6 所示），在有些情况下，这代表了一种很重要的方法。

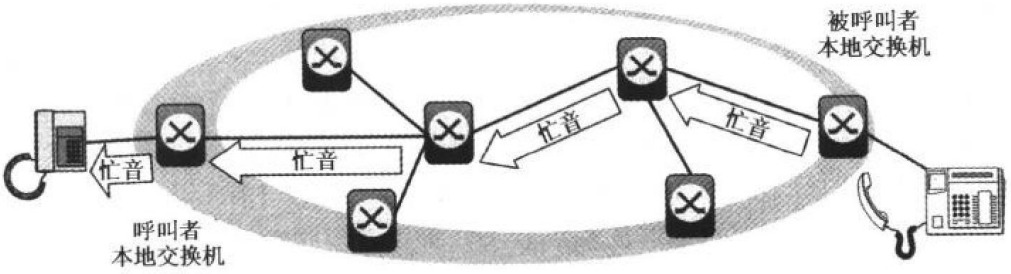


图 1-6　被呼叫者忙和只允许信号向前传输

呼叫者拨电话号码，呼叫者和被呼叫者之间路径上的每一个交换机都保留一个电路来传输这个呼叫的语音内容。当信号到达被呼叫者的本地交换机（也就是这个路径上的最后一个交换机）上时，交换机把这个连接定位到被呼叫者的终端。本地交换机检查这个连接的可用性，同时注意到此时被呼叫者的终端正在被一个其它呼叫使用。在一个不能向后传输信令的系统中，本地交换机就会产生一个忙音，这个忙音会传输给呼叫者以便让他（或她）知道而放弃这次呼叫。当听到这个忙音后，呼叫者会挂上电话，虽然有时候他不会立即那么做。当电话挂上后，表明呼叫者已经结束的信令会向前发送到被呼叫者的本地交换机上。最后，这个本地交换机会被触发而释放这个路径上的所有电路。

在上述系统中，即使当交换机已经通告呼叫者不需要资源了，但这个接续所占用的大部分电路仍然保持着。直流和交流模拟系统能够更好地利用网络资源。在通过控制平面发送忙的状态时，允许释放用户平面（也就是释放为这个呼叫而建立的电路交换路径）。呼叫者听到的忙信号已是他（或她）的本地交换机代为产生的了，并且当通知用户呼叫的状态并准备作出反应时，系统已经不再保留不需要的资源（如图 1-7 所示）。

因此，具有两个方向的信令可以使网络管理更加有效：控制平面能够接收更多、更准确的关于呼叫状态的信息。

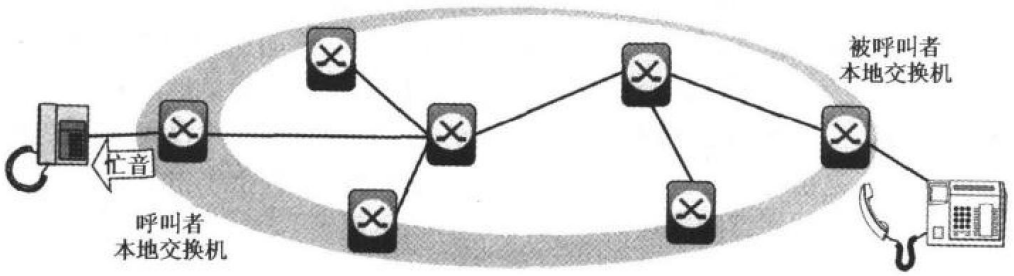


图 1-7　被呼叫者忙和有一个向后方传输的信号

1.3.1　FDM 和带内信令（In-band signalling）

电路交换在接下来的数十年间得到了持续发展。用于电话的每一个电路都由铜线组成，因此，一个交换机处理 500 个电路就需要 500 条电缆线。采用频分多路复用（FDM，Frequency Division Multiplexing）可以使数个同时发生的呼叫使用单条电缆。在 FDM 系统中，每一个语音路径占用不同的频谱（frequency spectrum），如图 1-8 所示，实际上是将一个电路重新定义为一条物理电缆中的一个频率而不是电缆本身。FDM 大约从 1910 年发展起来，但在 1950 年以前没有得到实际的应用。用那时的观点看，一条轴电缆线能够提供 1 000 个电路，确实是一个实实在在的进步。

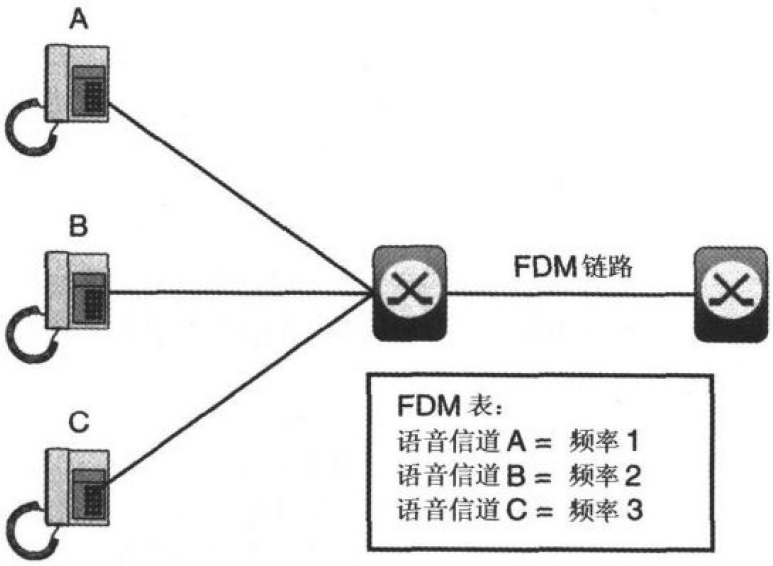


图 1-8　FDM 系统

FDM 系统需要一个新的信令过程：两个交换机间电缆中的的信号需要控制许多电路而不是一个电路。交换机需要用一种方法使得一个具体呼叫的控制平面同它的用户平面相关联。

已证明了的解决方法是带内信令系统，也就是信令以传送话音相同的频率传送——它们要在同一个电路中一起传输。

用最简单的术语描述就是：电路占线用一个脉冲来标示，这个脉冲发射信号，这个信号沿着在后面要传输语音的路径上传输。带内信令现在仍在一些国家中主要作为国际连接使用。

虽然 FDM 通过使大量连接共用同一条电缆而降低了电话系统的成本，但它仍被认为是一种昂贵的技术。FDM 使用模拟滤波器来分离不同的信道，维护这些滤波器的工作所花费的代价是很昂贵的。人们不得不定期检查它们——经常是一个月就得检查一次——使它们工作在要求的频率上。

1.3.2　模拟传输

从一开始，语音传输就是模拟的。换句话说就是：不考虑传输的内容，传输是由发送的连续信号组成的，比如语音（如图 1-9 所示）。用在电缆中移动的、负载着信息的电磁波来表示这个传输的信号。接收器需要从接收的电磁波中恢复原始信号以便掌握信息。

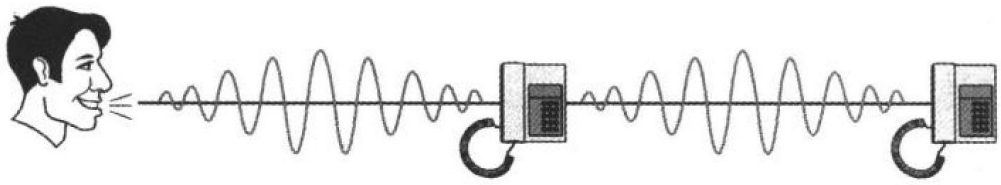


图 1-9　语音的模拟传输

连续的信号由不同的频率组成，但是每个信号总是将它大部分的能量集中于某个确定的频率范围——这就是所谓的信号的频谱（Spectrum）。频谱之外的频率成分只包含了很少的能量，因此它对确定信号的译码作用不大。结果是，通常只需要传输频谱内的频率成分就可以恢复原始信号，或者说精确些，是得到与原始信号非常相似的信号。

在模拟传输中，从一个信号获得的电磁波包含了频谱的所有频率成分。语音传输的标准频谱范围从 300～3400Hz，并且使用这个范围频率传输的语音能很好地被接受者理解。

模拟传输的主要优点是没有延迟。因为传输路径上的交换机不会对信号增加延迟，语音的传输速度实际上就是电磁波在介质中的速度。这个特性特别适合交互式通信——参与方包括活跃的发送者和接收者。

1.3.3　数字传输

此时读者可能已经注意到了一种发展模式，即用户平面产生的进步促使信令平面发展并利用它们，接着又返回来促进用户平面获得增长。电话技术的下一步进展是引入数字传输技术。数字传输改变了语音传送的方式，并使信令协议得到了显著的发展。

那么它是怎么做的呢？我们在前面已经解释了模拟信号具有连续的数值（Value）。与此

相反，数字信号的数值是离散的。一个数字信号的例子是文本（Text）。文本是由字母组成的，因为每个字母是从一组字母——字母表——中选出来的，因此，任何字母只能是“a”～“z”范围内的离散数值。

在语音数字传输中只使用两个数值：0 和 1。因此，系统间交换的信息都是由 0 和 1 组成的串构成的。目前正在使用的这种串的编码方法有多种，其中的一种是：指定一个电平常数为 1 而其它与此不同的电平为 0。接收器通过分析收到的电平来恢复原始数字流。

但是，语音不是一个数字信号。它是一个模拟的声学信号，在时间域内具有连续的数值，并且在它发送之前需要进行变换。为了在一个数字链路中发送模拟信号，需要一个编码译码器（Code-Decoder）。编码器以收到的一个模拟信号作为输入，并产生一个数字信号作为输出。可以用不同的算法来实现这个过程。把语音转换成数字流指的是音频编码（Audio Codecs）（如图 1-10 所示）。应用最广泛的是 G.711 编码译码器，它也被称作脉冲编码调制器（PCM，Pulse Code Modulation），它可以产生 64kbit/s 的数字流。

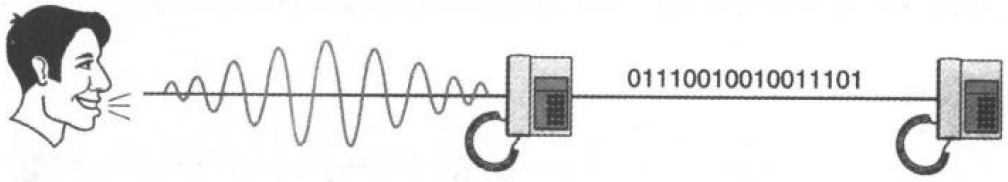


图 1-10　语音的数字传输

1．数字传输的优势

数字传输相对于模拟传输显现出许多优点。首先，数字设备便宜。流操作可以用相对比较便宜的计算机来执行，而模拟交换需要使用昂贵的机电设备。自从 1960 年在美国安装了第一台数字交换机后，计算机的价格已经急剧下跌。其次，数字传输质量高。模拟传输会产生信号衰减问题，也就是说，信号强度会随着传输距离的增加而减小。交换机使用放大器（Amplifier）来使信号强度保持在一个可以接受的水平之上，但是这样做就会不得不放大整个信号（记住，交换机不能检查模拟传输的内容）。放大整个信号会产生其他的问题，那就是把噪音也放大了，使得在目的地接收信号时会产生失真。

与此不同，数字系统使用中继器（Repeater）而不是放大器。中继器对收到的流进行解码，并在输出端重新生成流。因而输出的信号是全新的，并且它的强度不依赖于先前链路信号的强度。用这种传输过程，进行数字交换时不会把噪音加到信号中。

2．数字传输的弱点

如果考虑延迟的话，数字系统就不如模拟系统好。数字信号的传输过程总是比模拟交换中电磁波的传输速度慢。我们将会看到，数字传输中使用的多路复用机制（TDM）也会带来一些延迟，因为数据不得不在缓冲区中等待分配时隙。

虽然每个数字网络中都存在缓冲延迟，但是现代交换机具有极高的处理速度，执行这些操作非常快。因而在大多数应用中，数字传输带来的延迟可以忽略不计。例如，对于语音应

用来讲，往返传输延迟如果高于 300ms，会使一个正常的通话过程变得十分困难。而数字交换带来的延迟远低于这个极限。

对于传输语音的系统来说，更大的问题是终端的复杂程度，因为它不得不包含一个编码译码器来将声信号转换成数字流。这种机制使得数字终端更难于处理，并且比模拟终端更昂贵。为了避免终端设备过于复杂，许多数字网络都安装了一个面向终端的模拟接口，这个处于终端和本地交换机之间的接口被称作本地环路（Local Loop）或是用户线（Subscriber Line）（如图 1-11 所示），它使得用户可以保留可信赖的传统的模拟终端并且仍旧与一个数字式的本地交换机相连接。模拟信号的编码和译码过程在用户的本地交换机内进行。

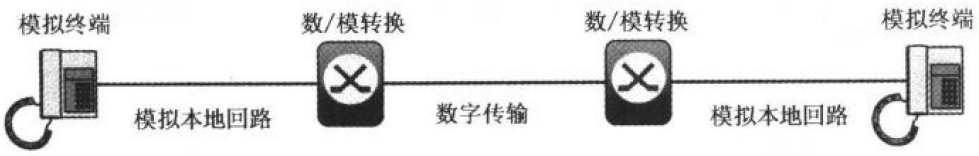


图 1-11　模拟本地环路

读过上述这些内容以后，读者可以看到，随着时间的推移，模拟正向数字迁移。已经存在的 PSTN 的基本原理是使用户设备尽量简单，而网络可以复杂一些。出现这种倒退有一些另外的原因，包括向后的兼容性和终端的价格。但是不可否认的是，这种设计风格一直延续至今。

1.3.4　时分多路复用

FDM 除了是一个比较昂贵的技术外，多路复用的能力也有限。在 FDM 中，不同的信道不得不使用分得足够远的频率以避免干涉，并且可用的频率受限于介质的带宽和衰减（衰减是频率的函数：频率越高，衰减越严重）。因为衰减越严重，在交换机中需要放大得越大，伴随着模拟信号的噪音也被大大放大了，降低了信号的质量。

多路复用能力的局限性是网络中一个实实在在的缺点，它激励了人们放弃 FDM 而寻找其他更好的复用方式。新的高带宽传输介质的出现，也可使得多路复用对能有效利用所有可提供的带宽。时分多路复用（TDM，Time Division Multiplexing）是现在在数字传输中使用最广泛的多路复用机制，它比“前辈们”更容易安装并且价格更便宜。

下面是 TDM 的简要工作原理。相互连接的交换机向位于它们之间特定的信道分配时隙（如图 1-12 所示），只有这种时隙才能在分配的信道上交换传输数据。因此，属于第一个信道的信息在第一个时隙中传输。第二个时隙分配给下一个信道，依此类推。当所有的信道都已分配了一个时隙——传输的时机——就会循环到第一个信道重新开始。接收方与发送方是同步的，因此可以标记一个时隙的结束和另一个时隙的开始。

TDM 必须要求在交换机中有一个缓冲区。交换机不得不在期望有适当的时隙到来进行传输之前存储信息。这个过程（像前面解释的，又增加了一点，但并不麻烦）导致了交换进程

的延迟。

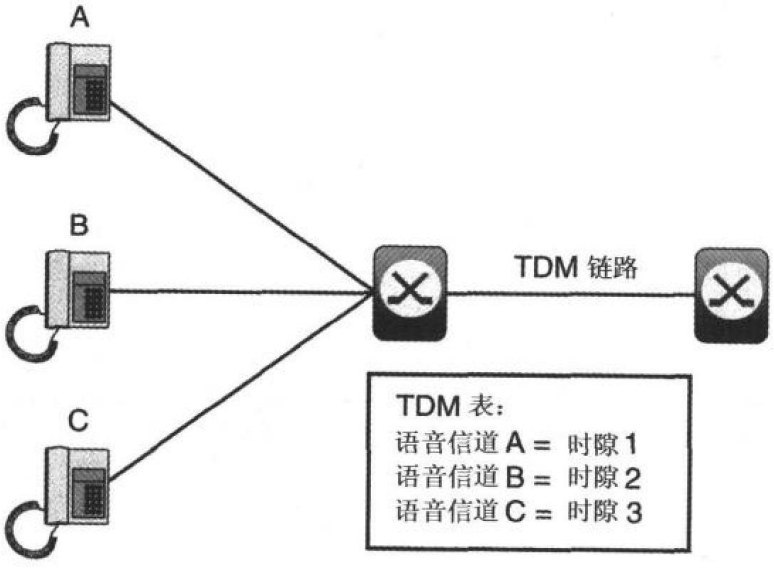


图 1-12 TDM 系统

注意到这项主要电话传输技术提供了 64kbit/s 的信道，恰恰与 PCM 编码译码器的数据输出率相同。这使得这些信道非常适合于 PCM 编码语音的数字传输。

1.3.5 数字信令系统

数字传输的出现使得信令系统不得不产生变革。对照模拟系统使用音（Tone）和脉冲作为信令的情况，一个占据了某个确定频率的脉冲意味着电路占线，而音被用来传输地址信息。用这种方法传输信令信息的数量非常低。因而，模拟信令系统不但价格昂贵而且交换机到交换机之间效率很低。当然，数字传输增加了控制平面传输信息的数量，数字信令使得交换机能够交换位流（Bit Stream）而不是离散的脉冲。正如依据一定的语法组合字母能产生单词并随后组合成句子一样，定义位流规则也能产生信令消息（Signalling Message）。一个信令消息中包含着比一个脉冲或是音丰富得多的信息。信令协议从位流和它们的语义出发，将这些规则编集成消息（Message）。这些协议清楚地说明了在交换机间已知的情况下要执行的动作。这些动作通常用协议的状态机（State Machine）来描述，因此状态机就可以想像成在一定状态下一定事件引起的一组动作。

接入信令（Access Signalling）和干线信令（Trunk Signalling）

第一个数字信令协议用在交换机间，但用户线仍然是模拟的，因此这个信令过程也得是模拟的。这导致了接入信令（通过终端和网络之间的信令）和网络信令的分离，这个网络信令也被称为干线信令（如图 1-13 所示）。后来，当用户线变成数字的了，接入和干线的区别仍旧保留了下来，并且在接入信令和干线信令中使用不同的协议。这样做的目的是

使用复杂的交换机和保持相对简单的终端，使网络操作员能够控制网络上发生的几乎所有事情。

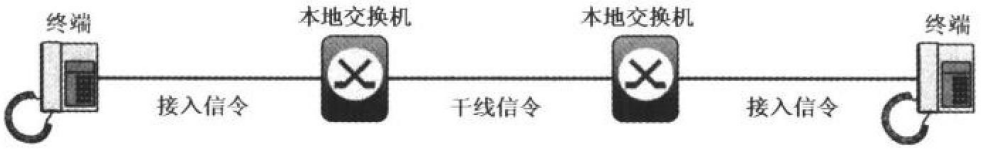


图 1-13　接入信令和干线信令

终端和网络间的接口叫做用户—网络接口（UNI，user-to-network interface），而交换机之间的接口称作网络—网络接口（NNI，network-to-network interface）。在后者中使用的协议更复杂，因为在交换机间流动的信息比流向终端的要多。终端仅仅被告知一些必不可少的信息，以便于用户更新呼叫的状态。这种模式由一个智能网络和哑终端组成，通常与 IP 模式，特别是 SIP 模式相对立。

1.3.6　接入信令

在模拟用户线上被告知的是什么类型的控制信息呢？它由摘机/挂机（off/on hook）信号、拨叫的号码以及通知用户呼叫状态的各种音组成。摘机/挂机信号用来申请一个想进行呼叫的拨号音，或是用来接收或停止呼叫（分别通过拿起电话和挂断电话产生）。所拨的号码通过脉冲或是音从终端向本地交换机发送（顺便说一下，老系统执行拨号脉冲（Decadic Pulse），而新系统实行双音多频（DTMF，Dual Tone Multi-Frequeny，）音）。音也向用户平面传送状态信息。已经定义了标准音（比如忙音或发信号音），但有时从一个国家到另一个国家的音也不相同。

使用数字终端使得发展数字接入信令成为必需。1 号数字用户线（DSS-1 Digital Subscriber Line NO.1，）应用最为广泛，它用在 ISDN 和移动网络中。全球移动系统（GSM，Global System for Mobile）通信移动终端和基站（BTS，Base Transceiver Station）间的信令就是基于 DSS-1 的。图 1-14 是 DSS-1 呼叫流程的一个例子。它显示了一个接收呼叫的终端和它的本地交换机间的消息交换过程。

DSS-1 的确比任何模拟信令系统使用得更广泛，但它并没有像流行的中继信令系统那样广泛，这一点我们将会在后面看到。

1.3.7　中继信令

电路交换网络中的交换信令（Interexchange Signalling）称作中继信令。存在两种类型的中继信令：随路信令（CAS，Channel Associated Signalling）和公共信道信令（CCS，Common Channel Signalling）。CCS 系统领先于 CAS 系统，因为它具有更大的容量、更高的可靠性和灵活性，现在 CAS 系统正逐渐被 CCS 系统所取代。

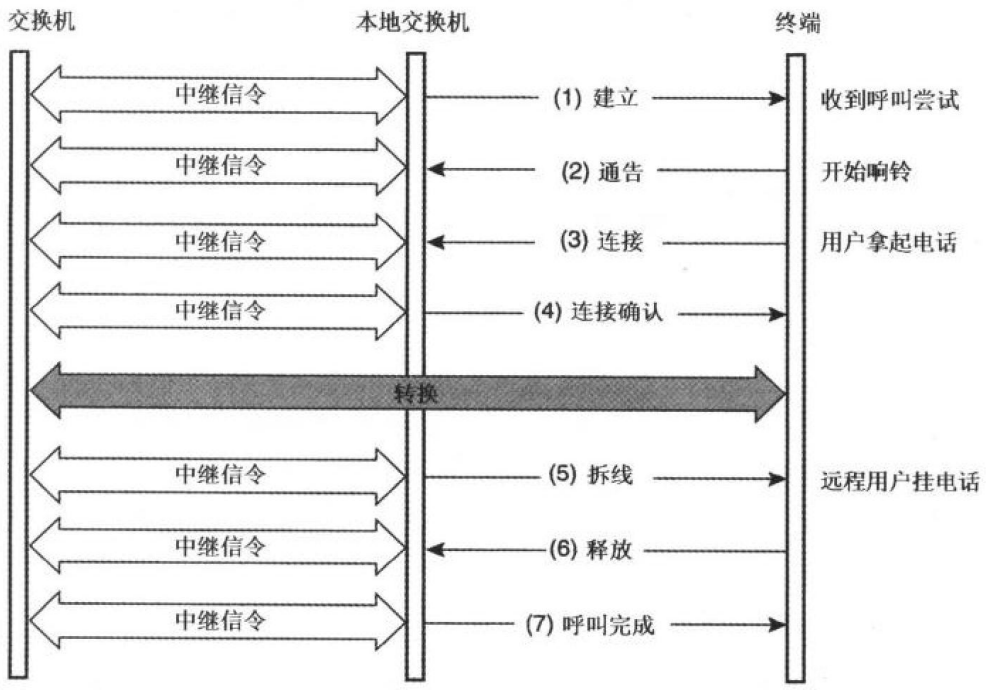


图 1-14　DSS-1 消息流

1．随路信令

在 CAS 系统中，与某一确定呼叫相关的信令和语音沿着同一个路径传输。前面已经描述过的带内信令就是 CAS 的一个例子。

CAS 既可以用于模拟系统，也可以用于数字系统。使用 CAS 的数字系统用一个特殊的信道实现线路信令（Line Signalling）。线路信令控制语音信道的建立和释放（如图 1-15 所示）。在中继线中，与语音信道相关的所有线路信令都在一个单独的信道中传输。例如，在欧洲系统中，为每个 PCM 链路保留的时隙数是 16。因此，第 16 时隙所运载的是线路信令，它将信道 1～15 和信道 17～31 联系起来（时隙 0 用于帧同步）。线路信令的一些例子是：空闲线、占线、响应和指示脉冲等。

另一方面，处理地址信息的记发器信令（Register Signalling）是通过语音路径传输的。一些记发器信令的例子是：被呼叫者号码、被呼叫者状态和呼叫者号码。因为记发器信令仅仅在呼叫建立期间发送，所以当没有语音传输时，记发器信令并不干预用户平面。

2．公共信道信令

在 CCS 中，语音和信令通常在网络中使用不同的路径传输（虽然这些路径在一定范围内仍然是相关的）。所有的节点既处理介质也处理信令，因此，如果一个语音路径通过两个交换机，这两个交换机也要收到相关的信令。但是，在 CCS 中，信令的路由方法不同于其他系统。在一个语音路径中，信令离开一个交换机到达另一个之前，可能要通过一组不能处理语音的中间节点，我们把这些中间节点叫作信令传输点（STP，Signalling Transfer Point）（如图 1-16

所示）。

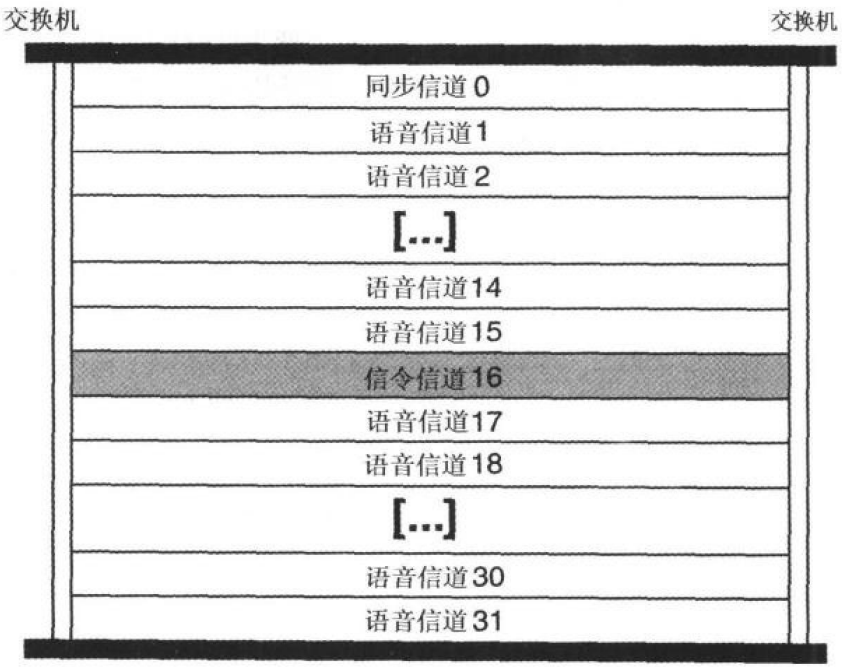


图 1-15　一个 PCM 链路中的信道

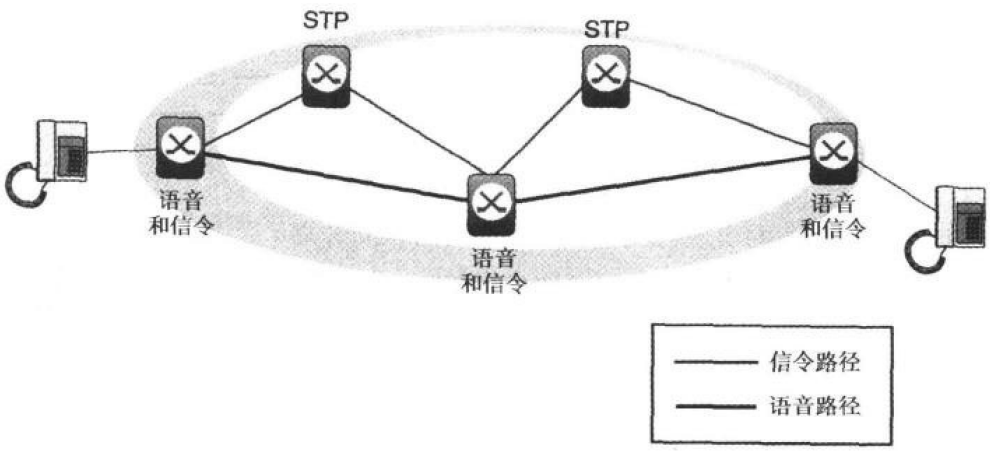


图 1-16　信令传输点

STP 是 CCS（译者注：原书有误，应为 CCS）的一个特殊特性，它迫使专用网络对信令传输实行优化，这也使得这些网络比共享语音/信号网络具有更好的性能和更高的可靠性。

3．专用信令网络

事实上，与一个呼叫相关的信令流是猝发的。在呼叫的某些阶段，这些信令流非常强烈，比如在呼叫建立和释放时；但一旦呼叫建立了，它又很微弱。在这里猝发性是有好处的，它

使得可以用一个单独信令信道去控制几个语音信道。在实践中，一个信令信道能够处理几千个语音时隙。

使一个信令网络从它的语音网络中独立出来，可以便于服务创建和使不同网络更容易地交互工作。SS7 是现在流传最广泛的 CCS 系统，SS7 的独特之处在于，它允许超出基本 CAS 连接之外进行服务的创建。不处理语音的专用网络节点间具有的发送信号的能力，事实上是当前 PSTN 所提供的所有高级服务的基础。通过使服务逻辑电路和交换逻辑电路分离，可以鼓励服务创建和用户平面操作独立地发展。

例如，当某人拨了一个免费号码，系统必须能够查寻用户所拨号码的电话（在美国，它可能是一个 1-800 的号码），并且把它传送到用户想要到达的公司号码上（地理上的决定）。这个数据库的查寻工作完全通过信令消息来实现（如图 1-17 所示）。

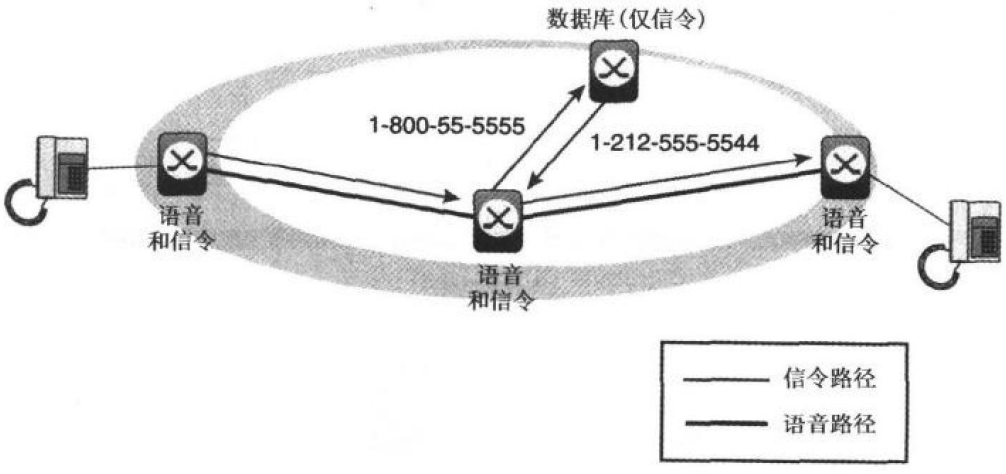


图 1-17　用户呼叫一个免费号码

免费电话和其他服务的实现得益于信令和语音路径的分离。否则的话，就必需向数据库路由语音路径，然后从数据库返回信息，以便查寻号码，导致呼叫者感受到语音质量严重降低。

今天，有两个主要的 CCS 系统在使用：SS6 和 SS7。SS6 在 1968 年标准化并打算用于国际连接。SS7 是 1980 年标准化的，现在仍在信令系统中占据统治地位。

1.3.8　SS7

SS7 提供了电路相关和非电路相关两种信令。就像它的名字所暗示的那样，电路相关的信令是与一个语音信道的建立相关的。但是常常没有语音信道建立——当系统查阅路由表时——这种类型的查寻可以用非电路相关的信令来处理。作为一个使能者（enabler），非电路信令对于可移动性来说是非常关键的。

首先看一下电路相关的信令。在这里看来，ISDN 用户部分（ISUP，*ISDN User Part*）和电话用户部分（TUP，Telephone User Part）是相关的 SS7 协议。在最后的规范中，ISUP 包括

TUP 所提供的全部功能，再加上更多的特性，比如具有传输用户到用户的信息的能力，因此在这里讨论 ISUP 就足够了。ISUP 实际上是一个协议的一般名字，它在不同的国家有许多不同的特点。当通信公司想要实现一个新特性时，他们通常使用非兼容的扩展，从而导致了互操作性的问题，并且这样做，也产生了多种本地 ISUP 特色。自然的，有国家差异的 ISUP 是不兼容的，因此用一个国际版的 ISUP 来连接有 ISUP 差异的网络。它们之间的网关交换机用来理解/支持国际 ISUP 和这个网络中执行的特定的 ISUP，并在它们之间执行传输（如图 1-18 所示）。

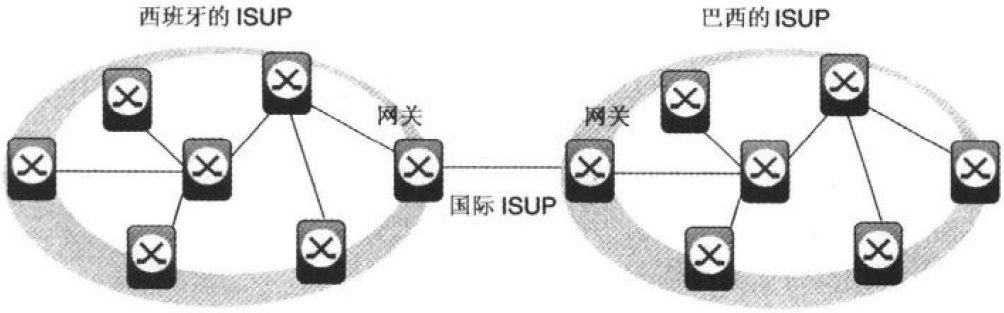


图 1-18　网关在具有国家特色的 ISUP 和国际 ISUP 之间执行传输

这种方案的缺点是国际 ISUP 的性能相当的低，限制了向连接在同一个网络中的用户的网络服务。例如，西班牙的用户必须在一套可用的呼叫服务间选择，因为这些呼叫使用的信令协议是西班牙的 ISUP。巴西的用户又有另外一套适合本国呼叫的服务。但是服务不能扩展到这两个国家间的呼叫，因为这两种 ISUP 特色之间没有共同的特性。

1．辅助服务

ISUP 具有的功能可以分成三组：基本服务、电路管理和辅助服务。基本服务包括提供语音服务的基本呼叫建立。

ISUP 电路管理功能包括阻塞电路和非阻塞电路，建立、释放和测试电路交换路径。用于建立呼叫的相同协议也用于管理网络资源。

辅助服务更加多样化，例如呼叫等待、呼叫转移和召开会议。这些辅助服务很有用，并且许多用户都在使用它。但是获得这些辅助服务是相当不容易的。因为为使用这些服务功能设置的接口通常是电话机的辅助键，用户不得不学习使用不同的代码来获得不同的服务。如果这个服务不经常使用，记住它的代码就会变得很困难。一个典型的服务请求的例子是：键入“*22#电话号码#”以开启呼叫转移。它非常复杂，因为用户几乎每次发送前可能都需要查找这些过程。我们可以看到，引用更好的用户接口不仅可以使用户更频繁地使用这些服务功能，而且也会促使这些协议更快地发展，因为用户能以一种更友好的方法去使用新引入的特性。

ISUP 辅助服务以一种分布式的方式实现。当与用户相连接的本地交换机升级到支持新特性时，用户就获得了使用新服务的权利。因此，服务的获得依赖于本地交换机和它的业务计划。此外，并不是所有网络中的用户都能获得同样的服务，因为这个网络中的交换机不可能

在同一时刻升级，并且具有同样版本的软件和同样的硬件。甚至用户可能会执行不同的操作来获得同样的服务，这取决于他们请求服务的交换机。

2．智能网络（IN）服务

服务提供商考虑到 ISUP 的局限性，因而促进了利用智能网络应用协议（INAP，Intelligent Network Application Protocol）的智能网络（IN，Intelligent Network）服务的发展。虽然 ISUP 服务是分布式的，IN 服务却集中于一个中心位置节点。这些称为服务控制点（SCP，Service Control Point）的节点能够被网络中的其他用户所访问，因此供应商只需升级这个节点就能够提供一种新的服务——这是服务管理的一个巨大进步。服务是在 SCP 中使用脚本（Script）实现的——这些脚本就是一组包含特殊服务逻辑电路的规则和指令。脚本分成两种类型：系统脚本和组脚本。系统脚本执行号码分析来响应用户键入的号码，然后调用正确的组脚本去执行实际的服务逻辑电路。

分布在整个网络中的是大量被称为服务交换点（SSP，Service Switching Point）的 IN 服务触发器。SSP 分析用户键入的号码，并决定是否需要提供智能网络服务，它然后与 SCP 联系，由 SCP 决定调用提供服务所需要的操作。

1.3.9 SS7 之后的模式

我们已经接触到了 PSTN 中的信令是如何从它的模拟原型向前发展的。现在来分析电话网络背后的结构原理，以及它们和用于 Internet 的结构有什么不同。这种比较将需要很长的篇幅，以解释为什么 SIP 是一个真正的变革而不是对已有的信令系统所进行的改进。

PSTN 和 Internet 之间在体系结构上有两点不同：

（1）在 PSTN 中，智能集中于网络而不是终端上；

（2）访问的网络和提供的服务紧密相关。

1．网络中的智能

SS7 将终端和网络进行了明显的区分，它在网络（干线信令）中执行操作，而其他的协议，像 DSS-1，要连接网络和它的终端。SS7 建议了一个具有有限响应性的、能够使用非常简单终端的体系结构。在 Internet 中，情况与此相反。智能被推入了最终用户的设备，而网络则尽可能保持简单。在后面的章节我们将会看到，这就可以更快速、更灵活地实现新的服务。

执行 Internet 模式的逻辑，一个信令应该支持智能终端系统而不是支持一个依据用户利益构建的网络。就像我们已经看到的那样，在 PSTN 中的协议不适合这种规范。一些使用 SS7 的系统（比如 GSM）类似于主/从体系结构，它网络中的一个节点——移动交换中心（MSC，Mobile Switching Center）——向 GSM 终端发送命令，GSM 终端无条件地执行这些命令。这种体系结构导致了这样一个网络的产生，它作出什么是用户需要的假定，而不是让用户选择自己需要的功能。当在这样的一个网络中实现端到端服务时，有时也会产生无意识的影响。

例如，考虑一个在两个 GSM 终端之间的呼叫（如图 1-19 所示）。GSM 终端使用 GSM

编码译码器来译码语音。MSC 代表终端处理呼叫者终端的决定，它需要从 GSM 编码译码器到 PCM（在固定的 PSTN 中使用的编码译码器）进行代码转换。当这个呼叫到达处理被呼叫者终端的 MSC，并在最后把这个语音传送到终端之前，执行相反的代码转换工作：从 PCM 到 GSM。然而，代码转换会显著地降低语音质量，而这种情况下执行了两次代码转换，所以这种影响变得尤为明显。

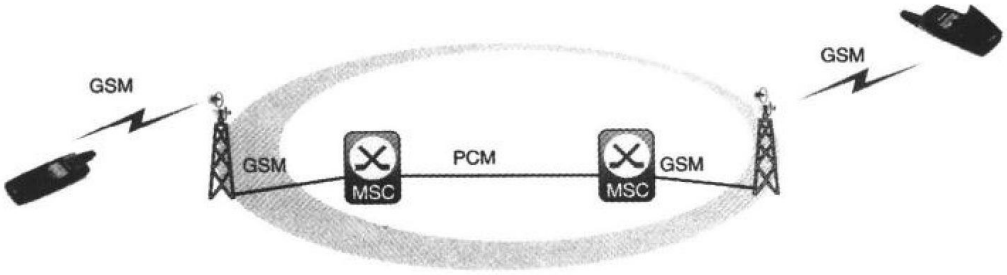


图 1-19　两个 GSM 移动终端间的呼叫

这个例子清楚地显示了使用端到端协议（它使终端就使用的编码译码器进行协商成为可能）能很好地执行端到端服务，避免了我们刚讨论过的情况的出现。

2．可靠性

因为 PSTN 将系统智能集中在一些网络节点上，如果这些节点之中的一个失效，就意味着对用户的服务会中断。因此，网络节点必须是高度可靠的。像处理器冗余这样的技术可以确保某个确定的网络节点不失效。但是，在一个重要的网络中要保证节点决不（几乎不）失效将会极大地增加设备的造价。

我们将会看到在 Internet 中，因为智能放在了终端，网络中的一个节点（比如一个路由器）失效并不危急，其它的节点可以接管它的工作。在这里，可靠性通过实现端到端的服务获得，在这个端到端的服务中，一个网络失效不会导致服务所必须的状态信息丢失。只要终端系统不崩溃，就能继续提供服务。

3．访问同服务设备紧密相连

我们已经看到这些年 PSTN 是如何发展的，以及为什么它的发展同用户平面紧密相连。当新的用户平面内的多路复用技术或传输机制产生时，信令协议的发展转向利用这些新的机制。传统上，新的信令协议的发展是为了传输新的用户平面的信息。

信令协议详细地规定了能与它们联合使用的用户平面的类型。例如，国际 ISUP 定义了 3 种类型的用户平面：64kbit/s 无限制用户平面、语音用户平面和 3.1kHz 音频用户平面。

这是因为在 PSTN 中，有权使用网络和服务设备是等同的。因而，如果一个用户访问 PSTN，他（或她）能获得的服务是用于访问的协议提供的。例如，如果使用 DSS-1，呼叫者会被提供一个语音信道。他（或她）就不能访问 PSTN 并请求其他类型的服务。

在下一章将会看到，在 Internet 环境中，访问和服务设备完全分离了。一个用户能够（并

且经常这样做）通过一个特殊的 Internet 服务供应商（ISP，Internet Service Provider）来连接 Internet，并能通过一个 Internet 入口获得 E-mail 服务。在这里，访问提供者和服务提供者是不同的。

就像可能想像到的，Internet 促进了一个不同于 PSTN 设计信令协议的方法。我们将会看到 SIP 不限制可建立的会话类型的数量。一个呼叫者可以使用 SIP 去建立一个 VoIP 会话，但他也同样能使用 SIP 建立其他的会话，比如，建立一个游戏会话。SIP 甚至能够用于建立一个传统的电话呼叫。因而，Internet 协议实现了一个比 PSTN 协议更加模块化的方法。

1.4 小　　结

我们已经看到 PSTN 在网络而不是在终端中实现了系统的智能。另外，与呼叫相关的信令的建立与用户平面的管理紧密相连。这使得协议负担过重，并在使用另一种类型的网络时缺少通用性。

考虑到上面这些情况，我们推断 SS7 之后的模式不适合于为获得最大的灵活性而设计的 IP 网络。SS7 是电路交换网络的优秀执行者，但它不能利用 IP 网络提供足够的灵活性。因而，在 IP 环境中使用的信令协议在设计时要紧记着 IP 模式，我们将会看到 SIP 非常适合它。

第 2 章　分组交换、IP 和 IETF

在这一章，我们将要介绍如何构建 Internet、分组交换的工作原理以及 Internet 协议的设计方法。这些内容是理解会话初始化协议（SIP，Session Initiation Protocol）工作方法及原理的关键。

另外，熟悉 Internet 工程任务组（IETF，Internet Engineering Task Force）是怎样工作的，并强调成熟的 SIP 和它的扩展真正是什么，这对于基本 IP 知识的普及（了解每一个 Internet 协议在成为 Internet 标准前所必须遵循的处理过程）也是十分重要的。

2.1　分 组 交 换

在第 1 章中读者已经看到，电路交换最初是在模拟传输的基础上发展起来的。电路交换提供了一个从源端到目的端的适合传输连续模拟信号（比如语音）的专用路径。频分多路复用（FDM，Frequency Division Multiplexing）是电路交换模拟网络中使用最广泛的多路复用技术。

数字技术的出现使得电路交换得以发展。信息在系统间以二进制的形式（即 0 和 1 的形式）传输。网络节点能够容易地存储和转发数字数据而不降低数据的质量。这种能力使得使用多路复用技术，例如时分多路复用（TDM，Time Division Multiplexing）比 FDM 更有效。

然而，即使是 TDM 系统，使用网络资源的效率也相当低：语音传输不能使用一个电路中的所有可用频段，并且这些频段也不能够再分配以用于其他目的。

仿佛这些还不够，数据通信带来了新的相关的通信模式。数据流具有突发性和非均匀性的特点。终端不是连续地传输，在大部分时间却是空闲的并且在某些点上却非常繁忙。数据率在整个连接期间也不是保持不变，而是变化非常剧烈。一个特定的数据传输有一个峰值数据率，平均数据率与它相关，而且它们通常是不相同的。

如图 2-1 所示，使用专用电路传输具有上述特点的信息流是一种资源的浪费。图中的电路连续地提供 64kbit/s，但是系统通信需要的数据率很少达到 64kbit/s，它通常低于这个值，而有时却又高于 64kbit/s。图中也显示了在某一时刻系统没有使用的可用容量被完全丢失了。

分组交换首先设计成用于满足出现突发数据流时的需要。虽然第一个关于分组交换的论文出现在 1961 年，但第一个分组交换节点直到 20 世纪 60 年代末才得以实现。在分组交换中，被传送数据的内容决定网络交换如何执行。源节点把将要传送的信息分成称为分组（*Packet*）的片，然后将每一个分组路由到达它的目的地所需的信息被附加在这个分组中，形成了分组

头（Packet Header）。分组被发送到了第 1 个网络节点——在分组交换中，网络节点称为路由器（Router）。如图 2-2 所示为一个由路由器形成的网络。当这个路由器收到了某个分组时，它检查这个分组头，然后将这个分组转递到下一个适合的路由器上。这个查寻分组头的过程在这个路径中的每一个路由器上都要执行，直到分组到达它的目的地为止。到达目的地后，目的地终端剥去这个分组头，就获得了发自源端的真正的数据。

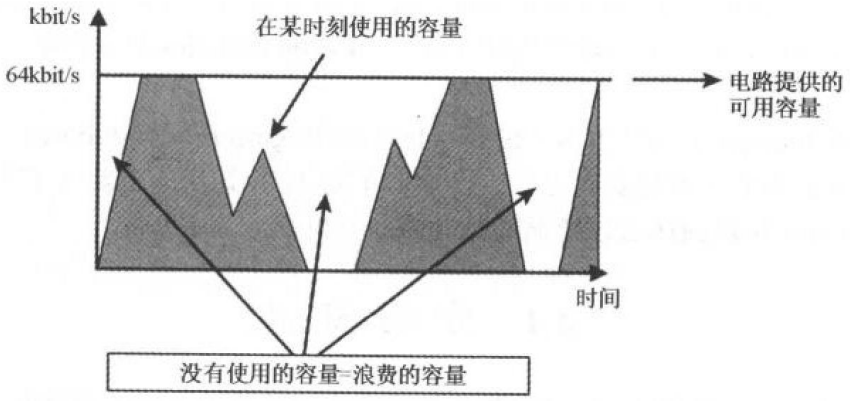


图 2-1　网络容量的低效使用

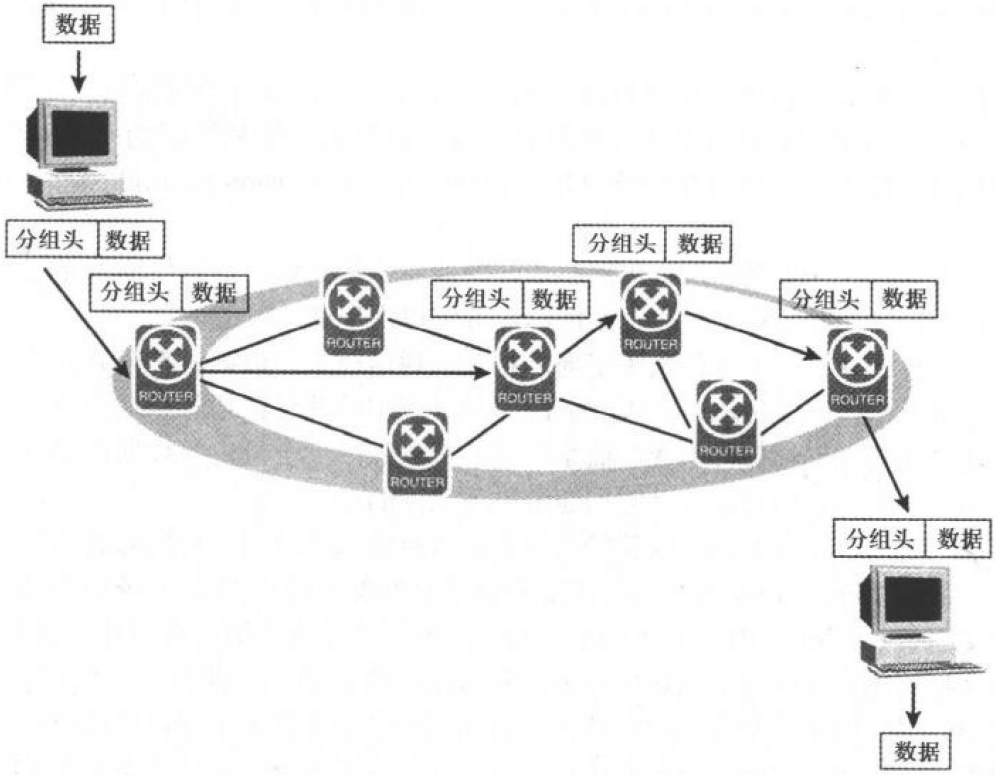


图 2-2　网络中的路由器

1．数据报

在一个分组交换网络中，可以用数据报（Datagram）或虚拟电路执行路由。在使用数据报的网络中，进行路由的决断只取决于将要被路由的分组的头部。网络将检查分组头并将分组发送到合适的下一跳（Hop）。发送到相同目的地的不同分组可能有穿过网络的不同路由，这可以使负载保持平衡（Load Balancing）。通过选择路径，分组可以避免网络中出现拥塞点。然而，分组经过不同的路由将会产生不同的延迟，并且就完全可能出现下面的情况：比分组＃1 晚发送的分组＃5 路由快，而比分组＃1 先到达。目的地对分组无序到达的补救措施是在将这些分组发送给用户以前对它们重新进行排序。

如图 2-3 所示为数据报在网络中运转的方式。从 A 传播到 B 的数据报向 B 路由是基于每一个数据报的分组头。我们已经认识到了一个特定的路由器可能使用可选择的路径向 B 路由数据报。这个就会产生如下的结果：到达 B 的数据报穿过了不同的路径，并因而可能会产生不同的延迟。

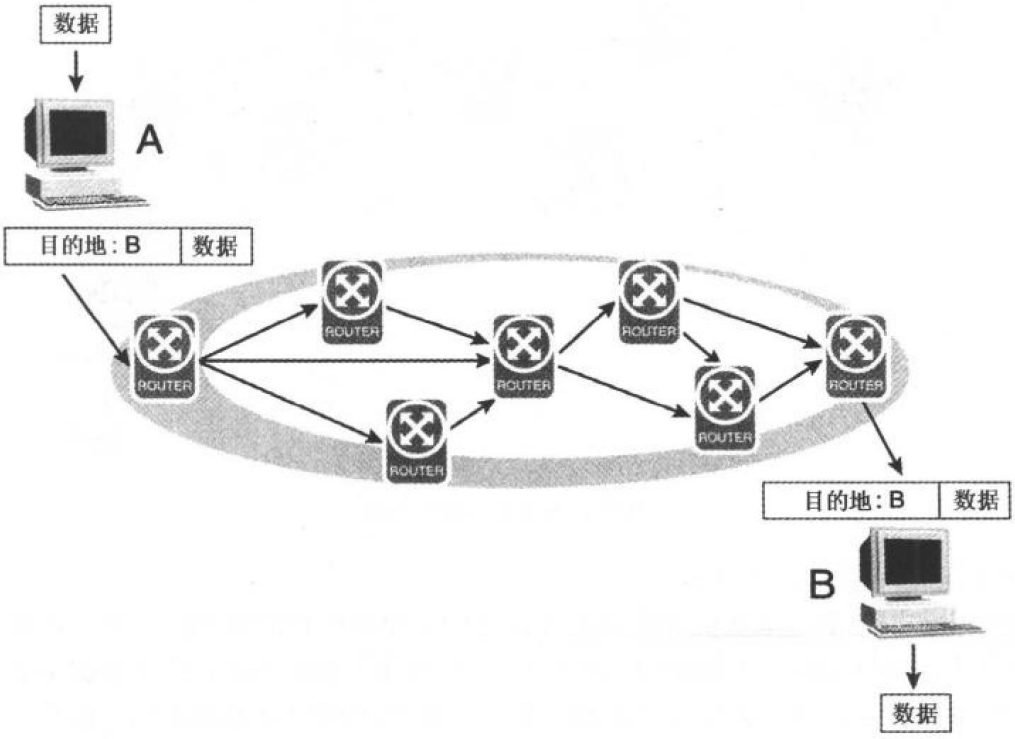


图 2-3　使用数据报的网络

2．虚拟电路

虚拟电路（Virtual Circuit）看起来更像电路交换。路由决定取决于将要被路由的分组的分组头和经过这个路由器的前面的分组。

在传输任何数据以前，源端向网络发送一个建立虚拟电路的请求。网络然后将包含这个

请求的分组路由到目的端，并向这个路径分配一个虚拟电路标识符。目的端接收到这个分组后，通过向源端发送一个确认分组来确认虚拟电路的建立。一旦虚拟电路建立了，所有将要传输的运载数据的分组都通过这同一个路径路由。路径中的分组头中合并了前面分配的虚拟电路标识符，路由器就依照它们转发分组。

如图 2-4 所示为一个基于虚拟电路的网络从 A 到 B 转发分组。所有的分组沿着第一个分组的路径传输。

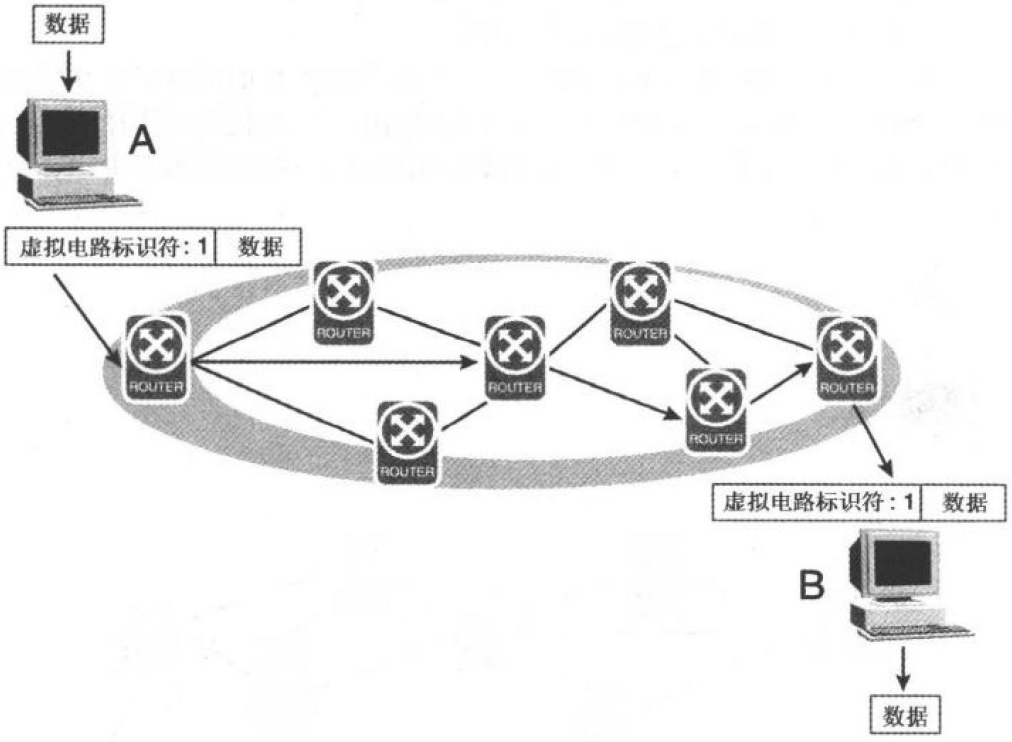


图 2-4　使用虚拟电路的网络

3．数据报和虚拟电路的差别

数据报和虚拟电路间最重要的差别也许是它们需要网络存储内容的不同。数据报路由不需要在网络中存储状态信息（除路由表以外）。一旦做出了路由决定，路由器就不存储这个结果以便于作后续决定，从这种意义上来说，处理数据报的网络是无状态的。相反，处理虚拟电路的网络必须存储虚拟电路的状态和标识符，以使得接下来的分组能匹配正确的虚拟电路。因此，虚拟电路网络在网络中比数据报具有更大的智能程度。

这些分组路由的方法都存在优点和缺点。数据报在传输用户数据之前不需要任何路径建立时间；而虚拟电路在发送任何用户数据之前不得不发送请求并建立虚拟电路。当一个终端希望向大量不同的目的地发送分组时，每使用一个新的目的地地址就会出现一次建立延迟（Establishment Delay）。在这种情况下，数据报则具有更大的灵活性。

另一方面，一旦虚拟电路建立了，通常它的总开销要低于数据报，因为头部携带的内容少——它们只需携带一个虚拟电路标识符就可以了。因而对于长程传输，网络使用虚拟电路是一个更好的选择。

数据报利用了网络中的动态路由：如果一个网络中的节点失效或者变得拥塞，分组能够立即选择路由以避免应力点，而不需要建立一个新的电路。负载平衡也成为可能：向着同一个目的地的数据报能够使用不同的路径，使得网络负载变得更均匀。虚拟电路也能处理一些负载平衡，但是当分组总是使用相同的路径时，再分配路径往往会使路由变得更加复杂。

2.1.1　分组交换的优势

具有有效性和适宜性是分组交换比电路交换更有前途的原因。它能迅速向网络返回没有使用的资源。分组交换也可以使不同的终端间的数据率匹配。路由器被设计用于对分组缓冲，直到它们在输出接口发送出去。因此，输入数据速率不需要与输出数据率相等。这使得终端可以使用完全不同的数据率进行通信。实际上，网络执行不同的数据率间的转化。

价格是另一个优势。通常，分组交换设备因为简单而比电路交换装置成本更低。一个路由器基本上就是一台计算机以及几个与它相连的网络接口。它从一个接口收到分组，然后分析分组头，最后把分组通过合适的输出接口转发出去。最近几年，计算机价格的下降使分组交换节点直接受益。

2.1.2　分组交换的弱点

前面已经提到，在某种情况下，分组交换不能像电路交换那样好地执行传输，而会产生更多延迟并使总开销更高。电路交换节点根本不检查要传输信息的内容，而分组交换路由器在转发分组之前要检查每个分组头。这个过程当然比仅仅从一个时隙到另一个时隙交换信息的 TDM 网络花费的时间要长。另外，将要通过路由器中的某个接口转发的分组必须在这个接口的输出队列中排队。在那里，分组必须等待，直到它前面的分组转发完毕。当网络负担非常重时，路由器队列通常是满的，就会出现这样的情况：分组不得不在它所经过的每一个节点中排队。虽然也可以采取措施缓和引入的延迟，例如通过实现区分不同流量的不同优先等级，但这些延迟是分组交换所固有的，不能完全消除。全部分组头带来的总开销的不利有多少呢？分组头当然浪费网络资源，并且如果分组的有效载荷足够小，分组头就可能会超过有效载荷。例如，用一个 20 字节的分组头传输 5 字节的用户数据，网络就必须传输 25 字节的分组，而其中只有 5 字节的用户数据。

2.1.3　X.25

在 20 世纪 70 年代，电信运营商发展了基于虚拟电路的分组交换网络。这些网络具有许多与公共交换电话网（PSTN，Public Switched Telephone Network）和智能网络类似的性质。X.25 规范定义了终端和网络间的接口（用户到网络接口）的技术标准，而 X.75 规范则对网

络中节点间的接口（网络到网络接口）进行了定义。实现了这些规范的网络统称为 X.25 网络。

X.25 网络向终端提供了高功能性。网络节点通过交换应答消息来执行错误检测，以确信通过这个链路传送的信息没有被破坏。终端只需释放数据；网络执行流量和错误控制，直到数据被安全地发送出去。就像设计电话网络所要求的那样，提高网络的功能性，使得在其中使用简单、可靠的终端变为可能。

2.2 IP 和 Internet 模式

X.25 网络也显现出了一些弱点。如果网络超载执行一些无关的任务（例如流量控制）会降低它的性能。因而，已实现的使用 PSTN 模式的分组交换网络没有利用这项技术所能提供的所有功能。接替它的模式应该能够最好地使用最新发展的技术。这个新模式，当然就是 Internet 模式，并且它的主要协议是 Internet 协议（IP，Internet Protocol）。

2.2.1 IP 连通性

Internet 不同于其他网络之处在于：它唯一目的是提供连通性，而其他网络不是这样，例如，PSTN 的目的是提供电话服务；TV 网络的目的是提供广播。各种服务（例如：E-mail、WWW 服务、视频会议和文件传输等）都是基于端到端的 IP 连通性实现的，如图 2-5 所示。

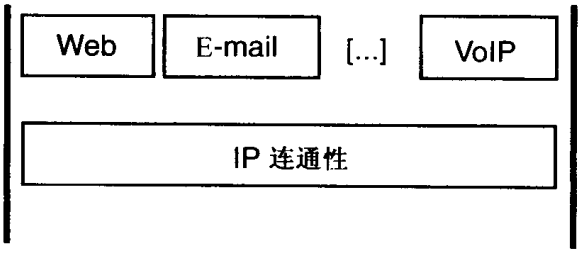


图 2-5 所有 Internet 服务都基于端到端 IP 连通性

为了具有真正的端到端的连通性，一个通用的端到端协议在网络层实现了——这就是 IP 协议[RFC 791]。在网络的所有系统中实现 IP 有两个优势：

（1）网络独立于底层技术；

（2）应用能够使用通用的 IP 基础结构。

从电信的观点来看，IP 基础结构差别相当大。特别地，使我们可以用不同的链路层技术而不管低层的不同类技术去构建一个在 IP 层的同类网络，从而实现网络连接（如图 2-6 所示）。这样做的另一个好处是，在网络层使用一个单独的协议，任何新的低层技术都能用来传输 IP 信息流。

当前，IP 信息流可以使用以下低层技术传播：异步传输模式（ATM，Asynchronous Transmission Mode）、帧中继、租用电路、光纤和以太网局域网（LAN，Local Area Network）等。在将来，IP 自然最有可能运行在新发展的高速数字技术上。

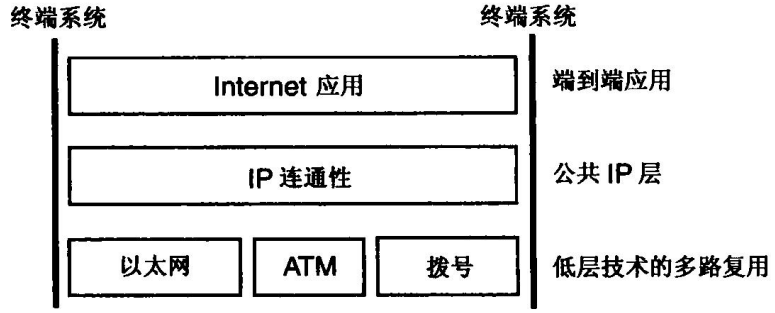


图 2-6　不同低层技术之上的端到端 IP 连通性

一旦通过在每个系统中使用 IP 而给予了端到端的连通性，IP 就变成了一个发展应用和服务的公共平台。因为有这么多的协议，开发者在能够较好地使用它们之前必须理解 IP 体系结构中的原理，这些原理总是围绕着一个值得研究的思路展开的。

2.2.2　增加终端系统的智能

一个 IP 网络是由一系列与仅提供数据报传输（数据传输在这里不可靠）的路由器组成的哑（Dumb）网络相连接的智能主机组成。IP 在网络层被路由器和主机使用，将智能移入终端系统，如图 2-7 所示。终端系统担负控制 IP 网络流量（包括端到端流量控制和错误检测）的职责。路由器网络仅仅需要尽可能有效地担负一个简单的任务：向终端网络目的地不可靠地发送数据报。终端系统仅仅知道端到端信息流的行为，网络不向终端系统发送任何指出分组是否已经被发送的通知。

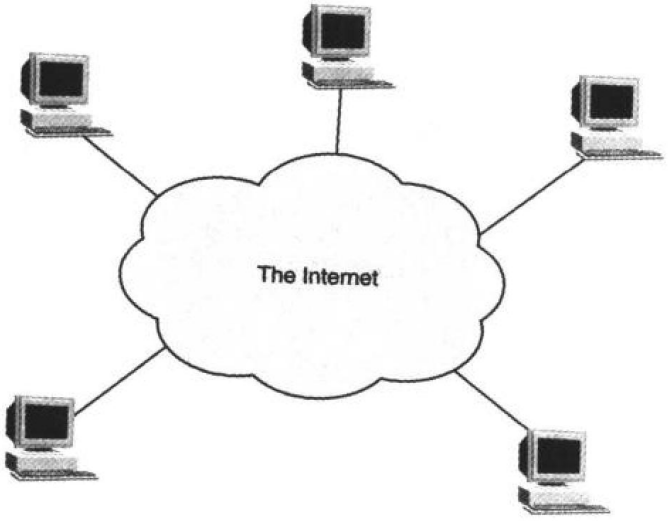


图 2-7　一个有智能网络终端系统的哑网

IP 网络不需要向目的地路由分组的状态信息，因而 IP 网络几乎总是无状态的。这种网络中的无状态使得节点失效的影响变得不那么显著，因为它们不再存储端到端通信必须的任

何信息。这代表了一种工程学上高可靠性的有趣方法。当一些地方的网络无论因为什么原因失效了，终端系统间的信息流只要简单地避开这些地区路由就可以了。其他的路由器能够接收以前被失效路由器处理的路由数据报的任务，因为在它们之间恢复路由不需要传输状态信息。因此，可以通过提供多条路径而不是试图实现故障安全的路由器（这是一个更困难的工作）来达到高利用率。

如图 2-8 所示为当 A 和 B 之间的一个路由器失效时的一系列过程和这两个节点间必须改道的流量。这个路由器的失效没有破坏两个终端系统间的信息流的流动。

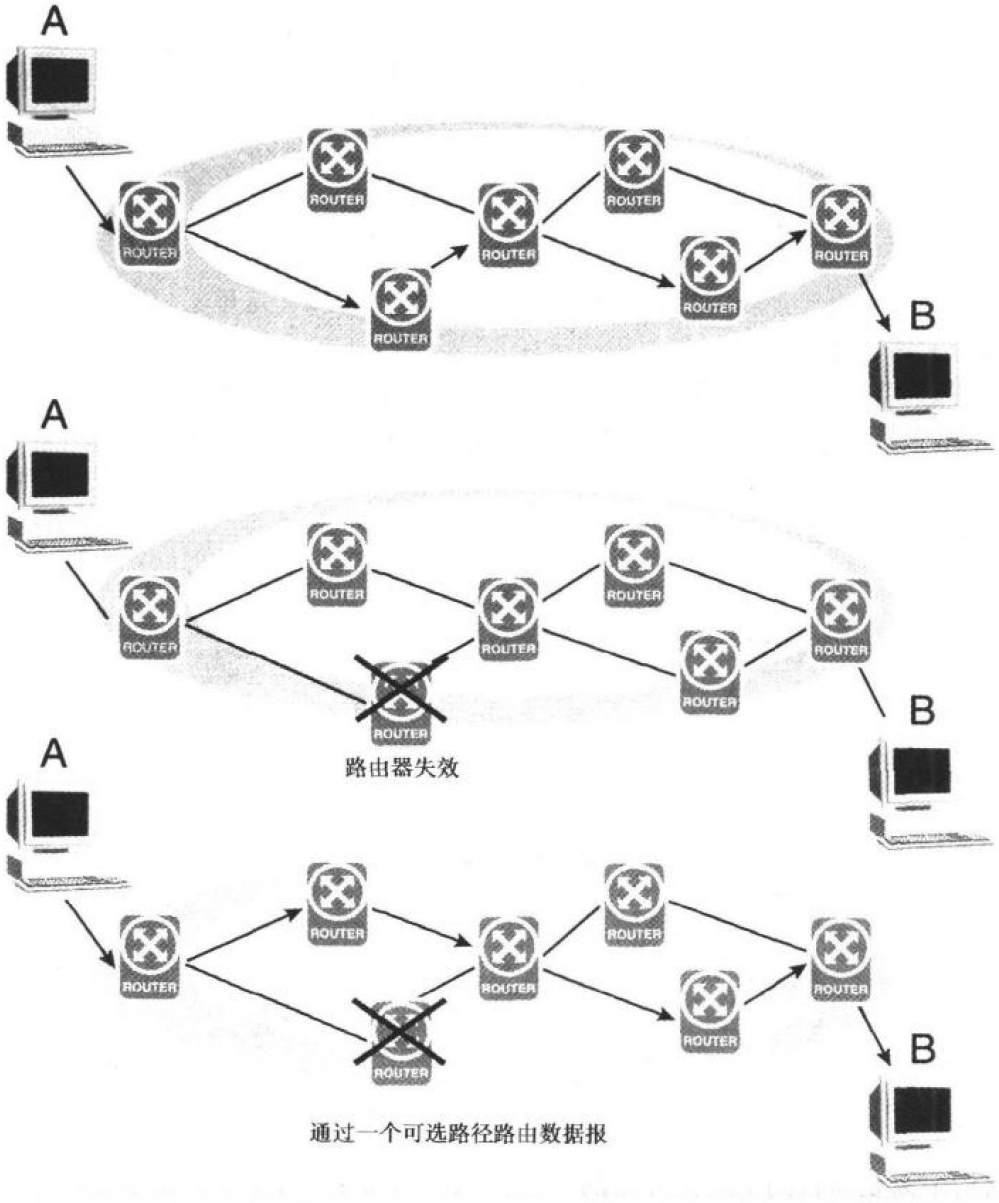


图 2-8　路由器失效

注意，虽然说 IP 网络不存储状态，但这并不意味着它们都是无状态的。为了能路由数据报，有些状态还是需要的。例如，当路由协议向不同的目的地分配新的路由时，路由器必须存储这些信息。另一个在网络中存储状态信息的例子是在低带宽点对点链路中存储的分组头压缩信息。链路的两端为了处理分组头的传输都存储了一些信息。例如当编码的语音通过一个音频接口传输时就会发生这种情况。但是，通常 IP 网络趋向于在网络中存储最少的状态信息以便使网络能更好地扩展和更健壮（Robust）。

2.2.3　端到端协议

一个对所有节点的公用 IP 层的出现并将智能集中在终端系统中，使得 IP 成为一个创建服务的极好平台，并且已经实现了基于 IP 的大范围的应用。应用的发展因研制者的不同而产生了许多不同的方法，但是为了充分利用 IP 作为一个平台所具备的全部特性，开发者经常被告诫要遵循某种通用的设计规则，这就构成了所谓的 IP 方法。

基于这种观点，我想强调，IP 和 Internet（最大的世界范围的 IP 网络）的重要性在于它的规则设计的过程和发展 Internet 标准的方法。IP 不仅仅是一个网络层的协议，也为发展新技术提供了一个全新的途径。

也许最著名的首要规则是试图通过实现智能终端系统和相关的哑网络来镜像 IP 的行为。这个实践鼓励了 IP 层以上（在这里可以更好地提供端到端的功能）端到端协议的设计。

通常，终端有系统中最好的功能以提供用户所希望的服务，网络不作不必要的关于什么是用户需要的假设，但它提供了在一个可清楚预知平面上的传输手段。使用上层端到端协议的另一个原因是端到端的安全——不依赖于网络提供的其他任何（易受攻击的或易犯错误的）机制的安全。为了这种方法的安全应用，终端间的信息流要终端自己加密和（或）签名。在信息加密的情形下，网络不能访问传输的内容。在已进行数字签名的情形下，除了终端以外，谁也不能更改信息。在这两种情况下，端对端协议仅仅意味着提供了端对端安全，而没有接触到每个应用所需的功能性。

2.2.4　一般设计问题

在 RFC 1958[RFC 1958]中，IETF 为协议设计提供了一系列的指导原则。这些指导原则表达的意图是提高 RFC（Request For Comment）协议文献的技术质量和节省标准化进程的时间。如果一个协议从一开始就正确设计的话，公众也就不需要给作者那么多的反馈，它在标准上也就能得到更有效地提高。

1．IETF 工具箱

遵从 IETF 的设计过程通常指的是一种自下而上的过程，以便同那种从一开始就先定义一个体系结构，然后再设计节点间更细小协议的过程区分开来。自下而上的过程提供了解决公众特殊需要的协议。一个具体的协议解决一个具体的问题，因而将所有的这些协议组合在一起就能被视为一个工具箱，如图 2-9 所示。它们能以不同的方法进行组合（并且经常这样

做），去解决下游更大的问题。

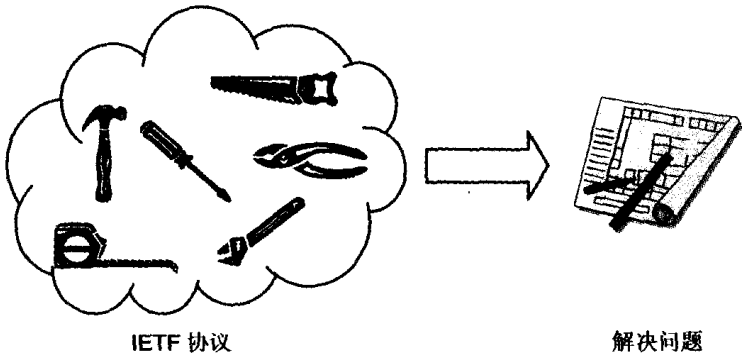


图 2-9　IETF 协议集像一个工具箱

这个过程的优势是：一个协议能变成在几种情况下使用的成分（如图 2-10 所示），它就是在 RFC 中宣传的特定的相干性（coherence）。因而，并不需要每研究一次新情况都设计一种新的协议。一种新协议，是在我们需要某种功能而又没有协议来提供它时出现的，但这个协议不需要变成一种新的解法。理想的新协议要解决将要或可能在未来的其他应用中出现的一个公共问题。

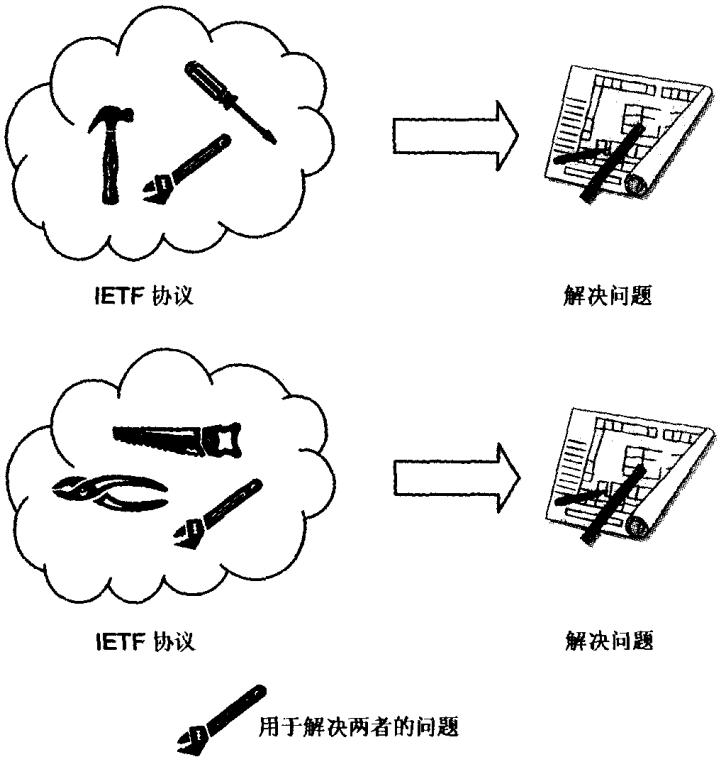


图 2-10　同一个 IETF 协议能用来解决不同的问题

例如：Web 服务需要鉴别要求访问某个 Web 页面用户的用户身份。IETF 会自动地产生对一个被许多种服务所使用的通用鉴别协议的方案，而不是对 Web 服务要求产生最直接的解决方法。用这种方法，尽可能地避免了协议在功能上的重复，增加了 Internet 工程技术的简洁性。IETF 设计的协议具有模块化风格。

2．IETF 协议特性

简单在一个协议设计的过程中是一个关键因素。简单的解决方法总是优于复杂的方法。有时为了得到或增加一个协议的功能特性要负出一定代价——使协议变得更复杂。在一定的条件下，引入一个新的复杂水平是不可避免的，但只要允许，选择简单的解决方案仍是规则。

因为所有来自 IETF 的协议都在 Internet 中使用，它们必定也面临着对可扩展性的迫切的要求。无论最初引出一个新协议的情形像什么，由此引起的协议在一个有数百万节点的巨大网络中应该是可用的。

健壮性是每个协议应该显现出来的另一个特性。IETF 实现协议健壮性的方法可用一句话来概括：发送时严格和接收时宽松。一个设备总是尽量遵从正确的协议句法和发送成型的信息，而不仔细检查一个设备从另一个设备上接收到信息的语法错误或句法问题。指导原则是任何信息只有能被理解才被处理，并坚信这个指导原则能完全使正确设备与错误设备交互工作。

例如，假定一个协议的特殊状态（一个包含时间戳（Timestamp）的特定信息）每 500ms 发送一次。一个正确的设备将要每 500ms 发送一次信息，而一个好的设备即使每 400ms 收到一个信息也能很好地工作。明显的，这个原则不是无止境灵活的，它的意思是指错误输入不影响系统的正确操作。

如图 2-11 所示是一个与此类似的图例，它是关于某人采用这个指导原则解决与某个不遵从严格英语语法的人的通信管理问题。

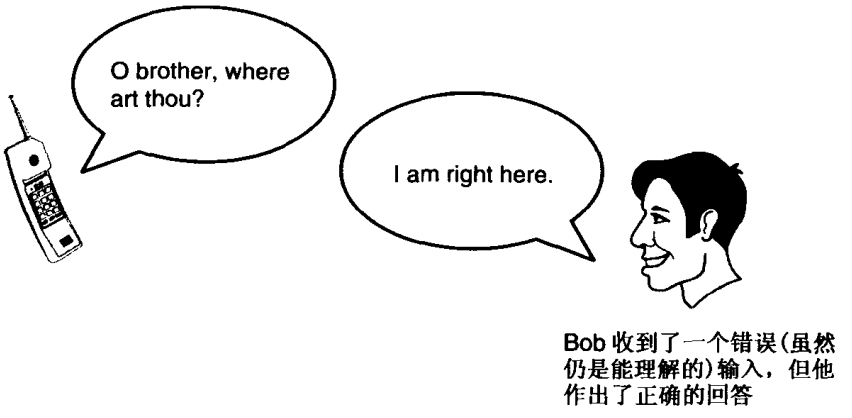


图 2-11　发送时严格和接收时宽松

为了理解所有的这些设计原则，我们需要简要地回顾一下 Internet 中新协议设计过程是

怎样发展成今天在 IETF 中使用的过程。下面讲述的 Internet 的简史就是为了这个目的。

2.3 Internet 协议开发过程史

Internet 的先驱——ARPANET，开始于对分组交换技术研究，它从 1964 年底开始进行工作，当时仅有 4 台相互连接的计算机。

从 20 世纪 70 年代初，ARPANET 使用网络控制协议（NCP，Network Control Protocol）进行主机对主机的传输。最早关于 IP 的工作也是在那时开始的，虽然 IP 不久就变成了 ARPANET 中网络层支配性的协议，但直到 1983 年 1 月才完成从 NCP 到 IP 的最终转化。今天，Internet 是世界上最大的 IP 网络。

2.3.1 RFC 的起源

伴随着其共享结果和思想的传统，ARPANET 在研究团体和学术团体中得到了发展。研究团体是一个产生快速发展的开放环境。ARPANET 发展的步伐极为迅速，这个团体很快发现学术出版物的模式（审阅和生产往往超过一年）因为太平庸而不能支持 Internet 样式的发展步伐和革新。

为了避免出论这种限制，这个团体在 1969 年制定了新的称为 RFC（Request For Comment）的文献。同样地，他们发布了 RFI（Request For Information）和 RFP（Request For Product）以便从厂家得到反馈信息。在 Internet 团体内的个人发布 RFC 以便从包含相同工作的其他人那里得到反馈和思路。这些信息能够被快速分发和自由获得。

这个想法是：在一个地方聚集某个特定 RFC 的所有反馈并公开地鉴定所有公开的问题。在进行了所有必须的讨论之后，达到了意见一致时，如果没有遇到连续的障碍，一个技术规范就形成了。

第一个 RFC 是工作在同地的研究者写出来的。后来，E-mail 的出现，使得 RFC 的作者变得更为分散。邮件列表在标准化进程中变得越来越重要，现在，RFC 的作者之间从没有见过面的现象是很普遍的。

这个故事有一个圆满的结局。ARPANET 的发展未受到抑制，所有这个团体中的个人都有权使用所有的协议规范和其他技术文献。有权使用这些文献的做法使得在已存在的应用和新兴的应用间容易进行互操作的新应用的迅速产生成为可能。

2.3.2 协作团体

基于这一点，ARPANET 发展的过程中需要进行一些协作。在 20 世纪 70 年代后期，一些团体为了这个目的松散地组成了国际协作委员会（ICB，International Cooperation Board）、Internet 研究小组（IRG，Internet Research Group）和 Internet 配置控制委员会（*ICCB*，*Internet* Configuration Control Board）。

这些团体在一定时期促进了 ARPANET 的发展，但不久后它们都被淹没了。那正是第一个任务组（Task Force）诞生的时候。每个任务组都由那些工作在特定专题领域内的人组成，并且每个任务组都设有一个主席，以对工作进展情况进行监督。

Internet 工作委员会（IAB，Internet Activities Board）接下来就诞生了。它的成员由各个任务组的所有主席组成。这些最早的任务组之一是 Internet 工程任务组（IETF，Internet Engineering Task Force）。如图 2-12 所示为 IAB 的初期结构。

在 20 世纪 80 年代中期，公众对 Internet 工程的巨大发展很感兴趣。越来越多的人参加 IETF 会议。作为响应，IETF 分成了许多工作组。每个工作组有一个或几个主席和一个预先确定的工作范围。这些工作组聚集在一起形成各领域工作组——每个领域由一个或几个领域主管管理。这些领域的全体主管形成了 Internet 工程指导小组（IESG，Internet Engineering Steering Group）。剩余的任务组演变为 Internet 研究任务组（IRTF，Internet Research Task Force）。

Internet 协会（ISOC，Internet Society）创建于 1991 年，用以管理它自身而不是其他任何特殊技术的标准化进程。在 1992 年，IAB 被重新命名为 Internet 体系结构委员会（Internet Architecture Board），于是形成了今天的 Internet 管理结构（如图 2-13 所示）。

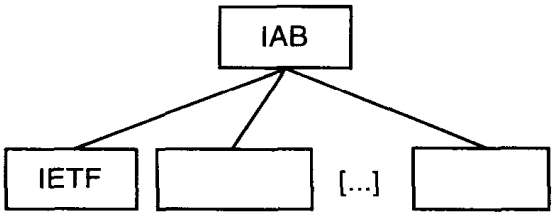


图 2-12　IAB 的初期结构

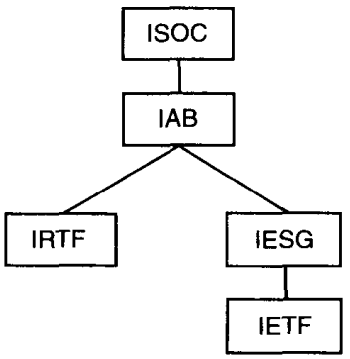


图 2-13　标准化团体

2.4　Internet 工程任务组（IETF）

Internet 标准化的过程[RFC 2026]总是在不断发展的。今天，Internet 标准在 IETF 中发展——IETF 成为一个关心 Internet 性能保护及相关技术的人们组成的开放性团体。IETF 下面分成了各个工作组（当前已经超过了 125 个工作组），指导实际的技术工作。

IETF 工作组通常只存在很短的时间，而 IRTF 中的研究组通常会长期存在。一个工作组集中在他们成立时定义的技术领域，当工作组解决了某个在它们授权中陈述的特定问题并令 IESG 满意（或者陷入僵局）后，这个工作组就解散。在本书中将要讨论的工作组的例子有：多方多媒体会话控制（MMUSIC，Multiparty Multimedia Session Control）、IP 电话技术（IPTEL，

IP telephony）和会话初始化协议（SIP，Session Initiation Protocol）工作组。工作组分属于 IETF 中的以下 9 个不同领域：

IETF 结构如图 2-14 所示。（1）应用（Application）、（2）通用（General）、（3）Internet、（4）操作和管理（Operation and management）、（5）路由（Routing）、（6）安全（Security）、（7）传输（Transport）、（8）用户服务（User service）和（9）子 IP（Sub-IP）。

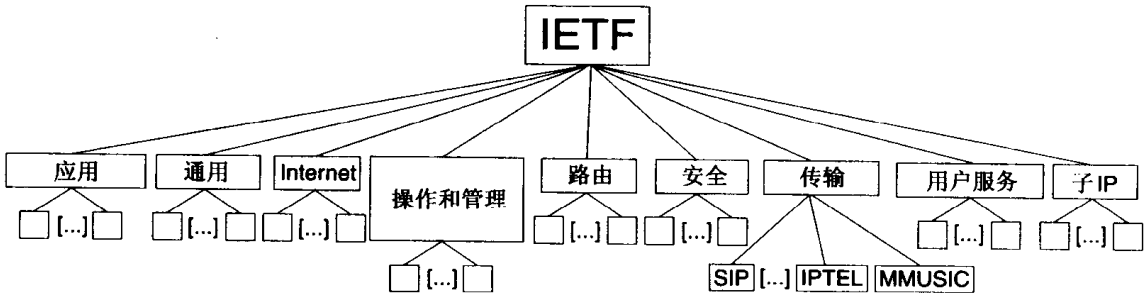


图 2-14　IETF 结构

2.4.1　Internet 工程指导小组（IESG）

由各领域主管和 IETF 主席组成的 IESG 审阅并通过各工作组提交的文档，然后授权使之成为标准。换句话说，IESG 是 IETF 的技术管理者。

IESG 也特许设立新的工作组。如果一个领域主管认为 IETF 已经足够关注并愿意成立团体来致力于某一提议的问题，就会将这一授权写入指定的工作组，并使之致力于这个任务的研究。授权中也包括提出这个工作组完成任务的时限。

有些课题也可能由 IETF 内的工作组机构之外的团体提出。当不清楚一个新的问题集是否会威胁 Internet，或是否要将它授权给一个工作组时，对此感兴趣的团体会召开一个 BOF（Bird Of Feather）会议。领域主管用 BOF 来研究即将提出的问题，并评估它们在这些团体中的重要性。

2.4.2　技术工作

每个工作组有一个或多个主席管理工作进展和工作程序。每个工作组也会创建一个官方的邮件列表，用于公众探讨与本工作组相关的问题。使用这个邮件列表是免费的和无限制的，并根据所报的数据维护存档文件。至今，一个工作组所作的大部分工作是通过邮件列表完成的。指导关于未解决问题的讨论、在程序上达成一致、提出已发展了的技术上一般性问题，这就是一个工作组的职责。工作组根据多数赞成的原则做决定。当只有个别人或个别团体不赞同某个事情时，占支配地位的观点获胜（怎样度量支配性地位是一个 IETF 正在争论的问题）。现在流行这么一个观点：作出一个决定比解决所有的不同观点更重要。

2.4.3　IETF 规范：RFC 和 I-D

IETF 以 RFC 系列出版物发表它的文献。但是，并不是 IETF 出版的所有文献都定义为标准。RFC 系列出版物有 3 个主要类型的规范：Internet 标准化规范、非标准化规范和 BCP（Best Current Practice）RFC。如图 2-15 所示为 RFC 系列的这 3 种规范。下面几节将要介绍每种类型的规范。

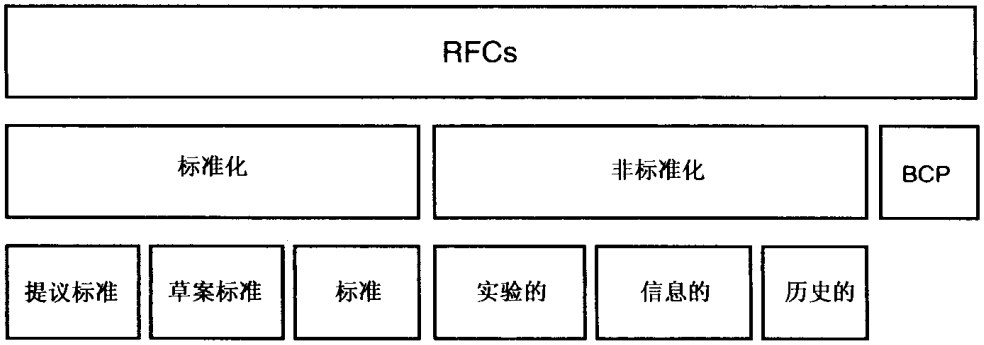


图 2-15　RFC 的类型

1．标准化 RFC

RFC 标准化的进程中要经历 3 个成熟的阶段：提议标准、草案标准和标准。当来自于工作组的规范被认为稳定和易于理解时，它们就会被分为提案标准这一类。在达到这一级别以前，提议标准必须经过兴趣小组的严格审议。虽然一些实现经验非常合乎需要，但它并不需要像一个规范那样正式地达到提议标准的状态。

对于草案标准来说就不是这样了。为了达到下一个成熟阶段，一个 RFC 必须指出至少两个在现实世界中能够互操作的实现。这些实现必须包含这个规范描述的所有特性，以便能从这个规范中移除非生产性的特性。也需要适当的时间以确保进行充分的讨论和审议。在变成一个草案标准以前，一个 RFC 必须在提案标准阶段至少停留 6 个月。

一个进行标准化的规范必须最终满足两个主要要求：大多数同意和运行代码。这两个要求是 IETF 进程中的关键点。

标准化的最后一个成熟阶段是 Internet 标准（STD，standard）阶段。只有当大量的实现和操作经验在这个团体中显现出来以后，这个规范才能成为标准。Internet 标准存储在 STD 子分类中，在这里，新标准被分配一个 STD 号。

每个 IETF 规范有一个相联的 RFC 号，并且当某个规范依它的轨迹进入下一个成熟阶段时，它会立即被分配一个新的 RFC 号。例如，一个提案标准“HTTP 扩展：摘要访问验证”原来的编号为 RFC 2069，当它达到了草案标准阶段，它就变成了 RFC 2617“HTTP 验证：要素和摘要访问验证”。RFC 将文献归档并以这种方法保存每个协议整个发展周期的文件。

当一个规范达到了 Internet 标准阶段并且被包含进了 STD 子分类中时，它仍保留一个 RFC 号。例如，IP 协议在 RFC 791 中得到了定义，但是因为这个协议规范的状态是 Internet 标准，它也出现在 STD 子分类中：STD0005。一个规范至少要在草案标准阶段保持 4 个月才能到达 Internet 标准阶段；并且一个草案 RFC 变成 Internet 标准前至少要开一次 IETF 会议对之进行讨论。

2．非标准化和 BCP RFC

就像前面提出的那样，一些 IETF 规范根本就不定义任何标准。这些规范遵从非标准化道路。实验型（Experimental）RFC 显示研究和发展工作中的结论和推论。它们记录了致力于一些技术领域的实现者和设计者的经验。

信息型（Informational）RFC 用于向公众展示通用的信息。实验型和信息型 RFC 都不需要像标准化文献那样要经过谨慎的审议，因为它们不能合并成一个提案。当 RFC 变得陈旧而被一个新的 RFC 所代替，或者它们不再被应用时，老的 RFC 就变成了历史的（historic）RFC。

BCP RFC 对实践进行标准化并赋予策略和操作上的指导原则。BCP RFC 不分阶段，它的过程同一个提案标准类似。BCP RFC 存储在 BCP 子分类中，但仍保留它们的 RFC 号。例如，RFC 2026“Internet 标准过程”也是 BCP0009。

3．Internet 草案（Internet Draft，I-D）

Internet 草案（与 RFC 相反）是在工作组内部用以驱使和收集反馈信息的草案文献。它们在规范的更有决定性的版本到达 RFC 状态以前发布。Internet 草案的有效期至少为 6 个月或者在此之前出现了一个更新的草案版本。Internet 草案不定义标准文献，并且除了作为过程中的著作外，IETF 不鼓励以任何方式引用它们。注意，本书中引用了一些 Internet 草案，读者应明白 Internet 草案的局限性和非官方性。

Internet 草案也在标准化过程中使用。当一个提案标准 RFC 发布时，与之伴随的新的 Internet 草案也就产生了。从提案标准 RFC 达到草案标准 RFC 所需要的所有的更改和补充都在新草案中执行，最后的形式就变成了草案标准 RFC。SIP 协议的基本规范[RFC2543] 当前是提案标准，它达到成熟阶段是在 1999 年。从那以后，一个 Internet 草案显示了从 1999 年 3 月公布以来，它为变成一个规范后所作的所有的更改。这个 Internet 草案[draft-ietf-sip-rfc2543bis]将要成为未来 SIP 草案标准 RFC。

如图 2-16 所示为 IETF 规范的生命周期——从最早的 Internet 草案直到作为一个 Internet 标准而发布的规范。几个不同版本号的 Internet 草案被发表，直到规范达到一种特定的成熟阶段。在此时，这个规范作为一个 RFC 而出版了，同时一个新的 Internet 草案也公布出来用以收集反馈信息，以便于到达下一个成熟阶段。

值得一提的是，一些规范在到达提案标准成熟阶段并在一个新的 Internet 草案中得到反馈后，重新作为一个新的提案标准 RFC 发布而不是草案标准 RFC。这种情况主要发生在：原来已发布的规范已经做了重大的修改，因此需要一个新的 RFC 来代替它，但新的 RFC 因为不够成熟而又不能达到草案标准状态。

4．命名 Internet 草案

正像前面描述的那样，工作组的授权包含工作组同意致力于其中的可交付使用的一系列限制。有了这些可交付使用的条件，工作组定义了一系列的工作组条目（item）。当这个工作组发布了一个关于某个特定条目的草案时，它依据下面的形式来命名：draft-ietf-工作组名-版本号。

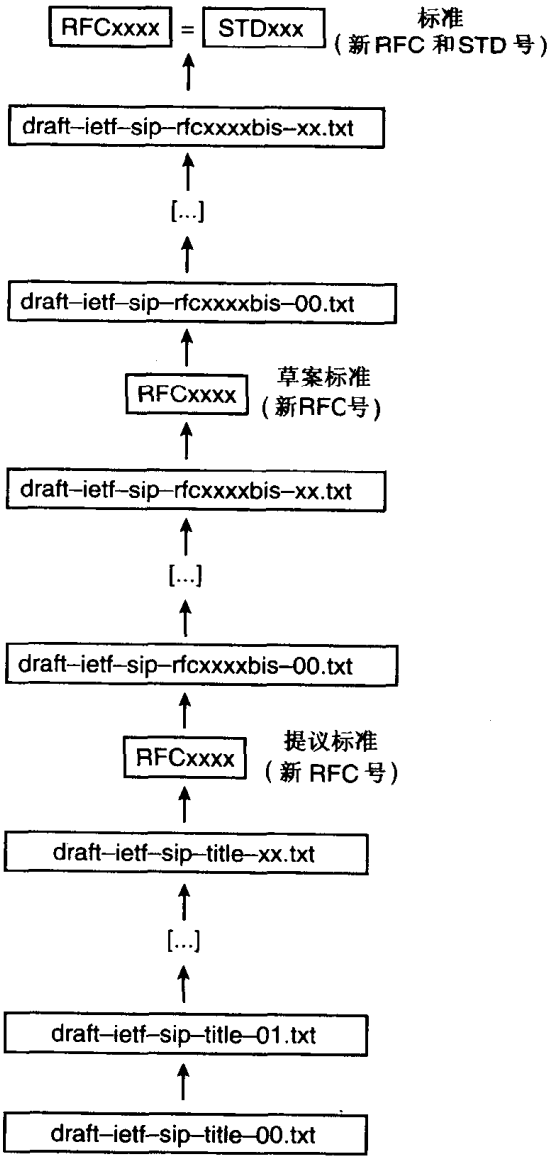


图 2-16　一个 Internet 的生命周期

命名习惯于把第一个发布的 Internet 草案版本号定为 00。因此，draft-ietf-sip-flows-*01*.txt 是属于 SIP 工作组的特殊的 Internet 草案的第二个版本（版本号 1）。

考虑到个人对一个工作组讨论的贡献，在命名时他们或者不直接涉及到工作组条目，或者是在一个工作组条目草案的初级版本上遵从一个稍有不同的命名惯例：包含作者的名字。例如：draft-schulzrinne-sip-911-01.txt。

最后，可以发现 Internet 草案和 RFC 的不同形式（Post-Script [PS]、Portable Document Format [PDF]和 txt），但是权威性的参考文献总是 ASCII 文本模式。它包括了所有的数字、表格、图例和包含在规范中的数据报。

图 2-16 概述了一个 Internet 标准的生命周期。一个新的思想（比如：一个新协议）首先出现在 draft-ietf-sip-title-00.txt 中。这个最早草案的发展，要归功于直到它到达标准（STD）成熟阶段以前团体所给予的反馈。

然而，并不是所有的草案都能到达如图 2-16 所示的标准化进程的末端。有许多草案到达提议状态，但只有很少提议 RFC 后来变成了草案标准。在变成了草案标准的 RFC 中，只有更少的 RFC 才能变成 Internet 标准。一个提议 RFC 在收集了反馈信息后，又被重新发布为一个提议 RFC（有一个新的 RFC 号）而不是一个草案标准 RFC，这是很常见的事。

在接下来的章节中，将要学习 IETF 规定的各种不同协议和协议扩展。记住，图 2-16 是很重要的，它可以帮助读者理解一个特定的规范目前处于哪个成熟阶段。

第 3 章　Internet 多媒体会议体系结构

既然已经约定 IETF 怎样工作并规定了由 IETF 组织所设计的所有协议都应具有的一般特征，对于其中的会话初始化协议（SIP），就应该注意其使用的环境。首先要弄明白在传输层中哪一种协议是用在 SIP 下，并且要知道这些协议能提供什么功能。

为了理解 SIP 在什么地方工作，将首先要分析推动 Internet 网络工程指导组（IESG）把 SIP 工作组放在第一位的原因。本章将重点讨论 Internet 网络多媒体会议体系结构。体系结构中包含一些协议，这些协议在 Internet 网络环境中共同提供多媒体服务。SIP 是这种体系结构中的一部分，这正是 SIP 的优点之一。因此，SIP 与其他协议灵活地交互并且利用它们的功能。也就是说，可以把 SIP 看成是这种体系结构中的工具包中的一个工具。

3.1　Internet 分层体系结构

前面已提到，Internet 的一个重要作用是它具有能够用来作为创建服务平台的适用性。IETF 坚持采用分层的方法来创建服务。Internet 网络分层模型包括 4 层，这 4 层处在物理层的上面，如图 3-1 所示，这也是我们常见的 TCP/IP 协议栈。

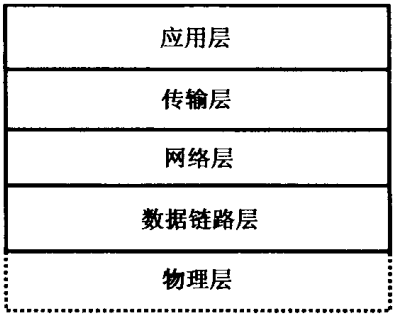


图 3-1　TCP/IP 协议栈的四个分层

传输层在普通 IP 层的上面实现，应用层利用各种不同的传输协议。为了提供某种特定服务，一种应用要选择实现服务的应用层协议。

假设我们要创建一个 Web 浏览器以使用户能够在网上冲浪和阅读 E-mail。为了建立这样一个浏览器，需要选择两种应用层协议，我们选择 HTTP［RFC 2068］进行网上冲浪，而用 IMAP[RFC2060]从服务器下载邮件。这两种协议都运行在 TCP 上面，当然也在传输层上面，如图 3-2 所示。

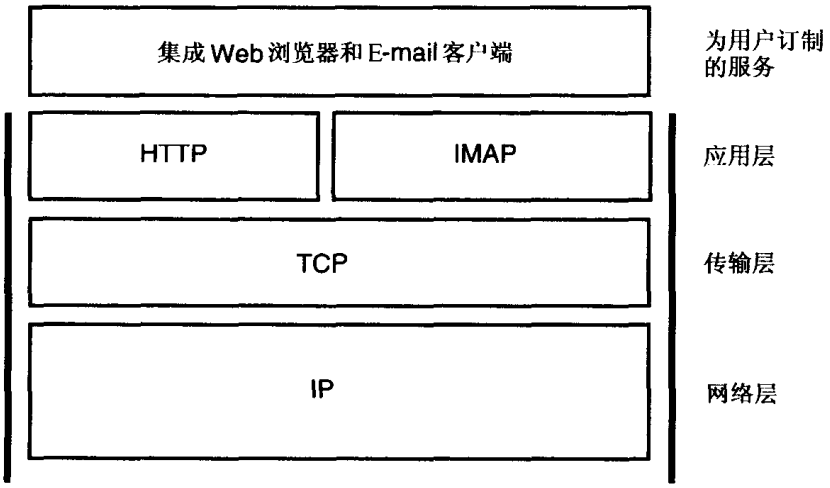


图 3-2　在分层体系结构上创建服务

3.1.1　传输层协议

网络为了提供各种服务必须利用应用层协议。反过来，应用层协议必须利用传输层协议，而传输层协议又运行在 IP 协议上。现在，有两种传输层协议得到广泛的应用：传输控制协议（TCP）[RFC 793]和用户数据报协议（UDP）[RFC 768]。

1．传输控制协议（TCP）

TCP 是可靠的传输协议，它按顺序在两台主机间传递字节流。它包含各种保证传输可靠的机制，如超时、重新传和顺序号等，这保证了接收者接收到的数据和发送者发送的数据一样。TCP 也执行流控制和错误检测。

因此，如果 Bob 通过 TCP 连接给 Laura 发送消息："Laura，你在干吗？"，TCP 会保证 Laura 接收到消息："Laura，你在干吗？"，因为 TCP 是端到端的协议，它通过观察端到端信息流的动态行为推断网络的状态。为了提供足够的信息，接收者将确认消息发送给发送者，这样才能进行错误检测和流控制。

TCP 能使端用户多路分解所接收到的 IP 分组，并将它们提供给不同的应用。因为到达主机的分组含有同样的目的 IP 地址，需要更进一步用标识符把到达的包与当前应用联系起来。这些标识符被指定一个 TCP 端口号，每一种基于 IP 的应用使用一个端口号。例如，带有 TCP 目的端口号为 80 的数据报由 HTTP 进行处理，而端口号为 23 的数据报由 Telnet[RFC 854]进行处理，如图 3-3 所示。

2．用户数据报协议（UDP）

UDP[RFC 768]是不可信赖的数据报分发协议，它不能确认所分发的数据报通过各种途径能否到达目的地。UDP 仅基于 UDP 端口号对 IP 流量进行多路分解。UDP 和 TCP 的端口号都是 16 位。通信流量进行多路分解后，UDP 提供检查机制，这种机制使目的主机能够检查

出所收到的数据报是否遭受网络破坏。

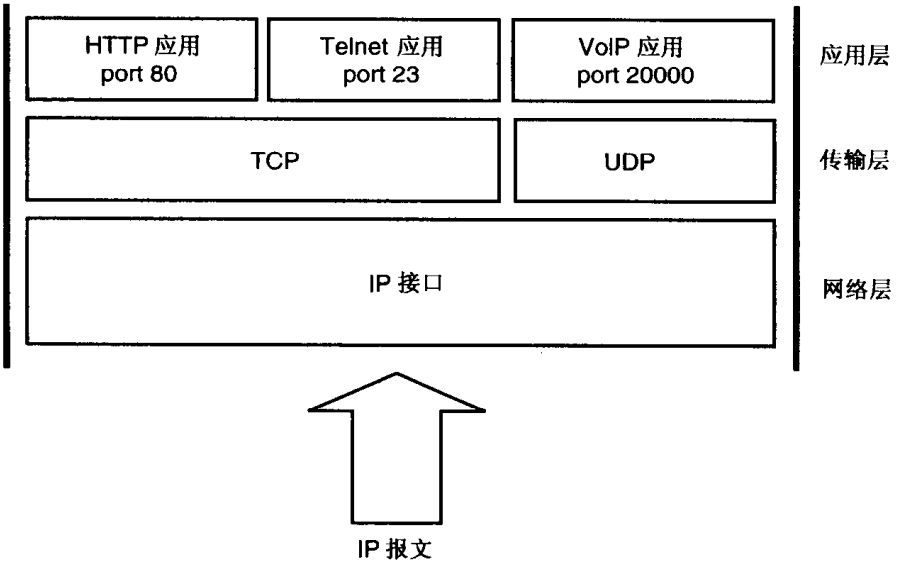


图 3-3　IP 分组分发给相应的应用

例如，如果 Bob 像上面一样通过 UDP 协议发送消息，他就不能确认是否 Laura 能够收到该消息。为了确认，必须要求 Laura 自己进行检查。如果含有 UDP 分组的数据报丢失了，Bob 必须亲自重发，UDP 本身没有提供重发这样的功能。

3.1.2　流控制传输协议

流控制传输协议（SCTP）[RFC 2960]是新发展起来的传输协议，可以预见到 SCTP 将会得到广泛使用，但从当前来看，它还没有 TCP 和 UDP 使用得广泛。

3.1.3　Internet 实时服务

用户服务是使用应用层协议实现的，两个常见的 Internet 服务是 Web 和 E-mail。Web 使用应用层协议 HTTP，而 E-mail 则使用其他的协议，如简单邮件传输协议（SMTP）[RFC 821]和 Internet 消息存取协议（IMAP）[RFC 2060]。

这些服务以及其他类似的服务促使了这些年 Internet 迅速发展。这些协议含有异步的信息交换功能。Internet 已被证明是发展此类异步服务的有力工具。但它也能提供同步（实时）服务，例如网络视频会议以及通过网络进行实况转播。实时服务对延迟很敏感。它们携带的信息分发到目的地有时间限制。如果由于网络导致的延迟时间过长，那么这个信息对接收者来说已失去作用或者服务质量会下降很多。

实时服务能够进一步分成流服务和交互服务两类，如图 3-4 所示。一般来说，流服务比交互服务的要求低。可以想象传输一场足球比赛作为流服务的例子，虽然相对地有较大延迟，

甚至延迟了许多秒，但只要图像和声音的质量得到保证，用户就可以接受；反之，如果图像和声音的质量很差，即使延迟很小也不行。业界普遍认为，如果在屏幕上看进球只比实地比赛的情况慢几秒，那么用户是可以接受的。

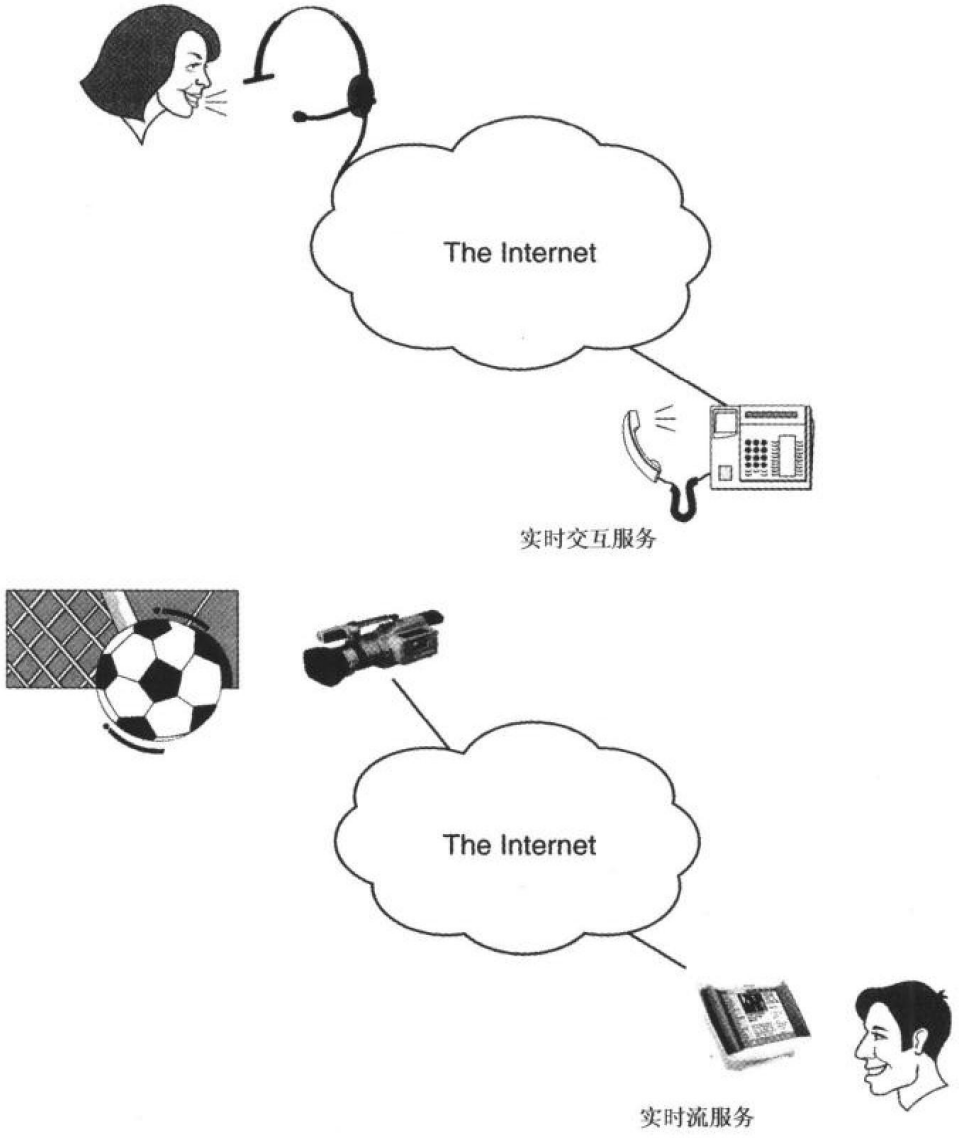


图 3-4　交互和流实时服务

带有交互方式的服务有严格的时间要求。在通过 Internet 进行对话的过程中，延迟必须保持很小，否则通话的双方就不能理解正常的对话。这种实时交互服务可能接受的最大延迟由很多因素决定，但一般来说，要低于流服务延迟。根据国际电信联盟电信标准化分会（ITU-T）的标准，语音会话中最大能够接受的延迟是：一个来回为 300ms。

网络多媒体会议体系结构

先进的实时服务包括几种媒体（如：视频会议包括视频流和音频流），同时它也被大家称作为多媒体服务。IETF 已为多媒体服务开发了几种具体协议。为了获得所需的功能，在应用中需要把这些协议中的一部分结合在一起。图 3-5（来自于[draft-ietf-mmusic-confarch]）展示了这些协议如何工作在一起。SIP 是多媒体会议体系结构中的一部分，实际情形如图 3-5 所示，如果你花点时间理解这种结构中每一种协议的作用，将会很好地了解：为什么需要 SIP 以及希望它能够提供什么功能。

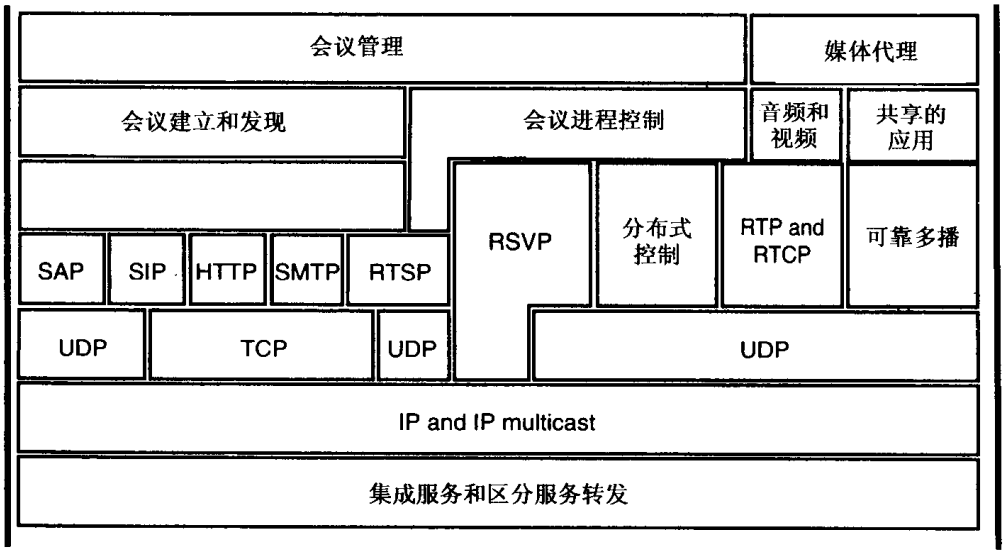


图 3-5　Internet 多媒体会议体系结构

3.2　多　　播

Internet 是为在两台主机之间能够尽力（Best-Effort）传送数据报而设计的。一个数据报中含有目的 IP 地址，网络中的路由器负责把数据报发送到目的地。一般来说，一个 IP 地址标识主机上的一个网络接口（如以太网卡）。只要一台主机作为发送者，另外一台主机作为接收者，这种机制能够让网络中任意两台主机进行双方的数据交换。接收者通过记录发送者的 IP 地址也能够将数据返回给发送者。这种路由方式，也就是一个 IP 地址标识一个主机接口，被称作为单播，如图 3-6 所示。

3.2.1　多址路由

当有两个以上的参与者要进行通信时，这种简单的情况就变得复杂了。在一个有 n 方参加的会议中，为了把 IP 分组发送到每一个主机，一个主机要传递 n-1 个 IP 分组，每一个 IP

分组中都含有指定接收数据主机的 IP 地址。

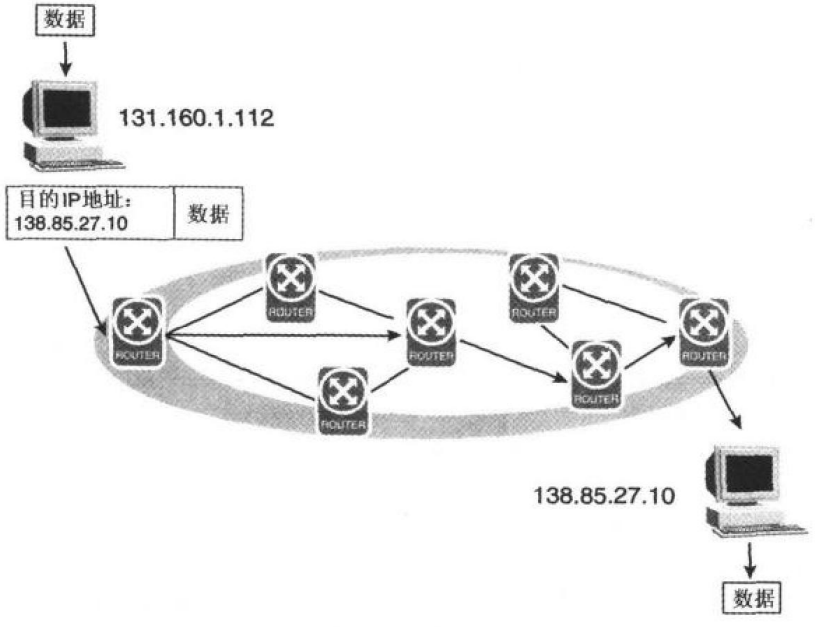


图 3-6 单址路由

很明显，对于大规模用户来说，这种数学似的系统不能很好地扩展。只要参与的主机增加一点，网络中的通信流量就会急剧增长，从而限制了主机的处理能力。每一个主机需要注册所有接收者的 IP 地址和建造同样信息的 *n*-1 个 IP 分组，而这些分组只是目的地址不同。在这些分组到达目的地之前，它们需要遍历许多相同的节点，需要网络中的链路多次发送相同的信息，如图 3-7 所示。

在图 3-7 中，从 131.160.1.112 到 138.85.27.10 发送的分组与从 131.160.1.112 到 153.88.251.19 的分组经历相同的路径到达路由器 R，因此，从 131.160.1.112 到 R 间的链接需要调用两次。

多播路由可通过对大规模的组提供多点到多点的通信来解决上述问题。这些路由器相互协作转发 IP 分组，而对于不同目的地址的分组被指定为同一个多播地址。一个多播地址代表一组主机，同一个组里的主机都愿意接收将被发送的 IP 分组。按照同样的标识，能够从一个多播地址接收到数据的主机也是一个特殊多播组里的成员。在这个网络里的任何一个主机能够将数据发送到多播地址，但为了接收到数据，它必须属于这个多播组。（在 IPv4[RFC 791]中，地址 224.0.0.0～239.255.255.255 是保留给多播用的）。

3.2.2 多播的优点

多播是可伸缩的，因为数据只需遍历链路一次，如图 3-8 所示。如果需要的话，多播路由器复制一份 IP 分组，并且尽可能的离多播路由器最近。

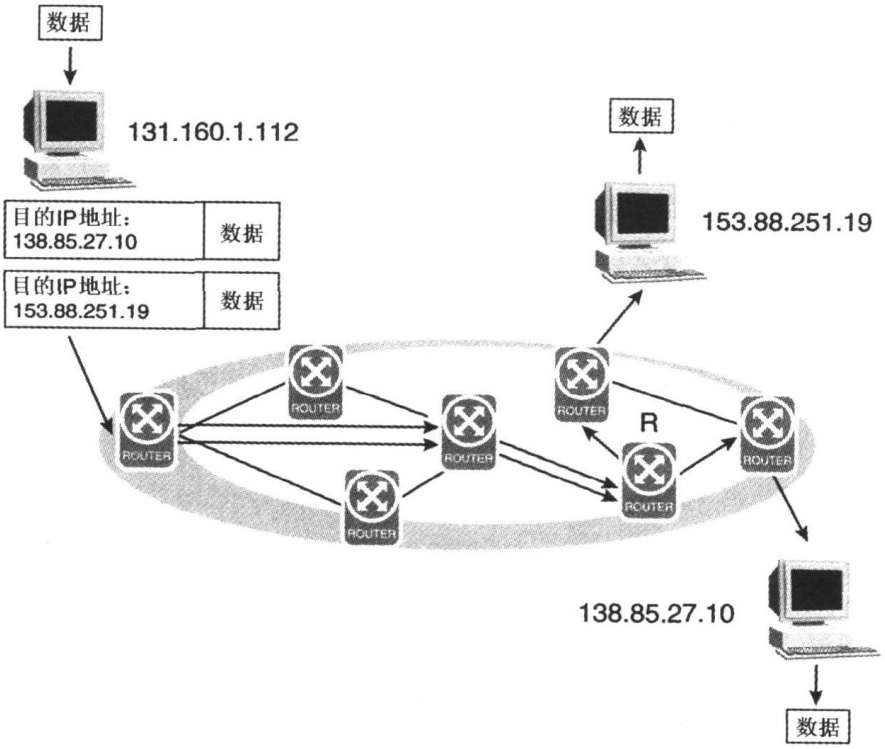


图 3-7　使用单播方式给不同主机发送相同数据

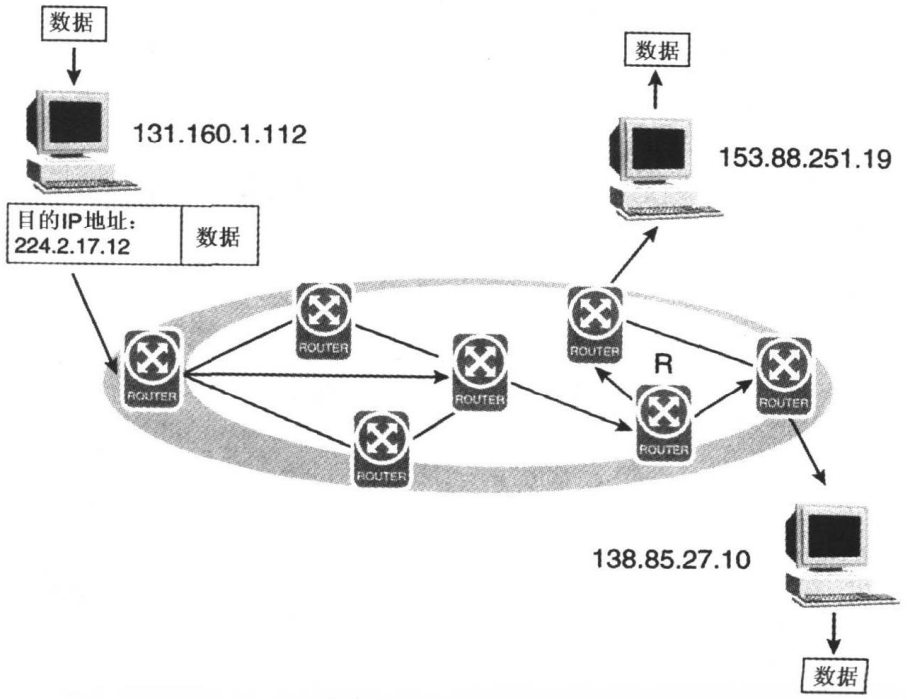


图 3-8　多播路由

图 3-8 中的配置与图 3-7 一样，仍然从 131.160.1.112 到 138.85.27.10、从 131.160.1.112 到 153.88.251.19 发送分组，但这次使用多播路由。携带相同的数据从 131.160.1.112 到路由器 R 仅需链接一次。为了把数据发送到 138.85.27.10 与 153.88.251.19，路由器 R 把数据复制成两份。这个图解例子说明了两个主要的优点：（1）131.160.1.112 到路由器 R 之间节省了一半的带宽；（2）如果只是发送数据，终端系统不必知道多播组的成员。

发送者把数据发送给多播地址，并不知道接收者的身份。只有离接收者最近的路由器才知道哪一个主机是多播组的成员。没有中心服务器跟踪多播组，所有的信息都在多播路由器中进行分发。

一个多播路由器所需要知道的是，使用路由器接口是否至少能够到达多播组的每一个成员。如果回答是“是”的话，数据将通过接口进行发送。

3.2.3 多播路由协议

多播路由器使用多播路由协议构建从发送者到接收者的分配树。这些协议可以分成两类：稀疏模式和密集模式。这两类协议使用不同的算法构建分配树。

1．密集模式多播路由协议

当大多数主机是多播组的成员时，密集模式多播路由协议能在网络中很好地工作。当主机接收多播数据的比例很高时，那么就很有这种可能，即在大多数情况下，任意一个特定的子网中至少有一个多播组的成员在接收多播数据。

在密集模式多播路由协议中，每一个发送者或者每一组发送者通常要建立一个最短的路径树。因此，使用不同的分配树主要取决于数据从什么地方来，这也被称作为基于数据源树。如图 3-9、3-10 和 3-11 所示为 3 个不同的发送者计算 3 种不同的分配树的方法。

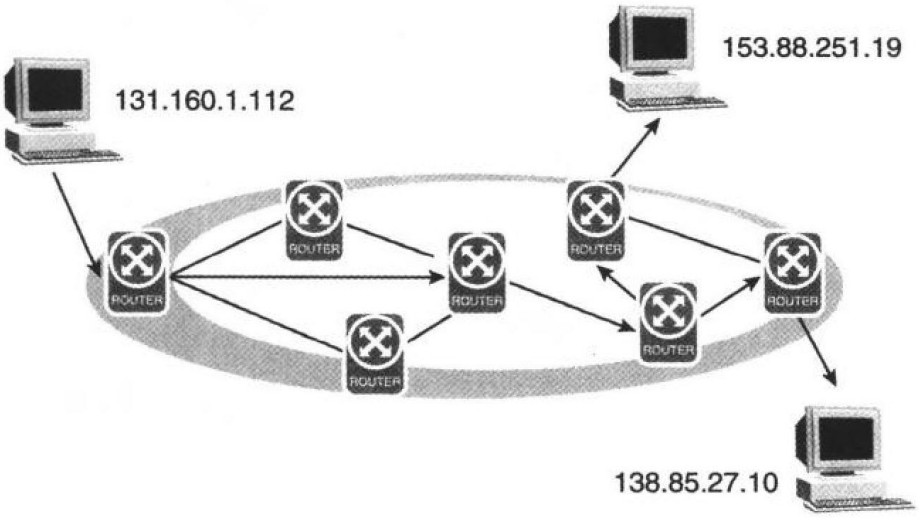


图 3-9　131.160.1.112 的分配树

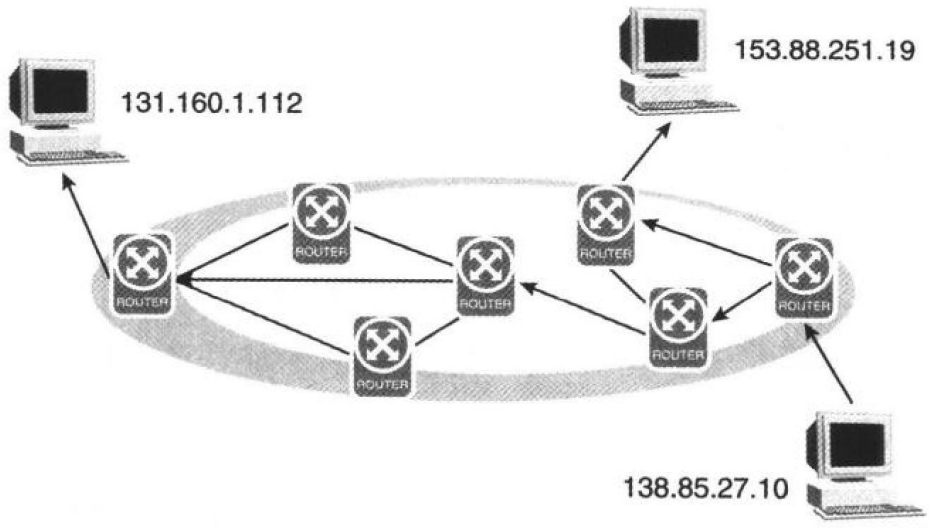


图 3-10　138.85.27.10 的分配树

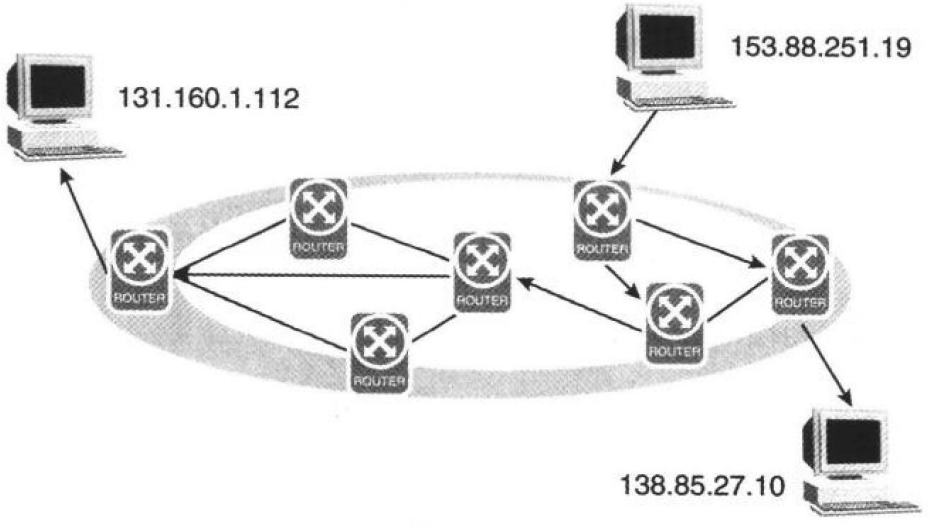


图 3-11　153.88.251.19 的分配树

密集模式多播路由协议的一些常用例子是距离向量多播路由协议（DVMRP）[RFC 1075]和协议独立多播—密集模式（PIM-DM），其中 DVMRP 现在应用最广泛。

2．稀疏模式多播路由协议

稀疏模式协议更适合用在主机接收多播流量比例不大的网络中。这些协议通常建立一个聚合点，基于这个聚合点建立共享树，从而被各个源节点使用。

如图 3-12 所示为基于聚合点的分配树建立的方式。当聚合点是发送者而其他组中的成员是接收者时需要计算树。在计算树的时候，将来自于组中发送者的分组封装，并将它路由到

聚合点，然后通过检查树发送给接收者。

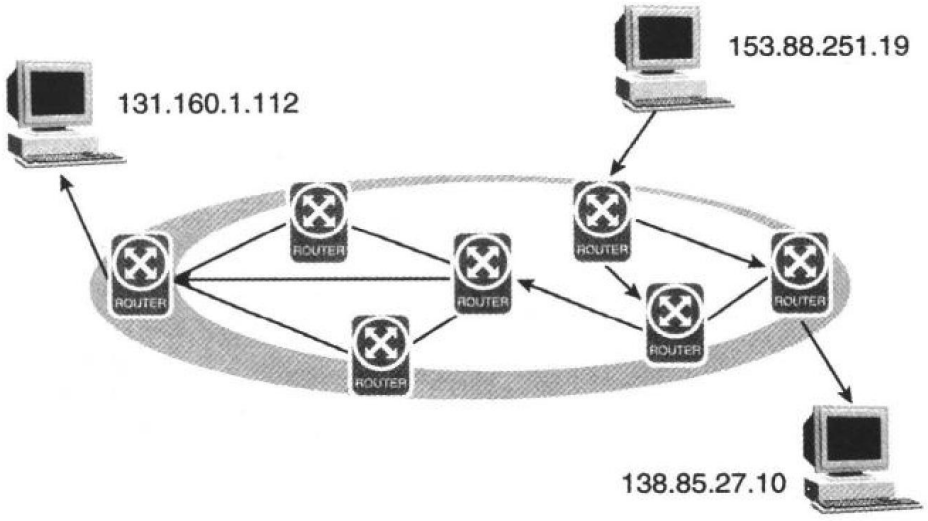


图 3-12　基于聚合点的分配树

一旦基于聚合点的分配树完成后，所有的节点都使用它。这意味着传送到聚合点的分组不需再进行封装，但需要通过树直接路由到接收者。在图 3-13 中，可以看出节点 138.85.27.10 和节点 153.88.251.19 使用的共享树不同于在图 3-12 中对它们进行的一次具体计算。

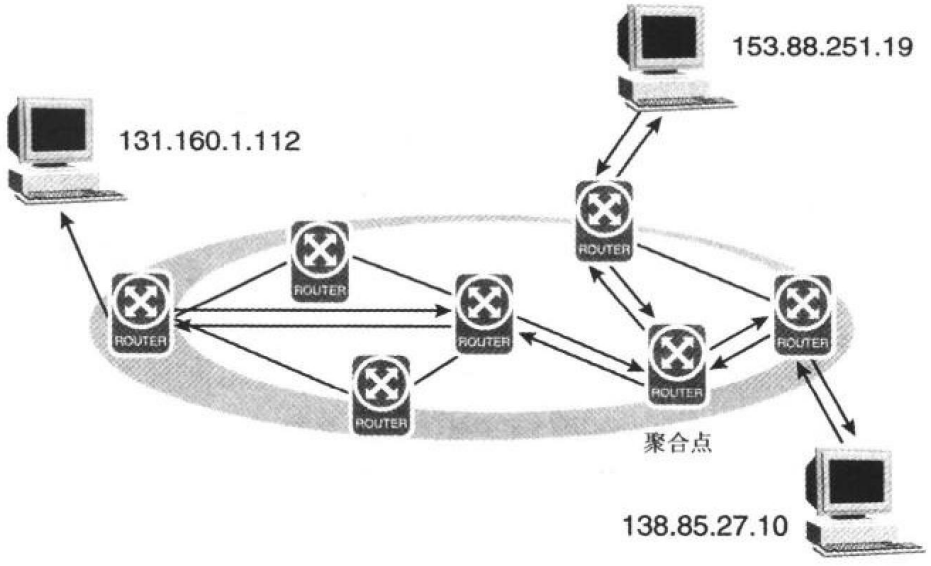


图 3-13　所有节点的共享树

稀疏模式的例子有协议独立多播—稀疏模式（PIM-SM）[RFC2362]和有核树多播路由协议（CBTS）[RFC2189]。

3.2.4　Internet 组管理协议

除了用来建立分配树的协议外，希望接收多播数据的主机也要成为特定多播分组中的一员，同一组中的主机设定相同的多播地址。Internet 组管理协议（IGMP）[RFC 2236][draft-ietf-idmrigmp-v3] 就是用来处理这个问题的（如图 3-14 所示）。主机发送请求加入或离开一个特定的组。由于处理一个特定子网的多播路由器拥有成员关系的信息，它就知道是否要接收多播的数据。如果该子网中没有成员，它就不必接收标有该组多播地址的数据报，并且同时使用多播路由协议把它自己从分配树中删除掉。如果以后它的子网中的一台主机希望成为该多播组中的一个成员，该多播路由器又将被加到分配树中。

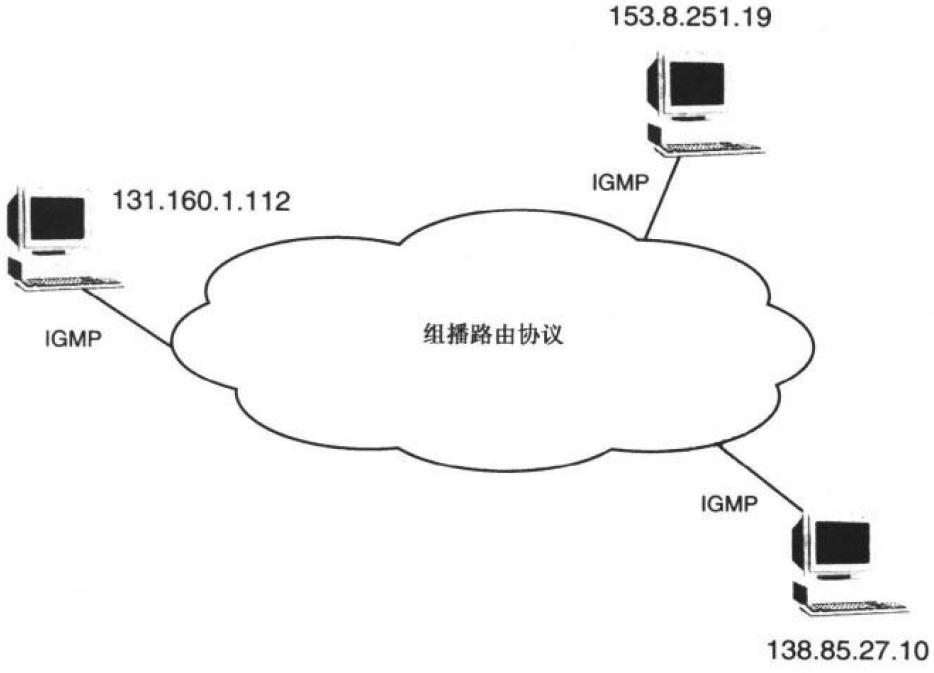


图 3-14　使用 IGMP

3.2.5　Mbone

Internet 中支持多播的部分被称作为 Mbone。Mbone 不是从 Internet 中独立的一部分网络。Mbone 路由器也是 Internet 的一部分，并且能够支持单播和多播信息流。当多播路由器孤立时，也就是它没有被直接连接到 Mbone 网上时，就在它和 Mbone 网中其他部分之间使用 IP 隧道。我们将会看到 SIP 首先被用于 Mbone 来使用户进行多播会话。

IETF 的发起者正在从事域间多播路由协议的工作，如边界网关多播协议（BGMP）[draft-ietf-bfmp-spec]，预计他们会提出一种具有更好伸缩性的分层的多播路由结构。域间信息流的聚合可以减少多播路由器保持的信息。

3.3 实时数据的传输：RTP

系统所需的各种类型的信息流通常是不同的。这种不同性要求根据不同的需求使用不同的协议来进行数据传送。对于需要通过网络进行可靠传送的数据，如 E-mail，发送者想要接收者能准确地阅读他或者她所写的信件，不等同于要致力于阻止分发延迟。对于大多数 E-mail 来说，延迟 10 秒钟与延迟 1 分钟没有什么大的差别。在这种情况下，有延迟但是确保分组正确是能够接受的，而保证了实时但分组不正确是不能够接受的。

根据以前的纪录，实时信息流的要求有明显的不同。假设两个人通过 Internet 进行对话，经过编码的音频通过需要回应的 IP 数据分组传送到接收者。使用过程中，普遍的感觉是，能接收正确的分组但是有较大延迟比没有接收到更难以被人接受：这简直没有任何作用。如果包含另外一个人所说的内容的乱序的数据报 5 秒以后才到达，那么它将会被丢弃。

3.3.1 数据分组抖动和排序

除了这些一般的实时需要外，还有一些其他问题，为实时传输信息流所设计的协议必须强调数据分组的抖动和排序问题。IP 网络会给每一个遍历 IP 网络的分组带来一些延迟，延迟量由许多因素决定。其中一个是接收分组路由器在这个瞬间的状态，如果路由器有很重的负载，分组将进入队列进行等待；如果队列是空的，分组就会得到立即处理。

处于属于同一个流中的每一个分组路由器的状态并不一样，有一个术语表示这种延迟的变化：抖动（Jitter）。如果抖动变化很大，那么后发送的分组可能会比先前发送的分组先到达。这样就打乱了实时传送，产生了乱序事件（如图 3-15 所示）。

为了减少抖动，业界广泛采用实时传输协议（RTP）[RFC 1889]。RTP 通过在分组头中指定时间戳和顺序号，以消除抖动的影响和乱序的分组接二连三地到来。RTP 头中的顺序号使接收者能够对接收到的分组进行排序。一旦排好序，通过时间戳，有效负载（如音频或视频）中数据的原始关系就能够得到恢复（如图 3-16 所示）。在已编码的音频例子中，时间戳告诉接收者什么时候通过扬声器播放 RTP 的分组有效负载。一个叫做有效负载类型的域描叙了 RTP 分组传送数据的类型（如通过 PCM 码进行音频编码）。

除了要传送有效负载的信息外，RTP 分组头也含有有效负载源的标识。请再回顾一下声音对话的例子，并且分解其包含的所有操作。发送者将编码的音频信号作为 RTP 分组的负载。接收者用一个缓冲区存储将要到来的数据分组，它们根据顺序号进行排序。当时间戳指示是回放有效负载时，该 RTP 分组将从缓冲区中删除掉。

如果将到达的 RTP 数据分组的时间戳指示它的有效负载已经被播放了，那么这个分组将被丢弃。因此，应该有足够的缓冲区保证分组有时间在它们被播放前到达。另外，缓冲区也要尽可能短，以防不可避免的延迟破坏正常的对话。

如果在这段时间内没有分组到达，那么应该播放什么？可以用一些其他的声音代替。可

以利用插入技术创建一个平滑的声音进行过渡，这样可以使接收方感觉不到这个时间间隙。

3.3.2　实时传输控制协议

先前的例子仅仅只有一个媒体流——音频流。可以利用时间戳恢复多媒体中数据的原始时间关系。它们分别为每一个多媒体流执行这个任务。例如，在一个视频会议中，视频流的时间戳保证了视频不会比原始图像快或者慢，而音频流时间戳对音频流进行同样的准确处理。然而，这需要同步流机制，因此就出现了实时传输控制协议（RTCP）[RFC1889]。它的作用是把时间戳和实时时钟联系起来。

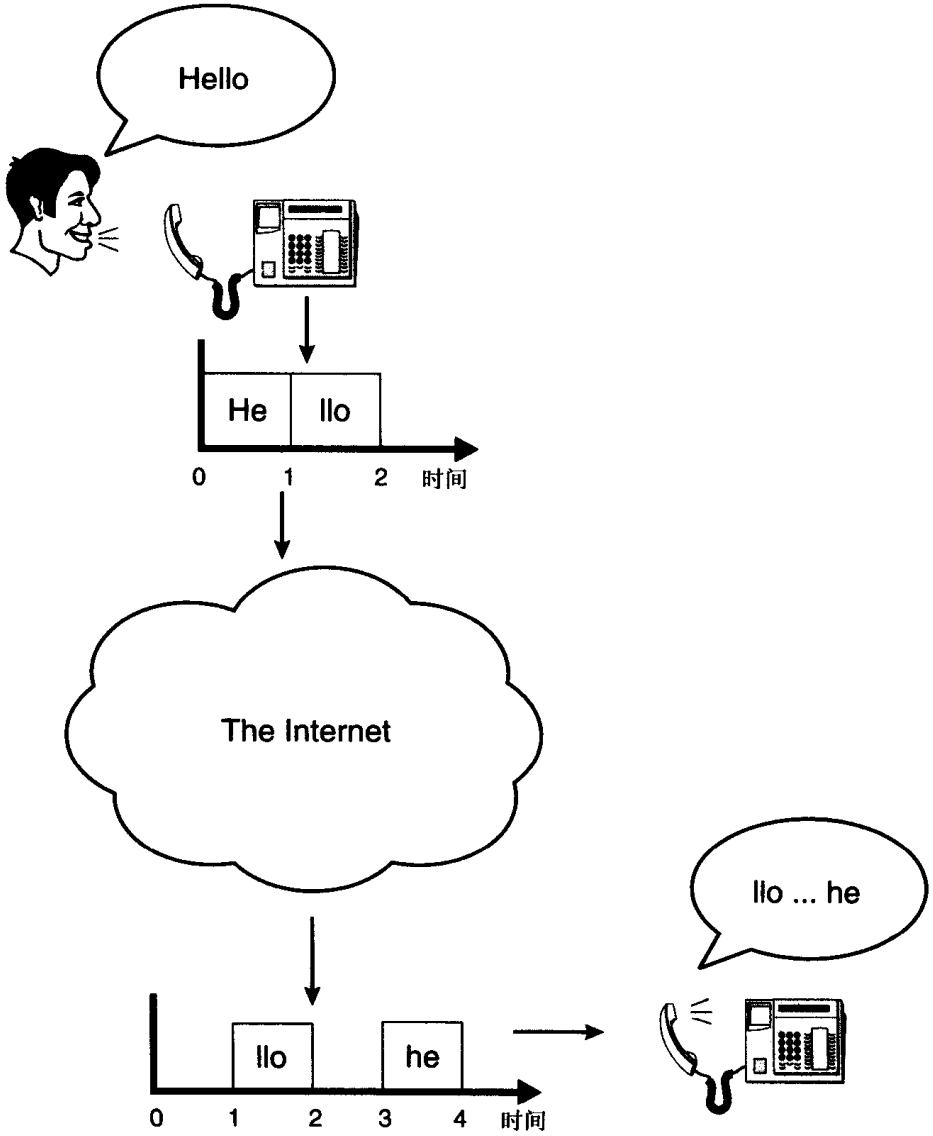


图 3-15　在话音传输中抖动造成的影响

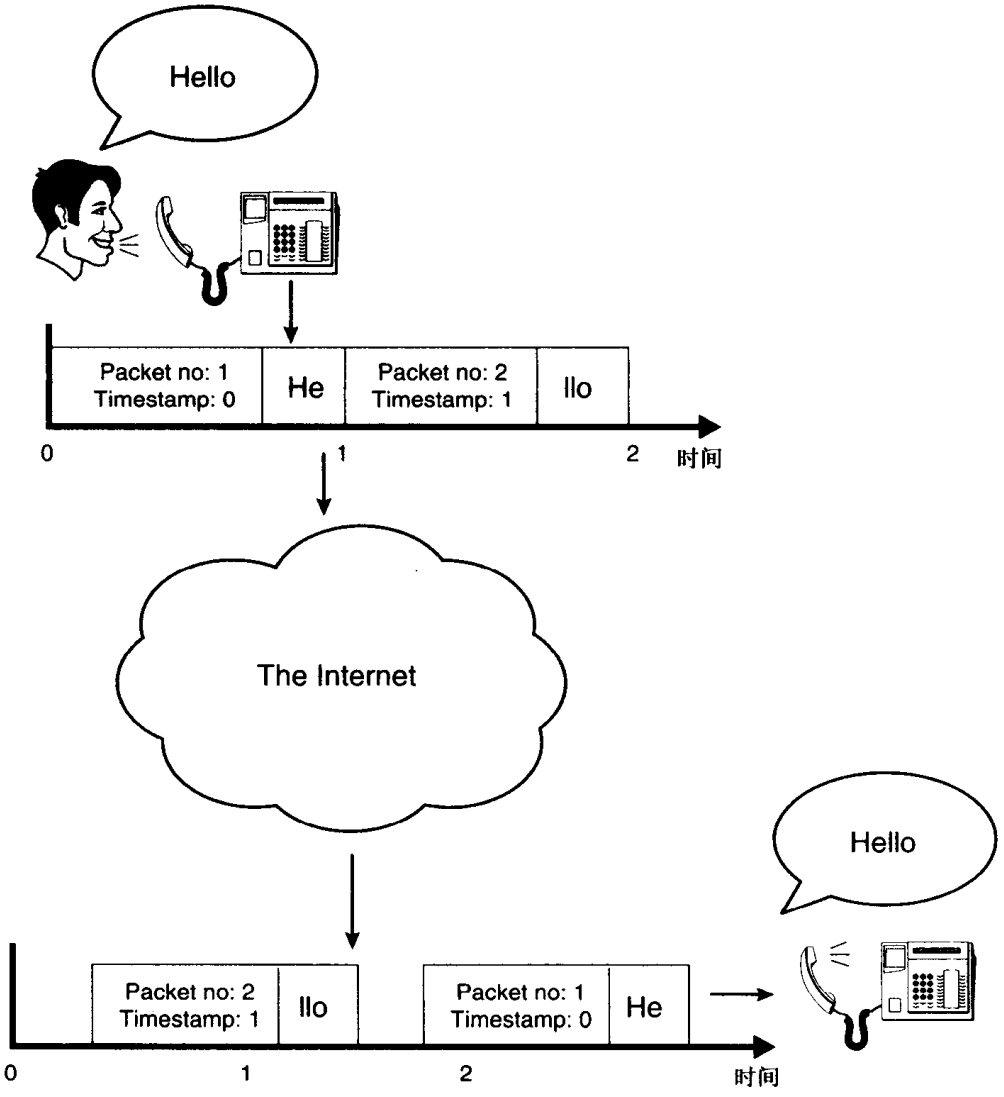


图 3-16　RTP 消除了抖动的影响和乱序包到达的影响

每一个 RTP 会话有一个并行的 RTCP 会话。除了媒体同步之外，RTCP 的会话成员提供信息并保证通信的质量。RTCP 向会话者报告网络在会话过程中丢失了多少分组，以便能让发送者知道接收者正在接收的质量。

3.4　服务质量提供：综合服务和区分服务

当网络处于正常负载时，Internet 的尽力（best-effprt）服务模型对大多数应用来说运行得很好。但是，当 IP 网络负载很重时，尽力服务模型可能就不能满足端到端的信息流的传输需求。当延迟增加时，网络就会成为有损耗的网络。IP 分组到达路由器的速度要比路由

器处理它们的速度快，这样就产生了排队现象。当排队超过队列大少的限制时，路由器就会丢弃分组。

对于 TCP 信息流，丢弃分组意味着要产生更多的重发，于是导致传输性能的降低和服务质量下降。用户开始关注端到端的传输延迟，甚至会放弃如文件传输之类的应用。

如果丢弃的数据报属于实时信息流，接收者将接收不到信息，并且数据报也不会被重发。因此，传输质量受到损伤。在声音传输的例子中，如果存在延迟，音频很快就不能被接收者理解。

对于需要比尽力服务更好服务的应用，有两个不同的方法能满足要求：综合服务（Integrated）[RFC 1633]和区分服务（DiffServ）[RFC 2475]。

3.4.1　综合服务

综合服务的基本思想是对属于不同数据流的分组在路由器中进行不同的处理。例如，可以将来自实时数据流中的数据报放在来自于低优先级的数据报之前发送。为了适当地提前发送数据报，处理特殊流的路由器需要得到附加的信息。也就是：怎样区别一个数据流中的数据报与另外一个数据流中的数据报以及怎样处理是合适的。

一个路由器可能根据某个一定的 IP 地址和一定的 UDP 目的端口号对所有的数据报进行优先排序，而同时继续对有不同目的地址的数据报使用尽力服务。在后面的例子中，一个路由器要处理几个数据流，因此，为了给路由器收到的数据报分类，就必须要实现对不同的数据报进行过滤。默认值是尽力服务模型，它是一些不属于任意一个路由器接收到的已知流的分组。

1．可用服务

当前综合服务体系结构提供两种服务：加载控制和保障服务。它们代表了优于尽力服务模型中的两层，如图 3-17 所示。

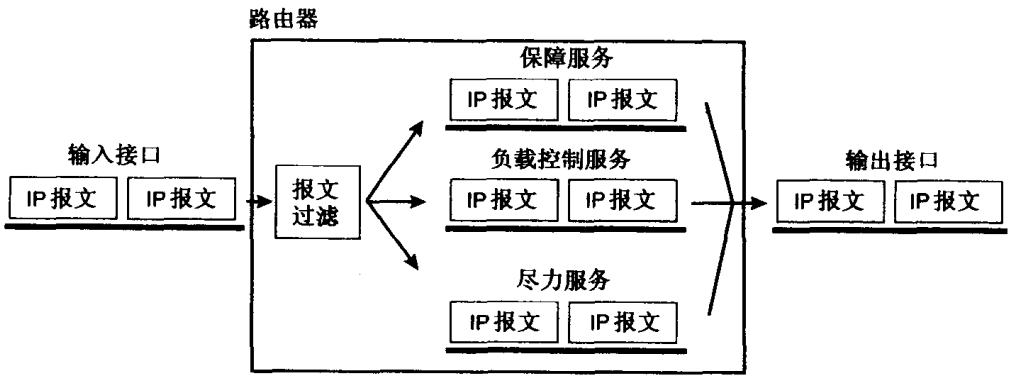


图 3-17　路由器中的包过滤

受益于加载控制服务的分组被赋予比尽力服务信息流更高的优先级。因此，即使网络中

充满了尽力服务信息流，只要网络处于中等的负载，加载控制服务就可以用来分发数据报。对于某一个特定的流，这种服务不能保证特殊的带宽和延迟，它仅能保证分组得到更高优先级的处理。

保障服务，正如其名字所示，能给特殊的数据流提供一定的带宽和延迟限制。因此，在保障服务的信息流中观察到的抖动是小到可以忽略不计的。

因为路由器不能给予无限制数量的数据流以高优先级，所以它需要具有允许控制和资源预留的能力。

一旦接收到处理新数据流的请求，路由器就要检查它是否有足够的资源接收它，而且不影响正在处理中的其他数据流。如果对于数据流的 QoS 请求能够满足，路由器就为下一个数据流预留资源。

2．网络中存储的状态信息

刚才已经看到，为了正确区分数据报，路由器必须存储流信息。这就意味着网络存储着状态信息。目前，我们正在讨论在 IP 机制上进行这样的处理，端点系统实现智能化，在网络中存储尽可能少的信息。放弃少量信息的系统是一个更强健的系统，这样的系统能够更好地容忍网络的失败。由于确认这种范例的值和需要即使在很好定义的情况下也会产生例外，用预留合并和软状态能帮助减少例外所能引发的问题。

3．预留合并

图 3-18，3-19 和图 3-20 展示了在多播组中预留合并的执行情况。在图 3-13 中，已经知道对于那种拓扑结构共享分配树是怎样计算的。在图 3-18 中，假定 131.160.1.112 是发送者，而 153.88.251.19 和 138.85.27.10 为接收者。

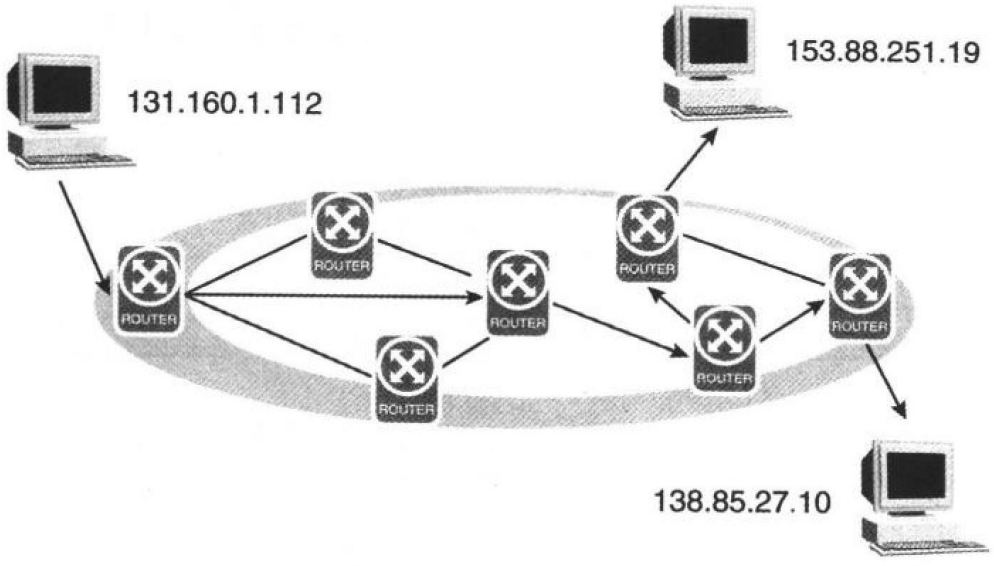


图 3-18　131.160.1.112 作为发送方时的分配树

138.85.27.10 为将要到达的数据流请求一个 QoS，这条路径中的路由器在分组过滤器中存储必要的状态信息，最后，所请求的 QoS 受到处理，如图 3-19 所示。

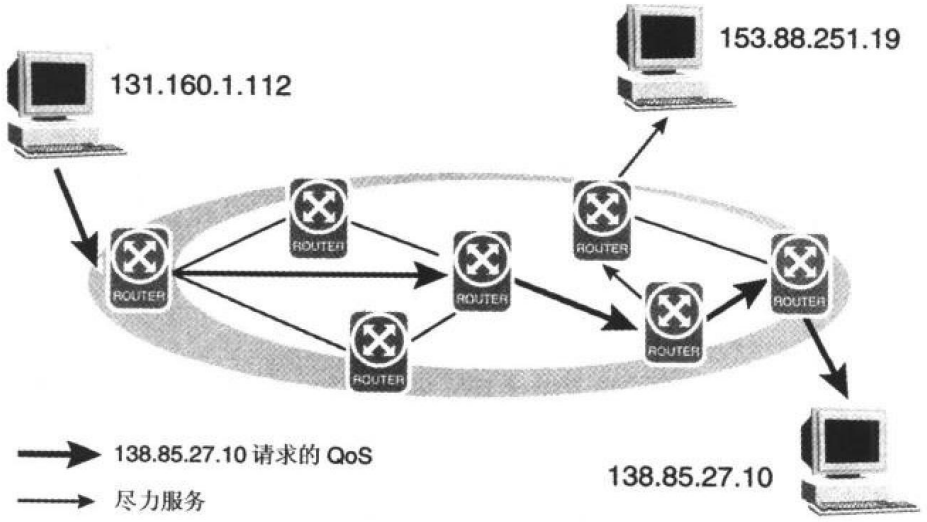


图 3-19　138.85.27.10 为数据流请求 QoS

现在，153.88.251.19 也为将要到达的数据流请求一个 QoS。然而，第二个接收者不必再从发送者开始的整个路径上请求 QoS，因为对于同样的数据流，在部分路径上已保留有 QoS 的信息。因此，当 153.88.251.19 发出 QoS 请求时，一直为第一个接收者提供 QoS 的路径上的路由器中的状态信息并没有增加。新的状态信息被限制到从 153.88.251.19 到主分配树中的这条路径上的路由器中，如图 3-20 所示。

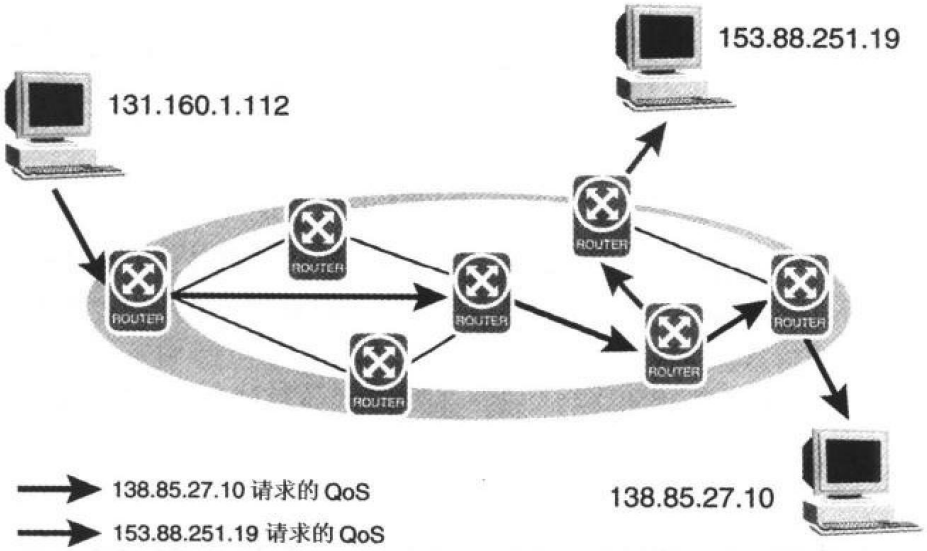


图 3-20　153.88.251.19 也请求 QoS

4．软状态

执行软状态能够增加系统的健壮性。当某一个服务器删除它已存储的所有信息时，软状态临时存储这些状态信息。在这个系统里，如果状态信息没有被定时刷新，它将会超时并释放路由器中保存的所有信息。一般通过给路由器发送一个合适的信息来刷新软状态，而硬状态则是永久存储的，它需要一个状态释放命令来放弃资源。

5．资源预留协议（RSVP）

RSVP 是用来在网络中预留资源的协议。RSVP 在路由器上安装了必要的状态信息，并且定时刷新它。通过接收者可以启动一个特定流的资源预留；从接收者到发送者存储状态的消息被称作为预留消息。

当数据报反方向传递时，从接收者到发送者的 IP 数据报一般情况下都不走同样的路径。因此，流中从发送者到接收者的 RSVP 消息（路径消息）必须被提前送到资源预留站，这样做的目的是分辨出这次流中的数据报将要走什么路径。路径消息中含有预留消息必须向后遍历路由建立状态的路径。如图 3-21 所示为一个 RSVP 消息流。

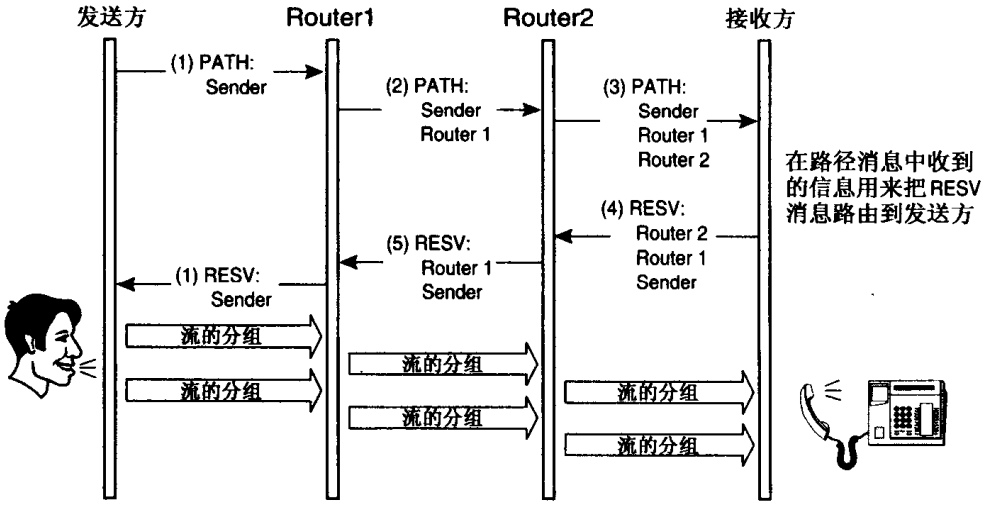


图 3-21　RSVP 消息流

软状态的使用也能够在路由发生变化时阻止网络保留不需要的信息。如果到达流目的地址的路由作为路由协议的运行的后果发生了变化，那么所有的数据报将进行重路由。周期性产生的路径（PATH）消息也会进行新的路由。在旧路由中的路由器将不会再接收到任何路径消息，也没有预留消息能被接收到。这时，这些路由器中存储的状态已超时并且会被删除掉。

3.4.2　区分服务

我们已经明白了 RSVP 和综合服务体系结构为路由器中不同流提供的不同方法。路由器基于所接收到的 RSVP 消息中的信息进行过滤分组。例如，只要流被分组的目的地址和目的

端口号所定义了，路由器就必须检查每一个数据报中的目的地址和目的端口号，并把它映射到状态信息上。区分服务模型根据优先级级别给信息流分类，从而简化了工作任务。在网络边缘上，分组被标识了所需的优先级级别。网络中的路由器从这些标识中获取相关信息；每一个标识与一个处理分组特殊的方法相联系（被称为每跳行为 PHB）Per-Hop Behavior，路由器仅需读取这个标识和查询它的 PHB。标准 PHB 包括加快转发[RFC 2598]和确认转发[RFC 2597]。前者模仿电路交换行为，后者提供降序优先（Drop Precedence）。

区分服务模型的伸缩性比集成服务模型好，因为它将路由器从维持每个流的状态的需求中释放出来。但即使有了区分服务模型，网络仍然需要获得允许进入网络的控制机制，否则没有办法阻止端系统把所有的信息流都标识为高优先级从而堵塞网络。RSVP 能够用来获得进入控制，所以希望能看到把 RSVP 和 DiffServ 一起使用，以获得最大的可伸缩性，如图 3-22 所示。

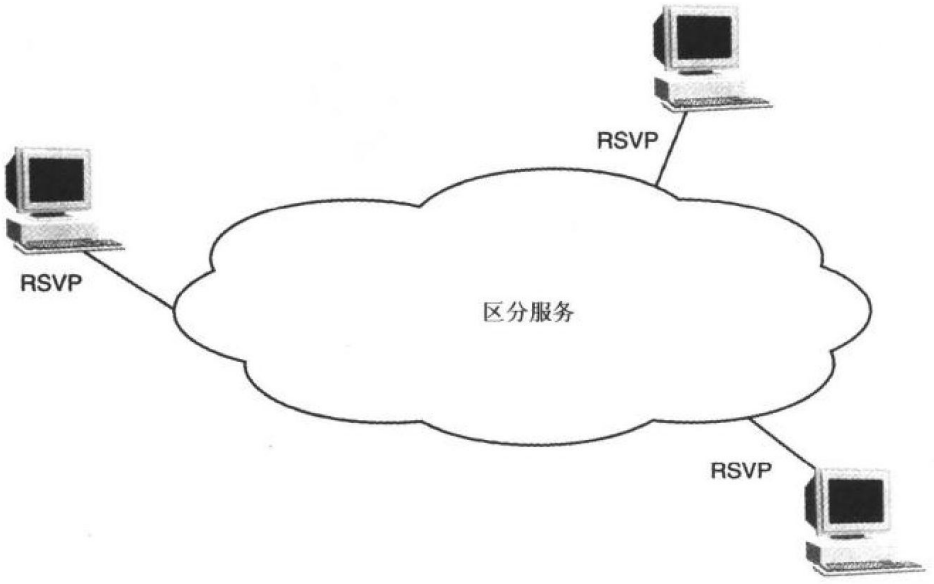


图 3-22　RSVP 和 DiffServ 共同工作

3.5　会话通告协议（SAP）

当一个人决定要看电视的时候，一般都要先看一下节目单，弄清楚哪个频道播放有趣味的节目。一旦一个人做了选择，就打开电视选择正确的频道。节目单中包含了播放节目的内容、播放的频道以及播放的时刻表。

Internet 利用了同样的程序。用户需在所有提供的信息中选择最有趣的多播会话，也需要知道怎样配置多媒体工具来接收所选择的会话。例如，他必须知道一个会话是否含有音频或者含有视频。

会话通告协议（SAP，Session Announcement Protocol）[RFC 2974]用来实现在潜在的接收者中分发有关多播会话的信息。SAP 在一个大家都知道的多播地址和端口号承担多播会话描述任务，如图 3-23 所示。因为多播技术并不提供可靠性，所以 SAP 是不可靠的，并且需要定时重发。SAP 使用固定数量的带宽，所以一台 SAP 主机能够承担起其他主机所发送有相同的地址和端口号的通告。根据通告的数量，主机为会话通告选择重发率。因此，当前的会话越多，重发的间隔就越长。

最后，SAP 通告能够被加密，也能利用鉴定机制。加密和鉴定提供必须的保密和检查一个特定会话创建者身份的功能。

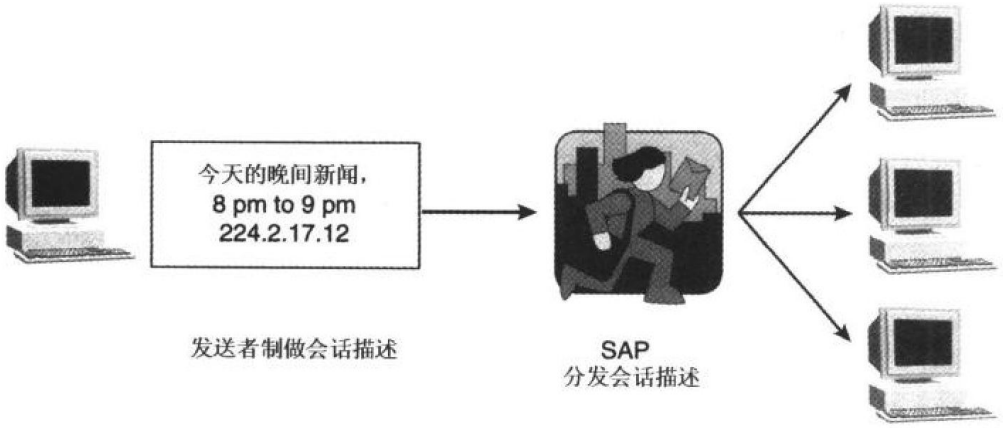


图 3-23　SAP 在替在用户间分发会话描述

3.5.1　会话描述

尽管 SAP 向潜在的接收者多播会话描述，但它没有定义这些描述的格式。采用 IETF 的方法，除了 SAP 外，还有其他一些协议也能够用来描述会话。（需要推荐格式，请查阅会话描述协议（SDP）[RFC 2327]）。SAP 携带所有的描述格式，但没有办法协商描述会话的协议。因此，当由于某些原因系统不能理解一个 SAP 通告时，它不起作用。它不能收到不同会话描述协议的相同通告。相反，SDP 作为普通的协议用来描述会话，并且所有的协议必须支持 SDP。

3.6　会话描述协议（SDP）

会话描述协议（SDP，Session Description Protocol）规定了对描述会话的必要信息怎样进行编码。SDP 不包括任何传输机制，也不包含任何种类的协商参数。一个 SDP 描述仅仅是能够被系统用表在一个多媒体会话中加入大量信息。例如，它包括 IP 地址、端口号以及会话处于激活状态的时间和日期等。

再回到上述节目单的例子中，在电视设备中，一个会话描述如下所述：“在 9 点钟调到 5

频道看一场足球比赛”或者“每天晚上 9 点钟打开 2 频道看新闻。”。在电视设备的会话描述中，应该含有一些关于怎样接收广播会话（5 频道），什么时候会话在广播（从 9 点到大约 11 点）以及会话内容的信息（足球比赛）。

我们能够通过许多方法得到喜爱的电视节目的信息，从报纸或节目预告中获取节目单，或者查阅电视文字广播，或者从朋友的电话里获知。然而，不管怎样获得我们自己喜爱的节目，我们都需要知道同样的信息：哪一个频道、什么时候和什么节目内容。Internet 接收多媒体会话所需的信息有点不同，但原理一样。不管会话描述怎样分发，SDP 都能够描述会话。它们能够使用 SAP 分发或者通过 E-mail 发送，也能出现在 Web 页上，或者通过一个普通的 Internet 浏览器检索到。我们将会很快地发现在 SIP 的消息中也能携带会话描述符。

3.6.1　SDP 语法

很值得花一些时间研究 SDP 会话描述采用什么样的形式，因为它们在许多 SIP 消息中出现。SDP 会话描述是基于文字的，这与二进制编码如 ASN.1 不同。一个会话描述由一些类似如下形式的文字行组成：

```
Type=value
```

类型域为一个字符长，而值域的格式取决于它应用哪一种类型。一个 SDP 描述含有会话级信息和媒体级信息。会话级信息应用于整个会话。例如，它能成为会话始发者或者会话的名字。媒体级信息作用于特殊的媒体流。例如，它能作为一个编码器给音频流编码或者是给视频流发送端口号。

一个 SDP 会话描述以会话级信息和媒体级信息开始，如果任意一个出现，另外一个就接着在后面出现。会话级部分以 v=0 开始，v 代表类型，0 为值，意思是协议版本号为 0（SDP 版本 0）。接下来的行直到媒体级部分或者会话描述的终点，提供了整个会话的信息。

媒体级部分以 m 行开始。下面的行直到下一个 m 行出现，或者直到会话描述的终点，提供了特定媒体流的信息。下面是一个 SDP 会话描述的例子：

```
v=0
o=Bob 2890844526 289084207 IN IP4 131.160.1.112
s=SIP Seminar
i=A Seminar on the Session Initiation Protocol
u=http://www.cs.columbia.edu/sip
e=bob@university.edu
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31
a=rtpmap:31 H261/90000
```

```
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

在这个例子中，会话级部分由前面 9 行组成，是从 v=0 到 a=recvonly。这个例子包括 3 个媒体级部分：一个音频流和两个视频流。o 行说明了会话的创建者（在这个例子中是 Bob）和他的站点的地址。s 行包含了会话的名字，i 行包含了会话的一般信息。u 行提供了统一资源定位器（URL，Uniform Resource Locator），在这个地址中能够检索到有关会话主题的更多内容。e 行含有会话联系人的 E-mail 地址。c 行描述了能够接收到会话的多播地址，t 行说明了什么时候会话是激活的。会话部分最后一行，也就是 a 行，说明了这不是一个交互式的会话：它只能接收。m 行的格式很重要，以媒体类型开始。在前面的例子中，第一个媒体流的类型为音频，第二个和第三个媒体流的类型为视频。

m=<媒体类型> <端口号> <传输协议><媒体格式>

端口号说明了在什么地方能够接收到媒体。传输协议域通常取值为 RTP/AVP，但如果不使用 RTP 协议，也能够取其他的值。RTP/AVP 指对于 RTP 的音频/视频描述文件；本例中，经过编码的音频和视频是在 UDP 上使用 RTP 传输。

媒体格式取决于媒体传输的类型。对于音频来说，它就是正在使用的编码解码器。本例中，值 0 意味音频是在单个信道中使用 PCM μ-law 进行编码和以 8kHz 的频率采样。

a=rtpmap 行传送有关媒体使用的信息，如时钟频率和信道数量。本例第二个媒体流中，媒体格式号 31 是指 H.261 协议并且使用 90kHz 的时钟频率。

SDP 定义的所有的类型和含意见表 3-1。

表 3-1　　SDP 类型

v	协议版本
b	带宽信息
o	会话的主人和会话标识
z	调节时区
s	会话的名称
k	密钥
i	会话信息
a	属性行
u	含有会话描述的 URL
t	会话激活的时间
e	获得会话信息的 E-Mail 地址
r	会话重复的次数
p	获得会话信息的电话号码
m	媒体行
c	连接信息
i	媒体行的信息

扩展 SDP

媒体属性行（也就是 a 行）提供了一种扩展 SDP 的方法。当应用需要一个在 SDP 中没有包含的特征时，可以通过增加 a 行来包含这个特征。例如，如果多播会话的创建者想要接收者以某种特定音量播放音频，他或她能够定义一个新的媒体属性，并把这种新媒体属性加在媒体级部分的末端。

```
m=audio 49170 RTP/AVP 0
a=volume: 8
```

随后，理解这个新 a 行的应用程序就以音量 8 播放音频。当应用程序发现一个它不能理解的 a 行时，就简单地忽视这一行并继续往下处理，就像没有遇到此行一样。尽管它不能以正确的音量回放音频，不能理解新行 a=volume 的应用程序仍然能够正确接收该媒体。

对于读者感兴趣的那些个别主题，IETF 已经开始评估一些被提议的 a 行的扩展方案，当 SIP 和 SDP 被一起使用时，这些扩展能提供 QoS。

3.6.2　下一代 SDP（SDPng）

最初，SDP 用来描述 Mbone 中的多媒体会话，但现在，它也在很多其他场合中使用。在其他的应用中，SDP 与实时流协议（RTSP）一起用于流服务、与 SIP 一起用于会议邀请、与媒体网关控制协议（MGCP）[RFC 2705]或者 H.248 一起用于主从结构的设备。因为最初设计 SDP 时没有考虑到在这些环境中工作，所以它并不很适合于这些新的环境，也缺乏一些应用所需的特征。

和一些将来的应用可能需要会话描述机制一样，这些新环境对 SDP 的后继者提出了新的要求[draft-kutscher-mmusic-sdpng-reg]。当前，SDP 的后继者被称作为下一代 SDP（SDPng）[draft-ietf-mmusic-sdpng],它正由多方多媒体会话控制（MMUSIC）工作组开发。与 SDP 相比，SDPng 将尽力提供更丰富的会话描述，用更好的办法解决容量协商问题。然而，在 SDPng 设计中的一个关键概念是简明性。因此，折衷的办法是在合理的复杂性的基础上上提供更多的功能。

3.7　实时流协议（RTSP）

实时流协议（RTSP）[RFC 2326]用来控制多媒体服务器，一般用于流应用。在用户和多媒体服务器之间使用 RTSP 和远程控制使用 VCR 相类似。例如，用户能够通过播放按钮告诉服务器开始某个音频或视频流，在一个特定的时候通过暂停按钮冻结流，或者通过向前或向后按钮在某个位置重放流。用户也能通过记录按钮命令服务器记录特定的媒体。举一个例子，RTSP 能够被用来实现一个分布式响应机器，或者用来记录在 Internet 上正被多播的内容。

3.8 Internet 多媒体会议工具包的使用示例

让我们看一下怎么样把本章分析的协议组合在一起在 Internet 上多播电影。控制电影播放的端用户使用 SDP 详细建立会话描述，其中包括电影什么时候播放、电影的内容以及接收媒体所需的一些变量。这些变量包括多播地址、端口号和最简的媒体格式。这个 SDP 会话描述通过 SAP 使用多播路由分发给潜在的接收者。

感兴趣的端用户将会检查他们所收到的 SDP，并正确配置他们的媒体工具，以能定时收看电影。当电影安排好后，会话控制者将使用 RTSP 通知存储电影的多媒体服务器进行多播，它使用先前分发的 SDP 会话描述。

媒体服务器将多播含有电影的视频和音频 RTP 分组。它将使用 RTCP 存储接收者所接收服务质量的统计量。RSVP 也可能用于确认媒体服务器与接收者之间的 QoS。

第4章　会话初始化协议：SIP

本章直接介绍会话初始化协议（SIP）。我们描述 SIP 提供的功能和协议中定义的实体，分析它具有哪些好的特性，同时解释 SIP 在通信发展前景中所处的位置。最后，我们阐述为什么 SIP 是一个好的协议设计的例子。本章说明我们希望能从 SIP 得到什么，但是并不讨论协议的细节。如果读者需要了解协议运行情况，包括消息、语法和具体使用例子的信息，请阅读下一章。将协议行为从协议操作中分离出来，使读者能够区分 SIP 的功能以及如何实现这些功能。

4.1　SIP 历史

在前面的章节中，我们看到 Internet 多媒体会议体系结构中包含很多协议。这些协议并不是同时开始发展的。从第一个多媒体系统开始，这个体系结构就是动态进化的。新的协议被设计出来而同时已存在的协议也得到改进。随着时间的推移，Internet 向着提供越来越多的多媒体服务的方向发展。

但是，这个体系结构仍然有不完善的一小部分：它没有办法能够明确地邀请用户加入一个特定的会话。例如，可以使用会话通告协议（SAP，Session Announcement Protocol）宣布一个多播会话，但是那些潜在的接收者必须周期性地检查所有发布的通告并从中找出他或者她想加入的一个。对于用户来讲，他不可能通知另外一个用户关于某一会话的信息，他也就不可能邀请他或她参与其中。

假设我正在看一部在 Mbone 上多播的有趣的电影，这时候我想起一个朋友可能也有兴趣观看它。我需要用一种简单的方式通知我的朋友，给她发一份会话描述符，邀请她加入这个会话，如图 4-1 所示。

邀请用户加入 Mbone 会话是 Internet 工程任务组（IETF）提交 SIP 的初衷。这个协议从那时起一直稳定向前发展，SIP 目前用于邀请用户加入所有类型的会话，包括多播和点到点会话。

据我们现在所知，SIP 并不是从头开始设计的，它是将两个由 IETF 为同一目的提出的协议进行合并的结果。SIP 从每个协议中汲取了其精华，从此开始，团体中所有的努力都汇聚到了 SIP 上。

4.1.1　会话邀请协议：SIPv1

尽管第一次在分组交换网络上进行语音传输大约发生在 1974 年，但第一个多媒体会议系统在 20 世纪 90 年代早期就已经出现了。Thierry Turletti 发展了 INRIA 视频会议系统（IVS，

INRIA Videoconferencing System)，这个系统用于在 Internet 上传输音频和视频信息。一个 IVS 用户可以呼叫其他用户，同时他们也可以建立起一个单播会话。IVS 也可以用于多播会话。由于在 Internet 上利用 H.261 视频编码传送信息，在 IVS 系统中的工作有助于为 H.261 视频流[RFC 2032]开发实时传输协议（RTP，Real-time Transport Protocol）的负载格式。

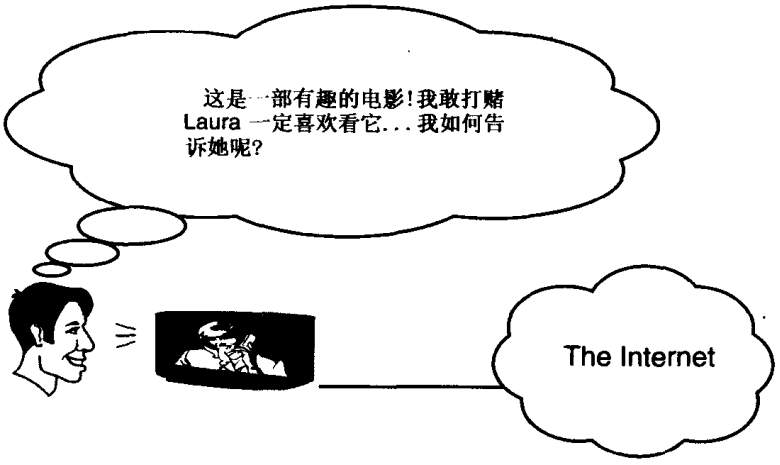


图 4-1 SIP 使我们可以邀请用户加入会话

这之后很短的时间内，Eve Schooler 开发了多媒体会议控制（MMCC，Mutimedia Conference Control）系统。MMCC 软件提供了点到点和多点电视会议功能，并且包含了音频，视频和白板工具。

为了连接不同的用户，MMCC 使用了连接控制协议（CCP，Connection Control Protocol）——一个面向事务的协议。一个典型的事务由一个请求（由用户发出的）和一个应答（从远端用户发出的）组成。为了传输信息，CCP 使用了用户数据报协议（UDP，User Datagram Protocol）对所传输的信息进行封装，所以它实现了超时和重传机制，以保证协议消息的可靠传输。

这两个最初的多媒体系统后来被 Mark Handley 和 Eve Schooler 开发的会话邀请协议（Session Invitation Protocol）取代。SIP 的第一个版本，即 SIPv1，于 1996 年 2 月 22 日作为一个 Internet 草案提交给 IETF。SIPv1 使用会话描述协议（SDP，Session Description Protocol），描述会话同时使用 UDP 进行传输。它是基于文本的。向会议地址服务器注册的概念在 SIPv1 中非常重要。一旦用户注册了他或她的位置，地址服务器就可以将邀请路由到适当的用户，同时使用户具有一定程度的可移动性。例如，如果某人离开了他或她常用的工作站而正在做商务旅行，这位用户可以选择注册他或她的临时工作站并接受邀请参加本地会议。

值得注意的是，SIPv1 协议仅仅处理会话的建立。一旦用户加入会话，则信令就终止了，同时会议的中间控制也有待将来实现。

4.1.2 简单会议邀请协议：SCIP

同是在 1996 年 2 月 22 日，Henning Schulzrinne 也将一份详细说明简单会议邀请协议

（SCIP，Simple Conference Invitation Protocol）的草案提交给 IETF。SCIP 也是一个邀请用户参与点到点和组播会话的机制。它基于超文本传输协议（HTTP，Hyper text Transfer Protocol）并因此利用传输控制协议（TCP，Transmission Control Protocol）作为传输协议。像 SIPv1 一样，它是基于文本的。SCIP 使用 E-mail 地址作为用户的标识符，为的是向同步和异步通信提供统一的标识。SCIP 信令一直持续到会话建立之后，以使参数可以在正在运行的会话间交换和关闭已存在的会话。与 SDP 反复使用一种机制用于会话描述不同，SCIP 定义了一种新的格式用于此目的。

4.1.3　会话初始化协议：SIPv2

在洛杉矶举行的第 35 届 IETF 会议上，Schooler 提出了 SIP，同时 Schulzrinne 提出了 SCIP。通过此次会议，以及在后来的 IETF 第 36 届会议上，在各种级别中对这两个协议开展了反复讨论。最后决定合并这两个协议。

最终的协议保留了“SIP”作为名字，但是将这个缩写的意思改变为会话初始化协议（Session Initiation Protocol）同时将其版本号提升到 2 如图 4-2 所示。

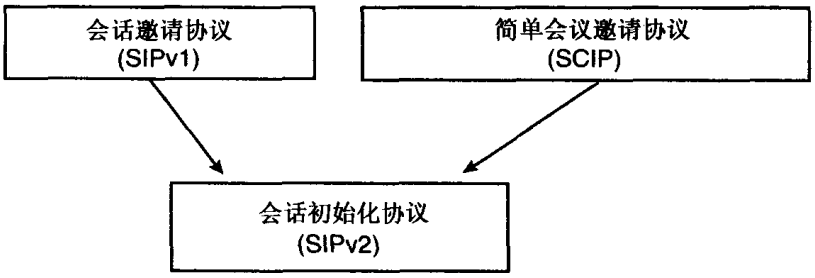


图 4-2　SCIP 和 SIPv1 合并为 SIPv2

一份 SIPv2 的 Internet 协议草案（作者为 Mark Hanley，Schulzrinne 和 Schooler）于 1996 年 12 月在圣乔治举行的 IETF 第 37 届大会上提交给 IETF。这个新的 SIP 基于 HTTP，但是可以使用 UDP 或者 TCP 作为传输协议。它使用 SDP 描述多媒体会话，同时它是基于文本的。它仍然是 SIP 协议的现行版本。

SIP 的发展成果属于多方多媒体会话控制（MMUSIC，Multiparty Multimedia Session Control）工作组的范围，MMUSIC 的主席是 Joerg Ott 和 Colin Perkins。第一个草案产生于作者收到的反馈和在 MMUSIC 邮件列表中的讨论。1998 年，Jonathan Rosenberg 由于为这些讨论做出了如此多的贡献而被加入到草案的合作者行列中，同时在接下来的 1999 年 2 月，SIP 达到了提议标准水平并且作为 RFC 2543 公开出版。

随着时间的推移，SIP 在 IETF 中逐渐获得重视，导致了在 1999 年 9 月一个新的 SIP 工作组的形成。这个工作组最初由 Joerg Ott，Jonathan Rosenberg 和 Dean Willis 担任主席。在 2000 年 8 月，因为 Rosenberg 和 Willis 将共同的从属关系改变到了同一个公司中，而在主席中希望存在多样性，这样 Rosen 代替了 Rosenberg 作为联合主席。

随着 2001 年 3 月份 IETF 会议的闭幕，SIP 工作组被分成两部分。讨论主要的 SIP 规范和它的基本扩展的任务由一个工作组承担，它仍然叫做 SIP，而关于使用 SIP 协议进行具体应用的讨论由另外一个叫做 SIPPING 的工作组执行。这次分工有助于对大量的要求 IETF 考虑 SIP 的投稿进行管理。

提议标准状态

到 2001 年 7 月为止，SIP 仍然不是一个完成的产品。它当时的状态是提议标准。在它被给予下一级成熟等级即草案标准以前，还需要通过更进一步的审议，并且它必须拥有至少两个不同的互操作实现。作者从社团中收到了大量反馈评议的同时，实现者的经验也被纪录下来。对协议所有的增补和修订都集中到一个 Internet 草案中[draft-ietf-sip-rfc2543bit]，这个草案注定在 SIP 真正成熟时将变成 RFC 草案标准。

值得注意的是，尽管 SIP 仍然是一个提议标准，但是它足够稳定，故可以在产品中得以实现。研发者周期性地举行互操作测试，以保证其互操作性，这一做法最初被称为 bake-offs，但是现在由于烘烤工业界的要求，有些乏味地改名为 SIP 互操作事件（参看 http://www.cs.columbia.edu/sip/sipit/pillsbury.html 了解这个故事）。首届 bake-off 于 1999 年在哥伦比亚大学举行，现在通常每年举行 3 次 SIP bake-off。SIP bake-off 已经证明对找出协议规范中的疵点和寻找这些疵点的精细的解决方案大有裨益。

4.2 SIP 提供的功能

RFC 2543 描述了 SIP 的核心，也就是协议的基本操作。除了基本规范之外，一些 SIP 的扩展功能在其他的 RFC 和 Internet 草案中已作了定义（如图 4-3 所示）。再次说明，本章的讨论仅限于基本协议规范中描述的功能。

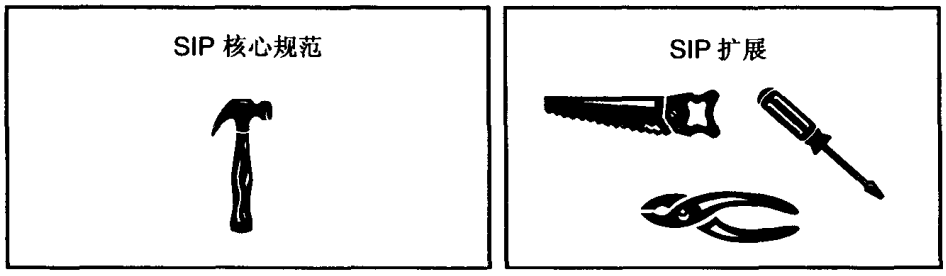


图 4-3 SIP 工具包由核心规范和几个扩展组成

4.2.1 会话的建立、调整和终止

SIP 建立、调整和终止多媒体会话。它可以用于邀请新的成员加入一个已经存在的会话或者创建一个全新的会话。当我使用 SIP 通知我的朋友 Bob 说 Internet 上一些正在多播的东

西他可能会觉得有趣的时候，我正在调用一个已经存在的会话。可是当 Bob 呼叫 Laura 来传播这个新闻的时候，这种双方的呼叫使用单个音频成分形成了一个新的多媒体会话。在这个例子中，Bob 邀请 Laura 加入一个需要创建的会话。另外，这个会话只有在两个条件得到满足的前提下才能被创建：（1）Laura 愿意同 Bob 交谈，同时（2）他们可以就会话将使用的媒体参数达成一致。

SIP 独立于它处理的多媒体会话类型和描述会话所使用的机制。它对于视频会议、语音呼叫、共享白板和游戏会话都同样有用。由传送音频和视频的 RTP 流组成的会话通常用 SDP 协议描述，但是某些类型的会话可以用其他的描述协议来说明。假设 Bob 想和 Laura 下国际象棋，他可以选择使用一个特定的国际象棋会话，这个会话可以用特定的国际象棋描述协议而不是 SDP 协议来说明。如果 Bob 和 Laura 将要在 Internet 上玩一个视频游戏，他们也可能使用一个协议而不是 SDP 协议来描述他们的游戏会话。

简单地说，SIP 用来在潜在的参与者中分发会话描述符如图 4-4 所示。一旦会话描述符分发完成了，SIP 可以用于协商和调整会话参数和终止会话。

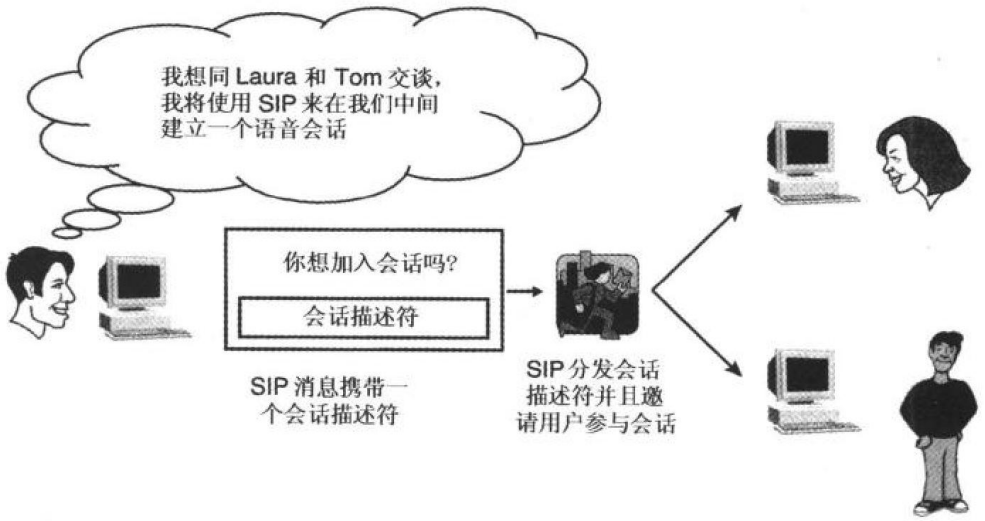


图 4-4　Bob 邀请 Laura 和 Tom 加入一个语音会话

下面的例子说明了 SIP 的这些功能。Bob 想要和 Laura 建立一个音频视频会话，并计划采用一个脉冲编码调制（PCM，Pulse Code Modulation）编解码器来对语音进行编码。在这个例子中，会话分发部分是 Bob 使用一个 PCM 编解码器用于会话的语音部分来向 Laura 发送一个会话描述符。而 Laura 喜欢使用一个全球移动通信系统（GSM，Global System for Mobile Communications）编解码器，因为它使用较少的带宽，因此她坚持 Bob 像她那么做。Bob 最终设置了一个 GSM 音频编解码器，但是会话在这个协商结束之前不能够被建立起来。

在音频、视频会话的过程中，Laura 突然觉得她今天的发型很糟糕，因此她想要去掉视频部分。她调整会话让它只剩下音频部分。当 Bob 觉得对话该结束时（我不能够猜想为什么），

这个会话就终止了。

就像电话系统通过发出不同的音（忙音或振铃音）来通知一个呼叫者关于他呼叫的他或者她的电话设置状态一样，SIP 向会话启动程序提供他或者她的会话设置程序的信息，如图 4-5 所示。

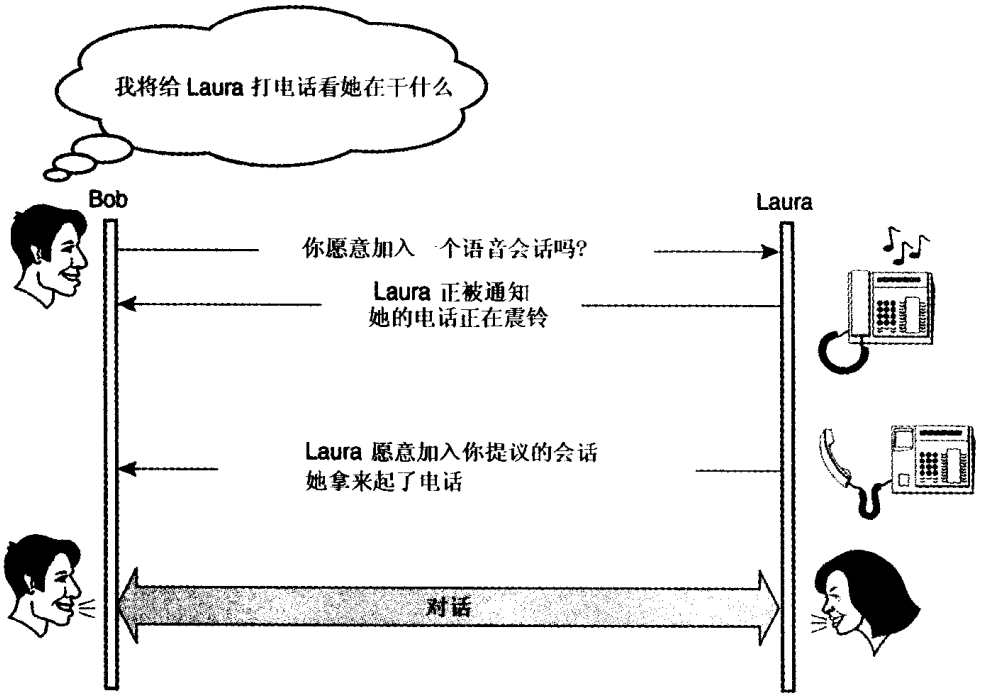


图 4-5 SIP 通告会话建立进展如何

4.2.2 用户可移动性

SIP 在潜在的参与者被定位之前不能向他们传送会话的描述符。经常出现这种情况，单个用户可能在多个位置被访问。例如，一个学生在大学的计算机房上机，通常他每天使用不同的工作站。这样，可以依据不同机器用不同的 IP 地址提供访问途径，并且他们想要仅仅只在他或者她当前所在的位置接收会话邀请。又如，另外一个人想要上午来到办公室的时候在工作站上，晚上回家时在桌面计算机上，旅行的时候在移动终端上接收会话邀请。

SIP URL

我们已经提到 SIP 可以给一些用户提供可移动性。在一个 SIP 环境中的用户使用 SIP 统一资源定位器（URL）标识自己。SIP URL 的格式和一个 E-mail 地址相似，通常由一个用户名和一个域名组成，它看起来像这样：SIP:Bob.Johnson@company.com。

在前面的例子中，如果向 SIP 服务器咨询处理 company.com 这个域名，将会发现一个名为 Bob.Johnson 的用户。Bob 的 URL 可能更改为 SIP:Bob@131.160.1.112，这表明 IP 地址为

131.160.1.112 的主机有一个名为 Bob 的用户。

注册

我们已经注意到，如果用户想被他人找到，则必须向一个服务器登记他当前所在的位置。在这个例子中，Bob 在他的笔记本上工作，笔记本的 IP 地址为 131.160.1.112。他的登录名是 Bob。他向公司的服务器注册他当前所在的位置，如图 4-6 所示。

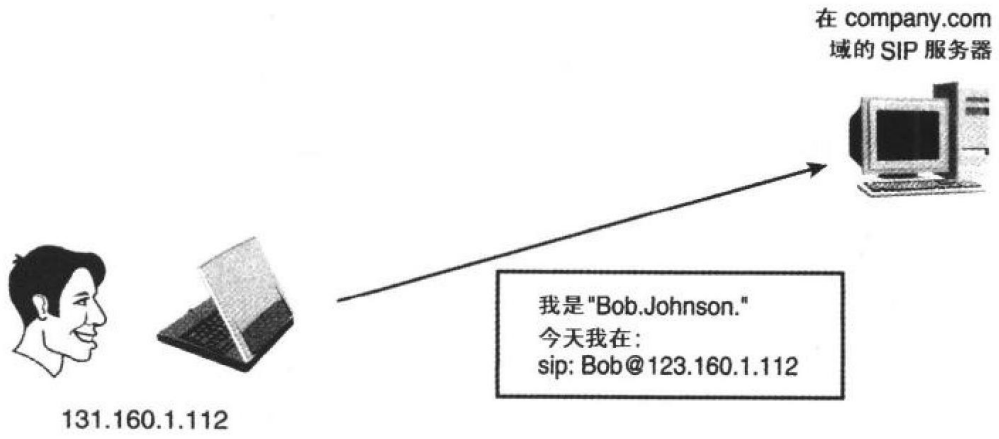


图 4-6　Bob 向服务器注册他当前的位置

现在，Laura 想要呼叫 Bob。她知道他的公开 SIP 地址（SIP:Bob.Johnson@company.com）因为这个地址印在他的名片上。所以当在 company.com 域的服务器被联系上并且查询 Bob.Johnson 时，它就知道 Bob.Johnson 在哪里，因而建立起一条与 Bob 的连接。

在这种情况下，SIP 提供两种操作模式：重定向和代理。在代理操作模式中，服务器用 131.160.1.112 这个地址与 Bob 联系，并且将 Laura 的会话描述符传送给他，如图 4-7 所示。

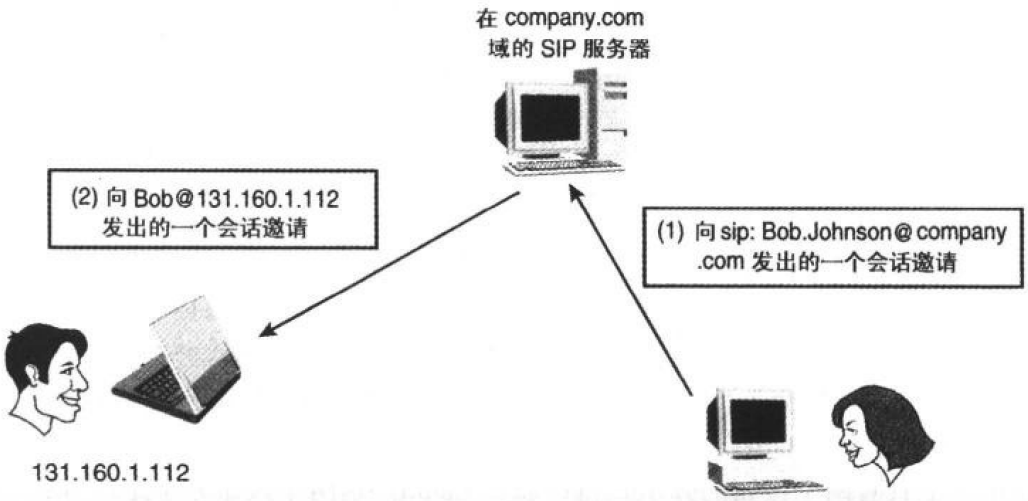


图 4-7　SIP 代理服务器

在重定向模式中，服务器告诉 Laura 尝试使用地址 SIP:Bob@131.160.1.112 与 Bob 联系，如图 4-8 所示。

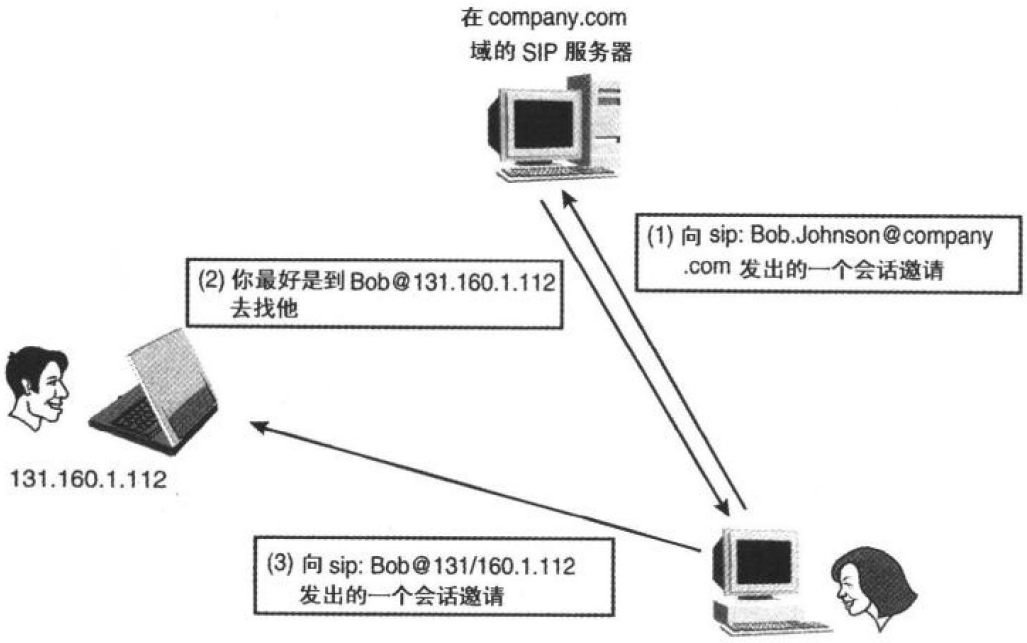


图 4-8　SIP 重定向服务器

用户完全可以在同一个服务器注册多个地址，或者将他或她的地址向多个服务器注册。在用户被最终找到之前，尝试同多个服务器和地址联系的情况是经常发生的。

4.3　SIP 实体

SIP 协议定义了多个实体，理解它们在使用 SIP 协议的体系结构中所起的不同作用是至关重要的。

4.3.1　用户代理

用户代理（UA，User Agent）是一个用于和用户交互的 SIP 实体。它通常有一个与用户连接的接口。比方说 Bob 想要用他的计算机在 Internet 上打电话。他运行一个包含 SIP 用户代理的相应程序。用户通过前面叙述的接口（通常是一个有着各种选择按钮的窗口）和 UA 交互。一旦 Bob 点击呼叫 Laura 的按钮，UA 就触发相应的 SIP 消息建立这个呼叫。

Laura 在她的计算机上也有一个 SIP 用户代理。当这个用户代理收到来自 Bob 的邀请时，它通过显示一个有着两个按钮的弹出式窗口通知 Laura，这两个按钮分别是：“接收来自 Bob 的呼叫”和“拒绝来自 Bob 的呼叫”。Laura 的用户代理依据她所点击的按钮向 Bob 的用户代

理回送消息。用户和 SIP 协议之间的所有交互都由用户代理作为中介进行处理的。

可是，请记住，有些使用 SIP 的系统没有直接面对用户。例如，Bob 可以将从午夜到上午 7 点之间收到的所有会话邀请重定向到他的 SIP 回答机器上。这个机器可以自动建立会话来记录消息。这一机器也包含有一个用户代理——它不必维持与用户的交互，但是它仍然可以作为 Bob 的代表对邀请进行响应或者发送邀请。

普通的唤醒电话就是一个自动建立会话的好例子。在酒店接待处的用户代理计划于时间 *t* 呼叫客人的用户代理。

媒体工具

总而言之，SIP 要给别的 SIP 用户代理发送一个会话描述符。如果这个描述的会话是语音会话，那么用户代理必须把它传送给处理音频的语音工具。对于其他类型的会话，用户代理将把它们交给相应的媒体工具。

SIP 用户代理有时候被并入使用媒体工具处理会话的同一用户界面中。一个音频/视频会话在缺少 SIP 用户代理、音频工具和视频工具的情况下是无法建立起来的。如果上面讲的这三者结合到一个用户界面中，那么在用户看来，它们就如同单个的应用：一个视频会议应用。

将处理会话描述符发送的用户代理和实际处理会话描述符内容的媒体工具区分开来是非常有益的。这一分离使 SIP 可以建立任何类型的会话。

一个 SIP 用户代理看起来像什么？

SIP 用户代理是在许多不同的系统上实现的。例如，用户代理可以在计算机中作为许多应用中的一个运行，或者可以在专有设备上实现，如 SIP 电话。设备类型不会影响 SIP。媒体工具依据产生的会话类型而从一个设备到另一个设备不尽相同，但是 SIP 协议的行为却始终是一样的。

然而，从用户的角度看，SIP 设备可能看起来彼此之间非常不同。这是由于用户界面随着设备的不同而变化。在计算机上运行的视频会议程序的用户界面几乎都是有着可以选择点击的多个按钮的窗口，但是 SIP 电话可能类似于有着 0～9、*号和#号按键的传统电话。SIP 设备可以是有着高带宽连接访问 Internet 的强有力的计算机，也可以是只有低比特率无线连接的小装置。如图 4-9 所示为一些 SIP 设备的例子。

我应该指出，目前将 SIP 用于家庭的改进工作正在进行。因此，将来有 SIP 用户代理的设备可能包括电冰箱、烤箱甚至电灯。

我们将讨论的例子集中于电话是因为它们很容易理解，并且与大多数读者直接相关；不过，请记住，SIP 是一个非常强有力的协议，因为它可以用于建立任何类型的会话。语音会话仅仅只是其中的一个例子而已。

4.3.2　重定向服务器

重定向服务器通过提供可以选择的位置帮助定位 SIP 用户代理，这些位置可以连通用户的

地点。例如，Laura 想要呼叫 Bob。她在她的监视器上点击了呼叫 Bob 的按钮。她的用户代理首先尝试 Bob 的公用地址，但是 company.com 这个域有一个 SIP 重定向服务器处理进入的邀请。因此 Laura 的用户代理改为和这个重定向服务器联系。这个重定向服务器知道 Bob 在办公室工作的时候在 SIP:Bob@131.160.1.112 或者 Bob 在写论文的时候在 SIP:Bob@university.com。因此，重定向服务器将建议 Laura 的用户代理尝试 SIP:Bob@131.160.1.112 和 SIP:Bob@university.com 这两个地址而不是 SIP:Bob.Johnson@company.com。重定向服务器也可以有优先排序的能力，因此它可能告诉 Laura 的用户代理，Bob 最有可能正在学校而不在工作。

图 4-9　有 SIP 用户代理的设备的示例

在收到通知后，Laura 的用户代理尝试通过上面建议的那些 SIP 地址和 Bob 联系。注意重定向服务器并非总是返回用户实际所在地的用户代理地址；它可能改为仅仅简单地返回一个知道更多 Bob 地址信息的服务器的地址，如图 4-10 所示。

这个例子说明了重定向服务器并不发起任何用于定位用户的行动，它只是返回用户有可能出现的那些位置的列表，而由用户代理去进行用户定位的所有尝试。在上面的例子中，是由 Laura 的用户代理尝试所有可能的地址，直到找到 Bob；这一点也就是重定向服务器和代理服务器之间的主要区别。代理服务器代替用户做后续的尝试而不是向用户发送新的联系信息。

组地址

重定向服务器也可以用于实现组地址。为了理解这是如何实现的，举一个例子进行说明。

假设 A 公司支持部的公用地址是 SIP:support@company.com，因为这个部门必须昼夜不停地提供支持，因此某一时段总有某人在工作。Bob 从上午 8 时工作到下午 4 时，Peter 从下午 4 时工作到午夜，而 Mary 从午夜工作到上午 8 时。在 company.com 域的重定向服务器能够依据一天之中时间的不同而返回不同的地址，因此如果它在中午收到了一个目的地址为 SIP:support@company.com 的呼叫，它就自动地以地址 SIP:Bob.Johnson@company.com 回复。

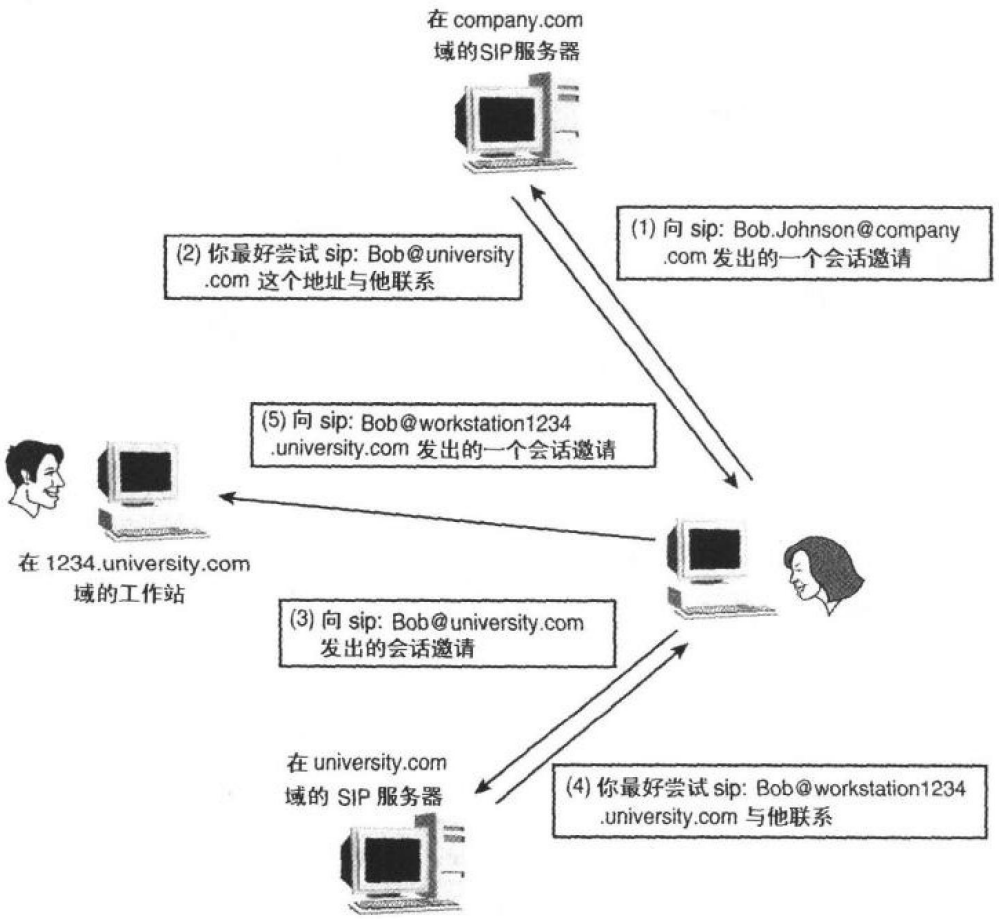


图 4-10　两个重定向服务器的示例

4.3.3　代理服务器

现在假设 company.com 域有一个处理进入邀请的代理服务器。当 Laura 的用户代理尝试地址 SIP:Bob.Johnson@company.com 时，它将到达 company.com 域的代理服务器，这个服务器将迅速代表 Laura 的用户代理尝试地址 SIP:Bob@university.com。如果 university.com 域也有一台代理服务器，那它将尝试地址 SIP:Bob@workstation1234.university.com，在这个地址最终联系到 Bob。这种情况下，Laura 的用户代理只尝试了一个地址，但是有多个代理处于用户

代理之间的路径上，如图 4-11 所示。

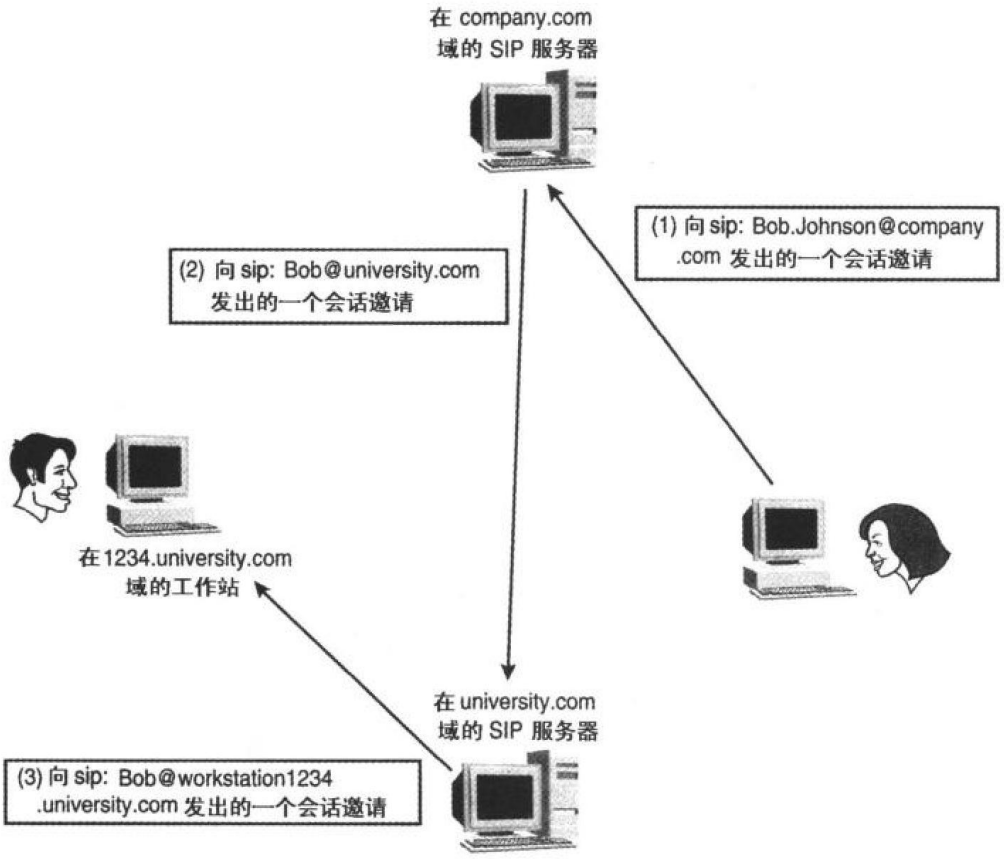


图 4-11　两个代理服务器的示例

派生代理

当代理服务器为用户尝试多于一个地址的时候，它就被叫做派生（Fork）邀请。派生代理可以按照它们的配置处理并行或顺序的搜索。一个并行搜索是同时尝试搜索所有可能的位置，而一个顺序搜索是每次单独地尝试搜索一个位置。

组地址

代理服务器也创建组地址。图 4-12 说明了一个派生代理接收到一个目的地址为 SIP:sales@company.com 的邀请并且尝试与销售部的所有人联系，直到它找到一个可以提供服务的人。

在会话建立期间，两种服务器（代理和重定向）都包含到中间的情况并非是不常见的。通常，术语 SIP 服务器是指这两种服务器而没有按照它们基本行为的不同来区分它们。实际上，同一个 SIP 服务器可以依据具体情形既可作为一个重定向服务器也可作为代理服务器。*例如*，一个 SIP 服务器可以为一个特定的人重定向所有会话邀请，同时使用代理功能处理其余的邀请。

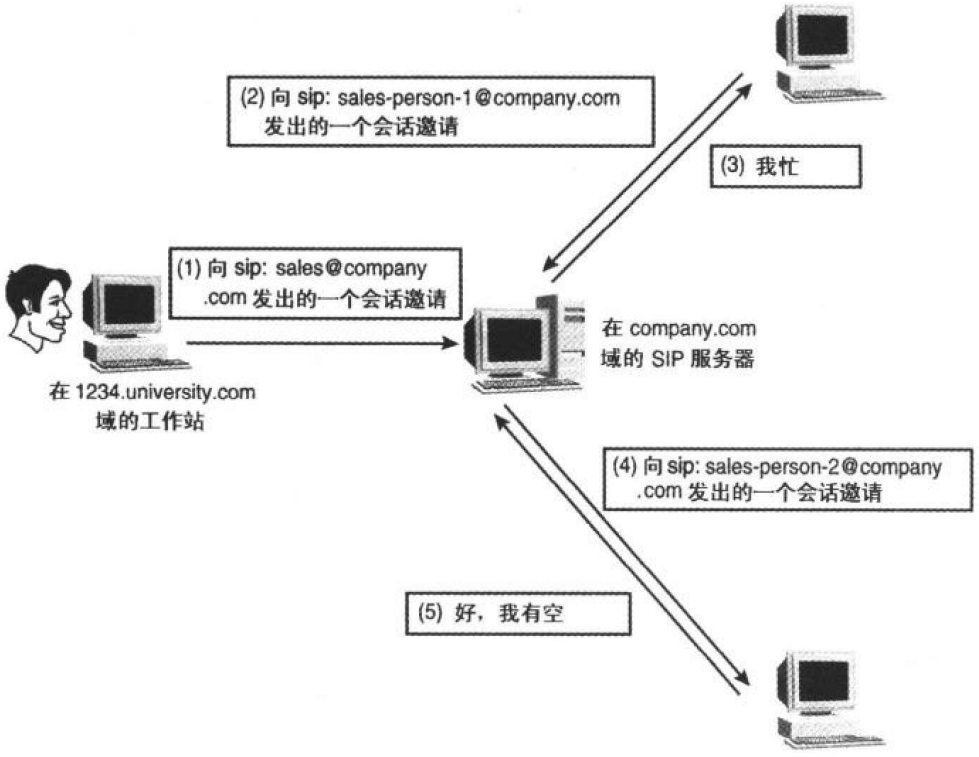


图 4-12　派生代理尝试联系不同的售货员直到它发现有一个不忙的售货员为止

4.3.4　注册员

注册员（registrars）是指一个接受注册的 SIP 服务器。一个注册员通常伴同一个重定向服务器或者一个代理服务器同时出现，如图 4-13 所示。

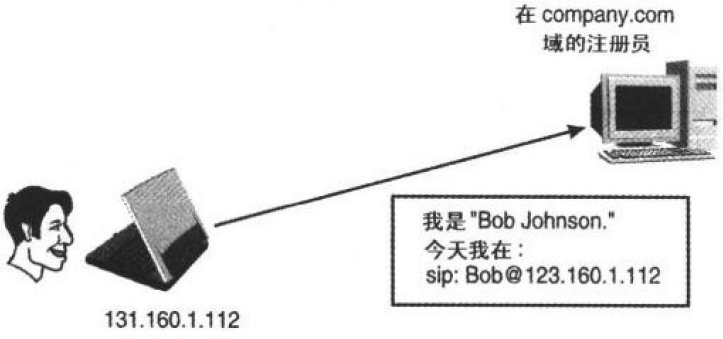


图 4-13　Bob 向注册员注册他当前的位置

4.3.5　位置服务器

位置服务器不是 SIP 实体，但是它们是任何使用 SIP 协议的体系结构中非常重要的一部

分。位置服务器存储并且向用户返回可能的位置信息。它可以利用从注册员或者从其他数据库得来的信息。大部分的注册员接收到位置信息时即刻将这些信息上载到位置服务器。图 4-14 说明了这一过程是如何发生的。在图 4-15 中，在 company.com 域的代理服务器向位置服务器询问 Bob 所处位置的 SIP 统一资源定位器。由于注册员在此以前已经上载了这些信息，所以位置服务器可以向这个服务器提供这些信息。

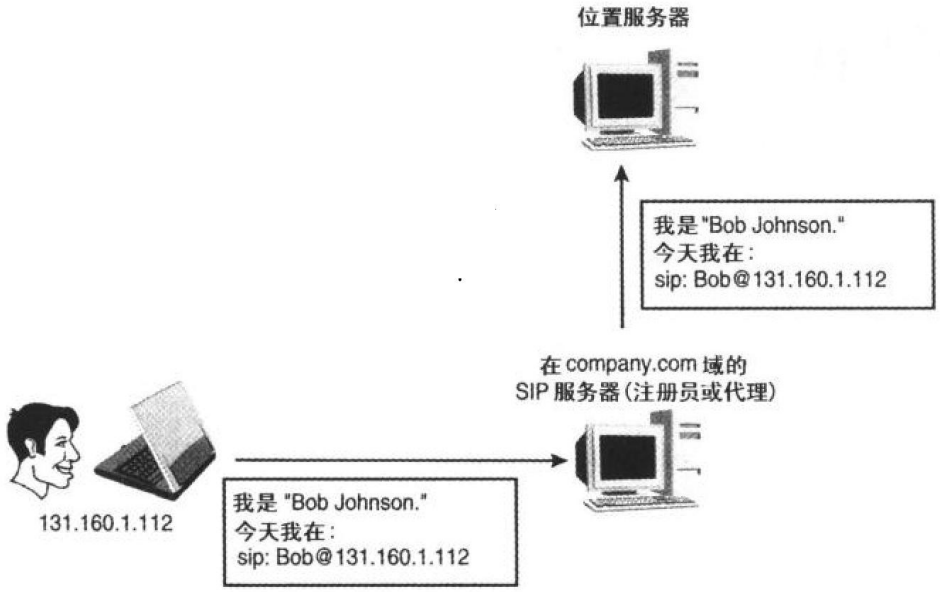


图 4-14　注册员上载信息到位置服务器

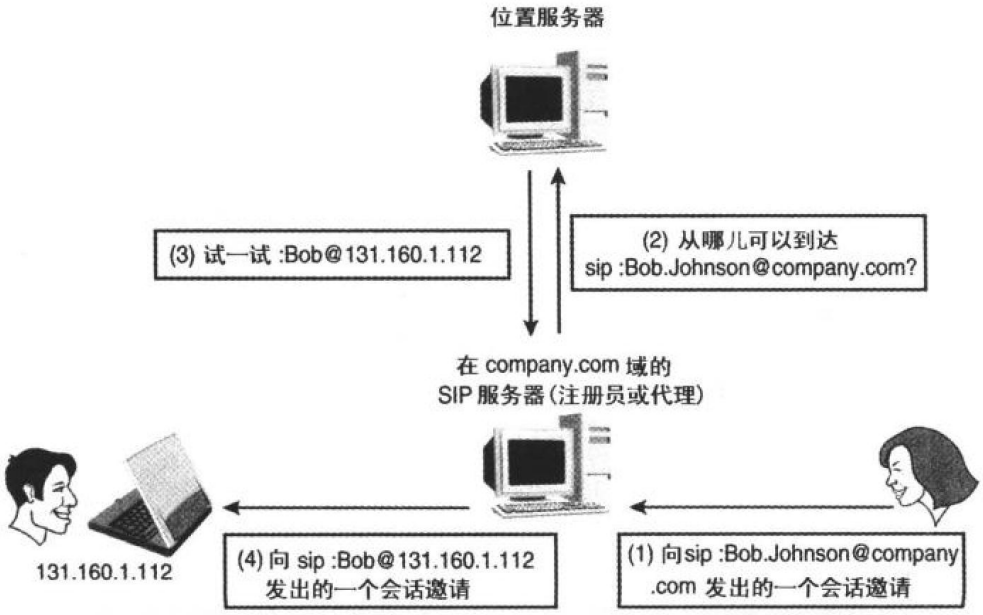


图 4-15　代理向一个位置服务器咨询

但是，在位置服务器和 SIP 服务器之间并不使用 SIP 协议。一些位置服务器使用轻量目录访问协议（LDAP，Lightweight Directory Access Protocol）[RFC 1777]和 SIP 服务器进行通信。

4.4　SIP 好的特性

我们已经看到从 SIP 核心规范可以希望得到的功能和核心规范定义的实体。现在让我们来看看是什么使得 SIP 与别的协议不同并且比它们更好。

4.4.1　SIP 是 IETF 工具包中的一部分

IETF 是以 Internet 作为范例来构思设计 SIP 的。作为 IETF 工具包中的一个工具，SIP 协议行使它的功能并且利用其他的 Internet 机制处理另外的任务。这就提供了很强的灵活性，因为使用 SIP 协议和其他 Internet 协议的系统可以以一种模块化的方式升级。

例如，如果一种新的鉴定机制在 IETF 组织中提议出来，使用 SIP 系统可以使用这种新的机制而无需对 SIP 协议进行修改。一个很好的例子就是 MMUSIC 工作组正在承当的下一代 SDP（SDPng，SDP next generation）的工作。当 SDPng 最后定稿后，今天的使用 SIP 的系统将可以携带 SDPng 会话描述符用以取代现在使用的 SDP 会话描述符。SIP 同样也可以利用新的服务质量（QoS，Quality of Service）机制。我们将这称为未来的验证（Future-Proof）。

4.4.2　建立一个会话和描述一个会话这两个功能的分离

SIP 协议清晰地将会话建立和会话描述区分开来。作为会话建立的一部分，SIP 协议按照要求定位用户，但是它对一旦会话建立后用户可能的行为不进行处理。它并不定义会话将如何描述或者定义会话类型。SIP 仅仅提供连接；至于用户怎样利用它则超出了 SIP 的范围。

这一区分使得 SIP 存在本质上的互操作性。例如，它现在就可以和 SDP 一起用来建立 IP 电话（VoIP，Voice over IP）会话，并且很快就可能将 SIP 协议和新的会话描述协议结合起来，用于建立目前还不存在的会话类型。

我们正在描述的概念并非与那些表征 IP 层特性的概念不相似。我们已经看到 Internet 所提供的最有价值的服务就是 IP 互联。所有其他的服务都是以 IP 互联为基础实现的。就像上面提到的一样，IP 互联如何被使用已经超出 IP 自身范围以外。因此，IP 服务的创建是模块化的、快速的和有效率的。SIP 甚至不假定它所建立的会话将在 Internet 中发生。例如，如果 Bob 想要邀请 Laura 加入一个发生在公用交换电话网络（PSTN，Public-Switched Telephone Network）中的电话会议，他所要做的就是使用 SIP 向 Laura 发送她可以拨打加入的电话号码。在这个例子中，会话描述符包含一个电话号码而不是 IP 地址和 UDP 端口号。当 SIP 将会话描述符传送给 Laura 时，她的反应就像她对待向她发出任何“砰”声音的设备一样。

SIP 提供恰如其分的信息来使被邀请者接受邀请；SIP 在处理它的任务方面是一个 IETF

规范的典范。当需要描述一个用 SIP 建立起来的会话时，应该使用为此目的设计的一个其他的协议而不是 SIP。

4.4.3 端系统的智能：端到端协议

IETF 社团相信用端到端协议能更好地提供端到端的服务，与此同时，IP 也是一个端到端的协议。IP 在一个由多个路由器介入其中的网络的端点之间提供连接。路由器尽可能高效地处理定义明确的数据报的路由任务。同样，SIP 也可利用 SIP 服务器在用户之间提供连接。SIP 服务器有一个相等的定义明确的任务：为基于 Request-URI 的 SIP 请求和基于报文头内容的 SIP 应答寻找路由。

SIP 服务器并不处理包含在 SIP 分组中的会话描述符，因为它们无需为了路由 SIP 报文而那么做。这一点连同这样一个事实，即 SIP 网络中所有的智能都存在于端系统——用户代理中，使得 SIP 成为一种高效协议。SIP 服务器事实上可以是不保留状态的并且可以忘记所有关于它们所移动过的事务，这是因为用于路由一个 SIP 消息的信息保存于这个消息的自身。

4.4.4 互操作性

SIP 协议被设计成协议核心部分的任一实现都可以同任何其他的实现之间实现互操作，并且任何一种实现都合并了协商协议扩展参数的方法，而这些扩展将被用于会话中。两个很高级的 SIP 用户代理之间建立一个会话可能会用到很多扩展和复杂的特性。然而，如果这些高级 SIP 用户代理中的一个需要和一个基本的用户代理之间建立会话，通常也是可以的。所有的 SIP 扩展都被设计成标准组件，这样它们可以被单独协商。用户可以为他发起的第一个会话选择一组特定的扩展，而为下一个会话选择另一组完全不同的扩展。

协商保证了网络中所有 SIP 用户之间存在真实的互操作性。那些是许多语音应用中出现的新东西，在这些应用中，协议（例如，ISDN 用户部分（ISDN User Part，[ISUP]））可能有一些不兼容的部分，需要网关在不能够以相同的 ISUP 协议进行对话的网络之间执行转换功能。任何用网关来对协议进行转换都是不受欢迎的，因为这样做破坏了端到端的模式，并且带有一个协议风格的某些特性在协议转换处理过程中可能会丢失。而与此不同，SIP 是一个真实的全局协议。

4.4.5 可扩展性

SIP 将智能推到了端系统来实现会话期间同时消除在网络中存储有关状态信息的需求。用户在会话建立期间一旦被定位，端系统间的端到端通信就可以进行，而无需服务器的协助。不需要监视会话持续期间信令的服务器就可以处理很大数量的会话。

网络中的某些状态

在下一章中，将看到 SIP 服务器如何按照它们存储的状态信息的数量分类。但是，即使

是保留状态时，存储有状态信息的 SIP 服务器也只是使用在网络的外围，而不保留状态信息的 SIP 服务器则处于网络的核心位置。在这里，它们将不得不处理很大数量的会话。可以说，SIP 网络有着很高的可扩展性是因为它们将不保留状态的操作迁移到了网络中的应力点。

4.4.6　SIP 作为一个创建服务的平台

这是 SIP 无可争辩的最重要的特性。到目前为止，我们说明的所有 SIP 的特性仅仅只是在它们将 SIP 转变为一个用户服务的好的平台时才有用。

组件重用

SIP（有意地）利用许多已经被其他 Internet 应用采用的 Internet 组件。这使得 SIP 成为一种为用户提供方便的各种不同服务的理想协议。特别是，它与 HTTP[RFC 2068]和简单邮件传输协议（SMTP，Simple Mail Transfer Protocol）[RFC 821]之间具有相似性，使得它可以很容易地将目前最成功的 Internet 服务（Web 和 E-mail）与多媒体结合起来。SIP 不仅整合服务，它还将它们传送到用户的真实位置。对于那些研究通信最好方法的人来说，SIP 代表了一个集成服务的解决方案，这是因为 SIP 应用以一种直接的方式整合了 Web 浏览器、E-mail、语音电话、视频会议、存在的信息和即时消息。一些人将 SIP 视为电信工业的下一个重量级应用。

SIP 基于 HTTP

由于显而易见的原因，即前者基于后者，SIP 实现与 HTTP 实现相当类似。两者都使用请求/应答模式，都是基于文本的，都有一种相似的协议消息编码格式。这些相似性使得协议实现可以在这两个协议之间进行代码复用。

对于那些不得不同时提供 Web 浏览器和基于 SIP 服务的设备来说，共享代码的功能是很有用的。一个有无线 Internet 特性的 SIP 移动电话将几乎无疑地实现这两个功能，如图 4-16 所示。这样的设备缺乏你所使用的桌面计算机中的大的硬盘和大量的内存；它们通常是很薄的设备并有着严格的覆盖区域限制，因而通过代码复用所获得的能力是巨大的。

SIP 协议使用统一资源定位器给出 SIP 资源的地址

幸运的是，SIP 用于给出 SIP 实体地址的格式和 Web 上以及 E-mail 系统中使用的格式是相同的。这给了 SIP 重定向以巨大的灵活性并且使得我们能够将多种通信格式整合起来。

一个重定向服务器通常返回一个可选择的用户可能出现的地址，这个重定向服务器实际上返回一个统一资源定位器。在我们大部分的例子中，这个统一资源定位器是一个 SIP 统一资源定位器，但是如果让服务器返回一个 Web 统一资源定位器或者一个 E-mail 地址是毫无问题的。

通过这种方式，Bob 可以配置他的 SIP 重定向服务器，将进入的语音会话送到他的 *E-mail* 信箱中。服务器通过返回mailto:Bob.Johnson@company.com的消息替 Bob 响应一个 SIP 请求，

如图 4-17 所示。Laura，她是以不耐烦出了名的，可以在给 Bob 发送一个语音邮件或者给他写一封 E-mail 这两者之间选择。如果 Bob 将进入的会话重定向到他的 Web 页上，那么她在电话排队等待时就可以看他的新的小狗、汽车或者是房子的图片来消遣了。

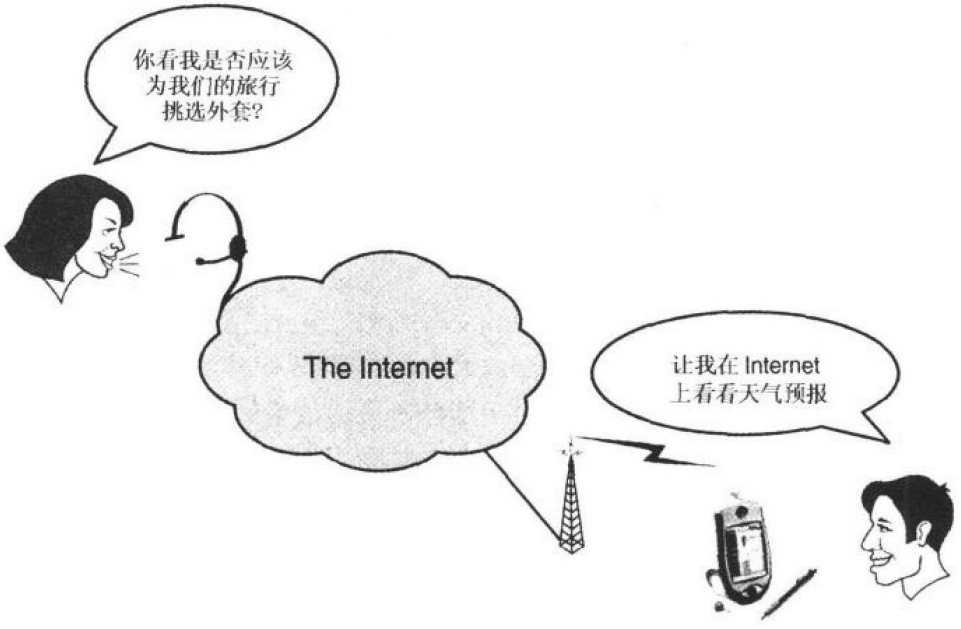


图 4-16　Bob 的终端实现了 SIP 和 HTTP

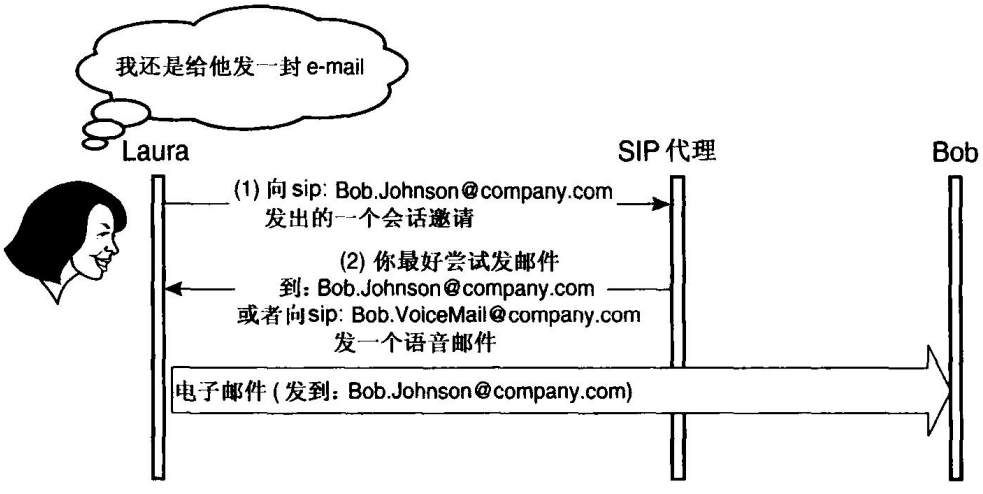


图 4-17　Laura 可以选择发送一个语音邮件或者发送一封电子邮件

Web 页也可以包含除 E-mail 地址外的 SIP 统一资源定位器，提供点击拨号特征。

某个人如果有一个可以重定向到其他任何通信方式的 SIP 统一资源定位器就可以显著地

减少他所需的不同联系信息的数量。当前，一张通常的名片上至少有一个固定电话号码、一个移动号码、一个传真号码和一个 E-mail 地址，如图 4-18 所示。SIP 将所有这些信息提取出来成为一个单独的统一资源定位器。

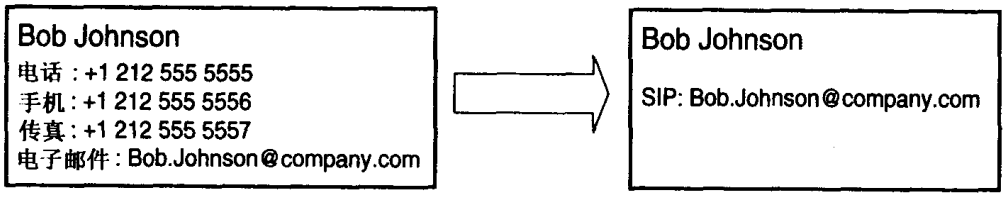


图 4-18　Bob 有一个 SIP 名片

用户可以和我们的 SIP 服务器联系，说明他们需要什么类型的服务（语音电话、电子邮件、传真等），而我们的重定向服务器将给他们提供合适的统一资源定位器。

和 SMTP 一样的路由概念

SIP 消息和邮件消息以几乎一样的方式被路由。它们也可以携带使用多用途 Internet 邮件扩展（MIME，Multipurpose Internet Mail Extensions）[RFC 2045]的多部分消息体。可是，SIP 并不擅长于传输大量的数据，它并不是作为一个传输协议来设计的。

另一方面，它确实很适合传输即时消息，这些即时消息按照定义非常小，可能是紧急的，而且想要在用户的当前位置通知到他们。这一特性也使得 SIP 适合作为一种用于存在（Presence）服务的协议。SIP 注册员应该知道一个用户何时在线，以便将消息传送给他或者她。将存在信息和即时消息结合起来是可以使用 SIP 实现的高级服务的另外一个例子。

SIP 使用现有的框架提供新的服务

例如，一个服务提供商为 IP 电话（VoIP）服务建立了一个 SIP 框架结构，这个框架包括 SIP 代理、重定向服务器和位置服务器。这个服务提供商的客户在新的框架上使用 SIP 用户代理建立语音电话。由于 VoIP 是客户想要的服务，所以服务提供商获得了成功

但是随着时间的推移，客户发现 VoIP 不再有趣，而现在他们真正想要做的是玩交互式游戏。在这一点上，服务提供商不必对网络进行修改或更新；他们仅仅需要获得一个使用游戏会话描述符协议而不是 SDP 协议来建立用户会话的更新的 SIP 用户代理。这是因为网络中的 SIP 服务器忽略了会话描述符的具体内容，以 VoIP 为中心建立的整个 SIP 框架结构可以立刻改变用途而不用对网络做任何改动。

如何对 SIP 应用编程的广泛传播的知识

对 SIP 应用编程需要特定的技巧，这些技巧已经在程序员中广泛传播。大多数程序员已经习惯于使用 Web 应用程序、文法分析程序和脚本语言，这些都是编写 SIP 应用程序代码时能确实起作用的。

另外，由于 SIP 消息是可读的，所以被设计用来揭示两个不同的协议实现不能够正确交互的特定协议分析器没有存在的必要的原因。实际上，为了创建一个 SIP 服务并不需要特别的知识。新从大学出来的程序员已经具备了开发新应用服务的一切知识，只要他们具有想象力。不久以前，这一点跟事实还差得很远。当时需要非常专业的知识，而且一旦服务被实现了，它会很难测试（并非所有的人能够轻易获得必须的电话交换）。今天，任何个人计算机都可以作为一个 SIP 服务器用于测试新的应用程序。

这有一个很重要的蕴含意义。它意味着创建一个特定 SIP 服务的人也是在那些特定服务方面有专长的人。也就是说，创建游戏应用的团体了解这些游戏，而开发消息应用的团体将会理解这些消息。今天的编程实践（非常令人感叹）就是找到一个了解协议的程序员并且教会他或者她应用程序是如何工作的。SIP 使得知道一个确定社团功能需求的人可以开发他们自己的服务。

SIP 使得应用可以分解

SIP 使我们能够将简单应用结合到很复杂的服务中[draft-rosenberg-sip-app-components]。例如，一个人想写一个应用程序，它用西班牙语以文本的形式输入，然后将输入文本翻译成英语，最后以语音方式将其输出，可以将这个解决方案看成是几个简单应用程序在一起工作。第一个应用程序将输入的文本从西班牙语翻译成英语；第二个应用程序将英语文本作为输入并将它转换成英语发音然后输出。这个例子包含两个只处理相对简单事务的应用服务器，它们可以被以很多种方式裁剪来满足用户的需求。用户使用 SIP 信令来协同双方的应用服务器以期得到希望的整体的结果。

第5章 SIP：协议操作

本章进一步描述 SIP 的细节问题。介绍 SIP 功能是如何实现的，即在不同的 SIP 实体之间交换什么消息以及这些消息的格式是怎样的。我们也将对多个关于 SIP 工作机制的例子进行分析。这些例子通常由一个消息流组成，这些消息流给出了 SIP 操作的全貌。

本章也选了一个使读者可以看到所有消息细节的例子，包括消息标题头、参数和会话描述。尽管分析所有的协议细节不是本书的目的，但是在此看到 SIP 消息究竟是怎样的对读者还是有益的。

5.1 客户端/服务器事务

SIP 基于 Web 协议即超文本传输协议（HTTP，Hypertext Transfer Protocol）。与 HTTP 协议一样，SIP 也是一个请求/应答协议。为了理解 SIP 中使用的请求/应答机制，先介绍一下关于客户和服务器的定义。

客户端是产生请求的 SIP 实体，而服务器是接受请求和返回应答的 SIP 实体。这个定义是从 HTTP 协议继承过来的，Web 浏览器包含有一个 HTTP 客户端。当在 Web 浏览器中键入一个地址，如http://www.accessmhtelecom.com的时候，就向一个特定的 Web 服务器发送了一个请求。这个 Web 服务器发回一个应答，包含着所请求的信息，也就是 McGraw-Hill 电信出版集团的 Web 页。

SIP 的过程与此相同。按照上述定义，当两个用户代理交换 SIP 消息的时候，发送请求的用户代理（UA）就是用户代理客户端（UAS，User Agent Client）而返回应答的用户代理则是用户代理服务器（UAS，User Agent Server）。SIP 请求连同一个它所触发的应答被叫做一个 SIP 事务。

5.1.1 SIP 应答

当服务器收到请求的时候，它就发出一个或多个应答。每一个应答都有一个代表事务状态的编码。状态码是 100～699 的整数并且被分成组，见表 5-1。

表 5-1　　SIP 响应类型

范　围	应答分类
100～199	报告
200～299	成功

续表

范　围	应答分类
300～399	重定向
400～499	客户端错误
500～599	服务器错误
600～699	全局错误

带有从 100～199 之间的状态码的应答被认为是临时的。从 200～699 的应答是最终的应答。处于端和一个服务器之间的 SIP 事务包括客户端发出的请求、一个或多个临时的应答和一个最终的应答。（读者在下一节将会看到，这条规则有一个例外。）

SIP 应答同状态码一起，也携带一个原因短语。后者包含关于状态码的人可读懂的信息。例如，编号为 180 的状态码的意思是通知用户加入一个会话。因此，这个原因短语可能包含“振铃”这一词语。当然，由于原因短语将由人来读，所以它可以用其他的语言来写而不是用英语。与此对照，计算机处理 SIP 应答时忽略原因短语。它可以从应答代码中找到足够的信息。可是，它可以向用户显示这个原因短语，这些用户将发现：知道远端的 SIP 电话正在振铃比知道收到了有 180 状态码的应答更有用。（从现在起，我们将使用状态码后紧跟着原因短语的形式来引用一个应答，例如，“180 振铃”）表 5-2 包括了目前定义的所有状态码以及与之对应的缺省的原因短语。

表 5-2　　SIP 应答编码

Trying　尝试	Request entity too large　请求实体太大
Ringing　振铃	Request-URI too large　请求 URI 太大
Call is being forwarded　呼叫正被转发	Unsupported media type　不支持的媒体类型
Queued　排队	Bad extension　坏的扩展
Session progress　会话进行	Temporarily not available　暂时没空
OK　好	Call leg/transaction does not exist　呼叫事务不存在
Accepted　接受	Loop detected　检测到循环
Multiple choices　多种选择	Too many hops　太多跳数
Moved permanently　永久清除	Address incomplete　地址不完全
Moved temporarily　暂时清除	Ambiguous　不明确
Use proxy　使用代理	Busy here　这儿忙
Alternative service　可选择服务	Request cancelled　请求取消
Bad request　坏的请求	Not acceptable here　这儿不能接受
Unauthorized　未授权	Internal server error　服务器内部错误
Payment required　需付报酬	Not implemented　没有实现
Forbidden　禁止	Bad gateway　坏的网关
Not found　没有找到	Service unavailable　服务不能提供

续表

Method not allowed 方法不允许	Gateway time-out 网关超时
Not acceptable 不能接受	SIP version not supported　SIP 版本不支持
Proxy authentication required	需要代理鉴定
Busy everywhere	各处忙
Request time-out　请求超时	Decline 下降
Conflict 冲突	Does not exist anywhere 不存在
Gone 离开	Not acceptable 不能接受
Length required	需要的长度

5.1.2　SIP 请求

SIP 核心规范定义了 6 种 SIP 请求，其中每一种都有不同的作用。每个 SIP 请求都含有一个称为方法（Method）的字段，它表示这个请求的目的。下面列出这 6 种方法：（1）INVITE（邀请）；（2）ACK（确认）；（3）OPTIONS（可选项）；（4）BYE（再见）；（5）CANCEL（取消）；（6）REGISTER（注册）。

我们将在后续章节中看到 SIP 核心规范的一些扩展是如何定义附加方法的。

请求或应答都可以包含 SIP 消息体。消息体就是它的负荷。SIP 消息体通常包含一个会话描述符。

1．邀请（INVITE）

INVITE 请求用于邀请用户参与一个会话。INVITE 请求的消息体包含有会话描述符。例如，当 Bob 呼叫 Laura 时，他的用户代理向 Laura 的用户代理发送一个包含有会话请求的 INVITE 消息。假设 Bob 的用户代理使用会话描述协议(SDP)描述会话，她的用户代理会收到有如下会话描述符的 INVITE 消息：

```
v=0
o=Bob 2890844526 2890842807 IN IP4 131.160.1.112
s=I want to know how you are doing
c=IN IP4 131.160.1.112
t=0 0
m=audio 49170 RTP/AVP 0
```

Laura 的用户代理收到的 INVITE 消息表示 Bob 正邀请 Laura 加入一个语音会话。根据 INVITE 消息中携带的会话描述符，Laura 的用户代理知道 Bob 想要在 IP 地址 131.160.1.112，UDP 端口 49170 接收包含 Laura 声音的实时传输协议（RTP）数据报。她的用户代理也知道 Bob 可以接收脉冲编码调制（PCM，Pulse Code Modulation）编码的声音（在 *m* 行的 RTP/AVP 0 表明了 PCM）。

Laura 的用户代理开始通知 Laura 并且回送一个“180 振铃”作为给 Bob 用户代理的应答。

当 Laura 最后接受了这个呼叫，她的用户代理就会返回一个“200 OK”的应答，其中包含有一个会话描述符。

```
v=0
o=Laura 2891234526 2812342807 IN IP4 138.85.27.10
s=I want to know how you are doing
c=IN IP4 138.85.27.10
t=0 0
m=audio 20000 RTP/AVP 0
```

这时，Laura 接受这个呼叫并通知 Bob 她将在 138.85.27.10，UDP 端口 20000 接收 RTP 分组，如图 5-1 所示。

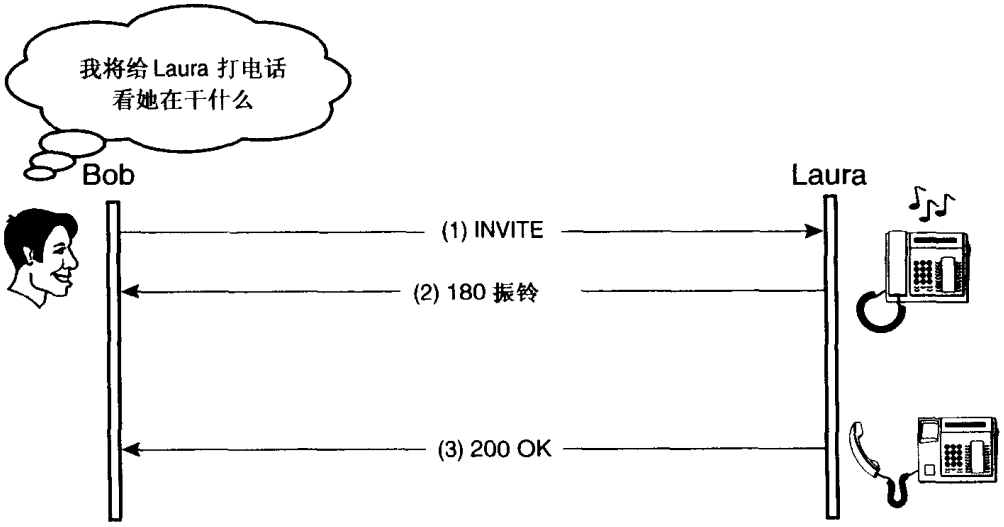


图 5-1　Laura 对她收到的邀请发出一个最终的应答（200 OK）。

如果 Laura 和 Bob 都处于会话进行中，他俩中的一个想要调整这个会话，他们仅需要发出一个新的 INVITE 请求。这种类型的 INVITE 请求叫做重邀请（re-INVITE）请求，它携带一个更新的会话描述符。它可能包含新的参数，如已存在媒体的端口号，或者它可以加入的新的媒体流。例如，Bob 和 Laura 可以通过重邀请请求将一个视频流加入到他们的语音会话中。

值得注意的是，SIP 仅仅处理对用户的请求和用户对请求的接收。所有的会话细节都由使用的会话描述协议处理（在此情况下是 SDP）。这样，SIP 可以使用一个不同的会话描述符邀请用户参与任何类型的会话。

2．确认（ACK）

ACK 请求用来确认对一个 INVITE 请求的最终应答的接受。这样，产生 INVITE 请求的客户端收到对 INVITE 请求的最终应答时，就发出一个 ACK 请求，提供一个三次握手过程：INVITE－最终应答－ACK，如图 5-2 所示。

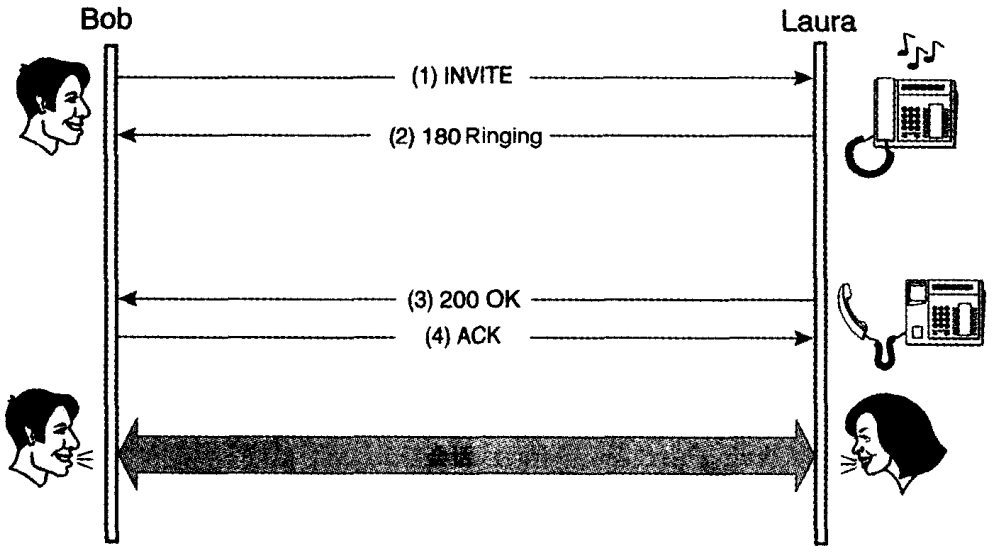


图 5-2　三次握手：INVITE－200 OK－ACK

为什么 SIP 使用三次握手机制？

INVITE 是仅有的使用三次握手机制的方法，与之对应的是两次握手机制（METHOD－最终应答）。这个确定的特性使得可以使 INVITE 方法同其他方法区分开来。当客户端发出一个不是 INVITE 请求的其它请求时，它希望从服务器得到快速的应答。可是，由 INVITE 请求引发的应答可能要持续很长一段时间。当 Bob 呼叫 Laura 时，她可能要从她的外套中掏出她的 SIP 电话，然后按键，因此将收到的“200 OK”应答多少有些延迟。当应答收到时，从客户端发送一个 ACK 到服务器，使得服务器知道客户端仍然在那里，并且会话已经成功建立。

三次握手机制也使派生代理的实现成为可能。当这些代理中的一个派生一个请求时，发出请求的客户端将从不同的服务器得到多个应答。向每一个发送应答的目的服务器发送一个 ACK 请求对于保证在诸如 UDP 等不可靠的协议上的 SIP 操作是至关重要的。

除了可以迅速建立会话并进行派生之外，INVITE 的三次握手也使我们可以在没有一个会话描述符的情况下发送一个 INVITE 请求，这个会话描述符可以在此之后包含在 ACK 中发送。这个特性是很有用的，例如，当 SIP 协议和其他使用不同消息顺序的信令协议交互工作时，就用到了这个特性。

使用三次握手机制的历史原因可以在旧的 SIPv1 草案中，即在关于如何提供会话邀请的可靠传输部分找到。这个草案引入了 ACK 方法来在会话建立过程中避免出现不同步的部分，这一情况可能发生在当一个二次握手被使用于不可靠的传输协议（如 UDP）之上的时候。考虑下面的实例，在这个例子中实现了一个二次握手过程。

Bob 将向 Laura 传送一个 INVITE 请求，并且重传它，直到从 Laura 处收到一个最终应答。在收到最终应答之前，Bob 并不能知道 Laura 是否收到了 INVITE 请求或是这个请求在网络

中是否丢失了。

Bob 等了一段时间，由于没有得到回答，他于是放弃并且停止重传 INVITE 请求。Bob 相信没有会话建立。

几乎同时，Laura 收到了 Bob 的呼叫，且回送一个“200 OK”应答。如果这个应答丢失了，Bob 将永远收不到它，因此，Bob 仍然认为没有会话已经被建立。因为 Laura 观察到 Bob 已经停止重传 INVITE 请求了，她设想 Bob 也收到了她的“200 OK”应答。因此，Laura 会误认为这个会话已经成功建立，如图 5-3 所示。

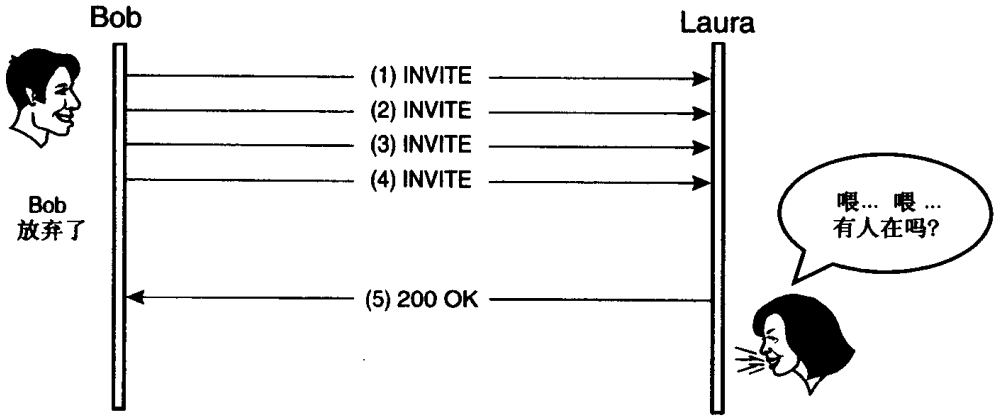


图 5-3　这种情况通过一个三次握手过程就可以避免（INVITE－200 OK-ACK）

在这种情形下，如果用一个三次握手过程代替的话，Laura 将不会收到对她的“200 OK”应答的 ACK，这个“200 OK”应答是 Laura 在 Bob 放弃之前发出的。因此，她将（正确地）认为会话没有建立。

3．取消（CANCEL）

CANCEL 请求用于取消悬而未决的事务。如果 SIP 服务器已收到一个 INVITE 请求但仍没有返回一个最终应答，那么它将停止处理这个 INVITE 请求，直至收到一个 CANCEL 为止。可是，如果它已经为这个 INVITE 请求返回了一个最终应答，那么 CANCEL 请求将对这个事务无效。

在图 5-4 中，Bob 呼叫 Laura，于是她的 SIP 电话开始振铃，但过了一段时间没人接电话。Bob 决定挂机。他为先前发送的 INVITE 请求又发送一个 CANCEL 请求。一旦收到这个 CANCEL 请求，Laura 的 SIP 电话便停止振铃。服务器为这个 CANCEL 请求返回一个“200 OK”应答，表明它处理成功。

注意到这一点非常重要，即在服务器对 CANCEL 请求作出响应之后，它也对先前的 INVITE 请求作出响应。它发送一个“487 事务取消”应答，而客户端通过发送一个 ACK 结束这个 INVITE 三次握手过程（INVITE－487，事务取消-ACK）。因此，*INVITE 请求的三次握手过程总是会进行的，甚至在事务被取消的时候也一样。*

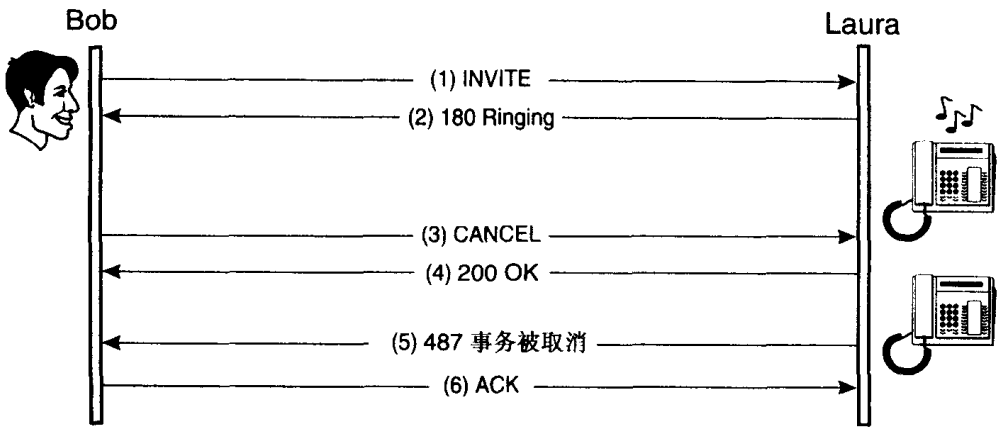


图 5-4　Bob 取消他的请求

当派生代理（指那些仅收到一个 INVITE 而发出多个 INVITE 的代理）存在于传送路径途中时，CANCEL 请求非常有用。当派生代理正在处理一个并行查找时，它尝试同时查找多个位置。比如，一个派生代理知道能找到 Bob 可能出现的 3 个地点：SIP：Bob@131.160.1.112、SIP:Bob:Johason@company.com 和 SIP:Bob@university.com。当这个代理从 Laura 那里收到一个 INVITE 请求邀请 Bob 时，它将并行地（在同一时间）尝试这 3 个地址。这个派生代理发送 3 个 INVITE，每个位置发送一个。Bob 现在正在 131.160.1.112 工作，回答这个呼叫。这个派生代理从 SIP：Bob@131.160.1.112 收到了一个“200 OK”应答，并且将这个应答发送给 Laura 的用户代理。由于这个会话已经在 Laura 和 Bob 之间建立起来了，派生代理需要停止已开始的其他查寻，所以它发送两个 CANCEL，每个位置一个，以关闭查寻，如图 5-5 所示。

请记住，一旦最终应答被送出之后，CANCEL 请求就对事务无任何影响了。因此，在我们的例子中，甚至假如派生代理向 SIP：Bob@131.160.1.112 也发送一个 CANDEL，在 Bob 和 Laura 间的会话仍将持续。CANCEL 不能够结束一个正在进行的会话。它会被已经完成的事务忽略掉。

4．再见（BYE）

BYE 请求用于放弃事务。在双方会话中，参与者其中一方如果放弃，就意味着会话结束。例如，当 Bob 向 Laura 发送一个 BYE 请求，他们的会话就自动结束了，如图 5-6 所示。但是在组播情况下，从某个参与者发出的一个 BYE 请求仅仅意味着某个特定的参与者自己退出会议，会话本身不受影响。实际上，在一个大型的组播会话中某个参与者自行离开会话而不发送一个 BYE 请求也是通用的做法。

5．注册（REGISTER）

用户发送 REGISTER 请求向服务器（在这里指一个注册员）通知他们当前所处的位置。Bob 可以向在 company.com 域的注册员发送一个 REGISTER，指示所有进入的目的地址为 SIP：Bob.Johnson@company.com 的请求都被代理处理，或重定向到 *SIP:Bob@131.160.1.112*，如图 5-7 所示。

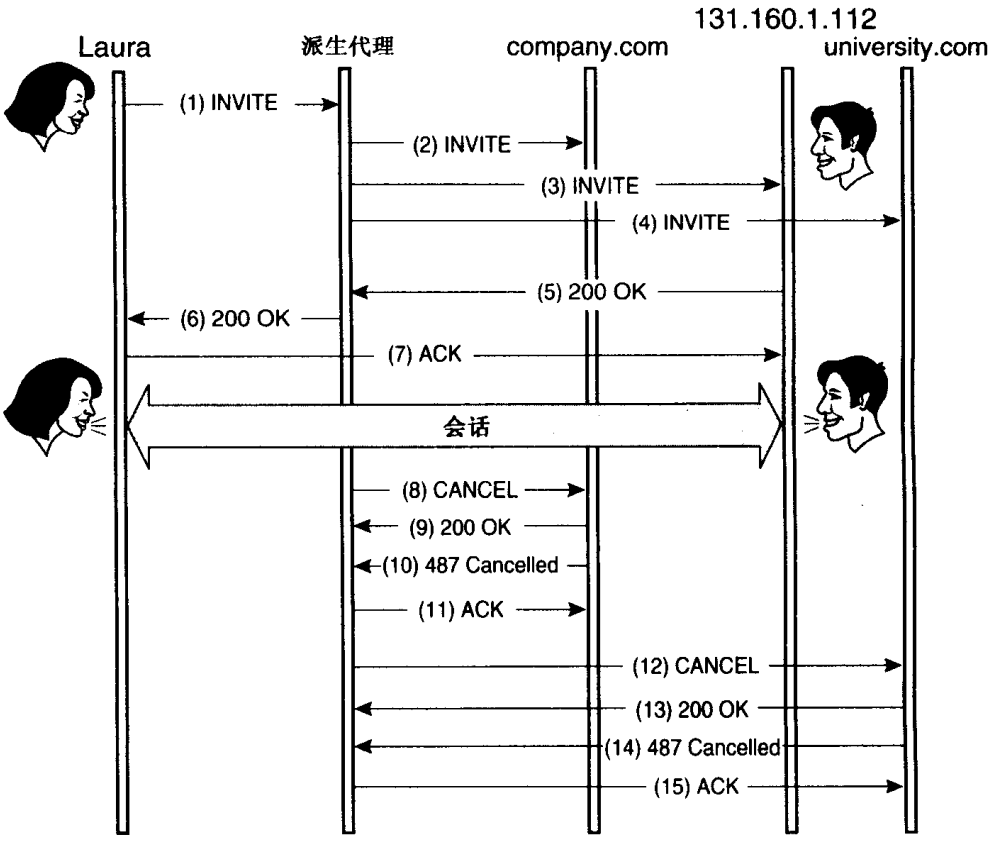


图 5-5　代理取消 INVITE 事务

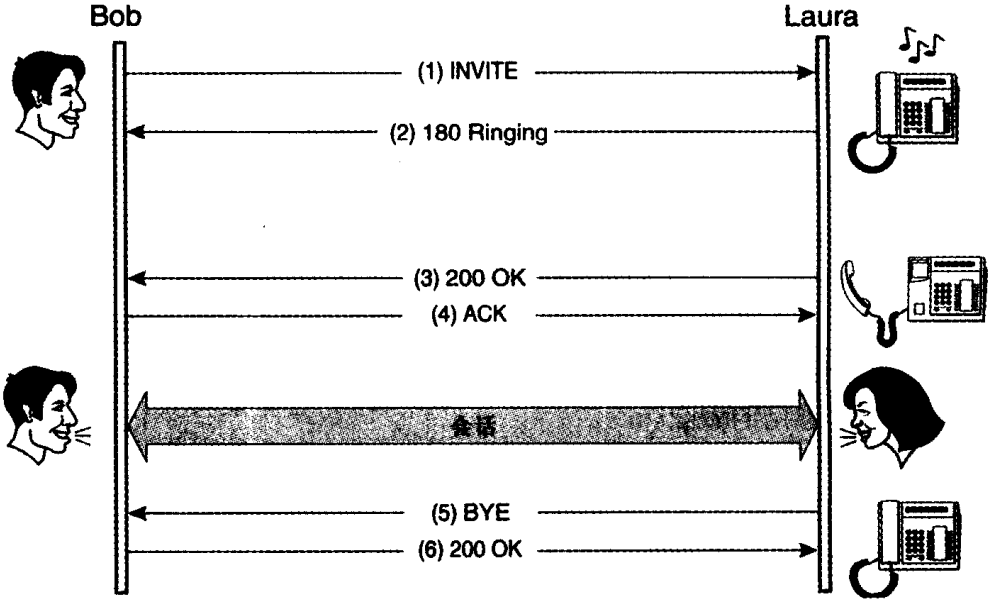


图 5-6　Laura 挂电话时发送一个 BYE 请求

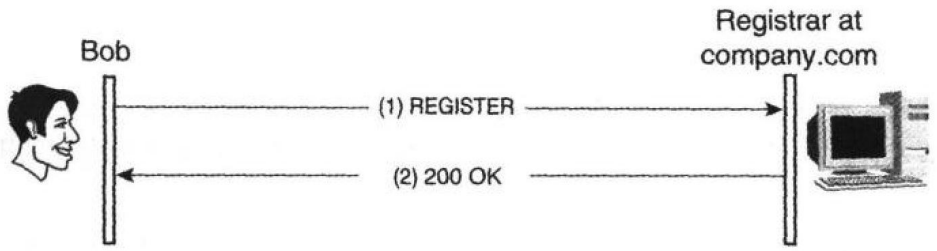


图 5-7　Bob 向在 company.com 域的注册员登记他当前的地址

SIP 服务器通常与 SIP 注册员在一起。SIP 注册员可以向单个的定位服务器发送所有的在各种 REGISTER 请求中收到的信息，使得这个定位服务器可以向任何寻找用户位置的 SIP 服务器提供服务。

REGISTER 消息也包含注册附属的时间。例如，Bob 可以注册他当前的位置直至下午 4 点钟，因为他知道那时他将离开办公室。一个用户也可以同时在几个地点被注册，通过这种方法告诉服务器：它应该在所有注册地点查找这个用户，直到这个用户被找到为止。

6．可选项（OPTIONS）

OPTIONS 请求用于寻问服务器的性能情况，如图 5-8 所示，包括这个服务器所支持的方法和会话描述协议。SIP 服务器可能回答 OPTIONS 请求它支持 SDP 作为会话描述协议，且它支持 5 种方法：INVITE、ACK、CANCEL、BYE 和 IPTIONS。由于这个服务器不支持 REGISTER 方法，故可以断定它不是一个注册员。OPTIONS 方法现在看来好像没用，但是当新的扩展将新的方法添加到 SIP 中时，OPIONS 方法就是一种发现特定服务器支持何种方法的非常好的方式。

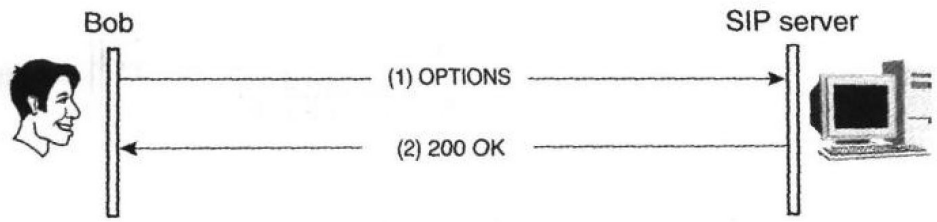


图 5-8　Bob 询问一个服务器关于它的性能

OPIONS 方法也返回用于描述服务器可以理解的消息体编码数据，例如，如果确定服务器懂得一种确定的压缩方法，那么客户端就可以发送压缩的会话描述符，从而可以节省带宽。

5.2　代理服务器的类型

代理服务器可以按照某个会话进行中它们存储的状态信息的数量分类。SIP 定义了 3 类代理服务器：保留呼叫状态代理、保留状态代理和不保留状态代理。

5.2.1 保留呼叫状态代理

保留呼叫状态代理需要知道在会话过程中发生的所有 SIP 事务，因此它们总是处于在端用户间传输 SIP 消息所选取的路径中。这些代理存储从会话建立时起直到会话结束那一刻为止的所有状态信息。

保留呼叫状态代理的一个例子就是实现了某一项呼叫相关服务的服务器，例如在每个呼叫结束后收到一封有关每个电话持续时间信息的电子邮件，如图 5-9 所示。为了计算电话持续时间，这个代理必须在初始化这个电话 INVITE 请求发送的路径的同时，结束这个电话的 BYE 请求的传送路径。

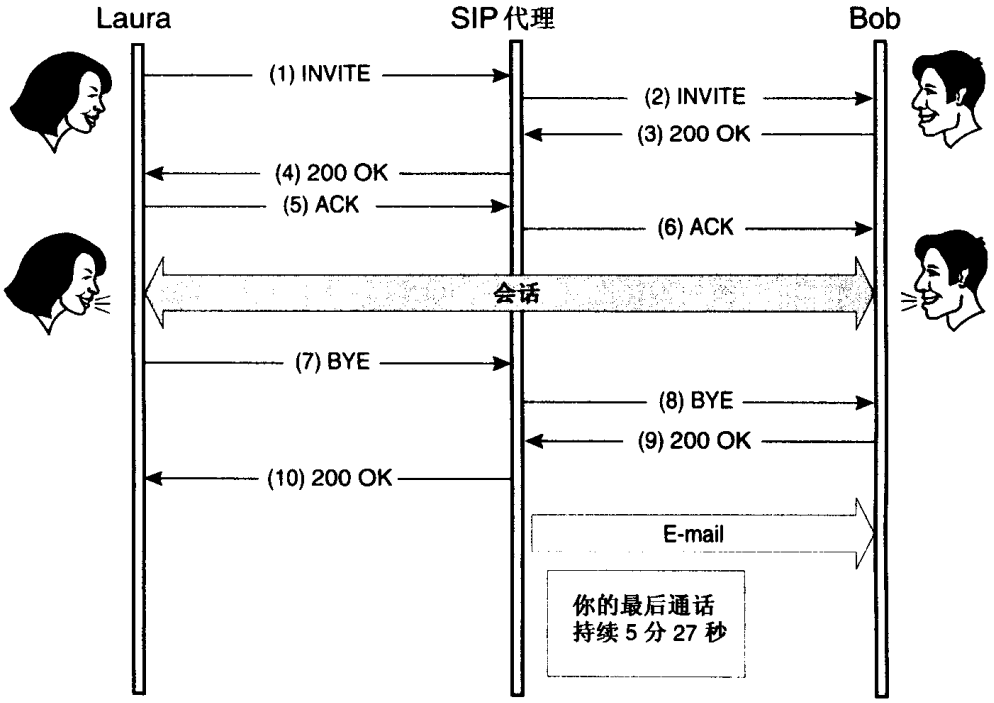


图 5-9 保留呼叫状态代理向用户提供服务

5.2.2 保留状态代理

保留状态代理有时被称做事务状态代理，这是因为事务是它们唯一关心的内容。保留状态代理存储与一个给定事务相关的状态信息直到这个事务结束。它不需要位于那些为后述事务发送的 SIP 消息传送的路径中。

派生代理是保留状态代理的好例子，如图 5-10 所示。它们往几个不同的地方发送 INVITE 请求，并且为了知道自己所试的所有地址是否都返回了一个最终的响应而不得不存储有关 INVITE 事务的状态信息。但是，一旦用户在某个特定的位置被找到了，这个代理就不需要

在信令路径上存在了。

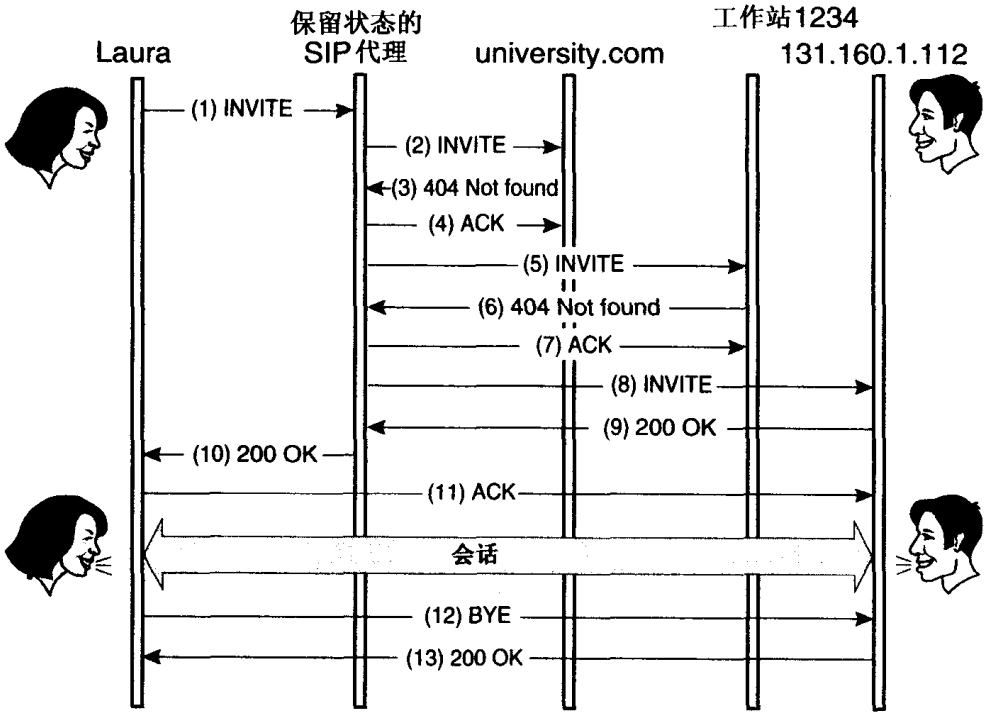


图 5-10　保留状态的派生代理

ACK 的产生

图 5-10 说明了在 SIP 协议中 ACK 是如何产生的。从图 5-10 中可以看出，ACK 是作为对 INVITE 的最后应答产生的。它是三次握手机制的一部分，并且依据服务器返回的应答类型的不同，或是由代理产生或是由 UAC 产生。

代理服务器只能确认非成功（Non-Successful）的最终应答，这些应答有一个大于 299 的状态码。成功应答（状态码在 200～299 之间）总是由初始化 INVITE 的用户代理确认。

如图 5-10 所示，代理服务器在消息（4）和（7）中确认非成功应答。而用户代理在消息（11）中确认“200 OK”应答。这使得一个代理在最终找到用户之前可以尝试多个位置而无需通知最初的用户代理关于定位用户不成功尝试的情况。一旦某个用户积极响应 INVITE 请求，源用户代理必须接收来自远端的会话描述符，以建立会话。

5.2.3　不保留状态代理

不保留状态代理不保存任何状态信息。它们接收一个请求，将它发往下一跳，并且立即删除与那个请求相关的所有状态信息。当不保留状态代理收到一个应答时，它仅仅基于通过 Via 标题头(Via header)的分析来决定路径，并且它不为之维持状态（我们将在本章稍后部分

讨论 Via header）。

5.2.4 代理分发

在网络中有关 IP 流量的分析总是表明，网络核心比网络边缘要拥挤得多。SIP 流量的真实情况也是如此。

在网络核心的 SIP 服务器需要能够处理许多消息，而处在网络边缘的 SIP 服务器就不需要支持相同的重负载。SIP 被设计用来在网络核心使用不保留状态服务器。它们基于请求-URI 或 Via 标题头处理路由，就像我们所知道的一样快速和高效。在网络边缘，保留呼叫状态和保留状态服务器可以基于更复杂的变量（如一天中的时间变量或在 From 字段中的发送者标识）处理路由，或者它们可以派生请求并且向用户提供服务。

分发服务器的这种方法使 SIP 成为一个扩展性能非常好的协议，它可以在增长的大型网络如 Internet 中实现。SIP 使网络核心快速、简单，同时将智能推向网络的边缘，如图 5-11 所示。

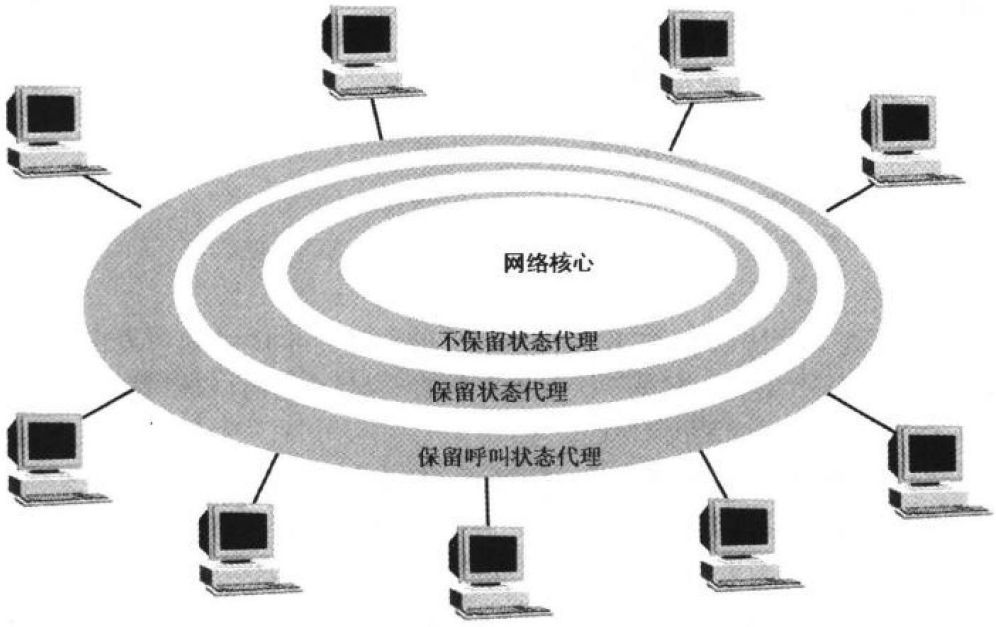


图 5-11 SIP 保持网络核心无状态

5.3 SIP 消息格式

协议设计以一种不连续的状态进行。当已决定在分布系统之间交换什么信息之后，下一步就是决定这些信息如何编码。这个决定有两种基本方法：二进制方法，它使用比特字段编码信息；文本方法，它使用字符串。下面的例子说明了这两种方法的差异所在。

用户需要在他们的计算机上明确当前的月份，而在网络中的一个服务器就有这个信息。我们需要一种协议将这个信息从服务器传送到桌面计算机上。当前月份这个字段可以确切地取 12 个可能值：一月、二月、三月、四月、五月、六月、七月、八月、九月、十月、十一月和十二月。

基于文本的协议将在系统之间传输月份名。假设消息的内容是一月（January），每个字符（字母）通常用一个字节（8bit）编码。这样，January 消息将使用 56bit（7 个字母乘以 8bit）编码。

相反，二进制协议将定义一个可能值的表和这些可能值的相应编码，见表 5-3。

表 5-3　　月份的二进制编码。

0000 January　一月	0110 July　七月
0001 February　二月	0111 August　八月
0010 March　三月	1000 September　九月
0011 April　四月	1001 October　十月
0100 May　五月	1010 November　十一月
0101 June　六月	1011 December　十二月

这样，为了传输当前的月份，二进制协议将要传送一个 4bit 包含有 0000 编码的消息。

SIP 使用与二进制相对的文本编码，这个问题引起了激烈的争论。文本与二进制之争看起来像一种类似于信仰的争论，在这个争论中无法找到一个适当的评判准则。文本方法的支持者宣称基于文本的协议可以更便于调试，因为它们可以由人直接阅读，并且文本协议更灵活、更容易用新特性来扩展。

二进制的支持者争辩说，二进制协议能够更有效地使用带宽，也可以在使用恰当工具的前提下很容易地调试和扩展它们。两种类型的编码各有其优点和缺点，我们不想在这里一一列举出来，不过请记住，SIP 是一个基于文本的协议，并且显示出了通常基于文本协议的所有正面和反面的特性。

5.3.1　SIP 请求格式

一个 SIP 请求由一个请求行、几个标题头（header）、一个空行和一个消息体组成。表 5-4 列出了一个 SIP 请求的格式。消息体是可选的，一些请求可不携带它。

表 5-4　　一个 SIP Request 的格式。

Request-line	请 求 行
Several headers	几个标题头
Empty line	空行
Message body	消息体

请求行

一个请求行有 3 元素：方法、请求-URI 和协议版本。方法表示请求的类型，这在前面的节中已经解释了一些。请求-URI 表明下一跳，就是请求消息将要被路由的地方。图 5-12 中，在 company.com 域的 SIP 代理收到一个有请求-URI 为 SIP:Bob.Johnson@company.com 的 INVITE 请求，这个代理知道可能在两个地方可以找到 Bob，因此它产生两个 INVITE。一个包含 SIP：Bob.Johnson@company.com 作为请求-URI 并且将被送给在 university.com 域的服务器；第二个 INVITE 包含 SIP:Bob@131.160.1.112 且将被送往 131.160.1.112。因此，请求-URI 包含了路径中下一跳的地址。

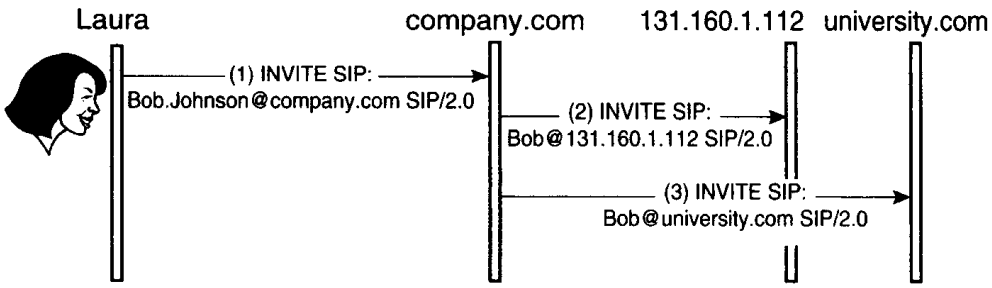


图 5-12　请求-URL 包含路径中的下一跳

最后，我们知道协议版本是 SIP/2.0，因此，在前一个例子中处于 company.com 域的服务器收到的 INVITE 的请求行看起来会像下面这样：

```
INVITE sip:Bob.Johnson@company.com SIP/2.0
```

5.3.2　SIP 应答消息格式

一个 SIP 应答由状态行、几个标题头、一个空行和一个消息体组成。表 5-5 列出了一个 SIP 应答消息的格式。消息体是可选的，一些应答可以不携带它。

表 5-5　　一个 SIP 应答的格式

Status line	状态行
Several headers	几个标题头
Empty line	空行
Message body	消息体

状态行

一个状态行也有 3 个元素：协议的版本号、状态码和一个原因短语，当前的协议版本为 SIP/2.0。状态码报告事务的状态，如早先描述的，状态码是从 100～699 的整数并且分组成 6 个不同的类(见表 5-1)。原因短语只是给人阅读的，它对机器处理 SIP 应答无任何意义。下面是一个状态行的例子：

```
SIP/2.0 180 Ringing
```

应答的可靠传输

最终应答在服务器和客户端之间可靠地传输，它使用重传机制或一种可靠的传输协议来保证发送。但临时应答并不这样，它们可能被客户端收到，也可能在网络中丢失。SIP 采用这种方法是因为，相对会话建立是如何进行的而言，它更关心一个会话是否被建立起来了，以及如果没有建立起来，其原因是什么。

因此在一个 SIP 呼叫中，例如，呼叫者被通知保证这个呼叫已经被接收了，但是可能不知道被呼叫者的通知信号是什么时候开始的，如图 5-13 所示。如果需要的话，可以对 SIP 进行扩展以保证临时应答的可靠传输。

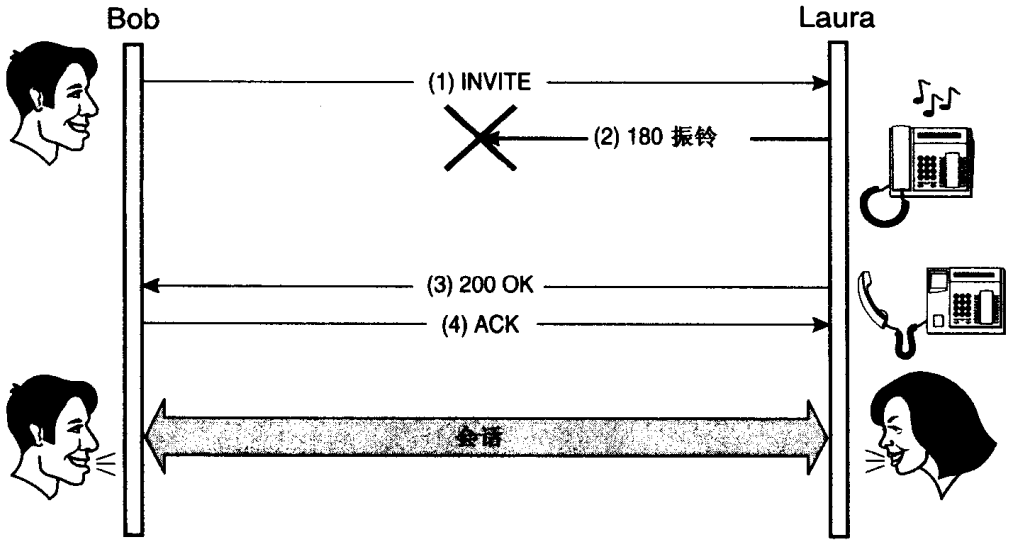


图 5-13　SIP 不能保证临时应答被接收

5.3.3　SIP 标题头

SIP 请求在请求行后包含有一些标题头，而 SIP 应答将它们放在状态行之后。标题头提供了关于请求（或应答）的信息和关于这些消息所包含的消息体的信息。一些标题头可以在请求或应答两种消息中使用，而其他的只能单独特定用于请求（或应答）。标题头由标题头名、后面跟着的一个冒号、再后面跟着的标题头值组成。

例如，一个标题头叫做 From,它表明了一个特定请求的始发者，看起来像下面这样：

```
From: Bob Johnson <sip:Bob.Johnson@company.com>
```

正如在这个例子中能看到的，一个标题头的值由几个字段组成，本例中，这个 From 头有两个字段：一个人名和他的 SIP URL。

表 5-6 包含了核心协议中定义的 SIP 标题头。

在后面的章节中，将解释最重要的 SIP 标题头的用途，并给出它们的简单的应用示例。

表 5-6　　SIP 头

Accept 接受	Content-encoding 内容编码	Max-forwards 最大转发次数	Route 路由
Accept-encoding 接受的编码	Content-language 内容语言	MIME-version MIME 版本	Server 服务器
Accept-language 接受的语言	Content-length 内容长度	Organization 组织	Subject 主题
Alert-info 通知信息	Content-type 内容类型	Priority 优先级	Supported 支持
Allow 允许	Cseq 命令序列	Proxy- authenticate 代理鉴别	Timestamp 时间戳
Also 也	Date 数据	Proxy- authorization 代理授权	To
Authorization 授权	Encryption 加密	Proxy-require 代理需求	Unsupported 无支持
Call-ID 呼叫标识	Error-info 错误信息	Record-route 记录路由	User-agent 用户代理
Call-info 呼叫信息	Expires 时间到	Require 需求	Via 通过
Contact 联系	From 从	Response-key 应答密钥	Warning 警告
Content-disposition 内容部署	In-reply-to 在答复中	Retry-after 此后重试	WWW-authenticate WWW 鉴别

呼叫标识（Call-ID）

Call-ID 代表了一种在两个或多个用户之间共享的 SIP 信令的关系。它标识一个特定邀请和与这个邀请相关的所有后续事务，它的格式如下：

```
Call-ID:ges456fcdw211kfgte12ax@workstation1234.university.com
```

为多个会话处理 SIP 信令的服务器使用 Call-ID 来将进入的消息与相应的会话联系起来。例如，Bob 使用一个特定的 Call-ID 邀请 Laura 加入一个国际象棋的会话。Laura 的用户代理接收了邀请，棋局很快开始。过一会儿，Bob 呼叫 Laura，想在他们下棋时同 Laura 讲话。这个从 Bob 的用户代理发出的 INVITE 请求与前者有不同的 Call-ID 号。

当 Bob 和 Laura 完成讲话时，Bob 的用户代理向 Laura 的用户代理发送一个 BYE 消息以结束通话。Laura 的用户代理使用 BYE 消息的 Call-ID 来决定是否终止国际象棋游戏或者终止谈话，如图 5-14 所示。

联系（Contact）

根据 Contact 标题头，可直接找到用户的 URL。这个特性非常重要，因为它卸载了 SIP 服务器，它们在路由第一个 INVITE 请求之后就不再需要存在于信令路径中。

例如，Laura 以 SIP:Bob.Johnson@company.com 呼叫 Bob。company.com 的代理将这个 INVITE 转发到 SIP:Bob@131.160.1.112，Bob 在这儿，他接受了这个呼叫。Bob 的用户代理返回一个“200 OK”的应答，其中有一个 Contact 标题头：

Contact: Bob Johnson <sip:Bob@131.160.1.112>

当 Laura 的用户代理收到这个“200 OK”应答时，它向 Bob 的用户代理发送 ACK 消息。因为 Bob 的位置可以在 Contact 标题头字段中找到，所以 ACK 被直接送往 SIP:Bob@131.160.1.112，且 ACK 消息不需要穿过 company.com 域的代理。

图 5-15 表明了接下来的请求，如 BYE 消息，在会话参与者间直接传送的情况。

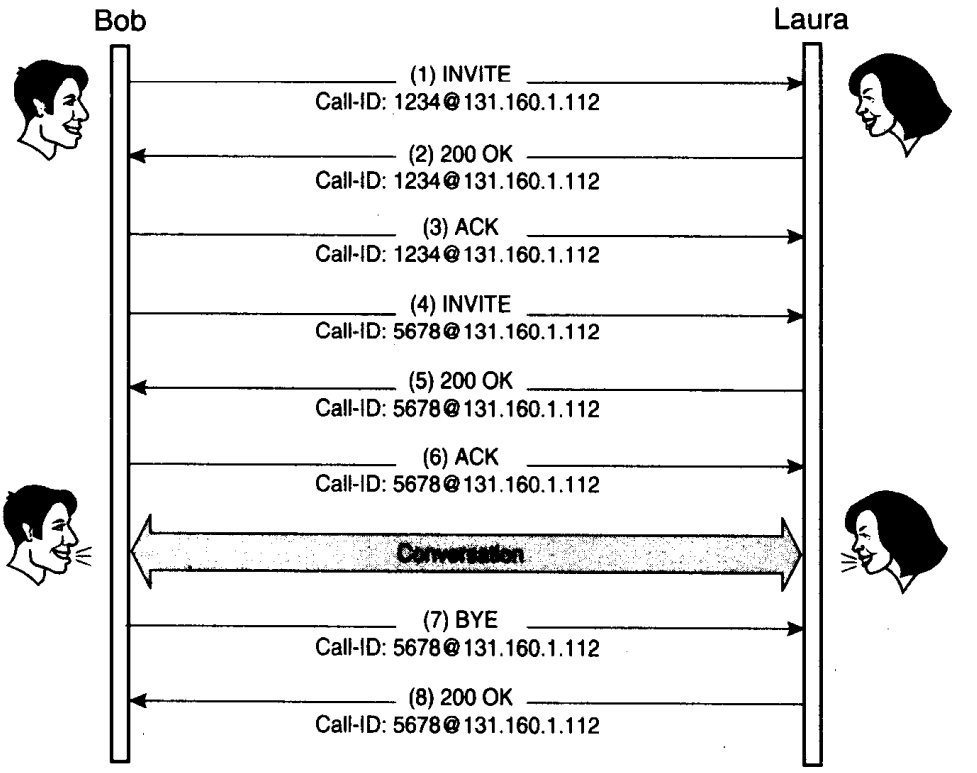


图 5-14　Call-ID 帮助区分不同的会话

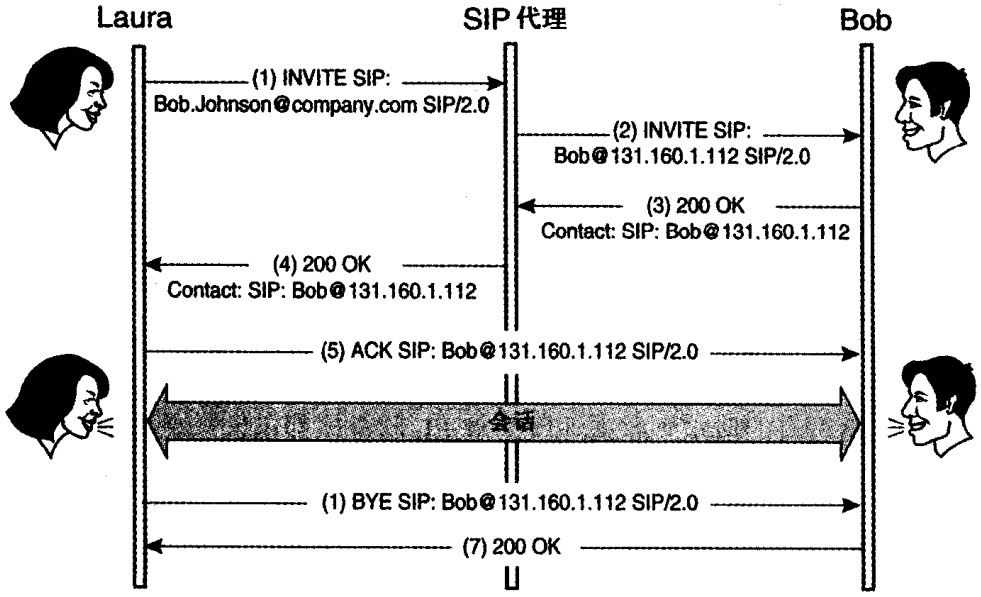


图 5-15　一个端用户被找到后 Contact 标题头可以跳过一个代理服务器

命令序列标题头（Cseq）

Cseq 有两个字段：一个整数字段和一个方法名。Cseq 的数字部分用于在同一会话（由特定的 Call-ID 定义）中对不同的请求进行排序。它也被用于将请求与应答相匹配。例如，Bob 向 Laura 发送一个 INVITE 消息，其中有这样的 Cseq：

Cseq: 1 INVITE

Laura 返回一个“200 OK”应答，中间有与 INVITE 相同的 Cseq。如果 Bob 想调整已经建立的会话，他将发送第二个 INVITE 消息（即 re-INVITE），其中有这样的 Cseq：

Cseq：2 INVITE

如果一个“200 OK”响应的重传过程被网络延迟了，并且在它产生第二个 INVITE 消息后才到达 Bob 的用户代理，根据这个 Cseq 头，它知道这是第一个 INVITE 的应答，如图 5-16 所示。

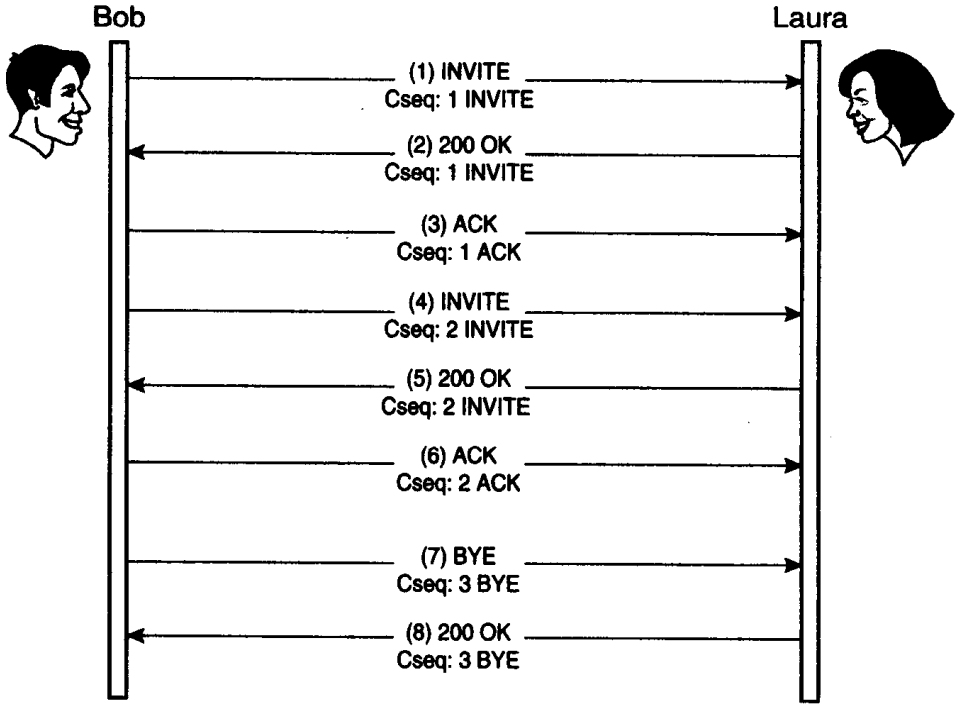


图 5-16　Cseq 帮助在一个会话中区分事务

在 INVTTE 之后所有的请求（ACK 和 CANCEL 除外）都包含一个 Cseq，它是最初请求的 Cseq 递增的结果。

ACK 中的 Cseq

一个 ACK 请求拥有与它所确认的 INVTTE 请求一样的 Cseq。这使得代理在为非成功最终应答产生 ACK 的时候，不需要创建新的 Cseq。实际上，新的 Cseq 可以仅被用户代理创建，

这保证了 Cseq 是唯一的。

CANCEL 中的 Cseq

一个 CANCEL 请求拥有与它所取消的请求相同的 Cseq。这也使得代理能在产生 CANCEL 时不需创建新的 Cseq。不仅如此，CANCEL 也是为什么 Cseq 头在数字部分后包含一个方法名的原因。

因为，INVITE 和 CANCEL 请求的 Cseq 号是相同的，一个 SIP 客户端如果没有附加字段就不能区分收到的是 CANCEL 的应答还是 INVITE 的应答。Cseq 中的方法名解决了这个问题，如图 5-17 所示。

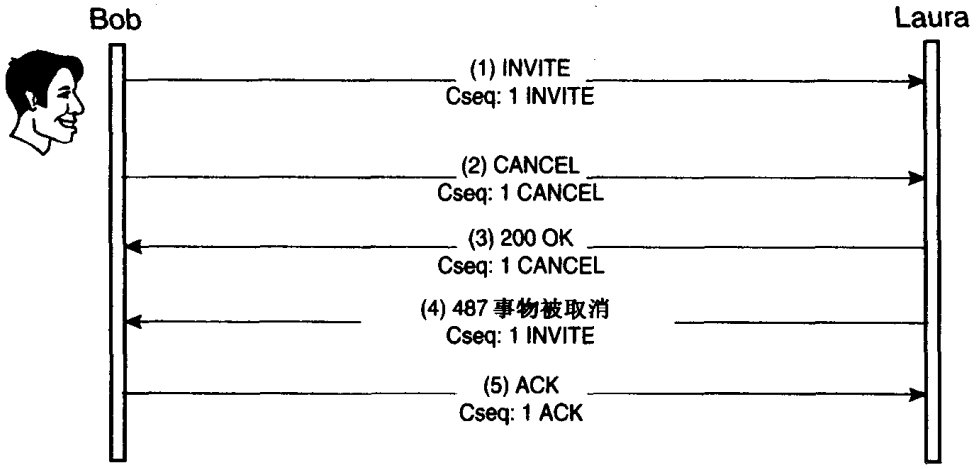


图 5-17　Cseq 中的方法名容许为 INVITE 和 CANCEL 请求产生不同的应答

记录路由和路由（Record-Route and Route）

这两个标题头被那些想要在整个会话过程中位于信令路径中的代理使用。我们看到，Contact 标题头使用户代理可以相互直接发送请求。这就在路径中创建了卸载代理，它们将第一个 INVITE 发往正确的目的地，然后开始让用户代理交换 SIP 信令。但是，有时候一个代理需要保留在信令路径中，这种情况下，必须有一种机制保证阻止用户代理在它们自己间交换 SIP 消息。这种机制由两个标题头组成：路由和记录路由。

代理可能因为某些理由想要在第一个 INVITE 请求之后仍然留在信令路径中。一个原因是出于安全考虑。有些域有安全代理，它有一个防火墙，用于过滤进入的 SIP 消息。不能成功穿越安全代理的 SIP 消息不会进入这个域。另一个原因是服务提供商。一个提供会话相关服务的代理理所当然地需要知道会话何时结束；就我们自己来说，也就是用户代理什么时候向另一个用户代理发送 BYE 请求。我们已经在图 5-9 中看到了这种服务的一个例子。例子中的保留呼叫状态代理为了给 Bob 发电子邮件，通知关于本次通话持续情况的信息，它必须看到从 Laura 向 Bob 发送的 BYE 请求。

如图 5-18 所示是这两个标题头的工作情况。Laura 向 Bob 发送一个 INVITE 请求，这个 INVITE 请求通过一个 SIP 代理，这个代理为了处理 Laura 和 Bob 之间后续的请求，想要处于信令路径中。这个代理在 INVITE 请求中增加了一个包含它的地址的记录路由标题头。Bob 的用户代理完整地接收到了包含记录路由标题头的 INVITE 请求，并且将它放在“200 OK”应答报文中。Bob 的用户代理也将它的 Contact 标题头加到应答中。

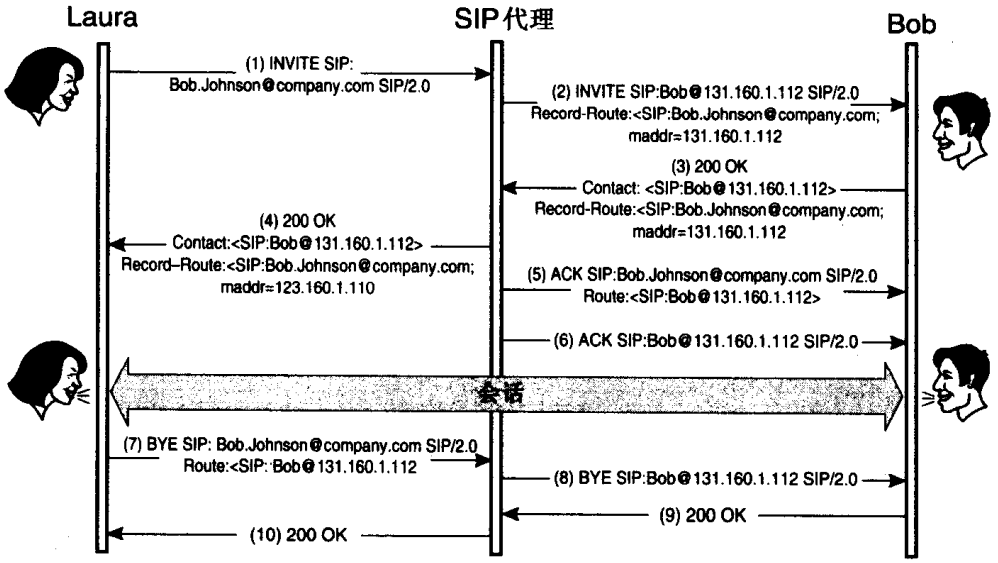


图 5-18　路由标题头有一个代理在整个会话过程中位于信令路径中

在记录路由标题头中出现的 maddr 参数仅包含服务器的 IP 地址，加上它是为了对将来的请求记录服务器的真实 IP 地址。

Laura 的用户代理收到“200 OK”应答并创建一个用于后续请求的路由标题头。路由标题头的创建源于在应答中出现的记录路由和联系标题头。由于仅有一个代理需要处在信令路径中，所有后续从 Laura 到 Bob 的请求（在这个例子中是 ACK 和 BYE）都将被送向代理并且将携有一个包含 Bob 的联系地址的路由标题头。通过这种方法，代理就知道将请求传给包含在路由标题头中的地址。

多个代理

前面的例子说明了记录路由标题头是如何起作用，以通知 Laura 的用户代理：后续的请求必须通过代理传送而不是直接送给 Bob。可是，它并没有显示需要路由标题头的原因。下面看一个有着更多代理的情况这将帮助读者理解路由标题头的作用。图 5-19 中有三个代理：P1、P2 和 P3。P1 和 P3 处在同一信令路径中，但 P2 不在。从图 5-19 中可以看出，从 Laura 到 Bob 的 ACK 是如何携带一个路由标题头来告诉 P1 将请求转发给 P3 的，*在路由标题头中最后的地址是 Bob 的联系地址。*

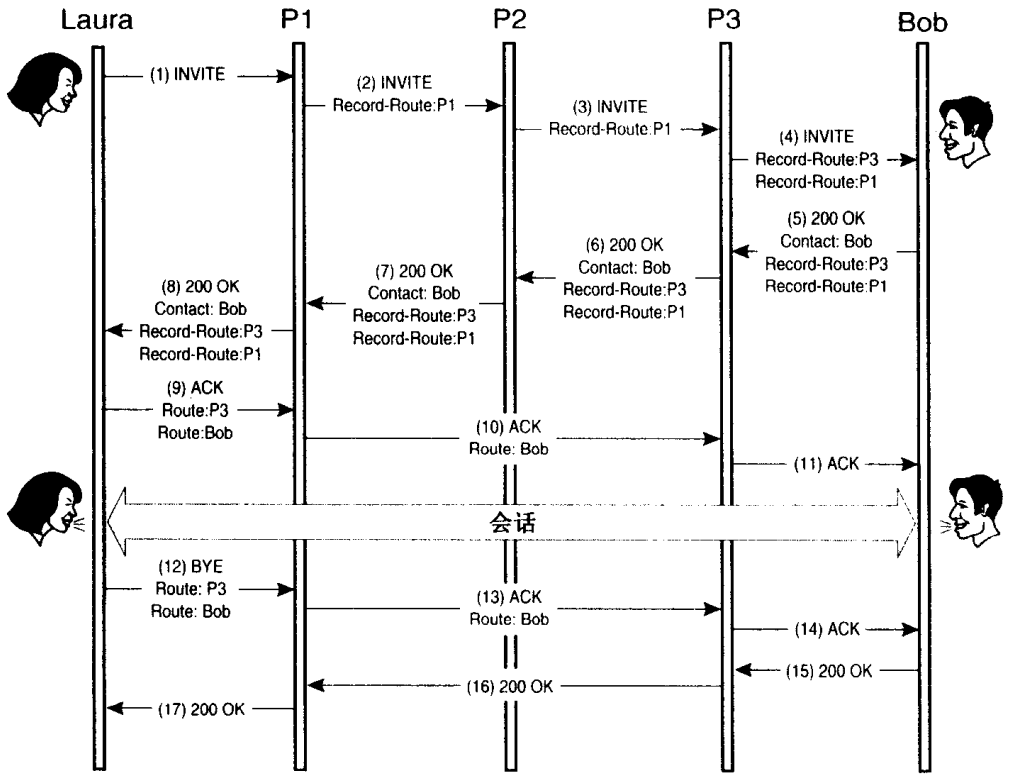


图 5-19　P2 不需要处在信令路径中

注意，图 5-19 没有像如图 5-18 所示的那样包含有联系、记录路由和路由标题头的真实格式。相反，它使用了一个仅仅表明有每个标题头中包含哪个地址的象征性的格式。

TO

TO 标题头总是包含请求的接收。它通常还包含目的方的公用地址。将一个请求的 TO 标题头和 Request-URI 区分开来非常重要。在整个会话过程中，TO 标题头含有同样的内容，它是打算用于远端用户代理的。它不能被代理改变。

Request-URI 含有在信令路径中下一跳的地址，并因此在路途中被每个代理改变。如图 5-20 所示为它们的用法。

Laura 用 Bob 的公用地址：SIP：Bob Johnson@company.com 呼叫他。这个 SIR URL 将插入 TO 标题头并且在整个会话中将不会变化；也就是，所有从 Laura 到 Bob 的请求将拥有相同的 TO 字段。

Laura 将同样的 SIP URL 放在 Request-URI 中，因此它将向 company.com 的 SIP 发送请求。这个代理通过尝试两个不同的 SIP URL：SIP:Bob@131.160.1.112 和 SIP:Bob@aniversity.com 进行一个并行查找。

所有在 company.com 域的代理发送的 INVITE 请求有同样的 TO 标题头，但它们有不同

的 Request-URI。

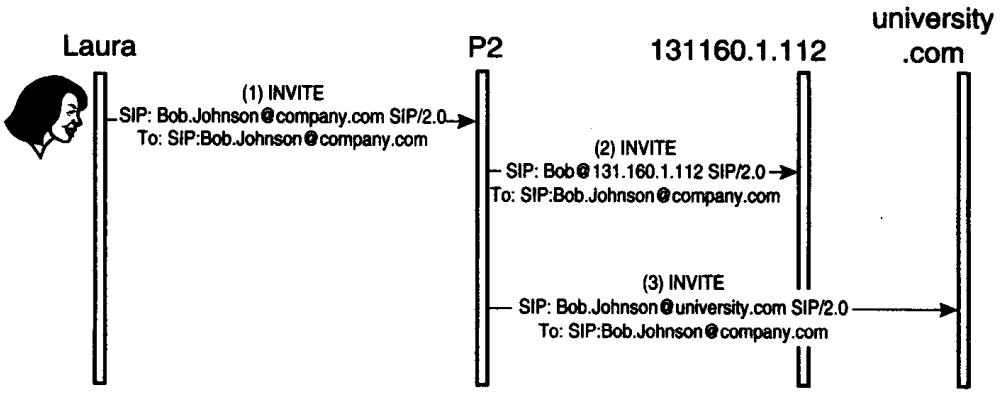


图 5-20　在一个会话中 TO 标题头不会改变

Via

Via 标题头存储所有处理请求的代理地址。因此，它们包含了请求所选用的路径，如图 5-21 所示。这个信息用于检测路由循环。如果一个请求在一个循环中转发，任何代理可以通过简单地检测 Via 标题头发现它。如果它发现自己的地址也在其中，这个代理就知道它已经处理了这个请求。典型的 Via 标题头的格式如下：

```
Via:SIP/2.0/UDP.workstation1234 company.com
```

Via 标题头也用于将应答路由给产生请求的客户端，通过这种方法，SIP 应答将和请求一样穿过相同的代理集，但方向相反。

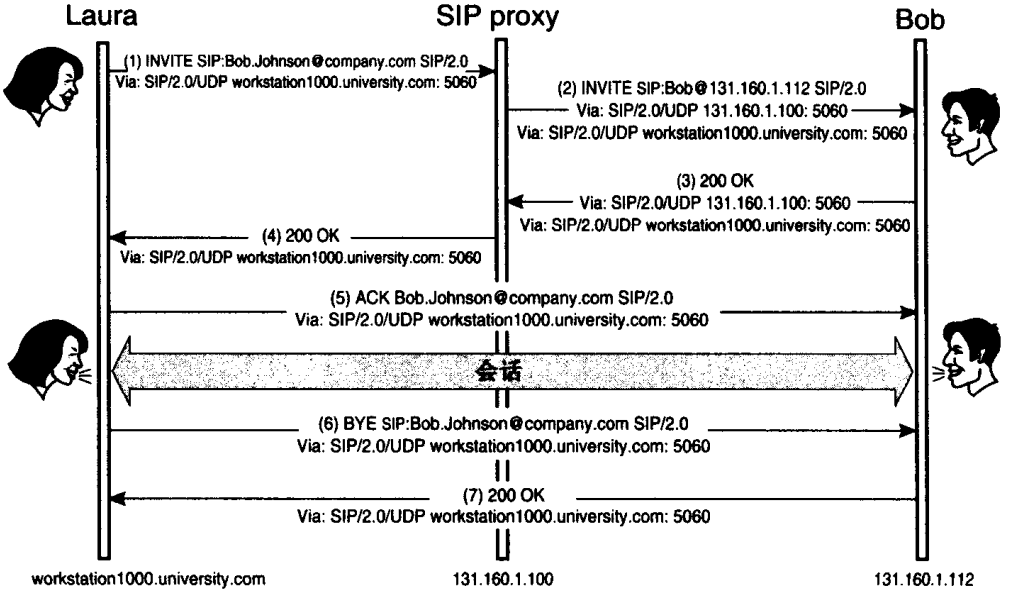


图 5-21　Via 标题头存储一个请求所选取的路径

5.3.4 SIP 消息体

请求和应答都可能含有消息体，它被一个空行和消息头分开。被 SIP 消息携带的消息体通常是会话描述符，但它也可以由任何不透明物体组成。因为 SIP 代理不需要检查消息体，消息体中的内容对它们是透明的。作为一个结果，会话描述符通过用户代理进行端到端的传输。代理为了路由 SIP 消息而需要的所有消息都包含在请求、状态行以及 SIP 标题头中。因为 SIP 消息体仅仅对用户代理有意义，所以消息体可被端到端加密而不会丢失任何功能。

可是，一些代理可能想检查会话描述符。一个例子就是安全代理（防火墙），它需要检查被交换媒体的信息，这使得它可以排除非授权的信息流。例如，如果一个公司决定它的雇员不能建立视频会议，防火墙就会阻止所有的视频流而同时仍然让音频流通过。

下面是一个在 SIP 消息体中的一个 SDP 会话描述符的例子：

```
v=0
o=Bob 2890844526 2890842807 IN IP4 131.160.1.112
s=I want to know how you are doing
c=IN IP4 131.160.1.112
t=0 0
m=audio 49170 RTP/AVP 0
```

就如 E-mail 消息可以携带多于一个附件一样，SIP 消息也可以携带多个消息体。例如，Laura 可以发送有两个消息体的消息，一个会话描述符和她的照片。这样，Bob 的用户代理可以在通知到 Bob 时在屏幕上显示她的照片。

5.4 传　输　层

我们从前面了解到 SIP 是一个应用层协议。因此，它利用传输层协议传输请求和应答。任何一个应用层协议的行为都会随着它所使用的传输层类型的不同而变化。如果它是可靠的，应用层协议创建一个消息并将它传向传输层，期待者这个消息达到目的的。应用层不知道传输层如何完成传输的；它仅知道任务完成了。

任务是如何完成的？通常，传输层将会重传这个消息，直到另一端收到它并且送回某种确认消息为止。这些重传对应用层透明。

另外一方面，如果应用层协议运行于一个诸如 UDP 这样的不可靠传输层协议之上的话，它不能保证传输成功。因此，应用层必须自己实现重传。它们通常如下面描述的一样实现这个过程。

应用层协议创建一个消息并将它传送给传输层。如果它不能在一个特定时段内成功接收来自目的地的接收确认消息，它将再次创建相同的消息并再次将它传向传输层。

通过在重传中利用应用层超时机制，一个应用层协议仍然可以利用不可靠的传输机制。

现在让我们看看 SIP 如何在所有这两种传输类型上运行。

5.4.1 INVITE 事务

由于 INVITE 事务包含一个三次握手过程和一个 ACK 请求，它们会采取与其他任何事务不同的处理方式。因此，SIP 实体以不同的方法对待 INVITE 和 ACK。

1．逐跳处理

请记住，当代理位于两个用户代理之间的路径中时，不同的传输层协议也可能在它们之间，如图 5-22 所示。用户代理使用一个可靠的传输协议向代理传输信息，并在到达远端用户代理之前不能保证端到端都使用同样的传输层。

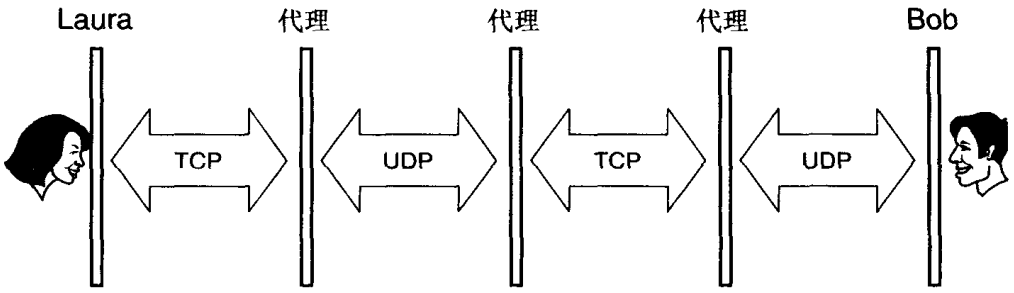


图 5-22　Laura 和 Bob 之间的不同类型的传输

SIP 提供了一种机制来保证 INVITE 被最终传送；也就是，它使代理担负起将一个 INVITE 请求在它的路径中向下一跳传递的任务。注意，不保留状态代理不能保证完成这个任务，因为它们不维持当一个 INVITE 报名丢失时进行重传所需的状态信息。因此，传输的下一跳是指下一个保留状态的代理（或目的地的用户代理）。

2．传输一个 INVITE 请求

由于用户代理（UA）和一个普通代理（proxy）（译者注：在 UA 和 proxy 同时出现的地方，都用“普通代理”表示 proxy）都有同样的责任保证 INVITE 到达下一跳，所以用于用户代理和普通代理之间的、用于两个普通代理之间的或者用于一个普通代理和一个用户代理之间的机制都是一样的。在这一节中，我们将解释用户代理向普通代理发送 INVITE 请求的行为，但那个普通代理将使用几乎相同的机制向路径中的下一个普通代理发送信息。

SIP 用户代理在可靠传输协议之上向普通代理发送一个 INVITE 请求，它并不需要处理任何特殊任务，但是如果使用了诸如 UDP 这样的一种不可靠传输协议，那么它必须准备重传，有时是重复的，直到前面的应答被收到为止，如图 5-23 所示。

收到 INVITE 请求的代理服务器总是产生一个“100 Tring”的临时应答，而收到 INVITE 请求的用户代理可能产生任何临时响应，如“180 Ringing”。

3．对一个 INVITE 传输应答

我们已经看到，临时应答用于阻止逐跳 INVITE 重传。可是，不能确保从被呼叫者的用户代理发出的一个临时应答都到达呼叫者的用户代理。路径中的代理通常将向前一跳发送一

次应答，但如果它没有成功到达也不再重传，如图 5-24 所示。

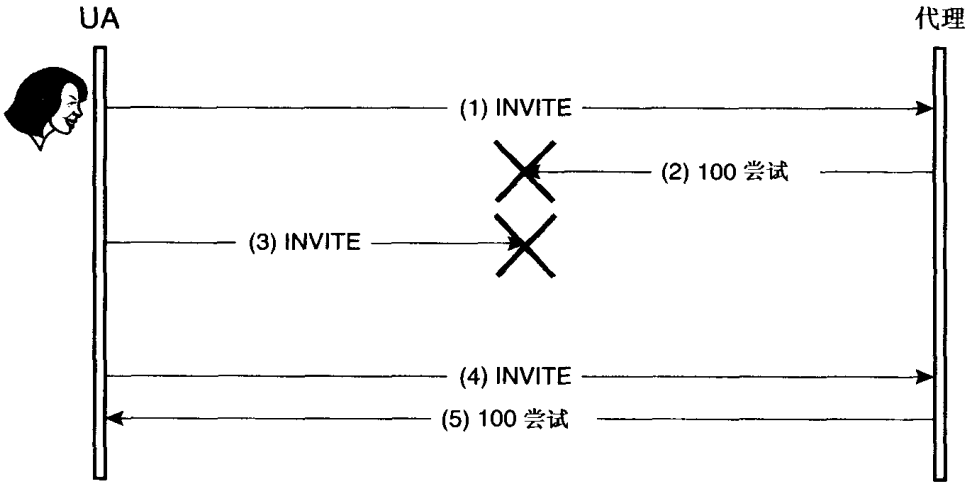


图 5-23　重传 INVITE 直到一个临时应答的到来

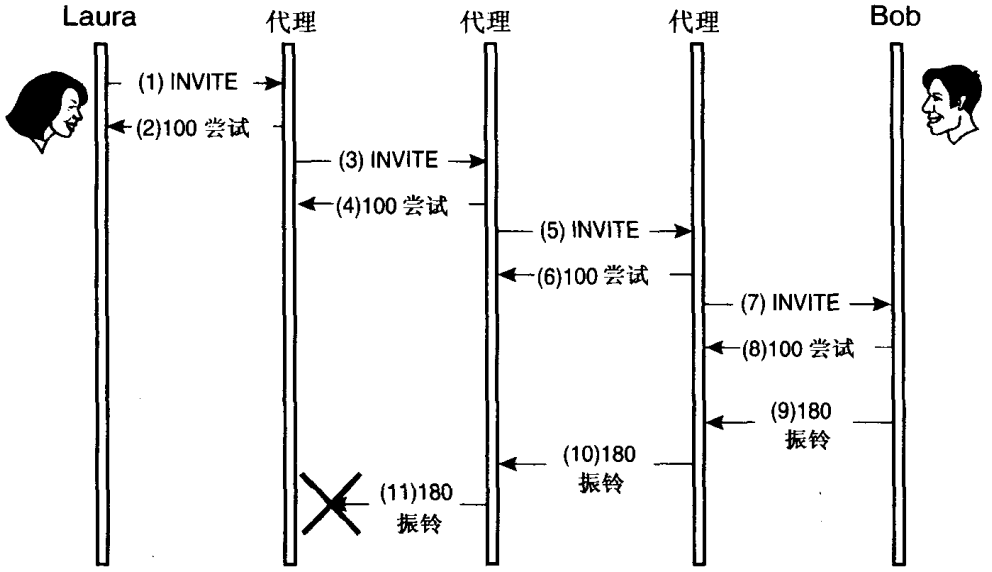


图 5-24　临时应答没有端到端地可靠传输

相反，SIP 能保证最终应答到达它们想要去的目的地。成功应答（200～299）被可靠地传送到源用户代理。非成功最终应答（300～699）使用和 INVITE 一样的逐跳机制。

4．非成功最终应答

处于传输非成功最终应答之后的想法和在传输 INVITE 之后的想法是一致的。每个服务器保证前一跳接收到应答并且前一跳保证承担处理应答的责任。

使用不可靠传输协议的用户代理将持续重传非成功最终应答，直至收到 ACK 为止，如图 5-25 所示。

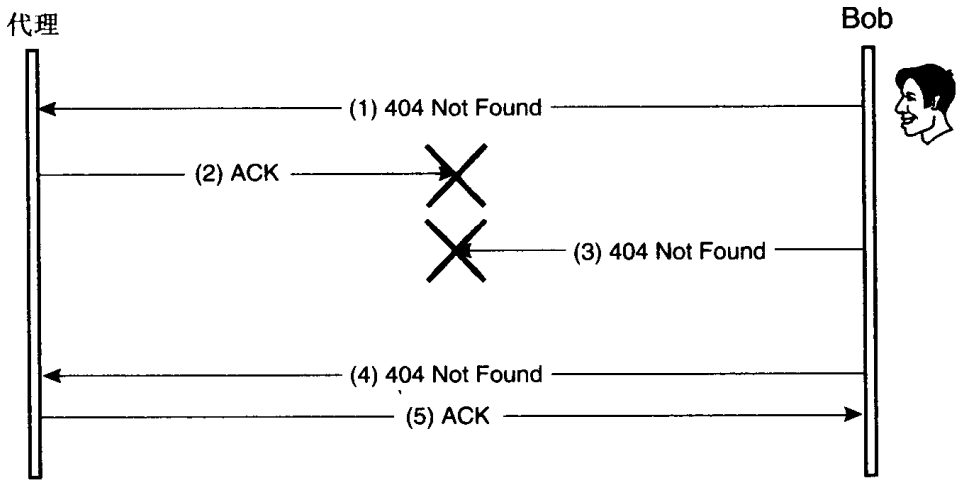


图 5-25 “400 Not Found”应答不断重传直至 ACK 到来

从理论上来说，使用可靠传输协议的用户代理没有必要使用 ACK。可是，为了使协议看起来相似地通过可靠的和不可靠的传输层，两者都使用了 ACK。

我已经提到，在某些情况下没有向源用户代理传送非成功最终应答。图 5-26 显示了在 company.com 域的派生代理如何接收“400 Not Found”应答，它没有把这些应答传送给 Laura 的用户代理，这得归功于使用应答中的逐跳传送机制。

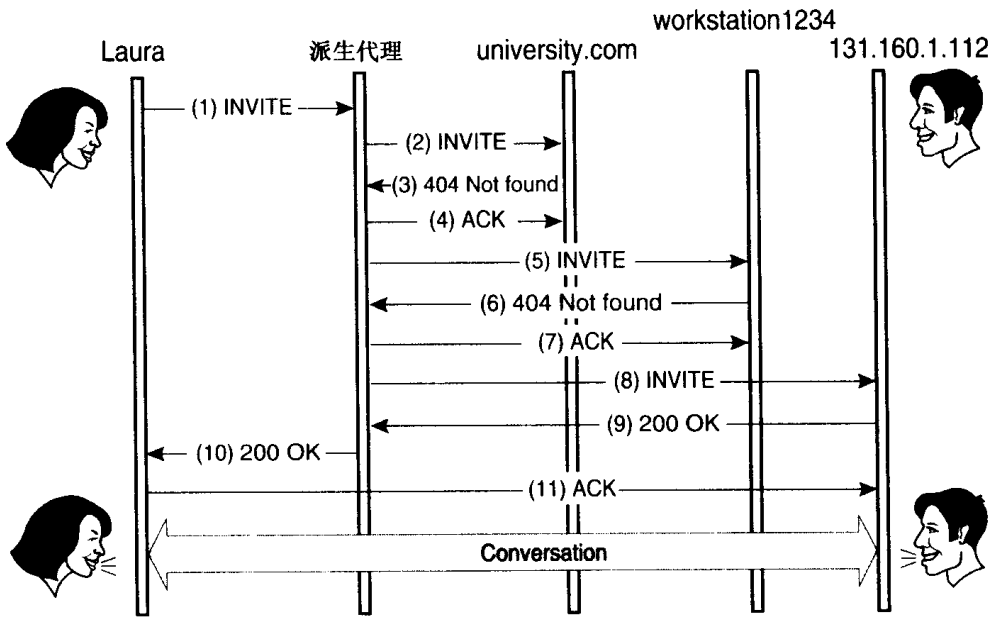


图 5-26 派生代理没有向 Laura 发送“404 Not Found”应答

5．成功的最终应答

成功的最终应答在用户代理之间被端到端地可靠传输并且不需要其他最终应答所使用

的逐跳机制，如图 5-27 所示。仅仅是最初产生 INVITE 的用户代理可以为一个最终成功应答发送一个 ACK。因此，不管使用的是什么传输协议（可靠的或不可靠的），用户代理将重传成功最终应答，直至它收到一个从最初用户代理发送过来的 ACK 为止。

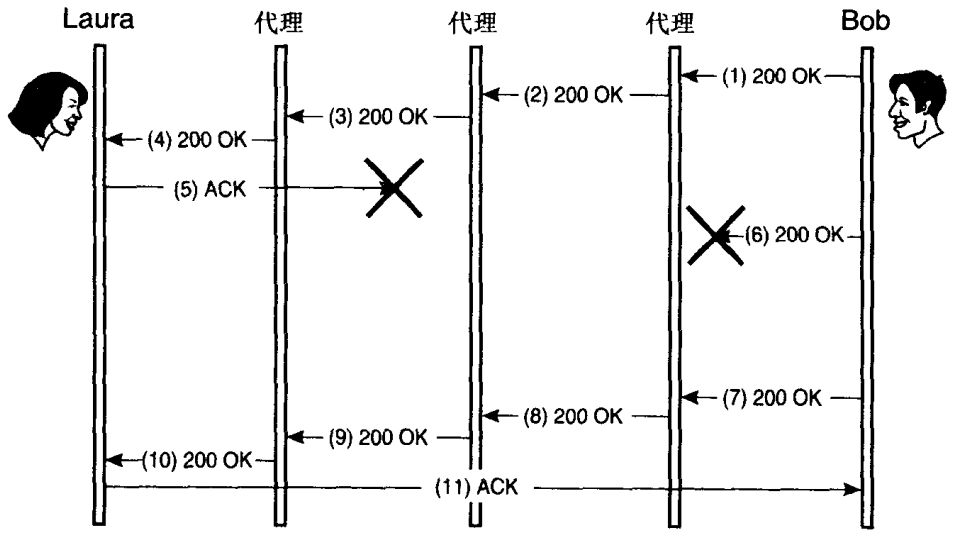


图 5-27　成功的最终应答被端到端地可靠传输

路径中的代理简单地将成功最终应答和它们的 ACK 同时转发。它们不涉及可靠性。图 5-28 显示了从 INVITE 直到实际会话发生的整个会话建立过程。

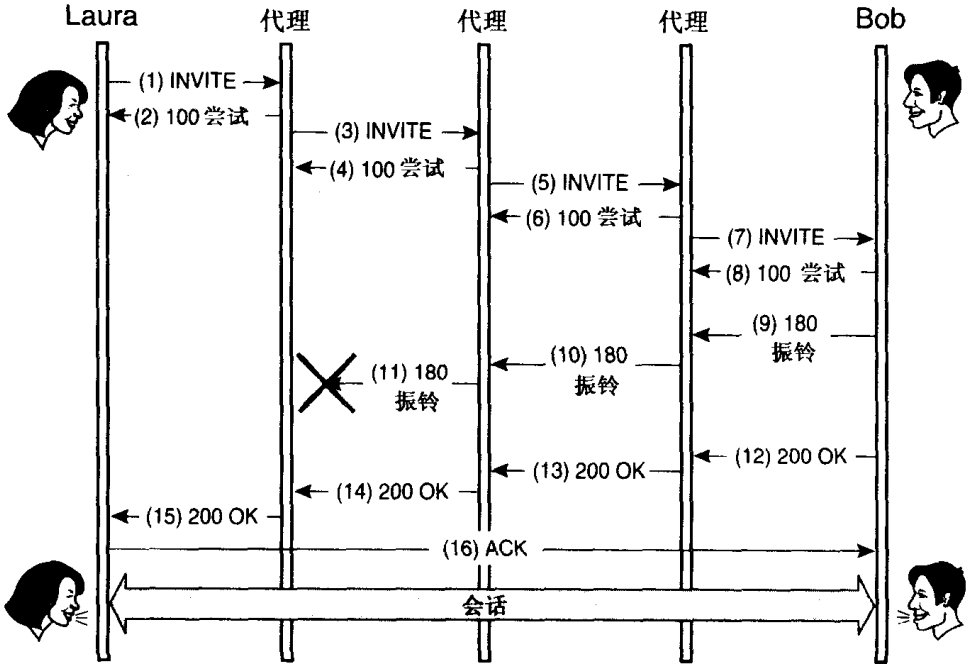


图 5-28　整个会话建立的过程

5.4.2 取消事务

作为逐跳事务，CANCEL 事务以一种特别的方法处理。当用户代理向普通代理发送一个 CANCEL 时，这个普通代理以一个最终应答来响应。在那一点上，CANCEL 事务为用户代理而完成。接着，普通代理将另一个 CANCEL 发送到下一跳，并且它也将收到一个最终应答。可以看到，CANCEL 请求的可靠性可以很容易地用重传机制实现，如图 5-29 所示。

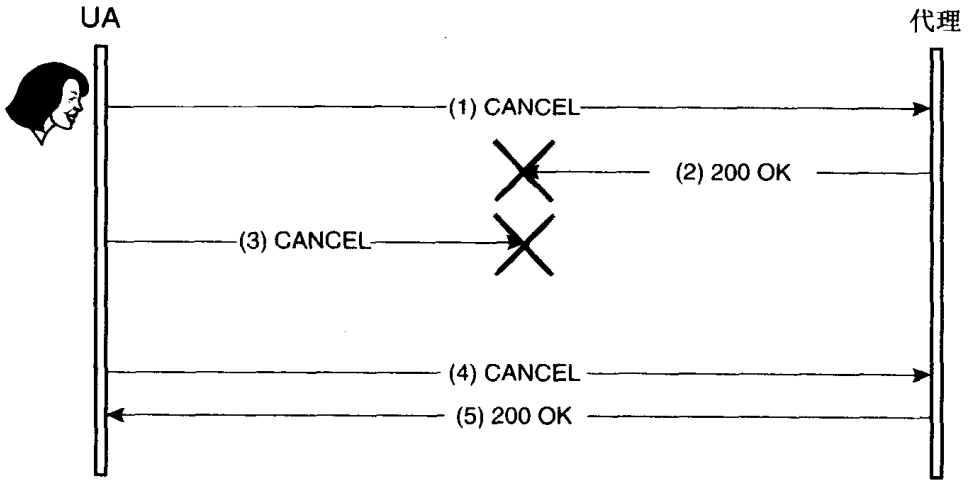


图 5-29 CANCEL 请求被重传直到最终应答的到来

5.4.3 其他事务

在 INVITE、ACK 和 CANCEL 请求的例子中，SIP 提供了适合于这三者的可靠传输机制。其余的 SIP 请求按照通用的原则实现。就可靠性方面来说，OPTIONS、BYE 和 REGISTER 被以同样的方法对待。在下一章，我们将看到它们具有通用性，使得协议能够以新的方法进行扩展；一个代理将通用可靠性规则应用到任何未知的方法上。因此，认为它们都是可靠的，在一个 BYE 和其他新方法间没有不同的表现。

通用可靠性规则也采用用于 INVITE 的逐跳机制。用户代理确信请求被下一个普通代理接收到了，并且下一个普通代理保证路径中接下去的代理也会接收到这个请求，如此等等。当最终的应答从远端传过来的时候，普通代理将保证它被传送到最初产生请求的用户代理那里。

对于可靠传输来说，用户代理向普通代理发送请求。当最终应答来到时，普通代理将把它传递到也使用可靠传输的用户代理那里，如图 5-30 所示。任何在最终应答前到达的临时应答也在按照可靠的传输协议被传送给用户代理。

对于不可靠的传输来说，用户代理不得不确认代理是否接收到了请求。当普通代理收到一个从远端传来的应答时，它不得不保证用户代理已经接收到它了。用户代理重传这个请求，

直到普通代理提出一个最终应答。普通代理只要不断接收到请求的重传，它就重传它的最终应答。当重传任务停止时，普通代理就认为用户代理已经收到了最终应答，如图 5-31 所示。

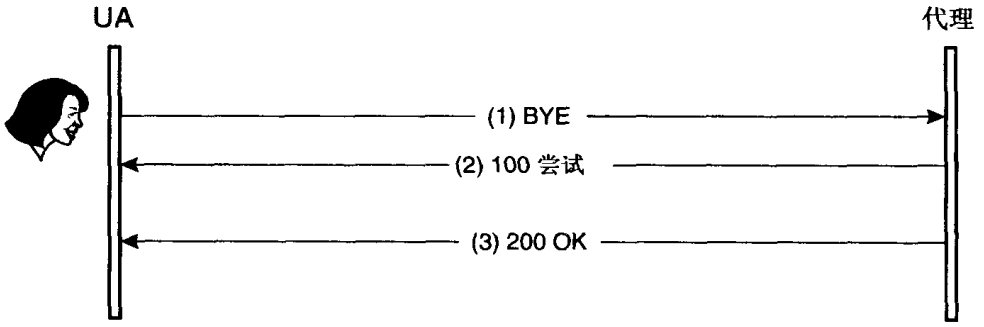


图 5-30　可靠传输机制保证请求和应答都到达它们的目的地

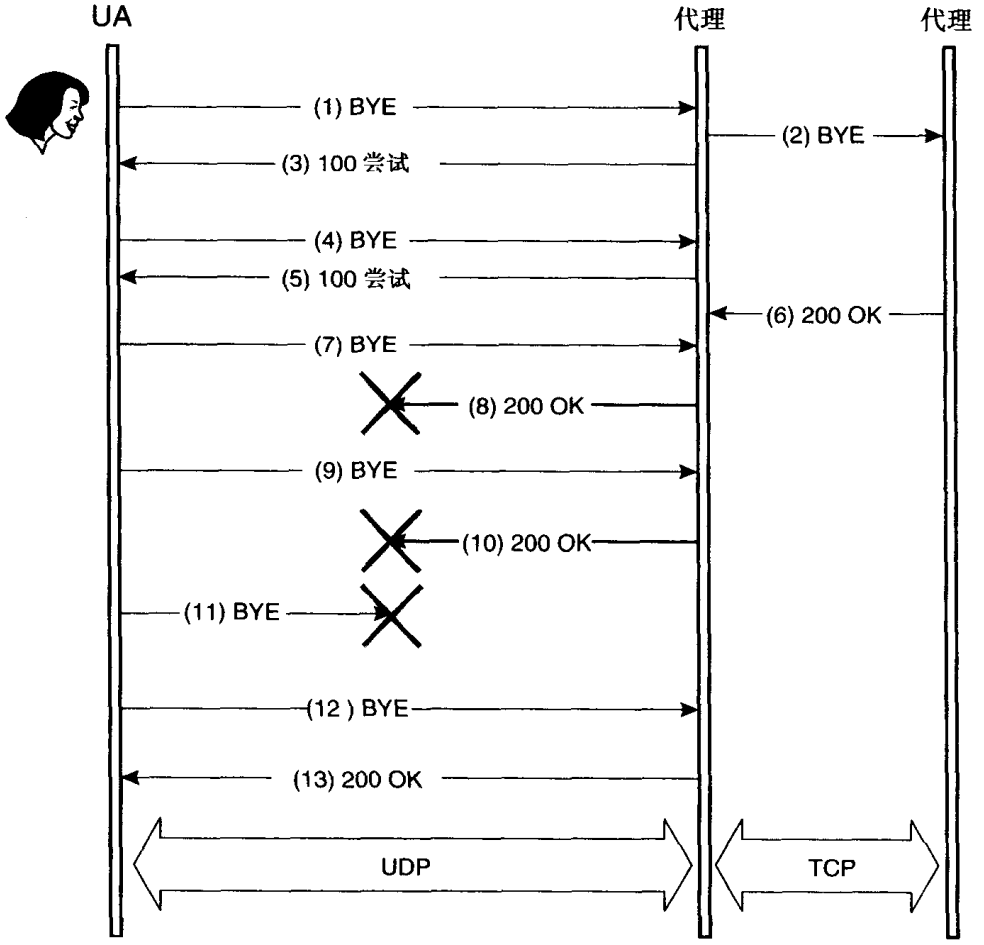


图 5-31　临时的应答没有停止请求的重传

5.5 详述的例子

综合上述内容，我们现在可以研究一个 SIP 工作细节的例子。就如前面提到的一样，例子通常由可以让我们一览 SIP 事务和消息交换的消息流组成。在一般情况下是不需要通过检查每个消息的所有参数去了解一个特定的情形的。在这个例子中，我们有意这么做，以显示 SIP 消息和所有它们的标题头和参数。

5.5.1 通过一个代理的 SIP 呼叫

在图 5-32 中，Laura 用 Bob 的公用地址呼叫他，但是 Bob 不在那里。在 company.com 域的代理服务器路由这个请求到他当前的位置：SIP:Bob@131.160.1.112。

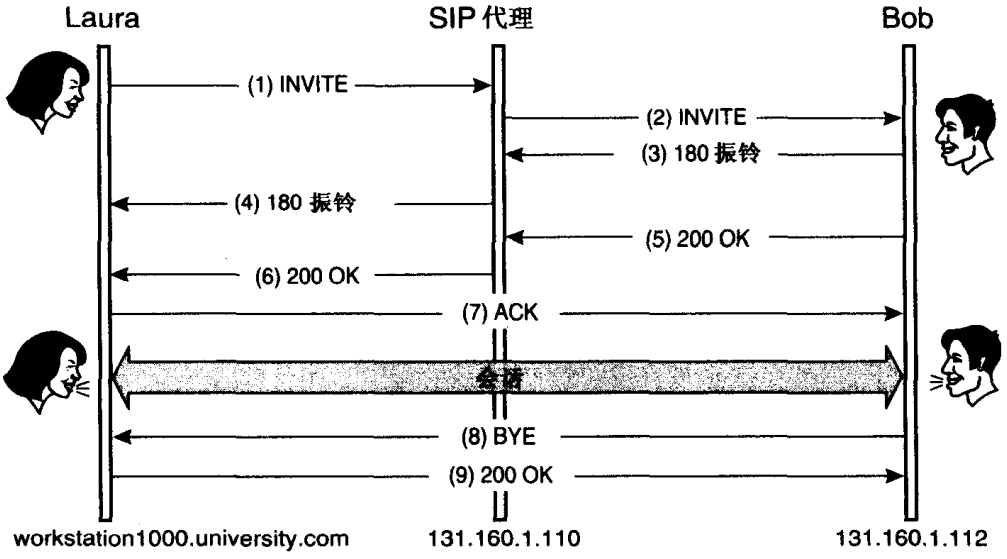


图 5-32 SIP 的呼叫通过一个代理

这个例子包含了 3 个不同的事务：INVITE，ACK 和 BYE。

1. 从 Laura 的用户代理到 SIP 代理的 INVITE

Laura 的用户代理将 Bob 的公用地址放在 TO 字段和请求-URI 中。用户代理增加一个有它自己地址的 Via 标题头，并创建一个有 SDP 会话描述符的消息体。Laura 想要在 UDP 端口 20002 接收包含 PCM 编码语音的 RTP 分组。这个请求被送往在 company.com 域的代理，因为请求-URI 域部分是 company.com。SIP 消息如下：

```
INVITE sip:Bob.Johnson@company.com SIP/2.0
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
```

```
To: Bob Johnson <sip:Bob.Johnson@company.com>
Call-ID: 12345678@workstation1000.university.com
CSeq: 1 INVITE
Contact: Laura Brown <sip:Laura@workstation1000.university.com>
Content-Type: application/sdp
Content-Length: 154
v=0
o=Laura 2891234526 2891234526 IN IP4 workstation1000.university.com
s=Let us talk for a while
c=IN IP4 138.85.27.10
t=0 0
m=audio 20002 RTP/AVP 0
```

2. 从 SIP 代理到 Bob 的 INVITE

在 company.com 域的 SIP 代理收到 INVITE 请求。请求-URI 的主机部分是 Bob.Johnson。代理知道可能在 SIP：Bob@131.160.1.112这个位置找到 Bob.Johnson。因此，它创建一个有 Bob 位置作为请求-URI 的新的 INVITE，并将它的地址 131.160.1.110 加到请求中作为一个 Via 标题头。注意到消息体仍没有变动。SIP 服务器一般不修改消息体。SIP 消息如下：

```
INVITE sip:Bob@131.160.1.112 SIP/2.0
Via: SIP/2.0/UDP 131.160.1.110
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>
Call-ID: 12345678@workstation1000.university.com
CSeq: 1 INVITE
Contact: Laura Brown <sip:Laura@workstation1000.university.com>
Content-Type: application/sdp
Content-Length: 154
v=0
o=Laura 2891234526 2891234526 IN IP4 workstation1000.university.com
s=Let us talk for a while
c=IN IP4 138.85.27.10
t=0 0
m=audio 20002 RTP/AVP 0
```

3. 从 Bob 到代理的临时应答

当收到 INVITE 请求时，Bob 的用户代理就必须开始报警，因此它返回一个临时应答，宣布开始待命了。Via 标题头是从收到的 INVITE 中拷贝过来的。[2]它们将保证应答首先穿过代理（131.160.1.110），然后到达 Laura 的用户代理（workstation1234.university.com）的 UDP 端口 5060。

Bob 的用户代理将一个 Contact 标题头加到含有 Bob 可以被直接找到地址 SIP URL 的应

答中；从现在开始，后续请求将被直接从 Laura 的用户代理传送到 Bob 的用户代理。

Bob 的用户代理也将一个标记参数加到 TO 标题头中，说明 Bob 当前使用的 SIP 用户代理。标记信息可帮助 Laura 区分那些在一个在路径中的派生代理尝试以多个地址联系 Bob 的情况下的应答。为了避免使 Laura 的用户代理产生误解，每个 Bob 的用户代理将用一个不同的标记来回答。SIP 消息如下：

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 131.160.1.110
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>;tag=314159
Call-ID: 12345678@workstation1000.university.com
CSeq: 1 INVITE
Contact: Bob Johnson <sip:Bob@131.160.1.112>
```

4. 从代理到 Laura 的临时应答

一旦收到这个 INVITE 请求，Bob 的用户代理就得去掉有它自己地址的 Via 标题头，并且将应答发送给包含在下一个 Via 标题头中的地址。这个代理不再做过多的动作。SIP 消息如下：

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>;tag=314159
Call-ID: 12345678@workstation1000.university.com
CSeq: 1 INVITE
Contact: Bob Johnson <sip:Bob@131.160.1.112>
```

5. 从 Bob 到代理的最终应答

当 Bob 接受了这个呼叫，他的用户代理返回它的 SDP 会话描述符。它将在 UDP 端口 41000 接收 RTP 分组。SIP 消息如下：

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 131.160.1.110
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>;tag=314159
Call-ID: 12345678@workstation1000.university.com
CSeq: 1 INVITE
Contact: Bob Johnson <sip:Bob@131.160.1.112>
Content-Type: application/sdp
```

```
Content-Length: 154
v=0
o=Bob 2891234321 2891234321 IN IP4 131.160.1.112
s=Let us talk for a while
c=IN IP4 131.160.1.112
t=0 0
m=audio 41000 RTP/AVP 0
```

6. 从代理到 Laura 的最终应答

代理服务器像它路由前面临时应答一样路由最终应答。换句话说，它去掉第一个 Via 标题头，并且将应答送往包含在下一个 Via 中的地址。SIP 消息如下：

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>;tag=314159
Call-ID: 12345678@workstation1000.university.com
CSeq: 1 INVITE
Contact: Bob Johnson <sip:Bob@131.160.1.112>
Content-Type: application/sdp
Content-Length: 154
v=0
o=Bob 2891234321 2891234321 IN IP4 131.160.1.112
s=Let us talk for a while
c=IN IP4 131.160.1.112
t=0 0
m=audio 41000 RTP/AVP 0
```

7. 从 Laura 到 Bob 的 ACK

当 Laura 的用户代理收到“200 OK”最终应答时，它发送一个 ACK 请求。这个 ACK 被直接送往 Bob 的用户代理，它的地址包含在刚收到的 Contact 标题头中。SIP 消息如下：

```
ACK sip:Bob@131.160.1.112 SIP/2.0
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>;tag=314159
Call-ID: 12345678@workstation1000.university.com
CSeq: 1 ACK
Contact: Laura Brown <sip:Laura@workstation1000.university.com>
```

8. 从 Laura 到 Bob 的 BYE

现在 Laura 准备结束这个呼叫了，所以她的用户代理发送一个 BYE 请求。使用不久前收到的 Contact 标题头，这个 BYE 请求也被直接送往 Bob 的用户代理。注意，Cseq 已经被增加了，这个 BYE 请求属于一个新事务。SIP 消息如下：

```
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>;tag=314159
Call-ID: 12345678@workstation1000.university.com
CSeq: 2 BYE
Contact: Laura Brown <sip:Laura@workstation1000.university.com>
```

9. 从 Bob 到 Laura 的最终应答

Bob 的用户代理收到了 BYE 请求，中止语音会话，并且为这个 BYE 请求返回一个“200 OK”应答。SIP 消息如下：

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP workstation1000.university.com:5060
From: Laura Brown <sip:Laura.Brown@university.com>
To: Bob Johnson <sip:Bob.Johnson@company.com>;tag=314159
Call-ID: 12345678@workstation1000.university.com
CSeq: 2 BYE
Contact: Bob Johnson <sip:Bob@131.160.1.112>
```

上面的例子说明了所有在本章前面描述的标题头是如何共同作用建立语音会话的。我们也能看到 SIP 协议操作是相当简单和易于理解的。

第6章 扩展SIP：SIP工具包

会话初始化协议的设计应使它的核心功能在每个实现中便于显现出来。这使得它在全局范围内提供了互操作性。每个SIP实现都能利用这样一个事实，即任何其他SIP实现都会理解所有在SIP RFC[RFC2543]中描述的机制。可是，一些实现还需要一些超出核心协议范围的功能，这意味着需要想办法增强SIP。

SIP很灵活并且易于扩展。于是，团体很快定义了一些扩展，对一些有特殊需求的应用设计一些扩展来满足它们的特殊需要。这些扩展以一种模块化的风格实现，并且它们的使用方法可以在会话建立期间协商，从而保证一个中心任务是：实现核心协议的简单的用户代理（UA）将总能和更高级的用户代理实现互操作。

SIP扩展可以视为是SIP工具包，其中每个扩展解决一个具体的问题。可以预见到，为了解决一个大的问题，比如怎样提供一种新服务，将需要把核心规范和适当的扩展结合起来使用。

这一章和第4章、第5章一样，首先通过解释每个扩展的功能，接着根据每个扩展是如何实现的来描述已存在的SIP扩展。一旦读者熟悉大多数常用的扩展的时候，我们在下一章将着重介绍其体系结构和使用SIP工具包（SIP和它的扩展）提供服务的应用。

6.1 扩展协商

一个给定的SIP应用总是假设另一个SIP应用能够理解SIP核心协议。可是，它不能作出关于远端支持何种扩展的假设。因此，为了决定在任何已给的会话中使用哪个扩展，一个协商过程是需要的，如图6-1所示。

在这个过程中，SIP实体要进行两件事情的协商：远端方支持哪个扩展和哪个扩展将在即将到来的会话中实际使用。例如，两个很高级的SIP用户代理可能支持多个扩展，但是如果核心协议肯定支持在一个特定时刻建立起来的会话类型，它们就不必使用任何扩展。可是在任何其他时刻，它们可能要决定使用它们所支持的扩展中的一个子集。SIP一直保持双方都可理解的扩展与那些只用于特定会话中的扩展之间的清晰差别。

6.1.1 它是如何完成的

使用两个标题头来处理会话扩展的协商过程：Require和Supported [draft-ietf-sip-serverfeatures]。客户端将它建立一个会话所需的所有扩展都列入到Require标题头中，同时将它支持的所有扩展列入Supported标题头中。服务器基于这两个标题头决定哪个扩展可以在会话中使用。

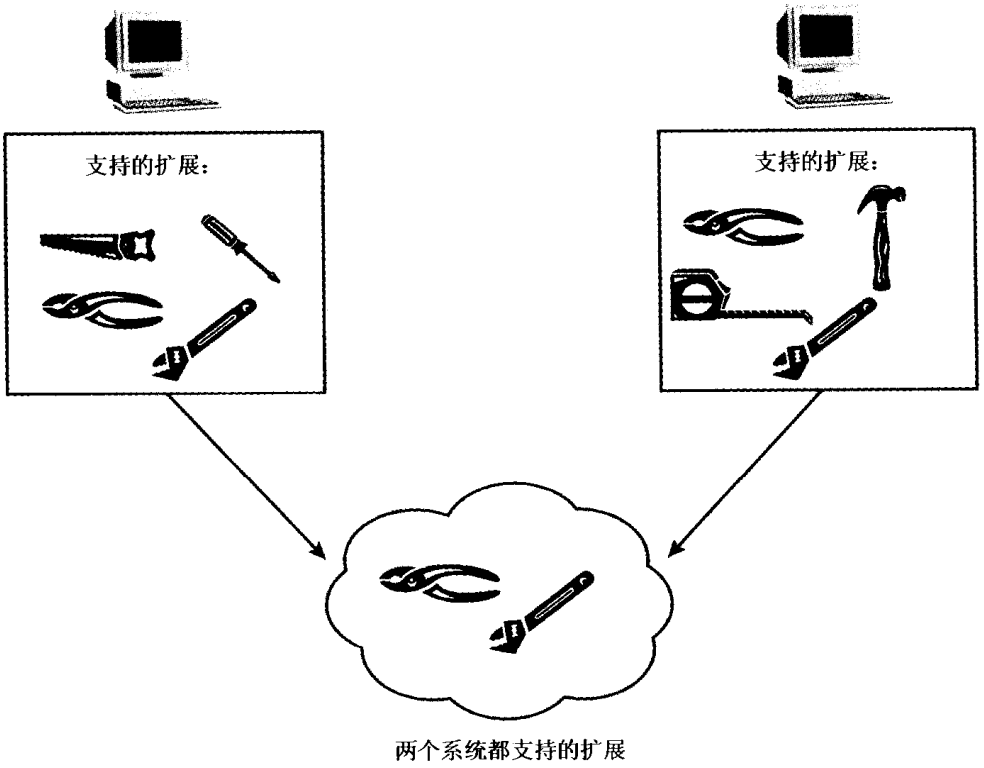


图 6-1　SIP 实体协商在一个会话中哪个扩展可以使用

在另一端，服务器把将被使用的扩展列入到 Require 标题头中，并且把它理解的扩展都列入到 Supported 标题中。这样，服务器与客户端就可以互换关于它支持哪个扩展的信息。

在图 6-2 中，Bob 的用户代理支持扩展 foo1,foo2,foo3 和 foo4。Bob 想要在会话中使用扩展 foo1，因此他将它加入到 INVITE 中的 Require 标题头中。Laura 想要使用扩展 foo2 。她知道 Bob 的 UA 支持它，因为它出现在了 INVITE 中的 Supported 标题头中。因此，她将 foo2 加到 Require 标题头中。除此之外，她还告诉 Bob 的用户代理她的用户代理也支持 foo4 和 foo5。现在，在从 Laura 返回的“200 OK”中的 Require 标题头包含了将在会话中实际使用的扩展。在 Supported 标题头中包含的信息在后面会被证明是有用的，如果 Bob 决定在会话中间也使用 foo4 扩展；他已经知道了这样的知识，即它是可用的且能使一个用户决定扩展他的会话能力。Foo3 和 foo5 扩展将不在会话中使用，因为它们只是被一个端系统所理解。

6.2　SIP 扩展的设计原理

对新 SIP 扩展的有效设计必须遵循一定的规则。已经定义了一些 SIP 扩展的设计原理 [daft-ietf-sip-guidelines]以保证新扩展不会改变 SIP 的精神。以 Internet 草案形式出现

的新的提议扩展在它们作为标准 SIP 扩展被接收以前在 SIP 工作组中被小心地分析。对那些将要设计扩展的读者来说，列举出一些为了能被 SIP 团体接受需要具备的特性是很有帮助的。

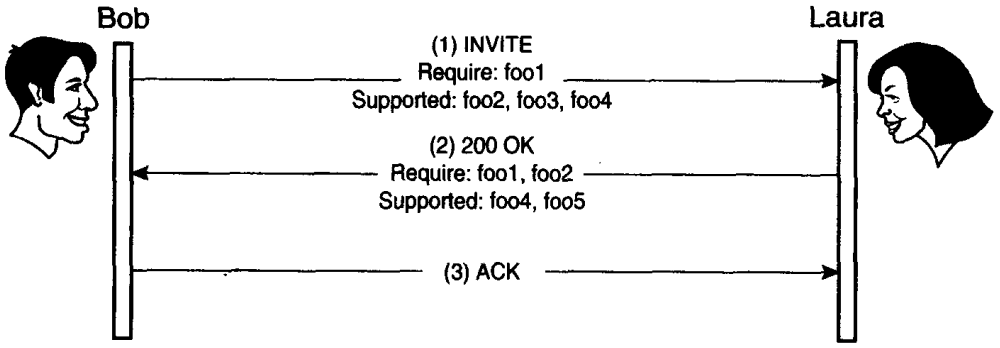


图 6-2　Bob 和 Laura 之间的扩展协商

6.2.1　不要破坏工具包方法

SIP 最大的优点在于它适合于 Internet 的多媒体会议体系结构。SIP 是 Internet 工程任务组（IETF）多媒体工具包的一部分。它按照它被设计的目标工作并且利用其他协议完成其他任务（例如会话描述协议（SDP）用于会话描述符）。SIP 扩展不应该扩大 SIP 的范围以使 SIP 用于能被其他 Internet 协议处理得更好的任务，甚至，此刻，看起来 SIP 将要做这项工作。

例如，某人可能想要用 SIP 从一个服务器上下载 Web 页面。为了实现这个目的已经有 HTTP 协议了。因此，应该知道，不要试图创建覆盖 HTTP 功能的 SIP 扩展。SIP 应该被用于定位一个特定的 SIP 实体和传送一个对象（如一个会话描述符），这之中可能有一个协商过程。SIP 应该用于这样的应用：它们支持用户可移动性、对象传送和 SIP 提供的协商机制。所有其他的应用落在 SIP 范围之外。如果我们尝试用 SIP 解决每一个它可能解决的问题，协议将很快变得大而且复杂，那将与 IETF 设计原理背道而驰，IETF 要求设计得精简、优雅。IETF 标准化过程确保了 SIP 保持简单和可管理。

6.2.2　对等关系

SIP 实体通常有一个对等关系。当服务器从客户端收到一个请求时，它处理一些任务，接着返回一个有请求结果的应答。客户端并不持续向服务器发送命令告诉它如何处理。因此，SIP 在一个主体对从属有很强控制的主/从体系结构中并不真正有效。

SIP 扩展不应该用于提供这样的控制功能，这种功能已经被更适合的协议如 H.248 [RFC 3015]所提供。相反，SIP 实体间的对等关系，使得协议非常适合域间通信。主 / 从协议被证明在域间通信中无效，那些域的所有者通常想要阻止不同域的所有者控制他们的资源。

6.2.3 会话类型的独立性

SIP 将会话建立过程和会话描述区分开来，只要将扩展添加到核心协议中，就应该维持这一分离原则，这是为了保证扩展能在将来证明（future-proof）（在 SIP 的例子中也就是，能够处理任何类型的会话）。例如，一个新的扩展将定义 SIP 如何同服务质量（QoS）机制交互，但可能没有定义为了提供 QoS，SIP 应如何与 SDP 和资源预留协议（RSVP，Reservation Protocol）相关联。那不是说后者在某些情况下无用，而是在任何情况下，通用机制应该定义为一个 SIP 扩展，并且这一通用机制对 SDP 和 RSVP 的固定应用将仅仅以一种报告文档的方式描述。

记住这一点非常重要，因为尽管 SIP 可能被用于建立所有类型的会话，SIP 在数据通信方面的发展已经十分集中在 IP 电话（VoIP）应用方面。这种集中在发展的协议的过程中是不寻常的，但是，我们也不得不防止这样一种自然倾向：仅设计那些在一个 VoIP 环境中应用的扩展。现在，SIP 扩展已足够用来覆盖不同类型的会话，尽管它们当前的用处只是一种 VoIP 服务。

6.2.4 不要改变方法的语义

一个 SIP 请求的目的由它的方法来定义（例如，一个 BYE 请求是为了结束一个会话）。因此，可能在第一眼看到一个请求时，通过检查它的方法就能知道它的目的。标题头和参数给出了关于请求的更多信息，但是请求的一般目的不会被任何标题头的内容所改变。

这一设计原则在计划进行新 SIP 扩展时仍然有效。让我们看一个不会被 IETF 接受的扩展的例子。下面的扩展破坏了通过方法识别一个请求目的的原则，因而将是一个坏主意。

某人可能定义一个叫做真实目的的标题头携带在 INVITE 请求中：

Real-Purpose: Tell me your capabilities 告诉我你的性能

有这个标题头的 INVITE 将用于访问一个远端系统，要求了解它的性能而不是建立一个会话。

可是，INVITE 请求的目的并不是询问。如果一个系统想要询问性能，它应该使用 OPTIONS 方法，它是专为此目的而定义的。

因此，无论何时需要一个有新功能的请求，SIP 团体就会创建一个新方法。他们不会去尝试改变一个已存在的方法的语义。

6.3 SIP 扩展

我们已经看到从每个 SIP 扩展可能希望得到的那些通用特征。在这一节，将分析一些已经提议的 SIP 扩展，并描述这些扩展所提供的功能，略述它是如何实现的，并且将试着尽可能将这两个概念分离开，以使读者可以很容易地区分它们。

6.3.1　SIP 工具包

核心协议和它的扩展一起可以被视为一个创建服务的工具包。为了设计一种新的应用或创建一种新的服务，设计者选取需要的扩展并将它们整合起来。这样，扩展就是十分通用的、可用于多种不同服务的机制。我们遵循同样的从底向上的方法来描述 SIP 扩展；关于使用不同扩展可以实现哪些特定应用和服务，将在下一章中进行讨论。

6.3.2　临时应答的可靠传输

当远端方同意加入会话时，核心 SIP 应保证将应答信息通知给会话的发起者。但由于目前没有被接受，所以关于会话建立过程的报告并不被认为是有用的。

可是，一些应用需要这种类型的信息。例如，设想一个公司的支持部实现了一个呼入电话的队列。当所有的工作人员都忙时，从客户来的呼叫被放入到呼入电话队列中。当有一个工作人员有空时，他或者她就回应在队列中的第一个电话。

呼叫了这个服务的客户将坚持要求告诉他们在队列中的排队情况。这既可以在呼叫者的 SIP 电话显示屏上显示一个信息（如“你现在在队列中排第二位”）来通知，也可以通过向呼叫者发出一个语音消息来通知。提供这个服务的相关扩展就是临时应答的可靠传输。

它是如何完成的

关于会话建立是如何进行的信息由一个 INVITE 请求发出的临时应答携带。从被呼叫者发向呼叫者的“180 Ringing”临时应答表明被呼叫者被通知到了。“182　Queued”临时应答表明呼叫被置入队列中了。

SIP 并不保证可靠地传送临时应答。当用户代理服务器（UAS）向用户代理客户端（UAC）返回一个临时应答时，它与 INVITE 穿越一样的代理，但是方向相反。即使在路径中的代理将这个临时应答转发到 UAC ，由于 UAS 不会重传临时应答，任何网络中的路由器都可能丢弃包含 SIP 应答的 IP 数据报。因此，可能发生 UAC 永远收不到临时应答的情况，如图 6-3 所示。

[Draft-ietf-sip-100rel]定义了用于提供临时应答可靠传输的 SIP 扩展。传输临时应答的 UAS 不断重传，直到收到一个从 UAC 发来的消息确认接收为止。这个机制和核心 SIP 中为“200　OK”应答使用的机制相似，那个请求也被 UAS 重传，直到收到 ACK 请求为止。

人们定义了一个新的方法用于确认临时应答的接收。这种新方法叫做临时应答 ACK（PRACK，Provisional Response ACK）；UAS 一旦从 UAC 收到一个 PRACK，便停止重传一个临时应答。

PRACK 请求和 INVITE 请求相比属于不同的事务。这样，UAS 也必须为 PRACK 发送一个应答。在图 6-4 中，UAS 为 PRACK 发送一个“200 OK”应答，表明 PRACK 请求成功了。

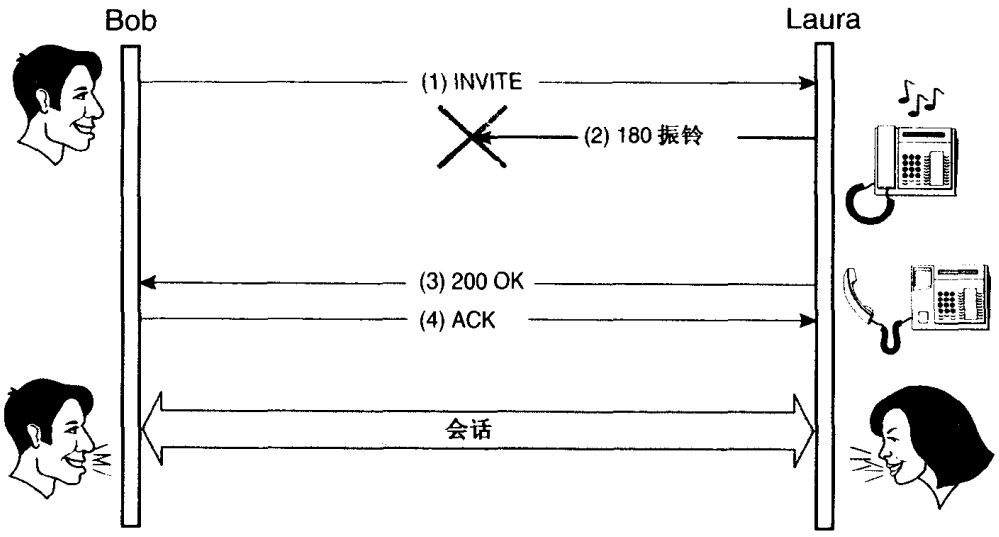


图 6-3　临时应答可能丢失

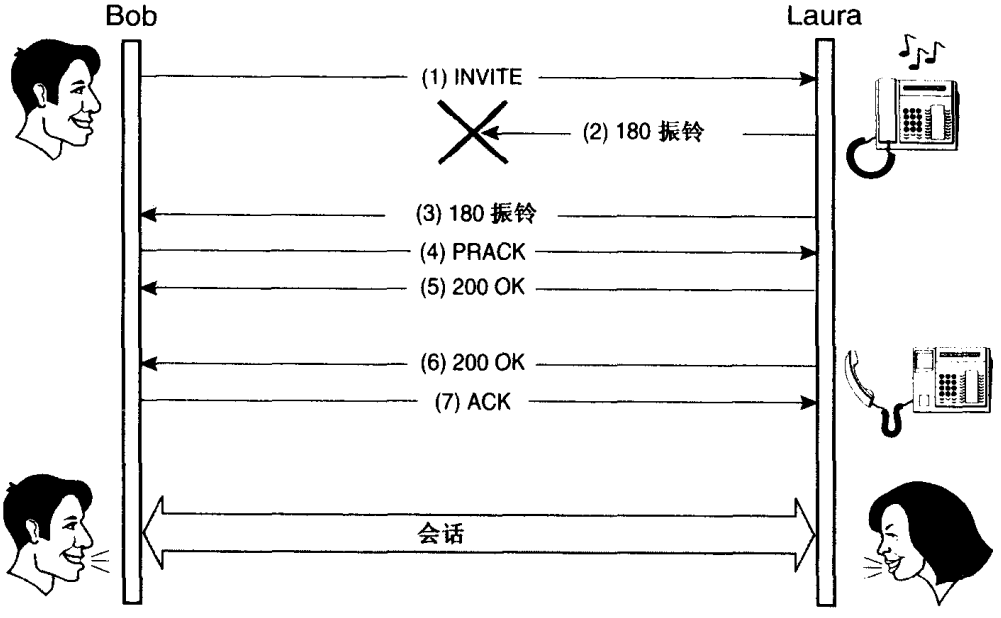


图 6-4　PRACK 保证临时应答被可靠传输

图 6-5 包含了前面描述过的例子。通过可靠的临时应答，一个顾客接收到了他在呼叫队列中位置的信息。每当一个新的应答到来时，他的 SIP 代理就在荧幕上显示他的新位置。

PRACK 是 SIP 工作组选择的一个通用解决方案的示例，它不是一个更高效的方案，而且通用性差。这个决定符合 IETF 范例和工具包方法。从可靠性观点看，为 PRACK 产生的“200 OK”应答真是没有用。UAC 知道 PRACK 已经到达了 UAS，因为临时应答的重传已经停止

了。因此为 PRACK 传送“200 OK”消息是为什么呢？

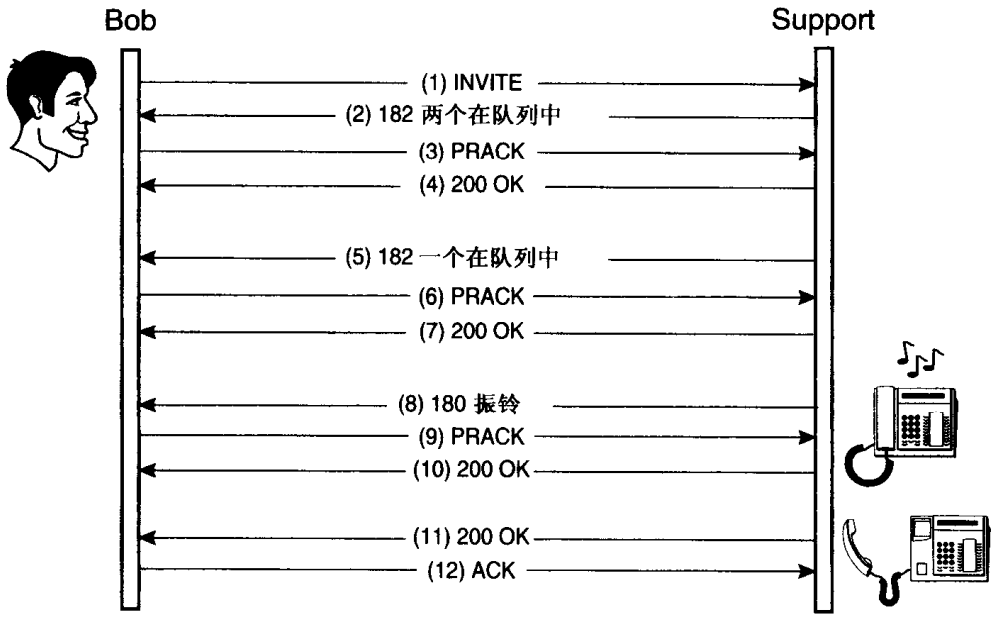


图 6-5　使用可靠临时应答实现的呼叫队列

这个问题的答案是 Internet 的一个关键特性：端到端的协议能更好地处理端到端的功能。SIP 代理了解基本的 SIP 方法。我们在第 5 章中看到代理收到一个它并不理解的方法时，这个代理按照处理 BYE 请求的相同规则对它进行路由。代理基本上就是转发这个请求，且如果它是保留状态的，它就会等待一个最终的回答。

使 PRACK 进入一个请求应答事务可以使不了解 PRACK 方法的代理服务器能正确地路由这个新方法。如果对 PRACK 没有应答，代理就将不得不为了进行特殊的路由而不是为了等待对 PRACK 的应答来学习一种新的方法。

这个例子说明了一个新的服务如何在不调整网络中任何事情的情况下被实现。当 PRACK 不存在时，两个实现 PRACK 方法的终端能够在 SIP 服务器组成的同一网络上交换可靠的临时应答。这些 SIP 服务器将不理解 PRACK，因为它们被安装在网络中的时候 PRACK 还在发展，但是它们仍然可以正确地路由 PRACK 请求。因此，端系统可以实现新服务并且不需要升级就可以在一个已存在的 SIP 网络上使用它们。这就是为什么为 SIP 网络创建服务的步伐比为其他网络或者为 Internet 创建服务的步伐快得多的原因之一。

6.3.3　不改变会话状态的中间会话事务

我们已经看到了一些 SIP 如何建立会话的例子。一旦一个会话被建立起来了，核心 SIP 规范通过 re-INVITE 提供一个手段来改变会话参数，并且通过一个 BYE 请求提供一种手段终止会话。当然，在某些情况下，参于会话的各方可能需要交换不会影响会话状态的信息。

可是，核心 SIP 没有提供一种用于向远端方发送信息而不影响会话状态的方法。在这种情形中，这种手段被证明是会话中信息传递的一个扩展。通常，在 SIP 同其他信令协议协同工作的时候可以看到交换这种信息。一个例子就是账单信息（Billing Information），会话提供者需要有规律地传输与会话相关但并不修改它的参数的信息。SIP 可以仅仅通过扩展来携带它。

它是如何完成的

[RFC2976] 定义了一个新的叫做 INFO 的 SIP 方法提供这个功能。INFO 传输会话中的不影响状态的信息。INFO 通常在消息体中携带信息，这些消息体在用户代理之间进行端到端的传输。图 6-6 说明了 Bob 和 Laura 在一个正在进行的会话期间交换 INFO。

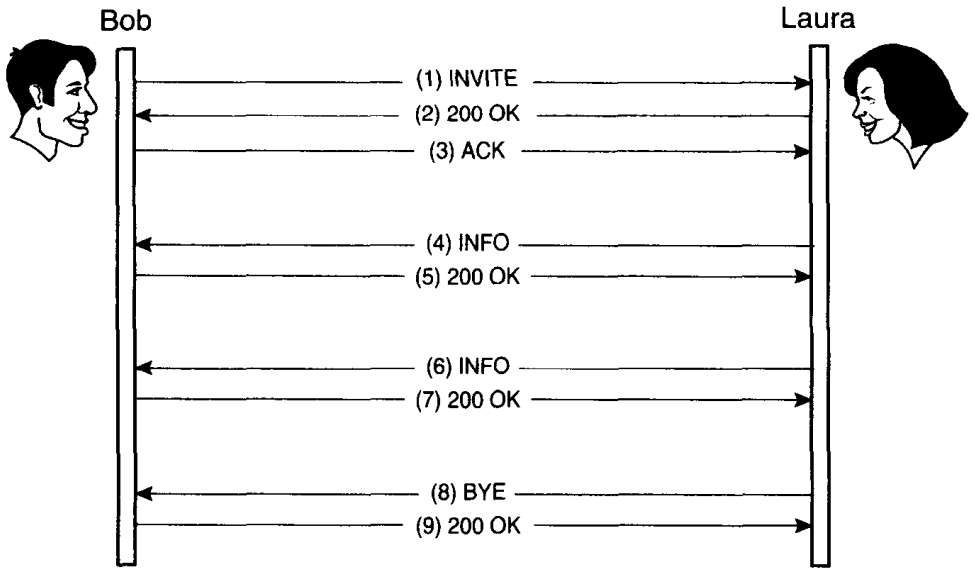


图 6-6　INFO 不改变会话状态

6.3.4　多消息体

读者已经知道，SIP 可以定位用户并且向他或她当前所处的位置传送会话描述符。可是，有时候需要发送不止一个会话描述符。例如，Bob 在他的便携式电脑中有一个 SIP 用户代理。当 Bob 收到一个呼叫时，除了收到使他可以建立会话的会话描述符之外，他需要看到是谁在呼叫他。SIP 可以在会话初始化时随着会话描述符一起发送呼叫者的照片，这幅照片将在 Bob 的便携式电脑上显示出来。Bob 可以根据呼叫者的标识来决定接受或拒绝这个呼叫。

它是如何完成的

为了能够显示呼叫者的照片，Bob 的用户代理需要接收一个照片文件（如 Laura.jpg）或者一个 URL，Bob 的用户代理可以在这个 URL 检索那幅照片（例如http://www.university.

com/~laura/photos//laura.jpg）。如果收到了一个 URL，Bob 的用户代理使用他的浏览器显示照片。为了接收这个信息，无论它是一个文件或一个 URL，Bob 的用户代理都需要收到一个 INVITE，它携带一个由两部分组成的消息体：一个 SDP 会话描述符和一张照片，如图 6-7 所示

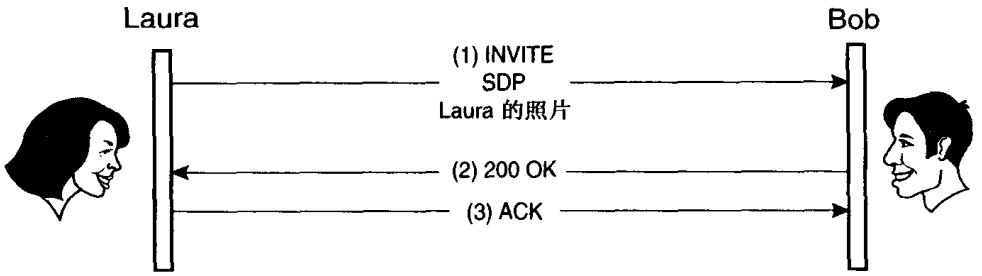


图 6-7　Bob 在 Laura 的 INVITE 中收到她的照片

核心 SIP 以多用途 Internet 邮件扩展（MIME）[RFC2045]的格式接收多部分消息体。SIP 不必为了支持它们而进行扩展。此外，SIP 使用 E-mail 系统传送附件的同样格式来发送多部分消息体。这说明工具包方法起作用了，因此新服务可以在已存在的机制上创建。

6.3.5　即时消息

为了建立会话，SIP 向呼叫者发送一个会话描述符。如果作为一个会话描述符的代替者，SIP 用于发送可读消息，那么即时信息系统就能很平常地用 SIP 实现。即时消息和电子邮件之间的主要不同之处在于即时消息很短且被即刻发送给用户。电子邮件通常长一些且它们将一直存储在用户的收件箱中，直到他或她访问收件箱才能读到它们。使用即时消息维持交互文本会话也很平常，但对电子邮件来说很少如此，因为应答总是不会在一个短的时段内被送出去。

很多应用使用即时消息，例如，一些网络游戏允许玩家在游戏正在进行时给他们的对手发送短消息。即时消息也可用于和语音整合，比方说 Bob 和 Laura 在电话中想要试着解释一个单词的正确发音。一个短消息比一个一个字母地拼那个单词更快。有一些扩展的 SIP 可以提供简单的即时消息服务。

它是如何完成的

[draft-rosenberg-impp-im] 定义一个叫做 MESSAGE 新方法用于在它的消息体中携带发送者写的消息。使用 SIP 来传送即时消息的最大优点就是，代理像对待一个 BYE 请求一样路由一个 MESSAGE 请求。因此，用于建立多媒体会话框架的配置可被提供即时信息的服务重用而不用作任何修改。代理为了路由 MESSAGE 请求不需要了解这个新的服务。

如图 6-8 所示为 Bob 和 Laura 处于一个会话中。他们使用一个 INVITE 建立一个 SIP 会话并且开始交谈。当他们讨论休假计划时，Bob 需要 Laura 解释怎样拼一个单词。他向她发

送一个即时信息，接着他们重新开始语音交谈。

6.3.6 用户代理的自动配置

SIP 的一个主要特性就是它提供用户注册的可移动性。用户代理向 SIP 服务器注册它们当前的位置，以便于这些服务器可以正确地路由 INVITE 请求。对 SIP 用户代理来说，为了向某一个 SIP 服务器注册知道某个服务器处理某个特定域这一情况是问题关键。用户移动性越强，对于他或她来说越难记住所有需要用于配置他或她的用户代理的、在每个不同域中工作的信息，用户在这些不同的域中可以被找到。

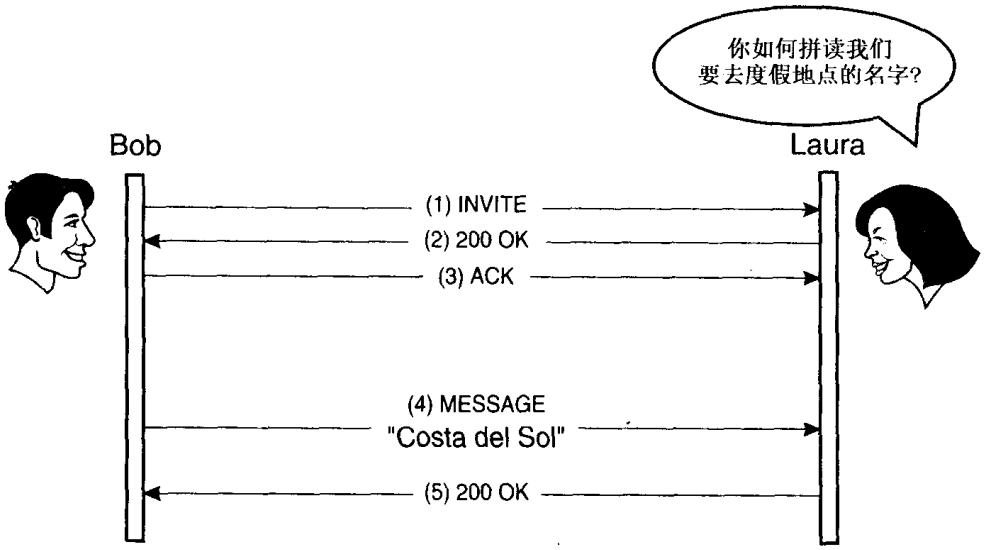


图 6-8 Bob 在一个语音会话中向 Laura 发送一个短消息

例如，Bob 在他便携式计算机上有一个 SIP 用户代理用于进行语音呼叫。当他在办公室工作时，他想使用在 company.com 域的 SIP 服务器。他知道域名：company.com，因此他为了向 SIP 服务器发送 REGISTER 而配置他的 SIP 用户代理。

Bob 经常下午在大学里工作。当他到达那时，他将必须改变他 SIP 用户代理的配置。他向在 university.com 域的代理发送 REGISTER，并且将这个代理配置为外出边界代理。所有从他的用户代理发出的请求将被首先发送到这个代理，这是因为这个大学有一个防火墙，仅接纳已经穿越了在 university.com 域中代理的 SIP 消息。这个安全代理起的作用很像很多公司使用的接入 Internet 的 HTTP 代理。

Bob 通常在 university.com 域和 company.com 域中工作，因此他用心记住了配置数据。可是，在任何一个其他的域，他可能就不知道如何去配置他的用户代理。

这个星期，Bob 正在访问一所其他的大学。在那里，他正与一个同事合作。当 Bob 用他的便携电脑上 Internet 时，他不知道他当前所处的域名或者他的外出边界代理。他只能亲自

找到这个大学的网管并且寻问这个信息。

利用发现 SIP 服务器的办法，可以使得 Bob 在外出旅行时使用他的 SIP 用户代理，使他的生活更舒适。有了这些办法，Bob 将不需要为配置他的 SIP 用户代理而烦恼。就像 Bob 在上 Internet 时一个 IP 地址自动分配给他的便携电脑一样，需要用于配置他的 SIP 用户代理的参数能自动传送到他的便携式电脑中。

它是如何完成的

这个配置过程可以通过两种方式自动进行。它们实际上是其他协议的扩展而不是对 SIP 的扩展，但是我们考虑到它们对 SIP 应用足够重要，因而需要提起它们。

一个选择是使用动态主机配置协议（DHCP，Dynamic Host Configuration Protocol）[RFC2131]来找回 SIP 服务器所在的域名。使用同样方式，一个便携机使用 DHCP 来从一个 DHCP 服务器得到它的 IP 地址，一个 SIP 用户代理也可以使用 DHCP[draft-ief-sip-dhcp]得到一个 SIP 服务器所在的域名。如图 6-9 所示为 Bob 如何从一个 DHCP 服务器得到他的 IP 地址和他的 SIP 域名。

一个更先进的用户代理可以使用服务定位协议（SLP，Service Location Protocol）[RFC2608]来找到一个有着一定特性的 SIP 服务器。SLP 按照需要的服务器性能提供匹配的服务器位置[draft-kempf-sip-findsrv]。

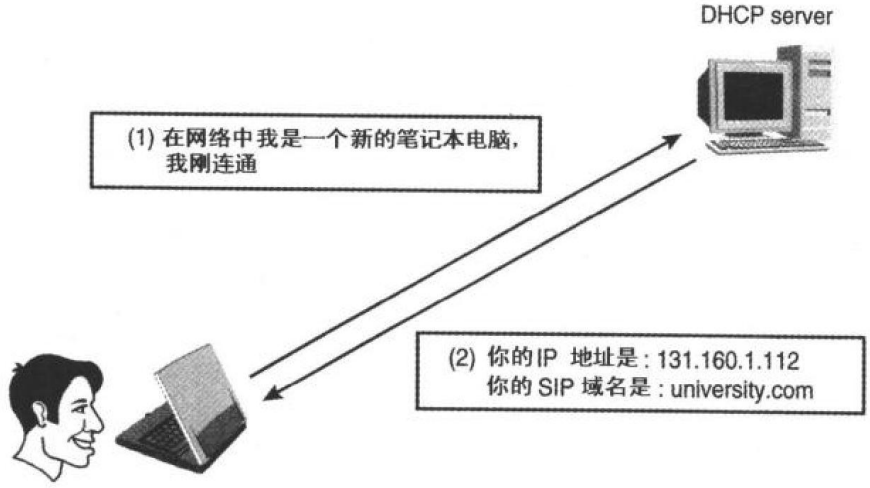


图 6-9　Bob 的便携电脑从 DHCP 服务器得到一个 IP 地址和一个 SIP 域名

6.3.7　通知之前必须满足的前提

到目前为止我们所看到的所有的例子中，UAS一收到一个 INVITE 就通知用户。如果用户接受邀请，会话就建立起来了。这个模式对那些可以被很快建立起来的会话工作得刚刚好。例如，在一个语音会话中，一旦用户接受了呼叫，他或她几乎立即就可以听到呼叫者的声音。

这个模式也可以为那些对建立时间并不是特别紧的会话工作。例如，如果一个用户收到了一个邀请（INVITE）来加入一个游戏会话，他或她并不希望一旦他或她按了接受按钮游戏就马上开始。用户希望（和指望）在接受和加入游戏之间有一些时间间隔。在游戏能够开始之前，将可能要从其他任务的应用中调一些游戏场景进入内存，同时在不同玩家的地图上同步每个玩家的位置。

但是，这种模式不适合于有着严格时间要求的会话。一个好的例子就是一个通过公用交换电话网络（PSTN）的电话呼叫。当在 PSTN 电话上收到一个呼叫时，我希望能够一拿起话筒就能说话。如果我不得不等上一会儿，哪怕只有 5 秒钟才能开始说话，那也是烦人的和不可接受的。

在 Internet 中，如果会话需要一个确定的 QoS 和 / 或一个确定的安全等级的话，会话建立的时间就会显著增加。在两个端用户间建立一个安全通道并提供 QoS（如使用 RSVP 协议）是需要很长时间的。

但是，即使可以提供一个能够满足各种变化的应用需求的特别快速的资源预留机制，仍然不能够预先知道网络是否将同意所需的会话 QoS。如果会话建立起来了而网络没有同意所需的 QoS，会话将失败。在一个语音会话中，这将意味着一个用户回答一个正在振铃的 SIP 电话，但是发现呼叫还没有建立起来，似乎是个幽灵在振铃。

这个问题既可以通过改变用户的期望又可以通过满足需求来解决。在使用蜂窝电话的早期，采用的是前者。固定电话网络的用户习惯于在他们拨完一个电话号码至呼叫者的电话发出第一声响铃之间只有非常短的延迟。蜂窝电话用户不得不为他们的呼叫的建立而等待相当长的时间。可是，他们仍然愿意接受这种延迟作为使用一种新服务特性——移动性的代价。一个经常使用 SIP 设备的用户可以接受较长的建立延迟，因为用户认为，为一个有着很广应用范围的新服务而作出一点点等待是值得的。

就像我们所知道的，有时需要简单地满足用户的一些需求。那些仅仅想要语音会话的用户很可能由于一个新技术（SIP）性能比他们当前使用的技术差而拒绝使用它。

因此，SIP 工具包中有一些方法在通知用户之前满足一个会话的前提条件，如安全和 QoS。这个机制保证了当用户同意参与进会话时，所有为会话建立所必须具备的条件都已经准备好了。

它是如何完成的

[draft-ietf-sip-manyfolks-resource] 定义了一个称为 preConditions MET（COMET）的新方法。COMET 被发送来表明所有前提条件都被满足了并且会话建立可以进行了。如图 6-10 所示是一个如何使用这种方法的例子。

Bob 想要和 Laura 建立一个需要 QoS 保证的会话。他向她发出的 INVITE 将包含 QoS 要求。Bob 详细说明他不想在网络已经同意一个确定的 QoS 之前通知 Laura。如果这个 QoS 不能被提供，Bob 宁愿终止这个会话。因此 Laura 的用户代理收到了这个 INVITE，并回送一个

有着 Laura 的 QoS 前提条件的“183　Session Progress”（这个应答使用本章前面描述的技术来可靠地传送）。接着 Bob 和 Laura 进行资源预留（如使用 RSVP）。Bob 在等待，直到他完成了为将一个 COMET 发送到 Laura 去的资源预留。当 Laura 的用户代理也完成资源预留之后，它通过通知 Laura 来恢复会话建立过程。

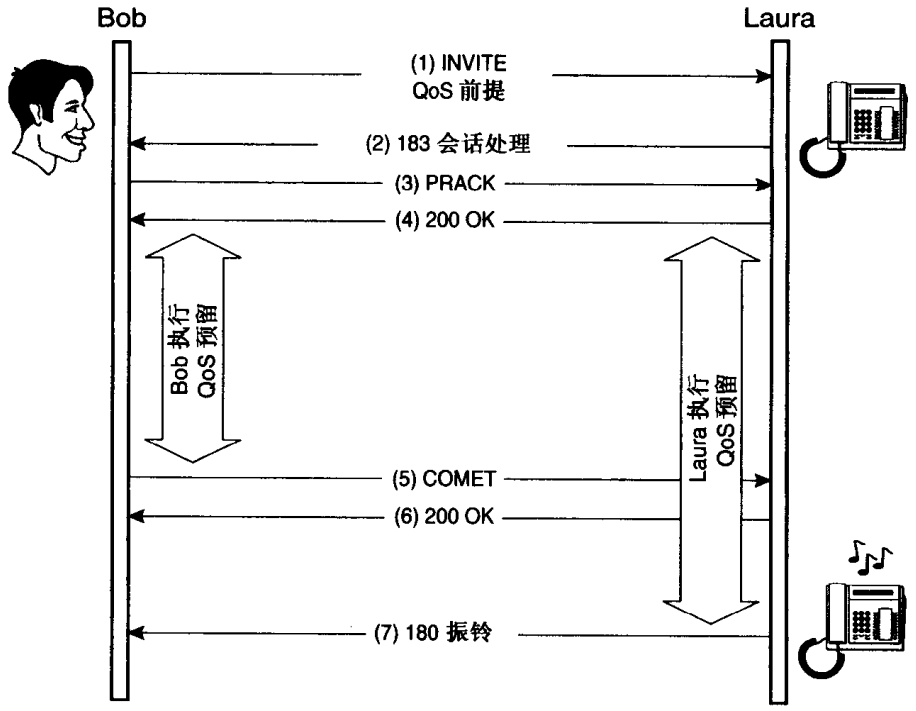


图 6-10　Bob 在 Laura 之前完成 QoS 预留

图 6-11 说明了如果 Laura 的用户代理在 Bob 的用户代理之前完成资源预留会发生什么情况。Laura 的用户代理等待直到一个 COMET 从 Bob 那里到达这边。一旦收到 COMET，它就通知 Laura。

6.3.8　呼叫者的喜好

我们已经看到 SIP 服务器以很多不同方式处理请求。例如，一个 SIP 服务器，可以在并行处理或顺序查找、派生或尝试单个位置之间进行选择。核心 SIP 规范将这些决定留给那些配置服务器的管理员。

可是呼叫者可能有不同的喜好。一些人可能想并行地创建所有请求而不是顺序地进行，这样节省时间，而另一些人可能想要他或她的请求被顺序地创建来节省精力。大部分人将希望他们的固定电话首先响，如果没人接听，那么接着他们的移动电话开始振铃。我的所有通信工具能够尽快被呼叫者接通这肯定是受欢迎的，但是如果我的所有通信工具同时响铃，这显然会让人讨厌。

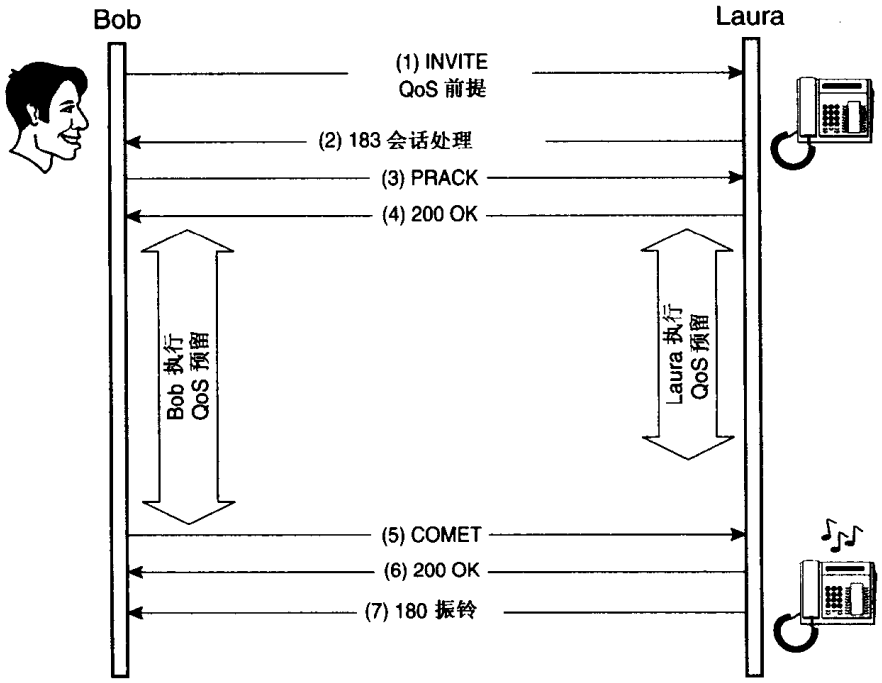


图 6-11　Laura 在 Bob 之前完成预留 QoS

除了影响呼叫者的请求是如何被处理之外，一个确定的呼叫者也可能有兴趣规定他或她想要拨打的终端。例如，当 Bob 呼叫 Laura 来谈论他们上个周末看的歌剧的时候，他不想打 Laura 的工作电话。他喜欢和 Laura 在私人线路上聊。而另一方面，当 Bob 的老板呼叫他谈论与工作有关的事情的时候，他不想拨打 Bob 的个人 SIP 电话。当 Bob 想要和 Laura 聊很久的时候，他不想拨打她的移动 SIP 电话，因为拨打这个电话比拨打她的固定 SIP 电话贵得多。因此，Bob 想要他的 INVITE 到达 Laura 的固定终端而不是她的移动电话。

了解呼叫者的喜好对提供服务来说也是有用的。一个呼叫税务所的用户可能英语讲得很勉强。因此，他想他的 INVITE 到达一个讲西班牙语的代办处来帮助他进行税务声明。

一些 SIP 的扩展使呼叫者能够描述他们的请求将如何被处理和他们想拨打何种类型的 SIP 用户代理。可是，注意到呼叫者的喜好必须同服务器的配置进行交互。这样，如果一个服务器被配置成将进入的呼叫转接到一个移动电话上，而呼叫者表明他或她想要 INVITE 到达一个固定终端，这样一个不匹配的矛盾就必须加以解决。在这种情况下，呼叫者将必须在请求中阐明他或她对固定终端的喜好是否可以被忽略，或者他或她根本不愿意接受连向一个移动终端。

它是如何完成的

为了完成这项功能，[draft-ietf-srp-caller-prefs] 定义了 3 个新的标题头和新的 Contact 标题头参数。新的标题头是 Atcept-Contact、Reject-Contact 和 Request-Diposvtion。Contact 标题头的新参数用于描述用户的终端。

图 6-12 举例说明了这些新标题头和参数的用法。Bob 向他的服务器发送一个 REGISTER，

表明他在 SIP:Bob@131.1601.112，并且详细说明这是一个用于商务的固定终端。因此，他的 REGISTER 的 Contact 标题头看起来像这样：

```
Contact:sip:Bob@131.160.1.112; mobility="fixed"; class="business"
```

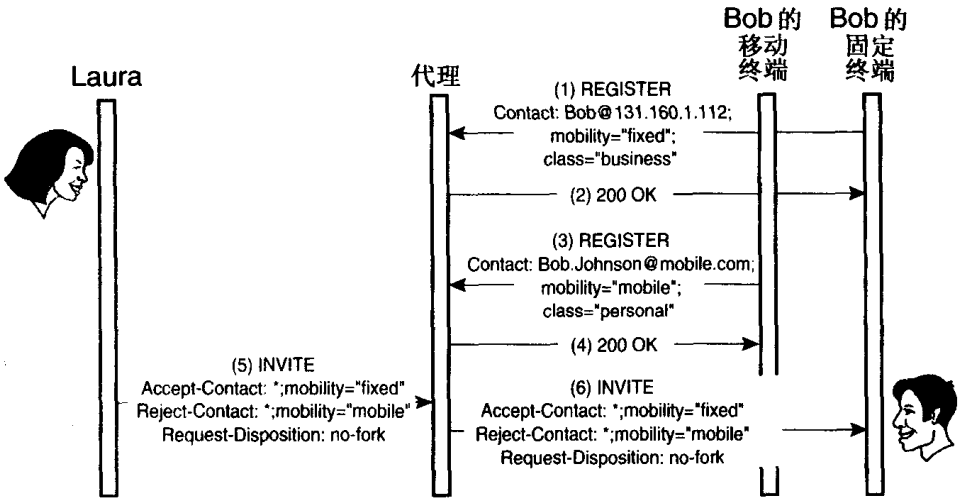


图 6-12　Laura 使用新的 SIP 标题头来表明她的喜好

Bob 发送第二个 REGISTER 说他也可能在 SIP:Bob:Johnson@mobile.com。他说明这是一个用于个人事务的移动终端。这个 REGISTER 的 Contact 标题头看起来像这样：

```
Contact: sip:Bob.Johnson@mobile.com; mobility="mobile"; class="personal"
```

现在 Laura 想要呼叫 Bob。她发送一个包含她的喜好的 INVITE。今天她要拨打 Bob 的固定终端并且在任何情况下不想拨打他的移动电话。为了表达这个愿望，Laura 使用了 Accept-Contact 和 Reject-Contact 标题头：

```
Accept-Contact*; mobility="fuxed"
Reject-Contact:*;mobility="mobile"
```

"*" 号是通配符。这些标题头合起来意思是 Laura 同意到达任何固定地址而同时她拒绝任何移动地址。Laura 也给她的 INVITE 添加一个 Request-Disposition 标题头，用于阻止她的 INVITE 被派生：

```
Request-Disposition: no-fork
```

6.3.9　事件的异步通知

到目前为止讨论的例子中，SIP 信令都是由一个行动触发的，这个行动是一个人直接发出的想要建立、调整或者终止一个特定会话的行动。当 Bob 拿起他的 SIP 电话呼叫 Laura（一个 INVITE）或当他挂上电话（一个 BYE）时触发了 SIP 信令。可是，除建立、调整或终止一个会话之外的其他事件也能触发 SIP 信令。SIP 事件通知构架（*Event Notification* Framework）使 SIP 能通知用户他们早先通过信令表示有兴趣的多种事件。这个事件通告机

制是创建服务的一个强有力的工具。假设 Bob 呼叫 Laura 而她正忙。即使他很着急，但他不想持续地每分钟拨一次号码来确定她什么时候有空。相反，他希望她有空的消息能传送给他。

另外一个使用事件通知机制的例子是存在服务的实现。一个存在服务通常由一个你的朋友和大学同学的列表以及他们是否可以与你通信的状态组成。Bob 可以从他的伙伴列表中看到 Laura 是否愿意在一个特定时刻接电话，或者她在那时仅仅想接收即时消息。

SIP 也可以被用于创建这种存在服务。每当 Laura 的状态发生改变，Bob 都被通知到了。这样，Bob 的伙伴列表反映了 Laura 状态的改变。

它是如何完成的

[draft-ietf-sip-events] 定义两个用于提供异步事件通告的新方法：SUBSCRIBE 和 NOTIFY。SUBSCRIBE 被一个 SIP 实体用来在一个特定事件中宣布它的兴趣。一个 SIP 实体预定一类事件中的一个特定事件。当这个被预订的事件发生时，NOTIFY 请求就被发送出去，其中包含着这个会话的信息。

图 6-13 说明了以前描述的自动重拨服务是如何使用 SUBSCRIBE 和 NOTIFY 来实现的。Bob 呼叫 Laura，但她正忙着。Bob 向 Laura 的 SIP 用户代理发送一个 SUBSCRIBE，表明他关心 Laura 的状态，Laura 的 SIP 用户代理以一个“200 OK”作为对 SUBSCRIBE 的回答，并且向 Bob 发送一个 NOTIFY，其中携带有 Laura 当前的状态：忙。过了一会，Laura 挂电话了，并且她的用户代理检测到了她的状态发生了变化，于是向 Bob 发送一个有着她新状态的 NOTIFY：有空。Bob 将向 Laura 发送一个 INVITE，以便和她交谈。这次他的呼叫将会成功。

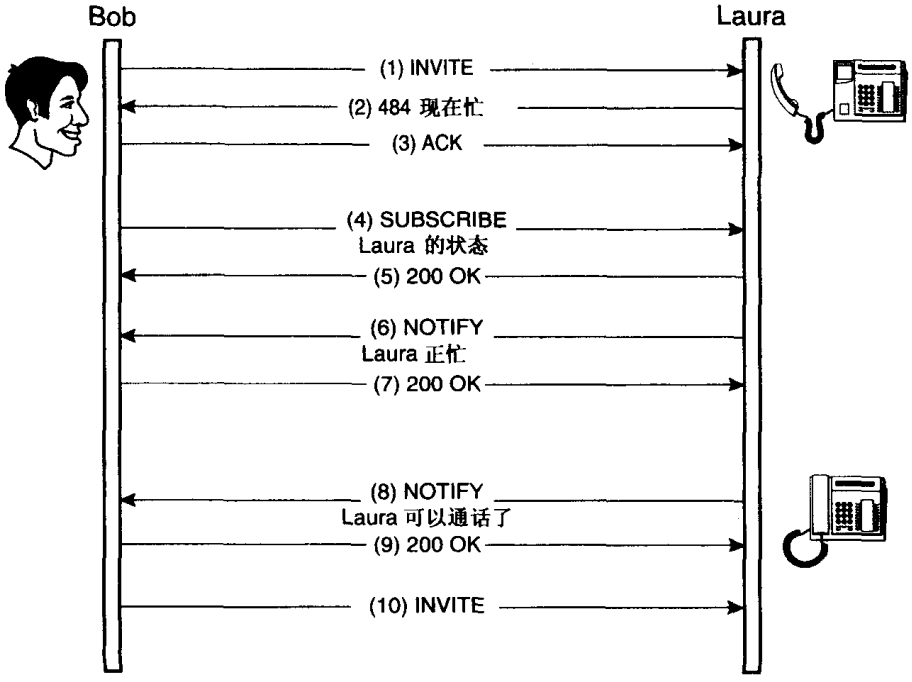


图 6-13 Bob 在 Laura 挂电话时被通知

图 6-14 说明用 SUBSCRIBE 和 NOTIFY 实现的服务的另外一个例子。Bob 正在参与一个使用了中央会议单元的 SIP 会议电话。所有参与者向会议单元发送一个 INVITE 来与它建立一个会话。会议单元就将所有进入的音频流结合起来，然后把它们发送给参与者。

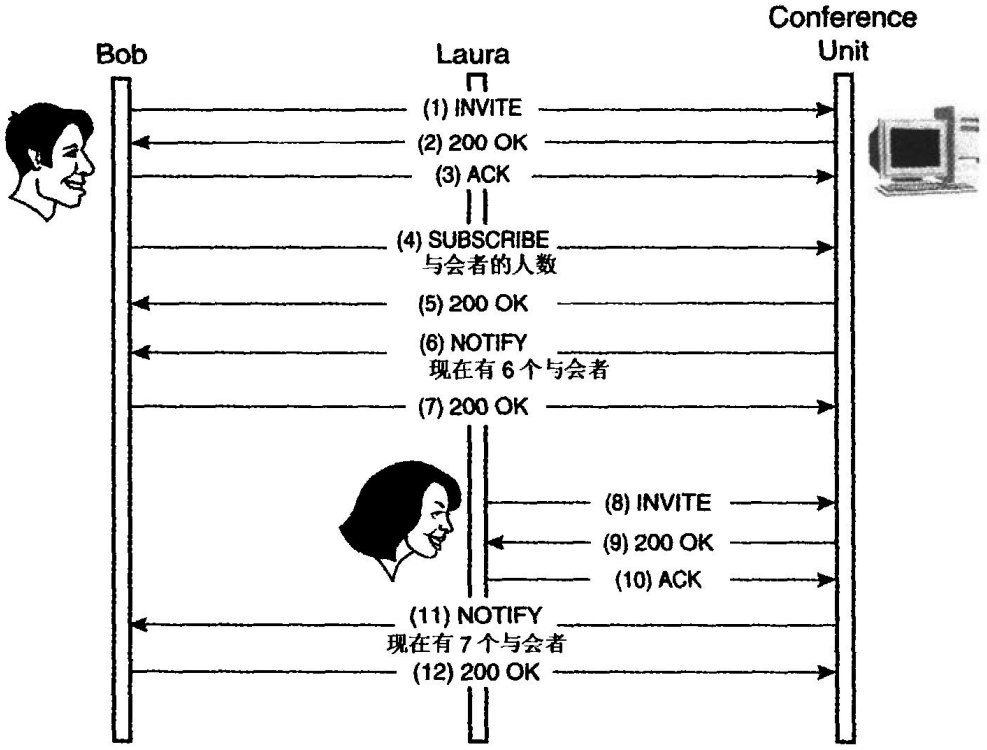


图 6-14　当 Laura 加入会议电话时 Bob 收到一个 NOTIFY

Bob 想知道在任何时刻有多少人正在出席会议。他向会议单元发送一个 SUBSCRIBE。会议单元通过在每当一个参与者加入或离开会议时向 Bob 发送一个 NOTIFY 的方法来使 Bob 更新参与者数量的变化。当 Laura 加入会议时，Bob 收到一个有着新的参与者数目的 NOTIFY。

6.3.10　第三方呼叫控制

我们知道，SIP 用户代理可以和另外的用户代理建立一个会话。可是，用户有时想要在其他用户代理之间建立一个会话而自己不想参与进去。当这些用户代理中的一个代表机器而不是人的时候，这种情况相当普通。如果会议电话被安排使用一种像 6.3.9 节描述的会议单元的时候，被邀请者可以选择身处其外而仍然保持与会议的联系。Bob 对在会议电话中说了些什么感兴趣，但他在那个特定时间很忙，因此不能出席会议。Bob 心想记录下这个会议内容将是个好主意。

可是，Bob 身边只带了 SIP 移动电话，它只有有限的存储能力。他将需要在会议单元和他的家用 PC 之间建立一个会话。为了解决这个问题，Bob 求助于第三方呼叫控制，它

使得一个 SIP 实体可以管理其他参与方之间的会话。一个在两个会话参与者之间建立会话的控制器将涉及到 SIP 信令中，但它并不需要出现在会话媒体中。在我们的例子中（如图 6-15 所示），Bob 变成了上述控制器，并且在会议单元和他的家用桌面机之间建立了一个语音会话。

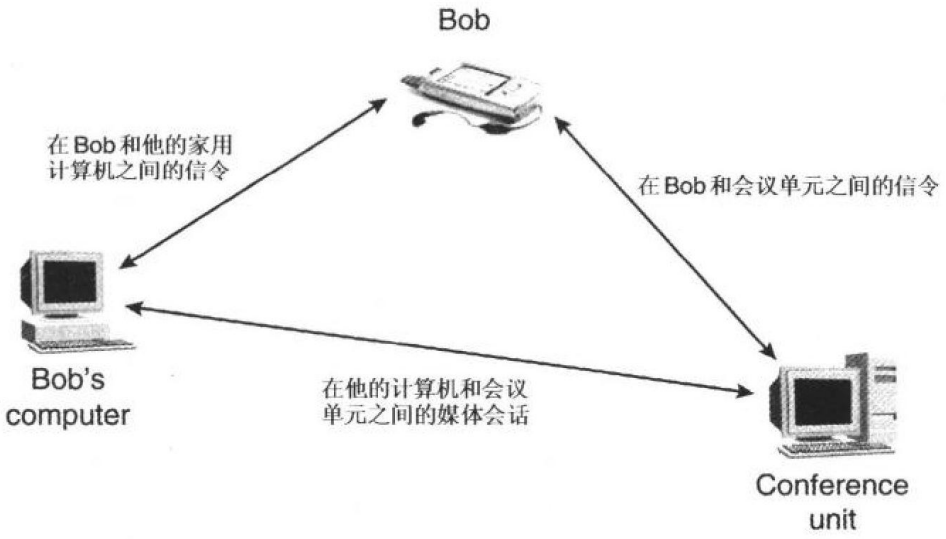


图 6-15　Bob 在会议单元和他家中的计算机之间建立一个会话

它是如何完成的

第三方呼叫控制[draft-rosenberg-sip-3pcc]不需要对 SIP 核心规范作任何扩展。控制器邀请参与者然后在他们之间交换会话描述符。图 6-16 说明了这个例子是如何实现的。Bob 发送一个没有任何会话描述符的 INVITE。由于 Bob 的计算机收到了一个无内容的 INVITE 请求，它希望会话描述符由 ACK 替代发送。Bob 的计算机按照通常的 SIP 处理过程向 Bob 返回一个在“200 OK”应答中的会话描述符，Bob 接着使用这个会话描述符来邀请会议单元。因为这第二个 INVITE 包含了由 Bob 计算机提供的会话描述符，所以会议单元将向 Bob 的计算机发送音频流。它以一个“200 OK”作为应答。Bob 收到了会议单元在这个“200 OK”中的会议描述符，然后将它放入一个 ACK 中发往他的计算机。现在媒体流在会议单元和 Bob 的计算机之间直接流动，而 SIP 信令通过 Bob 的 SIP 移动电话传输。

这个消息流最重要的特性就是它仅仅使用 SIP 核心规范。因此，受 SIP 支持最少的简单的用户代理可以被一个 SIP 控制器所控制。

6.3.11　会话传递

我们已经看到，第三方呼叫控制使一个 SIP 实体能够处于会话信令的控制中而同时媒体在其他实体之间进行交换。在某些情况下，控制器不想再监控（控制）会话的信令。作为代

替，他或她想要其他的实体独立地继续进行会话。在这时，控制器需要一种机制来将 SIP 会话传递给其他的实体。

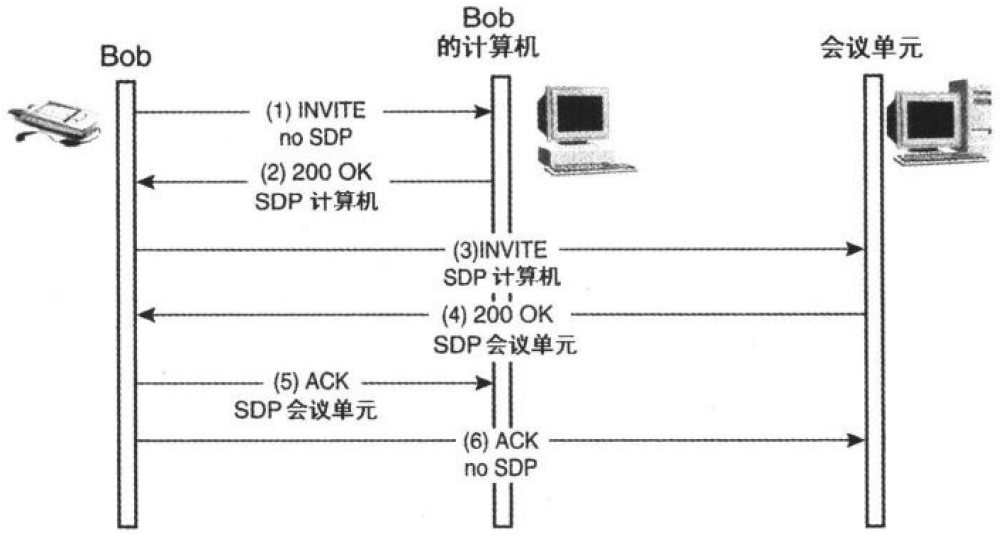


图 6-16　第三方呼叫控制消息流

最常见的会话传递的例子发生在当我拨一个电话找某人，而接电话的是我要找的人的秘书的时候。我告诉她我打电话的目的和我想要联系的人。接着这个秘书将我的电话转接给我要找的人或者是他的语音邮箱。这是一个会话传递的例子，因为一旦传递成功了，这位秘书就立刻停止接收任何与那个会话相关的信令或媒体。

SIP 扩展提供了会话传递功能。在我们的例子中，它使得秘书能够转接电话。不仅如此，它使她能够检查转接是成功还是失败了。在转接失败的情况下，秘书可以重新获得对第二个尝试电话的控制。

例如，Laura 用 Bob 的 SIP 统一资源定位器（URL）呼叫他，但他的秘书接了电话。为了转接这个呼叫，她指导 Laura 的用户代理在 Bob 的当前位置联系他。Laura 的用户代理使用新提供的 URL 向 Bob 发送 INVITE 并且终止前面的会话。

注意，如果 Laura 在这个结合点不终止第一个呼叫，他们可以使用相同的机制很容易地建立一个三方会议。

它是如何完成的

为了提供会话传递功能，[draft-ieft-sip-cc-transfer]定义了一个新方法：REFER。在这个机制背后的想法就是，一个 SIP 实体指导另一个去执行一个确定的行动。也就是，REFER 方法指示一个服务器向一个确定的 URL 发送一个明确的请求。图 6-17 说明了 REFER 机制。Laura 向 Bob 的办公室呼叫 Bob，而他的秘书接了电话。Laura 解释说她想和 Bob 通话。秘书通过发送一个有着表明呼叫保留会话描述符的 re-INVITE 来将 Laura 的电话置于保留状态。接着

她向 Laura 发送一个 REFER 指示她向 Bob 当前所处地点发送一个新的 INVITE。请求的 Refer-To 标题头包含有 Laura 要发送她的新 INVITE 的 URL。Referred-By 标题头包含有秘书的 SIP URL。Laura 将这个 Requested-By 标题头复制到她的新的 INVITE 中，然后向在 Request-To 标题头中包含的 SIP URL 发送。当 Bob 收到这个 INVITE 的时候，他知道这是一个由他的秘书所作的转接的结果，因为 Referred-By 标题头包含了这个秘书的 SIP URL。

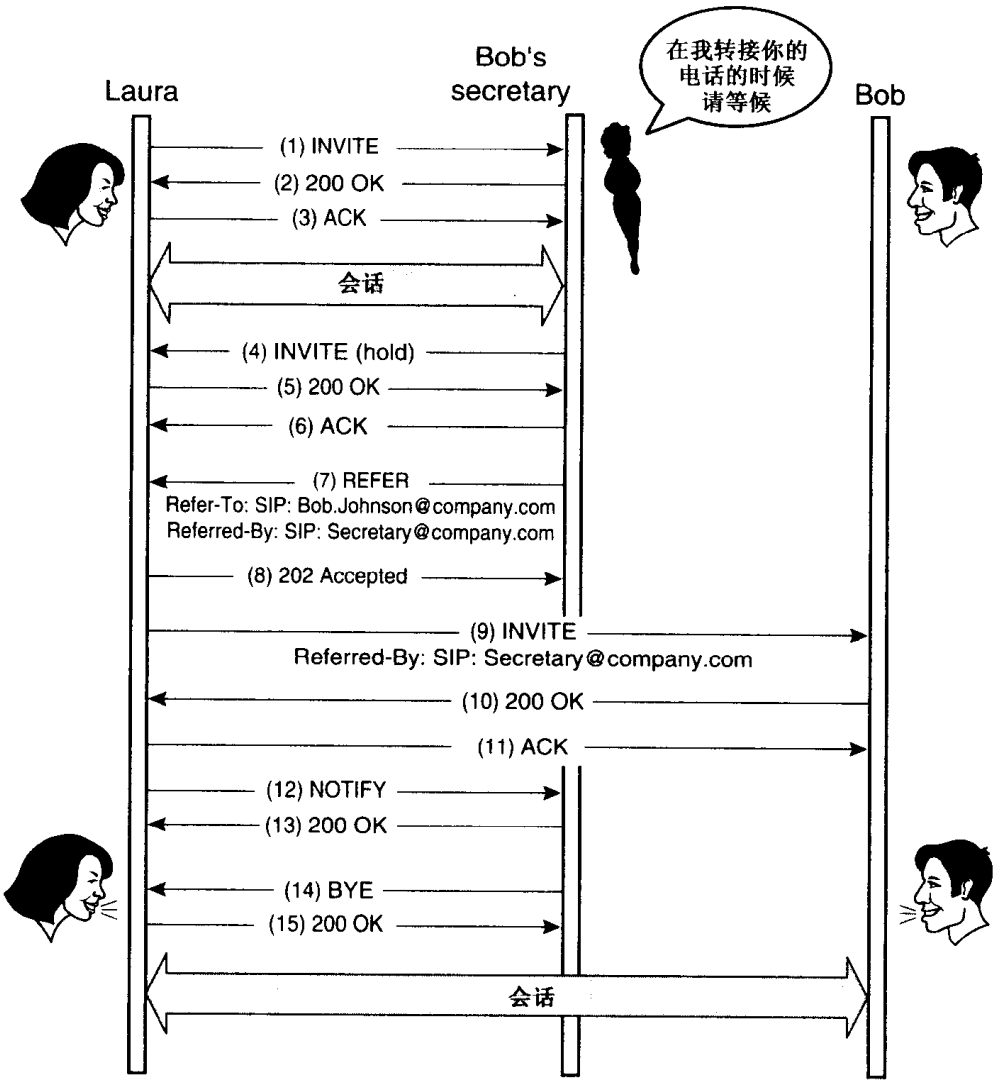


图 6-17　Bob 的秘书将电话转接给 Bob

当 INVITE 事务完成时，Laura 发送一个 NOTIFY 指明转接的结果。在我们的例子中，转接成功完成了，因此 Bob 的秘书发送一个 BYE 来终止她和 Laura 的对话。如果转接失败了，Bob 的秘书将能够查找到她和 Laura 的会话来继续同她说话。在 BYE 之后，*Bob 和 Laura 就*处在一个封闭的双方会话中，信令和媒体流都在他们之间直接交换。

6.3.12　发送命令

由于 SIP 不是作为一个主/从协议来运行，它不适合于控制紧耦合（Tightly Coupled）设备。在紧耦合设备的情形下，主设备向从设备发送命令，从设备以命令当前的状态作为应答。主设备控制从设备在每个时刻如何处理每个命令。如 H.248 或 MGCP [RFC 2705]这样的协议适合于解决这些问题，因为它们设计的时候考虑到了紧耦合设备。

可是，有时需要向处于一个主/从体系结构之外的设备发送一个命令。当发出命令和接收命令的实体是松耦合（Loosely Coupled）的时候，SIP 成为了一个主/从协议的有趣的可替代者。在这样一个环境中，当一个实体向另一个实体发送一个命令时，后者按它被请求的去做，然后报告命令的状态。发出命令的实体其行为并不像一个主设备，因为它不能每时每刻控制进程，并且得等待命令的最终结果。

它是如何完成的

[draft-moyer-srp-appli-auces-frameovork] 定义了一个新的叫做 DO 的 SIP 方法来携带命令。实体发出一个在消息体中携带命令的 DO 方法。收到 DO 的实体按消息所描述的执行某个动作。除了 DO 方法之外，一个用于描述命令的格式——设备消息协议（DMP，Device Messaging Protocol）被定义了。DMP 和 SDP 描述会话的方式一样，都用来描述命令。

使用这两个扩展，就可以控制能使用 SIP 的实体，且 Bob 可以用他的桌面计算机控制能用 SIP 的收音机。图 6-18 说明了 Bob 的 SIP 用户代理是如何使用 DO 来调节音量的。这样，用户代理可以在每次 Bob 接到电话时自动控制收音机使之静音，并且一旦通话结束了，通过另一个 DO 来使收音机继续播放音乐。

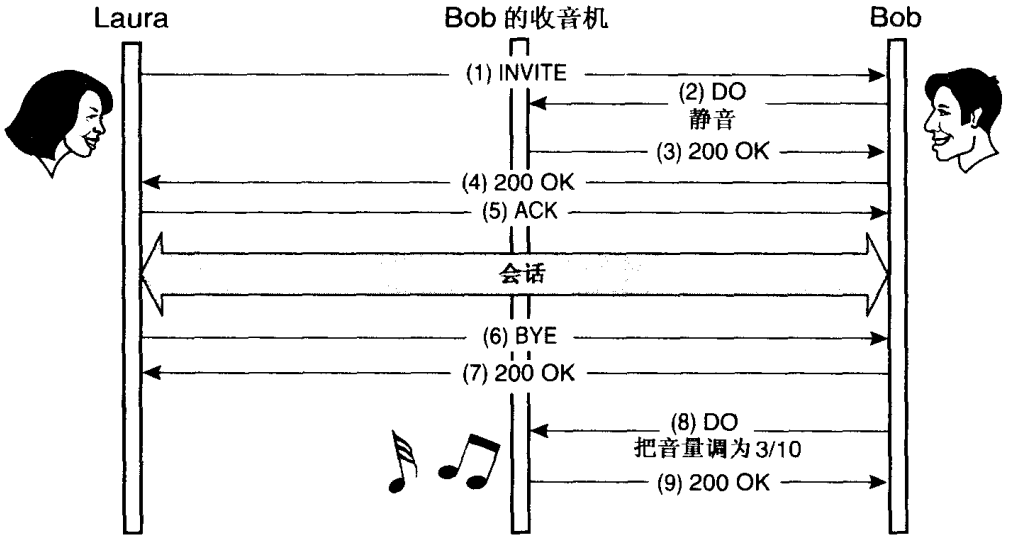


图 6-18　Bob 使用 SIP 控制他的收音机

6.3.13 SIP 安全

IETF 组织将安全视为任何协议必须具备的一个关键因素，因为 IETF 设计用于 Internet 中的协议，而 Internet 是考虑在一个敌对环境中运行的。Internet 用户通过很多方式保护他们的通信不受潜在的攻击者的攻击。SIP 的用户也不例外。可是不要将 SIP 安全和一个使用 SIP 的会话安全混淆起来也是很重要的。SIP 安全只关心 SIP 信令的交换。因此，Bob 可以向 Laura 发送一个加密的 INVITE，这样，没人知道他们正在建立何种会话，但是一旦会话建立起来了，如果他们传输没有加密的 RTP 报文，一个偷听者将能够听到整个会话。

例如，SDP 可以携带加密媒体会话的加密图形密钥。SIP 用户可以在会话建立期间交换密钥，然后使用它们以一种安全的方式交换媒体。本节的余下部分主要涉及 SIP 安全（验证，消息完整性和机密性）而不是媒体安全。

它是如何完成的

安全的一个主要方面就是验证。当 Laura 收到一个从某个声称是 Bob 的人那里发送的 SIP 请求时，她想确信 Bob 确实是发送这个请求的人。她需要一个机制来检查任何呼叫者的身份。

由于 SIP 基于 HTTP，它可以借用使用在 Web 中基于用户 ID 和口令用于消息验证的 HTTP 验证机制。当某人试着浏览某个受限访问的 Web 页时，HTTP 在决定是否让他或她访问之前得确定用户是准。在 SIP 中，这些机制用于接受或拒绝一个会话邀请。让我们主要分析两个验证方案：基本的和摘要的。在这两个方案中，一个服务器用一个“400-class”应答质疑客户端，向它要合适的凭证。客户端以一个包含有服务器所需要内容的新请求作为应答。质疑携带在 WWW-anthenticate SIP 标题头中，而凭证携带在 Authorization 标题头中，如图 6-19 所示。

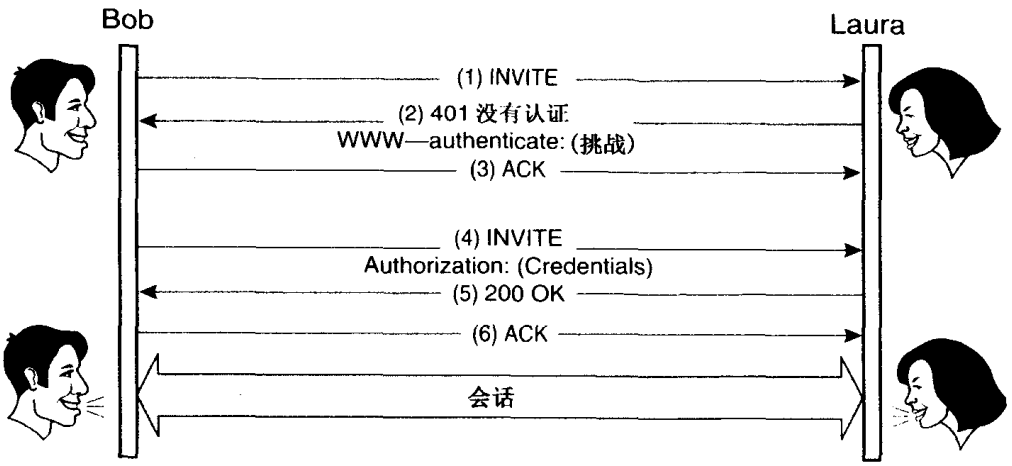


图 6-19 在 SIP 中使用的 HTTP 验证机制的例子

在基本验证方案中，客户端提供一个用户的 ID 和一个口令作为凭证。这个机制有一个很严重的缺陷，用户 ID 和口令都以明码传送。这样，任何偷听者仅通过窃听网络就可以轻

易得到它们。

摘要验证方案克服了那个不足。它也基于用户 ID 和口令，但不同之处在于，它们将永不通过网络传送。服务器通过传送一个现时值（Nonce Value）来质疑客户端。客户端计算出一个现时值、Request-URI、SIP 方法、用户、ID 和口令的校验和，然后送给服务器。服务器因此便可以确认客户端，知道用户的 ID 和口令而不必暴露它们。

这两种机制提供了一定等级的验证，但是对需要强有力验证的应用还是不够的。基本的和摘要的方案有着严重的限制。一个在 SIP 路径中怀有恶意的代理可以改变产生于 Bob 的 SIP 消息内容，然后转发给 Laura。当 Laura 检查消息的 Authorization 标题头时，她将会相信这消息来自 Bob。可是，代理已经改变了 Bob 请求的一些参数，所以即使 Bob 确实发出了这个最初的消息，但是 Laura 所收到的消息已不是真正由 Bob 发出的了。

为了克服这个在基本和摘要验证方案中的不足，有必要与验证方案一起提供消息完整性的证明。这是使服务器可以确信这样一个事实的唯一的方法，即收到的 SIP 消息在被客户端发出后没有被网络中任何实体修改过。

S / MIME 验证和消息完整性

这是一个通用的可用于传输 MIME [RFC2045]数据的安全传输机制。称为安全/多用途 Internet 邮件扩展（S/MIME，Secure/Multipurpose Internet Mail Extensions）[RFC 2633]，它足够用来交换安全电子邮件并同 HTTP 及 SIP 一同工作。S/MIME 提供了一种由此可以对一个消息内容进行签名（Signed）的格式，使得接收者可以验证它的完整性。

消息使用一种公开密钥加密机制进行签名。一个单独的用户有两个密钥：私有密钥（仅仅用户知道）和公开密钥（任何人都可得到）。用一个私有密钥加密的一些东西只能用公开密钥解密，反之亦然。

为了对一个消息进行签名，用户用他或她的私有密钥并将消息内容作为输入计算出一个数字签名。接着将签名加到消息中。这样，如果 Bob 用他的私有密钥对一个消息签名后，Laura 可以检验消息的完整性，并且确认它来自 Bob。她使用 Bob 的公开密钥来检验这个数字签名属于 Bob，他是知道他私有密钥的唯一的一个人。

S / MIME 机密性

可是，即使 Laura 可以确定 Bob 是发送者且消息在网络中没有被修改，任何偷听者仍然可以看到交换的 SIP 消息内容。S/MIME 提供了一种仅加密（Encrypted-only）的格式来保证消息内容仍然机密。Bob 用 Laura 的公开密钥加密他的消息内容。因为 Laura 是知道她私有密钥的唯一的人，所以 Bob 的消息只能由她解密。由 S/MIME 提供的两种格式，仅签名（Signed-only）和仅加密（Encrypted-only）格式，可以合起来提供验证、消息完整性和机密性。

端到端和逐跳安全

我们已经看到，代理为了路由请求消息和正确应答需要检查确定的标题头，如 Reguest-

URI、Via、To、From、Cseq 和 Call-ID 等。因此，这些标题头不能被端到端地加密。取而代之，可以使用逐跳加密。在一个用户代理和一个代理或在两个代理之间可以建立一个安全通道。在安全通道传送的所有内容都被加密了。然而，一个使用安全通道通向一个 SIP 代理服务器的用户代理不能够确信后者将使用另外一种安全通道通向下一跳。端到端和逐跳安全互为补充且应当一起使用。

然而，请注意，逐跳安全落在了 SIP 范围之外了，因为逐跳加密通常在底层使用 Internet 协议安全（IPSec，Internet Protocol Security）[RFC1827]或者传输层安全（TLS，Transport Layer Security）[RFC2246]来完成。在一个低层建立起来的安全通道对应用层仍然是透明的。消息体的端到端加密是关键，因为某些会话描述协议（如 SDP）携带着加密媒体的密钥。如果消息体没有加密，这些密钥将会暴露给任何潜在的偷听者。

在本章中我们已经描述了一组 SIP 扩展。在下一章中将分析这些扩展如何被不同的体系结构使用。

第 7 章　用 SIP 工具包创建应用

我们刚才已经看到几个对 SIP 进行增强的扩展的示例。每个扩展都提供了一个特定的可用于确定环境的功能。为了提供特别的服务，应用选择这些扩展的一个子集并将它们结合起来产生希望的结果。因此，对不同体系结构的不同服务使用不同的扩展集是一个规则，而不是例外。SIP 团体中一般的感觉就是现在 SIP 工具包对范围广泛的服务来说已经足够完备了。虽然一些扩展仍然处于 Internet 草案状态，但是把扩展集扩展为一个整体也被认为是事实上完备的。可能还会有一些较小的扩展在将来会被加入进来以补充目前还不具备的功能，但是最重要的扩展都已经定义了，并且已经准备用于任何服务体系结构中。

这一章描述在其他协议中使用 SIP 作为一个信令协议的体系结构。这些体系结构定义了不同的逻辑单元（Logical Box），并描述了有特定扩展集的 SIP 在它们中的使用方法。在很多例子中，我们将看到一个在体系结构中的特定的逻辑单元在行为上像一个 SIP 代理，但并不那样命名。一些人可能想知道这样做是否仅仅是用术语来迷惑我们，我也有同感。这样做的真实原因是，就 SIP 来说，一个逻辑单元可以从行为上看起来像一个代理，但是它有不同于 SIP 的其他协议使用的接口。因此，在某种情况下，一个特定的逻辑单元可能被视为是一个 SIP 代理，然而在另一种情况下，又可能被视为是一个 LDAP 客户端。被体系结构定义的新名字要能在总体上标识逻辑单元，即考虑它的全局功能性而不仅仅考虑 SIP 协议。

本章对设计者来说尤其重要，因为它展示了 SIP 是如何被用于解决多种实际问题的。SIP 能被不同体系结构采用标志着 SIP 的成功。一个没有日常应用的协议即使设计得很好也不会很有用。我们将看到 SIP 已经被一些体系结构，例如第三代移动通信系统（Third Generationof Mobile Systems，3G）和便携光缆设备系统（PacketCable）所采用，这些体系结构将会把 SIP 终端带入成千上万的用户之中。

7.1　第三代移动通信系统

第三代移动通信系统已经在电信世界中进行了大量宣传并引起了很多争论，它看起来像是将会把 Internet 世界和蜂窝世界融合在一起的技术。据称，3G 将对 Internet 拥有的所有成功服务提供无处不在的接入。仅在用户间提供语音和短消息服务的第二代（2G）移动电话将被具有 Internet 接入功能的先进的多媒体终端所取代。将 Internet 技术和广泛可用的蜂窝接入技术结合在一起，将使 Internet 变得具有可移动性。

第三代移动通信伙伴项目（3GPP，Third Generation Partnership Project）制定了基于演进的全球移动通信系统（GSM，Global System for Mobile communications）网络的第三代移动通

信系统的技术规范。3GPP 的成员有日本无线工业和商业协会（ARIB，The Association of Radio Industries and Businesses）、中国无线通信标准研究组（WTS，The China Wireless Telecommunications Standard）、欧洲电信标准化委员会（ETSI，the European Telecommunication Standards Institute）、美国电信标准化委员会（Committee T1）、韩国电信科技协会（TTA，the Telecommunications Technology Association）和日本电信技术委员会（TTC，Telecommunications Technology Committee）。由 3GPP 详细阐述的规范将被用于通用移动通信系统（UMTS，Universal Mobile Telecommunications System）。

尽管 3GPP 制定的 3G 规范基于 GSM 网络，而 3GPP2 制定的 3G 系统技术规范基于 ANSI/TIA/ETA-41 网络，但 3GPP 和 3GPP2 项目都是国际电信联盟（ITU，International Telecommunication Union）国际移动通信－2000（IMT-2000，International Mobile Telecommunications-2000）的一部分。

7.1.1 网络域

如 3GPP 定义的一样，3G 网络被分成 3 个不同的域：电路交换域、分组交换域和 IP 多媒体域（通常称为 IP 多媒体子系统）。

电路交换域采用电路交换技术在线路中提供语音和多媒体服务。这个域与 2G 系统大致相同，2G 系统中所有的服务也是通过电路交换来提供的。

分组交换域向终端提供 IP 连通性。终端通过此域获得 Internet 的接入。用户可在 Web 上冲浪、发送电子邮件和做在 Internet 上可做的大部分事情。这个域并没有在 IP 之上定义任何特殊的体系结构；它主要被视为一种接入技术。在第二代移动系统中，人们使用拨号连接或综合业务数字网（ISDN，Intergrated Services Digital Network）实现这个目的；3G 系统将通过分组交换域来提供 Internet 接入。用户使用这种连接来做什么只严格地由用户自己决定。

第三个域我们认为是最重要的一个。IP 多媒体域采用 SIP 作为主要的信令协议向用户提供基于 IP 的多媒体服务。在一个 3G 体系结构中，多媒体服务具有新的重要性并且被分离出来以便进行更详细的检查。

IP 多媒体域

读者可能要问的第一个问题是：IP 多媒体域与分组交换域有何不同。在分组交换域中的 3G 终端理论上可能使用蜂窝 Internet 接入与用户选择的 SIP 服务器联系起来，以便使他或者她获得所要的服务。尽管这种情形背后的逻辑是现实的，但蜂窝接入的特性使得这不太可能会发生。为了获得有效的数据传输，一个无线接口必须将要传送数据的所有已知的特性进行配置。否则会使数据传输速率慢下来并达不到实时流量传输的要求。除非网络知道使用哪种编解码器、端口号以及 IP 地址等等，否则语音就不能够通过无线接口传送。网络只有应用了如纠错码和标题头压缩等技术后才能使语音传输变得更高效。这些技术也提高了一个终端收到的语音在感觉上的质量，并超过了在分组交换中传输的质量。这就是将接入和蜂窝环境中

提供的多媒体服务结合起来的主要原因。

IP 多媒体体系结构

SIP 成为在 IP 多媒体域中使用的主要的信令协议。所有的 3G 终端都包含一个 SIP 用户代理（UA），且 IP 多媒体网络节点由 SIP 代理组成。可是，如我们前面解释的，它们实际上并不叫做 SIP 代理。它们被叫做呼叫/会话控制功能（CSCF，Call/Session Control Functions）。

3GPP 体系结构定义了 3 种类型的 CSCF，各有不同的任务。它们是代理 CSCF（P-CSCF，Proxy CSCF）、服务 CSCF（S-CSCF，Serving CSCF）和询问 CSCF（I-CSCF，Interrogating CSCF）。P-CSCF 是在网络和终端间的联系点。出去和进入的 SIP 消息都要通过 P-CSCF。一个 P-CSCF 运作起来像一个外边界 SIP 代理，因为所有的 SIP 请求不管它们最终目的是什么都被送向它。

S-CSCF 给用户提供服务。当终端注册时，它同 S-CSCF 联系，它向用户提供用户预订的服务。外出和进入的会话都将穿越同终端联系的 S-CSCF。这样，S-CSCF 可以为两种类型的会话提供服务。

I-CSCF 的任务是为特定用户找出相应的 S-CSCF。当 I-CSCF 收到一个请求时，它将它路由到处理会话的 S-CSCF。对于进入的会话，I-CSCF 是在服务提供商网络中的联系点；它在它的域中接收用户的请求，并将它们路由到相应的 S-CSCF；对于外出会话，I-CSCF 从用户那接收请求，并将它们路由到相关的 S-CSCF 那去。

服务提供商网络的 IP 多媒体域包含有许多这 3 种类型中任意一种的 CSCF。

另一个在 IP 多媒体域中的关键节点，不是一个 SIP 节点，叫做家庭预定服务器（HSS，Home Subscriber Server）。HSS 包含有与一个单独用户相关的 S-CSCF 和他或她的简介。作为一个结果，它知道用户在哪儿和他或她预订了哪种服务。CSCF 在它们需要这些信息时向 HSS 咨询。

7.1.2　呼叫流的例子

下面的例子将帮助我们理解在这个体系结构中不同节点的任务。

用户注册

图 7-1 显示了在一个 3G 网络中终端注册的过程。这样一个注册的要点是给用户分配一个 S-CSCF。这个 S-CSCF 将负责向用户提供服务。这个从终端来的 REGISTER 消息像任何从终端来的请求一样被送给 P-CSCF。P-CSCF 是处于终端和网络之间的联系点。P-CSCF 向 I-CSCF 发送 REGISTER 请求。I-CSCF 得为用户选择一个 S-CSCF。为了完成这个任务，I-CSCF 向 HSS 咨询。注意，在它们之间使用的协议不是 SIP。HSS 可被视为一个定位服务器，它和 SIP 实体交互，但是不使用 SIP 协议。

从 HSS 来的应答包含了足够的信息，因此 I-CSCF 可以为用户选择一个特定的 S-CSCF。这时，REGISTER 请求被转发给所选择的 S-CSCF。一旦收到这个 REGISTER，S-CSCF 就从 HSS 下载一个用户的简介，这个简介会告诉它用户需要哪种服务。最终，它用一个“*200 OK*”作为对这个 REGISTER 的回答。

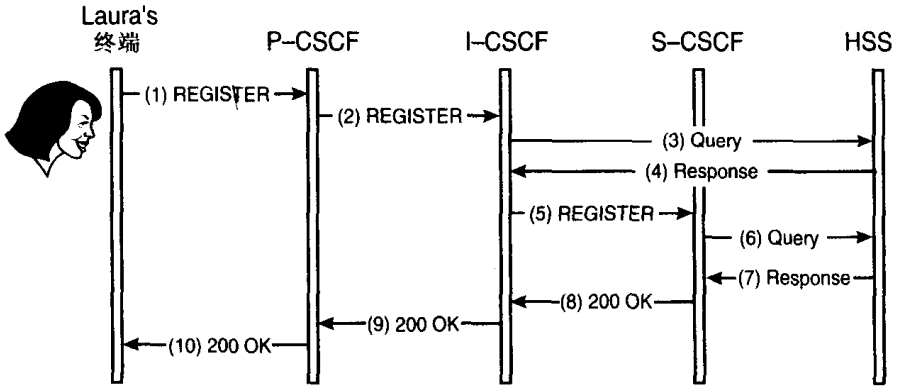


图 7-1　在一个 3G 网络中注册

从一个正在访问的域来的注册

当用户不在他或她自己的本地网络中时，前面的情形就会变得更复杂。3G 系统的一个核心功能就是保证用户可以从他们的服务提供商的网络漫游到任何其他网络（全球漫游）。这样，用户就可以在休假或者作全球旅行时使用他们订购的服务。只要用户开始漫游，两个不同的域（本地域和被访问域）就开始起作用。前者是用户的服务提供商的 IP 多媒体域，而后者是他们当前使用网络的 IP 多媒体域。

当用户建立了一个会话，他们的 SIP 消息通常穿越一些在被访问域和本地域中的 CSCF。图 7-2 说明了用户如何从他们正在访问的域向他们的本地域发送一个 REGISTER 请求。消息流和图 7-1 几乎是一样的，但在这个例子中，可以区分这两个域。

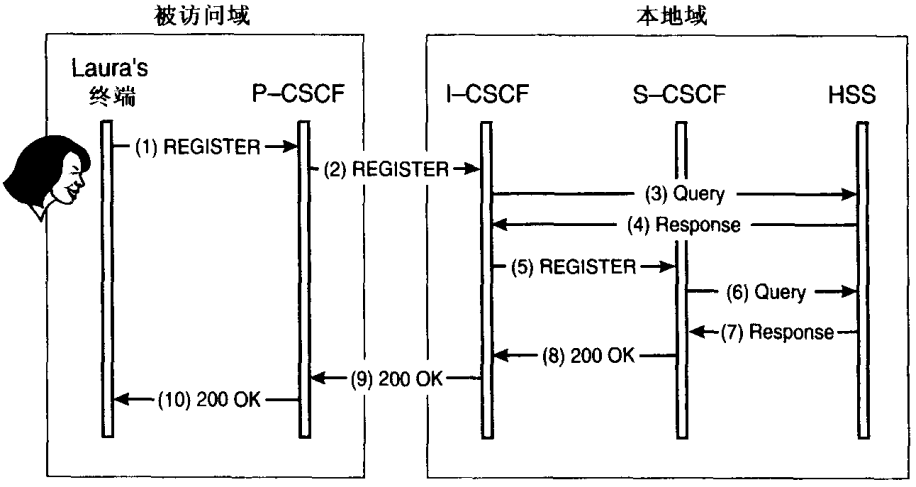


图 7-2　从一个正在访问的域中进行 3G 网络的注册

当被访问域的 P-CSCF 收到了 REGISTER 请求时，它检测到用户不在他们的本地域中，P-CSCF 向用户的本地域转发这个 REGISTER。从这时开始，以后的处理就像在本地域中进行注册一样了。I-CSCF 通过咨询 HSS 来选择一个 S-CSCF，接着 S-CSCF 下载用户的简介。

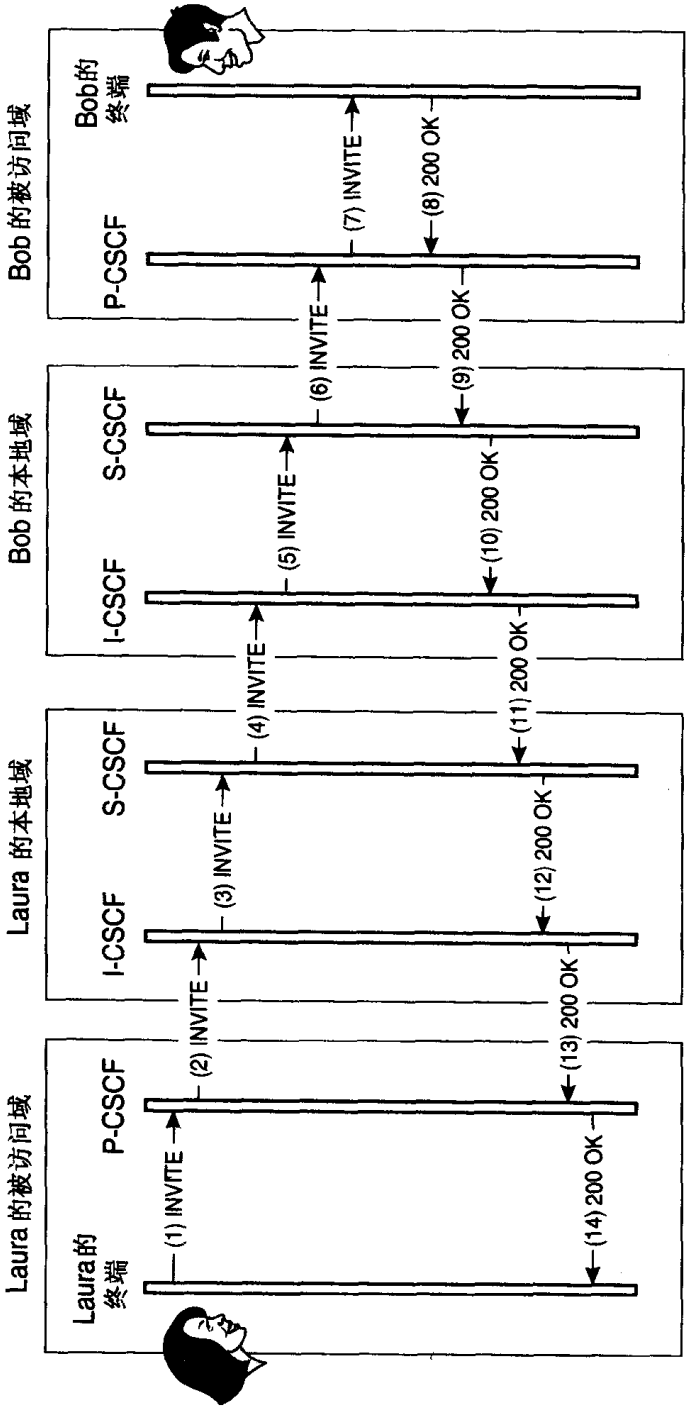


图 7-3　漫游用户间的会话

注意，在注册以后，HSS 知道了哪一个 S-CSCF 已经与用户关联起来了。因此，在一个进入的 INVITE 请求邀请那个用户的情况下，HSS 将可以提供足够信息来定位管理用户的 CSCF。S-CSCF 将向用户转发这个 INVITE。这种情况在下面的例子中继续描述。

在漫游用户间的会话

如图 7-3 所示是在两个正在漫游的用户之间会话建立过程的消息流。这是最普遍的现象，因为它说明了从一个被访问域出和进的会话的过程。

让我们分析一下在 Bob 和 Laura 之间建立会话过程中的不同网络节点的作用。Bob 正在到国外作商务旅行的途中。当他到达目的地时，他像如图 7-2 所示的那样注册他当前的位置。Laura 正在休假，所以也不在她的本地域。

Laura 向 P-CSCF 发送一个 INVITE 请求。P-CSCF 注意到 Laura 属于另外的服务提供商并向相应的网络转发这个 INVITE。Laura 的本地域的 I-CSCF 向 HSS（没有在图中显示出来）咨询，并向 Laura 注册时分配的 S-CSCF 发送这个 INVITE。这个 S-CSCF 将把这个 INVITE 发送给 Bob 的本地域。

从 Bob 的本地域看来，这是一个正在漫游的用户的入会话。I-CSCF 向 HSS（没有在图中显示出来）咨询并向处理 Bob 事务的 S-CSCF 转发这个 INVITE。这个 S-CSCF 知道 Bob 当前正处在一个被访问域中，因为 Bob 早先已经注册了他当前的位置。因此，这个 INVITE 被向 Bob 身边的 P-CSCF 转发，在那里 Bob 最终收到了这个请求。

7.2 即时消息和存在消息

即时消息和存在信息是使用 SIP 工具包来提供集成服务的非常好的例子。我们在前面的章节中看到了如何使用 MESSAGE 消息发送即时消息。也看到用户是如何使用 SUBSCRIBE 和 NOTIFY 方法来预订特定事务的。一个结合使用了这两个 SIP 扩展的应用可以提供即时消息和存在信息。

正被讨论的服务向用户提供了关于他或她朋友的状态信息，并使用户可以向他们发送即时信息。SIP 是使用相同的 SIP 服务器组成的网络框架来建立多媒体会话的另外一种方法。

这样一个应用的用户接口通常是显示在屏幕上的伙伴名单（Buddy List）。名单中的每个人有一个标记，表明他或她当前的状态。因为 Laura 已经把 Bob 加入到她的好友名单中了，所以他的名字将在名单中显示出来。如果 Bob 正在他的工作站上工作，他的名字将标记上“有空”（Available）。当 Bob 去吃午饭时，他的标记变成“没空”（Unavailable）。这样，Laura 就知道是否可以在任何一个给定的时刻给他发短消息。

现在 Laura 点击 Bob 的名字，选择“发送即时消息”，然后完成一条消息发送。如果在交换了一些消息之后，Laura 又想同 Bob 进行语音通话以澄清某些事情，她仅需要再次点击 Bob 的名字，并选择“建立一个语音呼叫”。

7.2.1 SIMPLE 工作组

即时消息传送和存在协议（IMPP，Instant Messaging and Presence Protocol）工作组特别创建了一种能使即时消息和存在信息的应用相结合的协议。可是，在提出这样一个协议的需求之后[RFC2779]，工作组不能取得一致意见。IMPP 工作组中的不同小组提出了不同的建议。所有这些不同的建议都有它们的赞同者也有反对者。

在合并了一些建议之后，工作组以 3 个主要的方法来提供即时消息和存在服务，以此来结束这场争论，这三个方法是：应用交换（APEX，APplication Exchange）、存在和即时消息（PRIM，Presence and Instant Messaging）和用于即时消息和存在影响的 SIP 扩展（SIMPLE，SIP for Instant Messaging and Presence Leveraging Extensions）。

我们将重点介绍第三个方法：SIMPLE，因为它是基于 SIP 的。SIMPLE 工作组第一次会议是 2001 年 3 月在美国明尼阿波利斯举行的第 50 届 Internet 工程任务组（IETF）会议。它的任务是定义一组可以使 SIP 提供即时消息和存在服务的扩展。使用包含有这些扩展的 SIP 将可以满足 IMPP 工作组提出的要求，这些要求为的是让每个为此目的而设计的 IETF 协议都可令人信服地被接受。当考虑一个问题的不同解决方法时，最低的要求是所有为即时消息和存在信息设计的协议都应能实现互操作性。

7.2.2 存在体系结构

一个即时消息和存在服务的实现使用基本上两种 SIP 扩展：MESSAGE 方法和 SUBSCRIBE/MODIFY 结构。用于存在服务的体系结构基本由两个节点组成：存在用户代理（PUA）和存在服务器。

存在用户代理与我们呼叫一个用于存在服务的 SIP 用户代理一样，向存在服务器注册用户的状态。这样，当 Bob 在他的笔记本电脑上作了注销时，Bob 的 PUA 将向存在服务器发送一个 REGISTER，报告 Bob 现在没空。

记住，单个用户可能有多个 PUA。例如，Bob 有一个 PUA 在他的工作站上，而另一个在他的笔记本电脑上。这样，存在服务器从这两个设备处收集 REGISTER 来追踪单个用户。

假设另一个用户对 Bob 的存在信息感兴趣，于是他预订 Bob 的存在服务器，这之后，每次 Bob 的存在信息改变时，存在服务器将一个有着新状态信息的 NOTIFY 请求传送给他。Laura 通常对 Bob 现在正在干什么感兴趣，于是她的 PUA 收到了一个 NOTIFY，说 Bob 已经注销了。Laura 将看到在她的电脑屏幕上 Bob 名字边的标记从“有空”变成“没空”。

在图 7-4 中，Bob 的两个 PUA 通过 REGISTER 向存在服务器发送存在信息。Laura 为了更新 Bob 的状态而预订他的存在信息。

这个例子说明了 Bob 的 PUA 发送一个存在信息的简单格式：有空或没空。可是 SIP 可以提供比这更多的信息。Bob 可能不愿意接受语音呼叫，但是想接收即时消息。或者 Bob 不

愿意接收他老板的语音呼叫，但很乐意接收 Laura 的电话。SIP 的 REGISTER 方法和 SUBSCRIBE/NOTIFY 结构给 SIP 提供了足够的灵活性以便在某种程度上实现这种服务。

7.2.3 即时消息

尽管存在信息可以用于提供范围很广的服务，但即时消息是与存在信息联系最紧密的服务。简单地提供有空或没空状态的存在系统通常用于告诉用户谁接收或不接收存在信息。

我们已经规定 MESSAGE[draft-rosenberg-impp-im]方法可用于在 SIP 中发送即时消息，如图 7-5 所示。使用 SIP 的好处是，它不仅仅能使存在信息和即时消息结合在一起，它还使得存在信息被用于建立任何类型的会话，包括即时消息和多媒体会话。因此，对提供合成服务的应用来说，SIP 是作为存在和即时消息协议的决定性的选择。

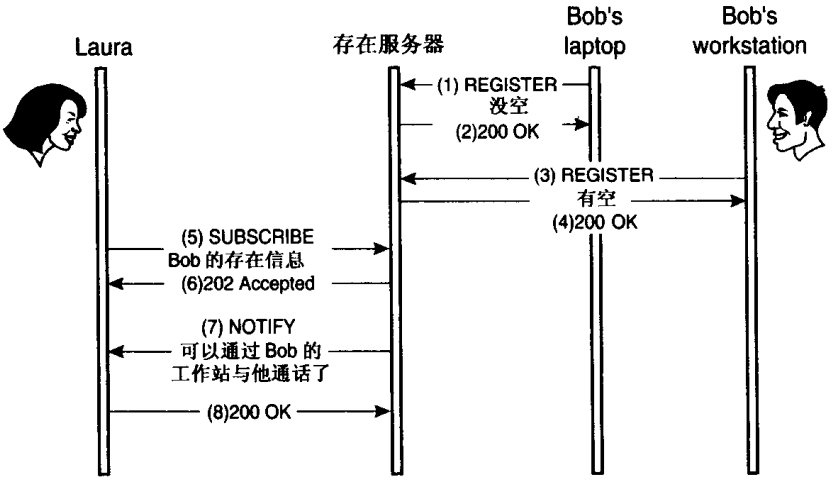


图 7-4 Bob 有两个 PUA 向存在服务器发送存在信息

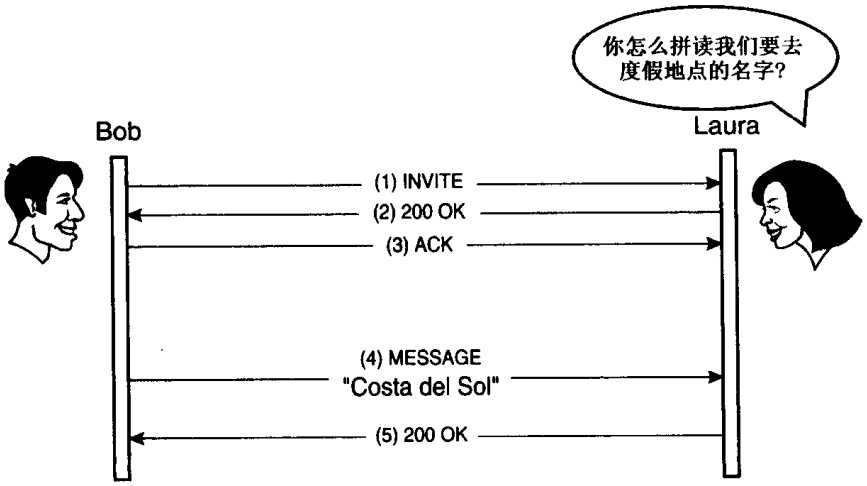


图 7-5 Bob 向 Laura 发送一个即时消息

7.3　便携电缆设备

便携电缆设备（PacketCable）是一个有线电视实验室（Cable Television Laboratories）和它的成员公司开发的项目。它的目的是通过电缆接入网络提供音频、视频和多媒体服务。例如，它预想有线电视的用户将能够使用他们的电缆调制解调器拨打语音电话。这个概念就是通过同一接入技术提供合成服务。

这个项目象征了对 SIP 的一个很大的认可，作为一种 3G 网络，PacketCable 使用进行了一些扩展的 SIP 协议满足现实的客户需求。这种类型的项目是对 SIP 团体的一种至关重要的回馈和新思想的来源。新的需求导致新扩展的产生。

电缆调制解调器可最大限度地用于电话所能用的地方，PacketCable 项目可将 SIP 多媒体服务提供给很大数量的用户。

7.3.1　体系结构

PacketCable 的体系结构定义了一组节点和这些节点之间使用的协议。SIP 是用于这个体系结构的协议中的一种，为此，SIP 增加了一组扩展。在 PacketCable 体系结构中与 SIP 操作相关的最重要的节点是多媒体终端适配器（MTA）和呼叫管理服务器（CMS）。

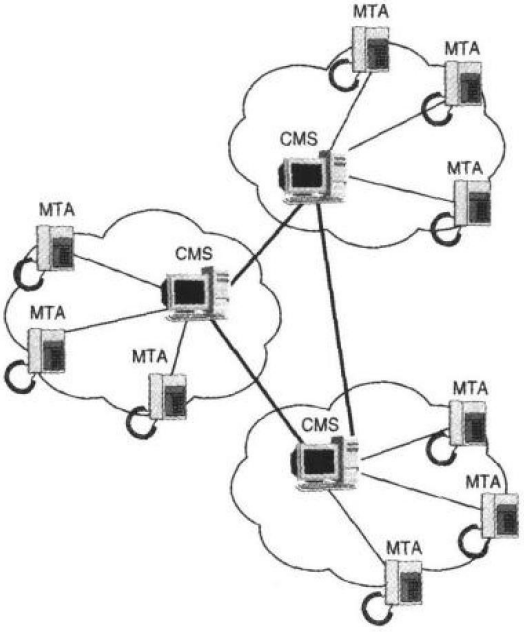


图 7-6　PacketCable 体系结构

MTA 基于用户的需求并且有一个类似于传统电话（POTS）的用户接口。在 CMS 和 MTA 之间，使用一个叫做网络呼叫信令（NCS）的协议。NCS 是一个主/从 IETF 协议，它是媒体网关控制协议（MGCP）[RFC2705]的扩展的变种。MTA 运行起来像从属：发送事务并响应 CMS 发出的命令。

例如，当用户挂机或者他或她敲入一些数字的时候，MTA 就通知 CMS。CMS 可能命令 MTA 通知用户，或者打开语音媒体通道。PacketCable 体系结构将可能会进一步发展，这样用于 MTA 和 CMS 之间的协议将可以用 SIP 取代 NCS。

CMS 基本上是一个 SIP 服务器。CMS 处理一个域并且通常控制多个 MTA，如图 7-6 所示。用于 CMS 之间的协议是呼叫管理服务器信令（CMSS）协议，它基本上是进行了一些扩展的 SIP 协议，这些扩展如临时应答的可靠传输和服务质量（QoS）前提。

7.3.2 呼叫流例子

图 7-7 说明了两个 CMS 之间的 SIP 操作过程，从图中可以看出 PacketCable 系统中实现的不同 SIP 扩展是如何在一个呼叫中一起工作的。PacketCable 实现了临时应答的可靠传输和 QoS 前提，因此当被呼叫者拿起电话时，网络将总是有足够的可用的资源来满足呼叫请求。

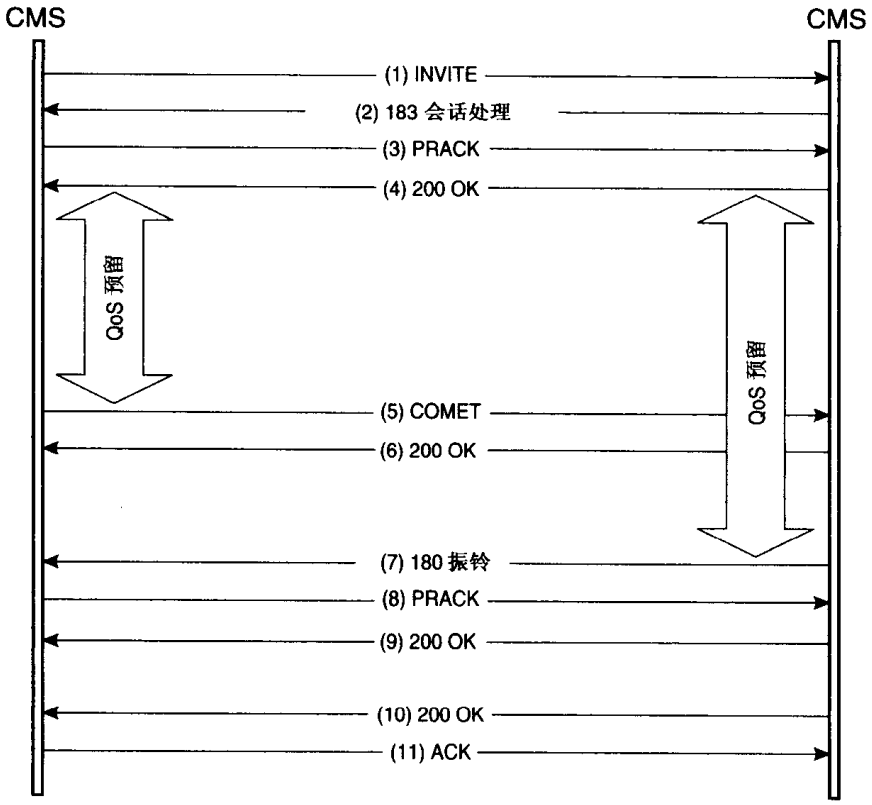


图 7-7 PacketCable 使用 QoS 前提和可靠的临时应答

7.4　PSTN 与 SIP 交互

电话应用是 SIP 的主要功能中的一个。在这个舞台上，很清楚，SIP 是创建服务的能手。传统的电话提供商现在为了利用 IP 电话（VoIP）服务的灵活性而开始建造语音传输的 IP 网络。

当 IP 网络被恰当地配置好并且采用了相应的 QoS 特性时，VoIP 可以提供能与公用交换电话网络（PSIN）相比（或更好）的相当好的语音质量。当前，许多 VoIP 提供的国际长途电话业务比 PSTN 便宜。因此，看起来 VoIP 将势必成为电话网络的未来。

可是，无论 SIP 实现电话服务有多么高效，也无论它可以达到何种等级的质量，它与任何新的电话系统一样，必须满足一个基本要求：必须可以与 PSTN 交互。

PSTN 是世界上最大的电话网络，即使将来它最终让位于 IP 网络，但它仍将存在很多年。一个巨大的电话框架不可能轻易扔掉并全部被新技术代替。

由于作为 SIP 的一个驱动因素的 VoIP 的突现，SIP 团体很快就认识到了实现与 PSTN 中使用的协议之间的兼容是使 SIP 能存活的主要努力目标。但就像我们在第 1 章中论述的一样，PSTN 的体系结构和设计原理都与 IETF 大体上不一致，与 SIP 也部分不一致。只能通过在 PSTN 和 IP 网络边界上通过网关进行协议转换才能实现兼容性，如图 7-8 所示。

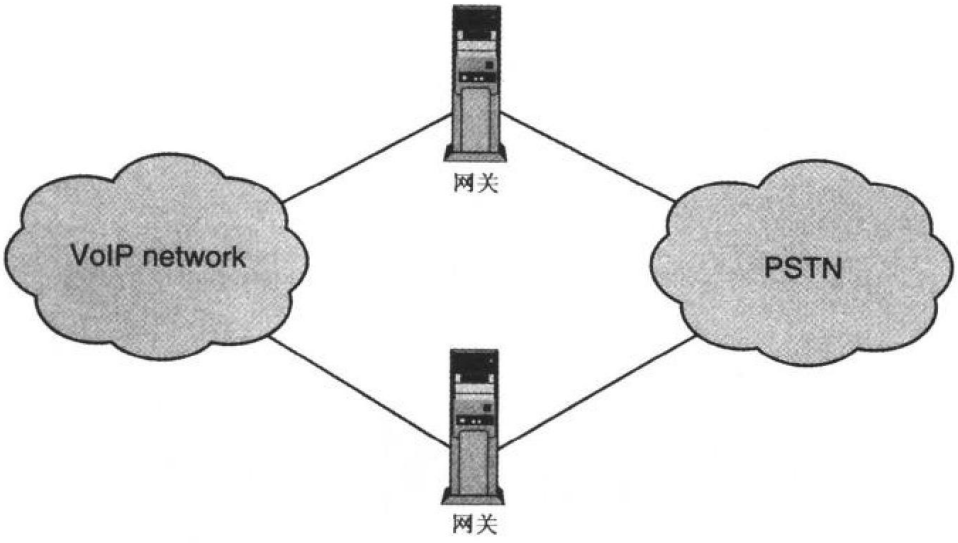


图 7-8　网络间的网关提供交互功能

网关在 PSTN 中是一个网络节点，而在 SIP 网络中是一个端点。用这种机制来替代，SIP 体系结构就不会在根本上受与 PSTN 协议交互的影响。一个网关通常看起来像 SIP 网络的一个 SIP 用户代理。这样，网络向产生于 PSTN 中的呼叫做出应答，就 SIP 而言，它只是由一个其

他 SIP 实体产生的呼叫。网关让 SIP 在与 PSTN 相连的情况下保持了它好的特性。实际上一个 SIP 与其他系统交互的通用规则就是保证 SIP 不用改变自身的行为来完成它的任务。在网络边界建立一个复杂的网关用于协议转换总比将更复杂的东西带入协议本身要好一些。

PSTN 使用多个不同信令协议，因此在 PSTN 和 SIP 之间存在多种类型的网关。

7.4.1 低性能网关

与特定网关适合的体系结构与它的性能和需求密切相关。低性能网关通常在单一的机器内集成信令和媒体处理。它们通常和 PSTN 接入协议交互，如数字用户线路 1 号（DSS-1），并且它们将支持住宅或小办公室的需求。

例如，好几年前，Bob 在家有一个 ISDN 电话使用 DSS-1 和它的 PSTN 本地交换机通信，如图 7-9 所示。随着技术进步，他决定用一个小的支持 SIP 和 DSS-1 的网关替代它的老式 ISDN 电话，如图 7-10 所示。现在，他将他的新网关连接到他在家中建立的一个小的 IP 网络（一个局域网）上。这样做的好处就是 Bob 现在在家的时候可以从他的计算机或 SIP 电话上接听或拨打电话。他升级了他的住宅通信来使它们和他将要依赖的商务通信手段匹配——这一切没有太多的麻烦。

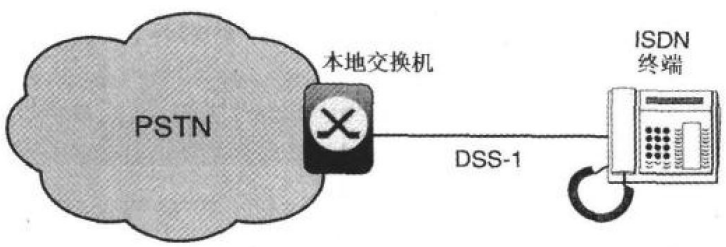


图 7-9 Bob 使用 ISDN 电话的老配置

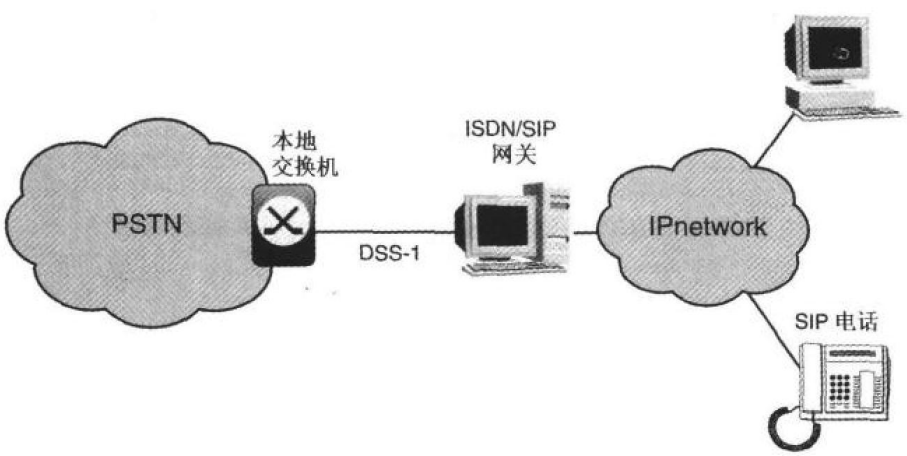


图 7-10 Bob 使用 ISDN/SIP 网关的新配置

让我们看另外一个例子。Laura 的办公室有几部电话连接到一个专用分组交换机（PBX）上，如图 7-11 所示。因为一个新提供商正在以更低的价格提供更多的服务，所以她的老板想要更换电话服务提供商。问题在于 Laura 公司的所有雇员已习惯于使用他们的传统电话（他

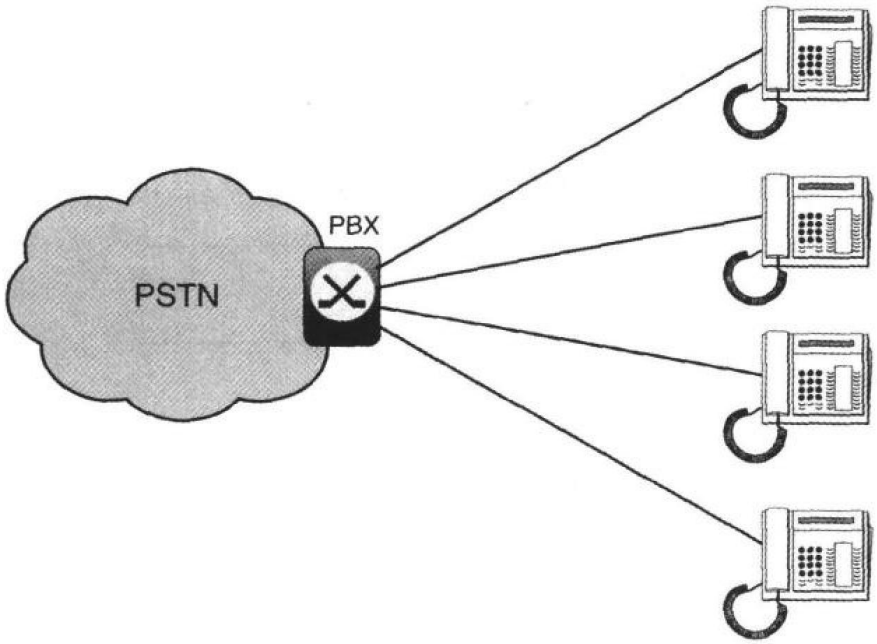


图 7-11　Laura 装备传统电话的办公室

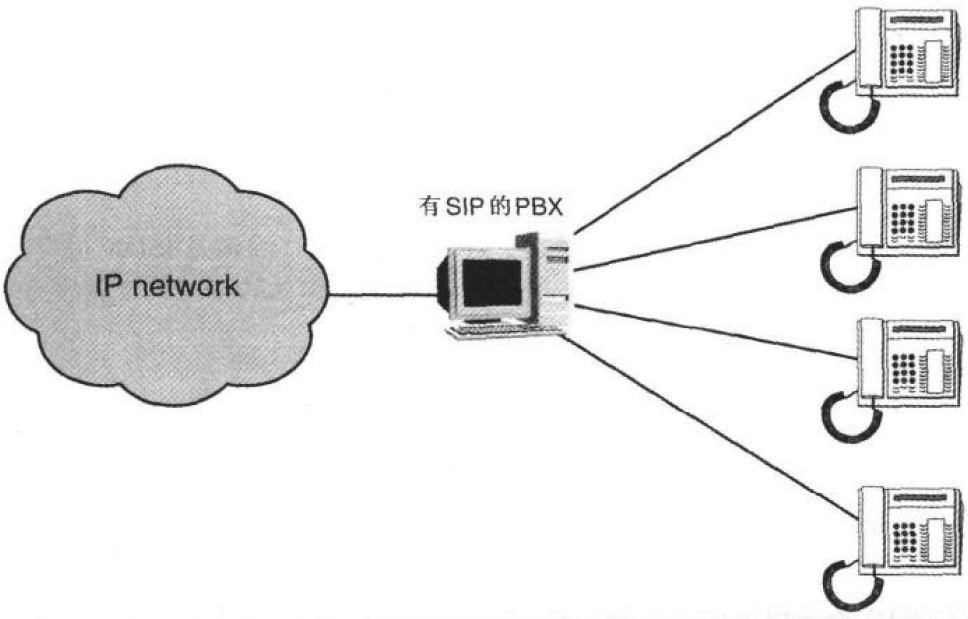


图 7-12　Laura 的有着新的 PBX 也有着传统电话的办公室

们纯粹只是喜欢），因而不想向 SIP 电话迁移。为了减少用户改变他们行为的需要和投资于新技术的成本，新服务提供商安装了一个网关，现在公司职员可以在没有真正注意到变化的情况下使用一个 SIP 网络，如图 7-12 所示。他们使用相同的电话机，按同样方式拨号，像从前那样接电话。

7.4.2 高性能网关

相对于单个主机——低性能网关，高性能网关通常是分布式的。也就是说，处理不同功能的网关的不同部分是分离的。如图 7-13 所示为高性能网关最通用的体系结构。

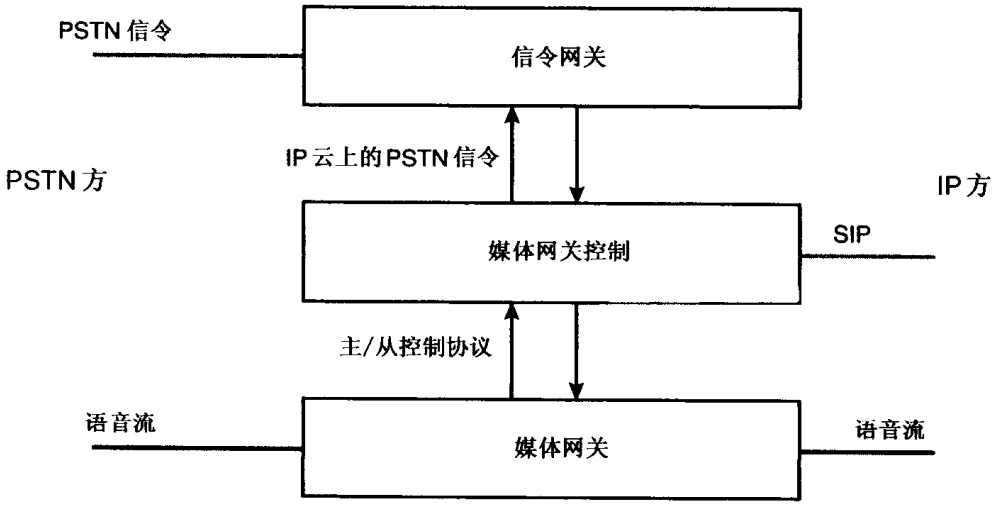


图 7-13 一个分布式网关的体系结构

信令网关（SG）接收 PSTN 那边传来的信令并封装在 IP 之上（反之亦然）。用于此目的的传输协议通常是流控制传输协议（SCTP）。SG 不修改这些信令消息，它只是将它们路由到相应的媒体网关控制器（MGC）。

一个 MGC 处理两个任务：（1）转换在 PSTN 信令协议（通常是 ISDN 用户部分 ISUP）和 SIP 之间的信令；（2）控制媒体网关（MG）。MG 负责媒体转换。它合并线路上的语音和 VoIP 接口。MGC 向 MG 发出如打开语音通道或关闭语音通道之类的命令。它们用于通信的协议是与 MGCP 或 H.248 类似的主/从协议。

7.4.3 用于与 PSTN 交互的 SIP 扩展

在 SIP 和 PSTN 之间的网关通常利用一些常用的 SIP 扩展。它们通常支持临时应答的可靠传输来向 PSTN 一侧传送进展报告。网关通常也支持 QoS 前提。在这些常用扩展之外，还有两个专为 PSTN 交互而设计的其他扩展：INFO 方法和用于 ISUP 和 QSIG 对象的 MINE 媒体类型。它们用于一个呼叫，这个呼叫是在两个网关之间，产生于 PSTN 之中，穿越一个 SIP

网络，最终再回到 PSIN 中。这种现象称为 SIP 桥，如图 7-14 所示。

在两个 PSTN 电话间的呼叫可能因为很多原因而要穿越一个 SIP 网络。它可能得利用一些目前仅在 SIP 网络中可用的服务。如果它不穿越 SIP 网络，所有这些服务对 PSTN 用户来说是禁止使用的。另一个呼叫可能被通过 SIP 路由，因为对提供商来说更便宜。一个服务提供商也可能使用他或她的 IP 网络来缓解他或她的电路交换网络的负担。在这种情况下，一个最初是传向一个 SIP 电话的呼叫可能被重定向到一个 PSTN 电话上。

在 SIP 桥的情形中，没有 PSTN 端用户愿意仅仅由于电话穿越了一个 SIP 网络而失去任何由 PSTN 提供的特性。由此，SIP 桥必须提供一定程度的特性透明性。在 PSTN 信令消息中包含的信息当在 IP 网络中转换或 SIP 消息时应予以保护，这样出口网关可以在远端用户终端上正确地重新生成 PSTN 消息。

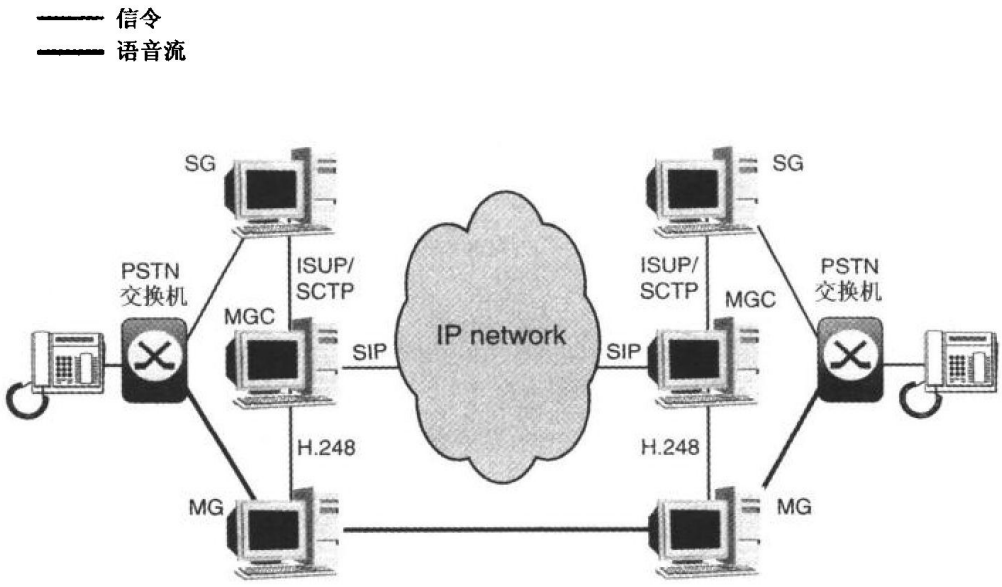


图 7-14　SIP 桥

多方消息体

问题：PSTN 协议携带某些信息不能够映射到任何 SIP 标题头上。这样，明显的从 ISUP 到 SIP 的一个简单映射的解决方案不起作用了，因为它将导致在 ISUP 消息中丢失某些信息，特征也会丢失。

为了阻止信息丢失，由 SIP 消息体携带 PSTN 协议消息。为 ISUP 和 QSIG 对象定义的 MIME 媒体类型被用于此目的[draft-ietf-sip-isup-mime]。QSIG 是一个 PBX 使用的 PSTN 信令协议。一个特定的 SIP 消息携带多部分组成的消息体，例如，它包含有一个 ISUP 消息和一个使用会话描述协议（SDP）的会话描述符。对于具体的 ISUP 与 SIP 交互的例子来说（这可能是最有趣的现象之一），用一组准则[draft-ietf-sip-isup]来帮助实现者在两个协议间作一致的

映射。

INFO 方法

MIME 媒体类型解决了在 SIP 消息中不能映射到任何 SIP 标题头的信息的发送问题。但那还不是整个问题。一些 PSTN 协议有着不能映射到任何 SIP 消息的消息。例如，如果一个网关在一个电话中间收到一个 PSTN 信令消息，没有 SIP 消息可以被送往出口网关去将它从 PSTN 传送过来。INFO[RFC2976]方法就是为此目的而创建的。INFO 方法用于在它的消息体中携带呼叫中的 PSTN 消息。如图 7-15 所示为在一个交互情况下的 MIME 媒体类型和 INFO 方法。

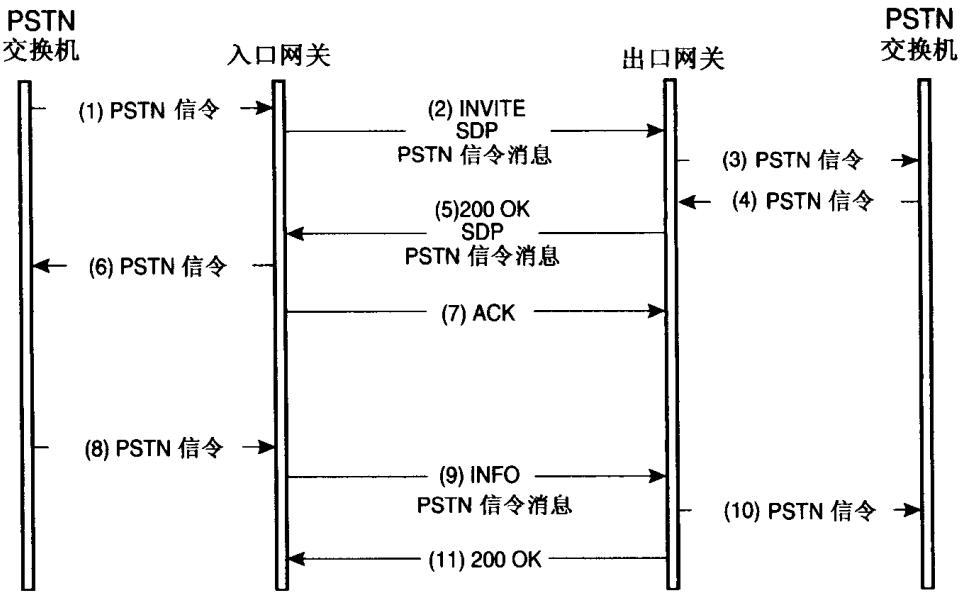


图 7-15　用于 SIP 桥的 MIME 媒体类型和 INFO 方法

INVITE 和“200 OK”

INVITE 和“200 OK”在它们的消息体中携带一个 PSTN 信令消息而使它们与会话描述符分开。ACK 可能携带另一个 PSTN 信令消息，但它通常不这么做，因为 PSTN 信令协议很少使用三次握手机制。

一旦一个会话建立起来了，入口网关会收到一个 PSTN 信令消息。它使用 INFO 方法将这样的信令传送到出口网关去。为了能在网关交换 PSTN 消息，INFO 方法不得修改 SIP 会话中的任何参数。

7.4.4　PINT 服务协议

使用 PINT 可定义与 PSTN 交互的另一种类型。IETF 的 PSTN 和 Internet 交互（PINT）

工作组研究用 Internet 设备请求 PSTN 电话服务的方法。例如，用户可能点击 Web 上的一个链接来请求支持部门的某人给他或她拨打 PSTN 电话。或者用户可能请求通过 PSTN 向一台机器发送传真、数据或者一个指示。

PINT 工作组提出了 PINT 协议[RFC2848]，由它提供此类服务。这个协议由 SIP 和带有扩展的 SDP 组成。在其他扩展中，PINT 使用 SUBSCRIBE/NOTIFY 方法接收关于调用服务的状态报告。

PINT 的目标是以一种受限的方式使用 SIP 来建立这样的会话，这些会话不属于 Internet 会话的范畴。例如，PINT 定义了 SDP 格式来描述一个传真会话，一旦这个格式被定义，SIP 就像平常一样使用它来建立会话。仅有的不同在于正被讨论的会话是在 PSTN 上的一个传真而不是在 Internet 上的实际传输协议（RTP）分组。我们又一次可以看到把会话描述符从会话建立的过程中分离出来的作用。同样的 SIP 协议可以被用于建立任何类型的会话。仅有的要求是可以在一个 SIP 消息体中携带的会话描述符格式必须存在并用于描述它。

图 7-16 说明了 Bob 是如何使用 SIP 给 Laura 发传真的。Bob 在他的办公室中没有传真机，但是她所需的信息在 URL: ftp://ftp.com.pany.com/Bob/document.txt。使用 SIP，他发送一个 INVITE 到 PINT 网关，它说明了将要被发送的信息（通过一个统一资源标识符（URI））并带有 Laura 传真机号码（1－212－555－5555）。PINT 网关处于 Internet 和 PSTN 的分界面上。它从提供的 URI 下载信息并发往 Laura 的传真机，同时，它向 Bob 发送传真状态的报告。

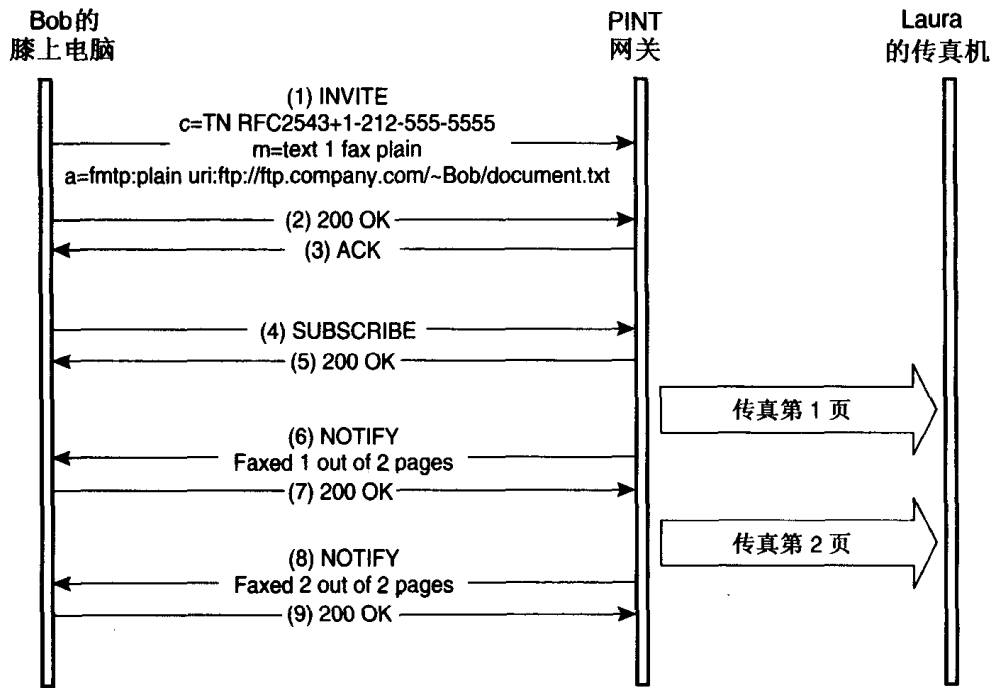


图 7-16　由 SIP 建立的在 PSTN 上的传真会话

用于描述这个服务的 SDP 的语法相当简单。一个新的类型（C 行）被定义作为电话网络（TN）。它包含有一个 PSTN 电话号码。M 行描述了媒体流（语言、传真或页面）的类别。

7.5 用于会议的 SIP

我们知道很多使用SIP实现的服务是由点到点的会话组成的。当Bob建立一个和Laura的会话时，他们加入这个双方的会话，信令和媒体直接在他们之间交换而不通过其他的人。可是，双方会话并不是 SIP 所能做得最好的。实际上，我们了解 SIP 和 SDP 最初是为MBONE 上的会议设计的，在 MBONE 上一个组播组的成员在一个组播地址上接收媒体发送的信息。

多方会议的形式是许多服务的基础。可能最简单的例子就是有着 3 个或多个成员的视频会议。在这个例子中，在参与会议的人们中间交换媒体信息和真正在一起开会是一样的。

没有什么规定要求与会的所有成员必须是人。会议在大多数情况下只有两个人参与。例如，Bob 和 Laura 可能通过他们的 SIP 电话开一个商务会议并记录他们的谈话，这样 Laura 的秘书随后就可以准备会议记录。在这种情况下，Bob 和 Laura 实际上处于一个多方会议中，因为用于记录的机器是一个会话的第三方成员。

当会议机制[draft-rosenberg-sip-conferencing-models] 被看作是服务使能者而不是仅仅是用于一大群人通信的手段的时候，它具有更强大的力量。无人会议成员的引入可以提供许多不同的服务，包括背景音乐和图像处理。

7.5.1 多播会议

第一个要注意的问题是尽管多播组的成员以多播地址接收媒体传递的信息，SIP 信令仍是会话参与方间的点到点信令。

例如，Bob 可以向 Laura 发送一个带有组播会话描述符的 INVITE。Bob 和 Laura 使用传统的单播来发送和接收 SIP 消息，但使用组播来发送和接收媒体信息。因此，一个像 SIP 的点到点协议可用于信令组播会议。

组播会议通常要预先安排事务，因此 SIP 不用于创建它们或终止它们。相应的，一个用于一个组播会议的 SIP INVITE 仅仅用于邀请用户而不用于建立会议会话。当一个用户退出会议时，SIP BYE 请求也不被使用。注意，每个参与方通常与和另一个参与者之间有 SIP 信令关系，但不是和会议中的所有参与者都有这种关系。

组播会议在有大量参与者并预先作好约定的情况下可工作得很好。可是，对于少量事务或特别会议，获取组播地址去传送媒体是不值得的。因此，SIP 支持另一个会议模式，它不能很好扩展但也具有其他一些优点。

7.5.2　端用户混合模式

对于一个双方会话来说，随着时间的推移，经常会不断有参与者加入进来。将一个参与者加入到会议中的最直接的方式就是一个当前会话成员邀请他或她。发出邀请的成员将收到从前面的成员和新成员发来的媒体信息。他或她将混合这个媒体信息并将结果发送给他们。

当处理这种混合媒体信息的人也是最后离开会议的人的时候，如图 7-17 所示的这种机制将起作用。但是当参与者的数量增加了，需要用来混合从所有参与者发送来的媒体信息的处理能力将很快超过这个用户设备能力的上限。因此，端用户混合模式很适合于小型特殊会议，但不能够扩展到大型会议。另一个 SIP 支持的会议模式，特别地称为多点控制单元（MCU），与端用户混合模式相比更具优越性。

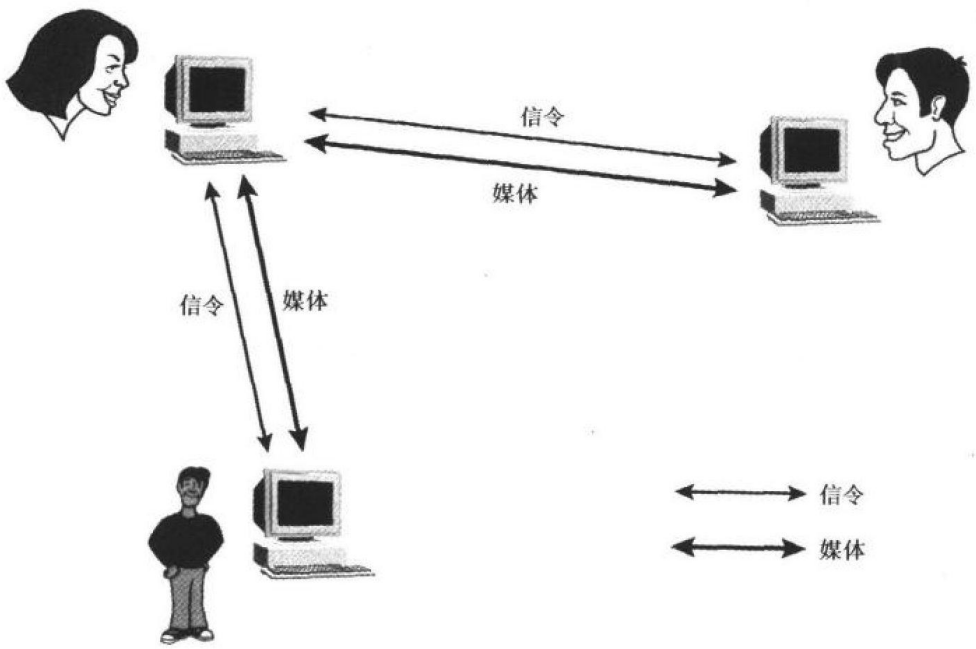


图 7-17　端用户使用混合会议模式

7.5.3　多点控制单元

多点控制单元（MCU）用于预先安排的或特殊的会议。在一个预先安排的会议中用户邀请 MCU，然后 MCU 混合媒体信息并向与会的所有人发送结果，这样就建立了在 MCU 和每个参与者间的信令关系。

还有一种情形，用于代替用户邀请 MCU，MCU 可以邀请参与者加入一个会议—— 一

个特别适合于预先安排事件的策略，MCU 可以立即邀请所有的用户。在两种情况下，MCU 都处理会议的信令和媒体，如图 7-18 所示。

基于 MCU 的会议可以比端用户混合会议扩展得更好。MCU 准备用于处理媒体混合并以高处理性能著称。可是，端用户混合会议可以被转换成利用 REFER 方法的 MCU 会议。为了从一个模式迁移到另一个模式，用户被指示去邀请 MCU。

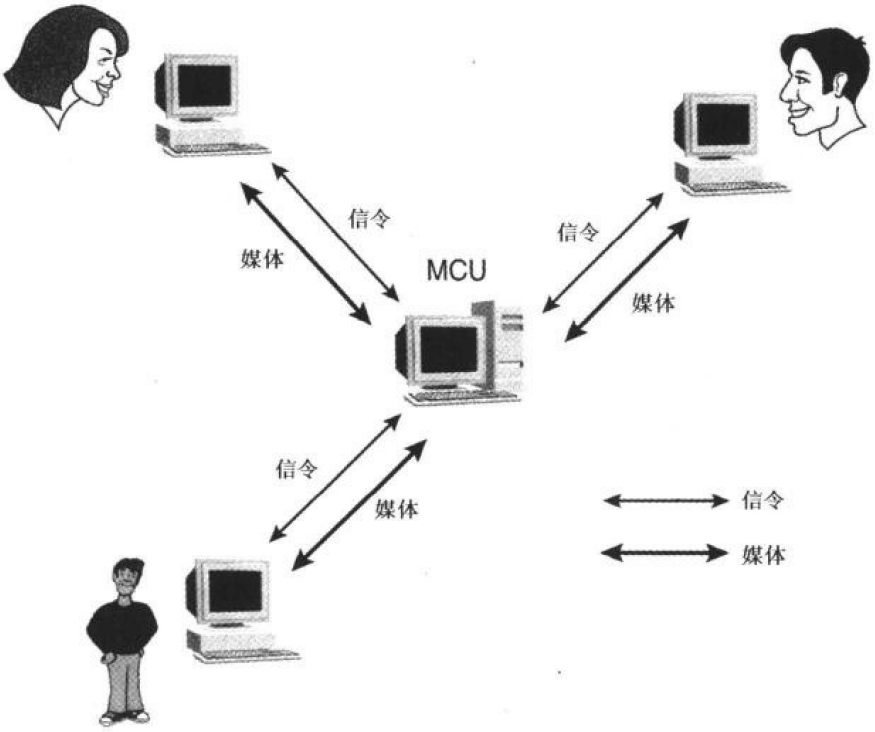


图 7-18　MCU 处理会议的信令和媒体

7.5.4　分散的多点会议

当媒体信息以一种分散的方式发送时，它形成了一个使用中央信令点（如一个 MCU）的特殊例子。当用户使用单播或组播发送媒体信息时，会议单元仅处理信令，如图 7-19 所示。当使用组播时，会议单元将一个 SDP 描述符返回给每个参与者，其中有将要被使用的组播地址。在单播的例子中，会议单元每当一个新参与者加入到会议中来的时侯，在一个 re-INVITE 中加入一个新的 m 行，来触发所有当前的用户向新参与者发送媒体。

这个模式简化了会议单元的创建过程，因为它避免了处理媒体的需求。在组播媒体的例子中，这种模式的扩展性相当好，并且会议可以处理很多信令关系。当使用单播媒体时，端系统必须并行将相同的媒体发送到几个不同的地点。低处理能力端系统可能当参与者数量增加时会遇到困难，并且因此选择迁移到处理媒体效果更好的 MCU 方式中去。

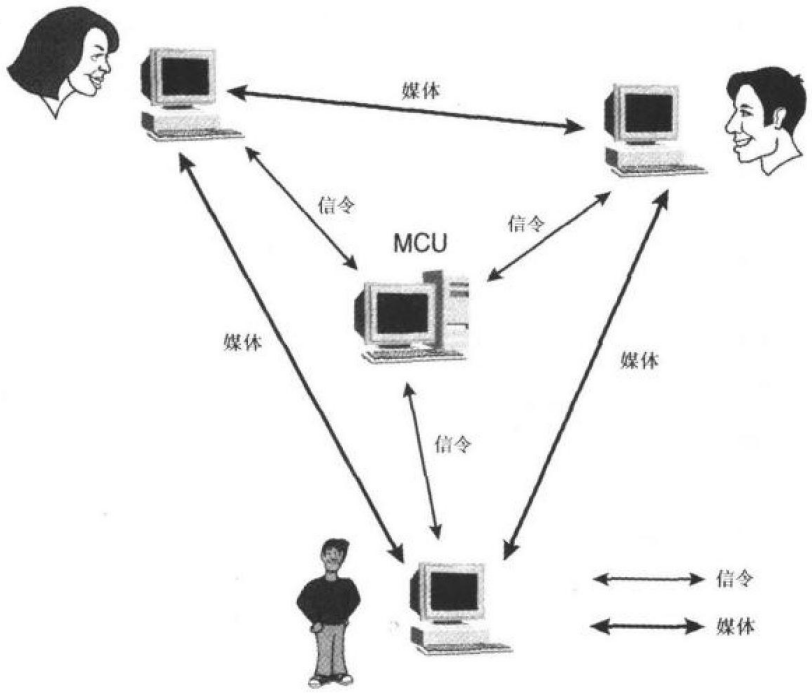


图 7-19　分散多点会议情况

7.6　网络应用的控制

当前，具有网络接口的家用设备的数量相对来说还是比较少的：一台桌面计算机，可能还有一个笔记本电脑，一个打印机，如此等等。这种情况预计在将来的几年中将会显著改变。当日常用具，如冰箱或闹钟都有一个网络接口的时候，它们就变成网络用具了。网络用具的主要好处是它们具有交互的能力和同其它网络设备交互的能力。这样，当冰箱中没有蛋黄酱时，它可以通过超市的 Web 页面订购新鲜的予以补充。床头钟可以得到关于交通状况的信息并设置相应的警报。如果 Bob 在清晨离家时忘了关走廊的电灯，他可以在他的办公室里关掉它。

已经设计了一些协议来控制家中的网络化的用具，可是没有可行的解决方案用于处理网络用具的中间域通信[draft-tsang-appliances-reqs]。恰当扩展后的 SIP 协议是一个扮演这种角色的候选工具。图 7-20 说明了各种 SIP 扩展是如何协同工作来控制一个家中用具的。假设一位修理工计划在清晨 Bob 工作时修理 Bob 的洗衣机。Bob 简单地预设他的门铃系统。当修理工按门铃时，他自动引起产生一个 NOTIFY 请求并发往 Bob。Bob 接着就与他家门上的照相机建立一个视频会话，并检查按门铃的是否是修理工。他也和他家门上的音频系统建立一个音频会话并告诉修理工没人在家。Bob 向修理工解释他洗衣机的毛病并决定让他修理。这时，Bob 向他的门发出一个 Do 请求使它开锁。现在修理工可以干活了，而同时

Bob 通过安装在修理地点的照相机监督他。当修理工完成工作并离开房间时，Bob 发送另一个 Do 请求来锁门。

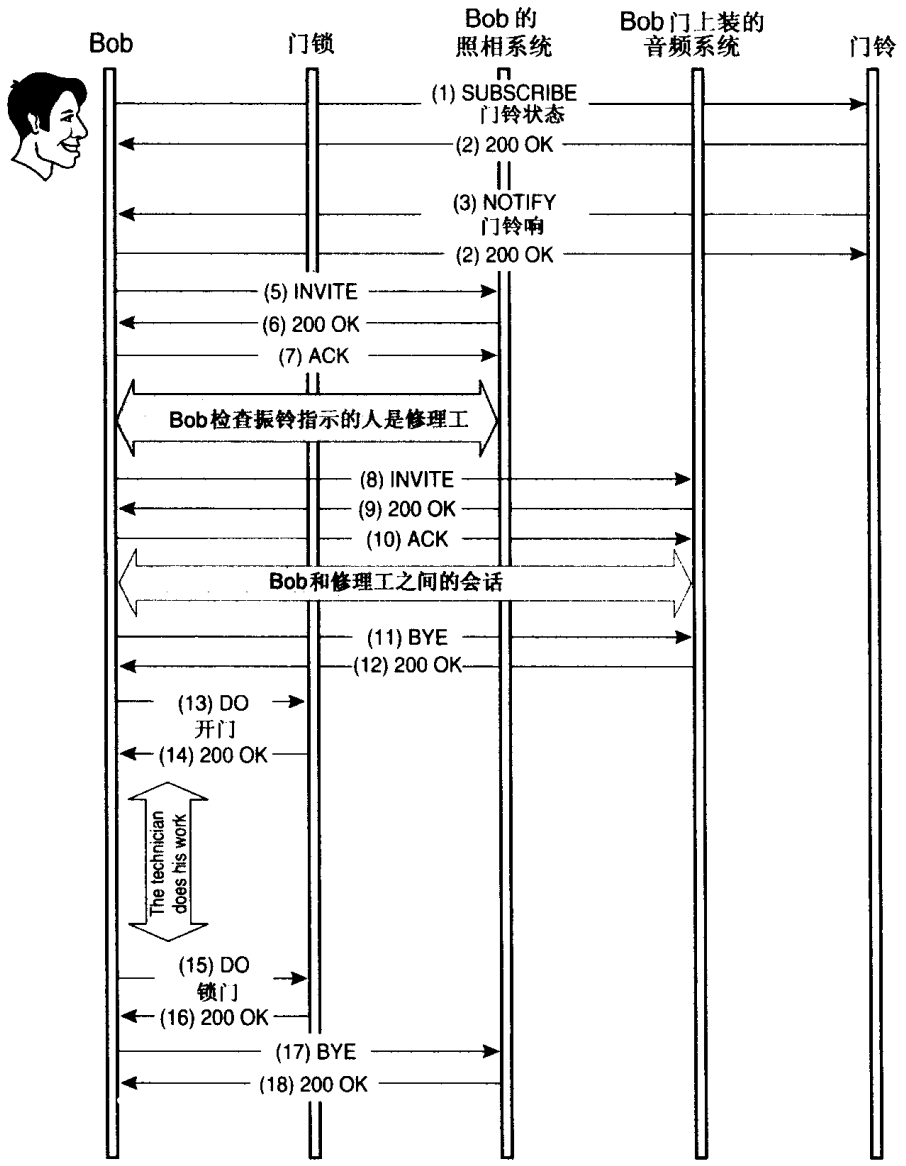


图 7-20　Bob 从他的办公室控制他的网络化的用具

在这个例子中，SIP 提供的一个主要服务就是安全。Bob 发送的每个命令在它执行前都被严格鉴别。Bob 明确地不想任何其他人使用他住所的照相机或打开他的房门锁。SIP 安全结构提供了控制网络化用具的机制，同时保证了用户隐私的安全。

本章提供了一些使用 SIP 作为信令协议的体系结构的例子。现在读者清楚了怎样结合使用不同的 SIP 扩展，甚至不同的协议来为真实的用户提供现实生活中的服务。

附　录

更多的 SIP 信息

此附录适用于已读完此书并想更多地了解 SIP 的读者。寻找 SIP 信息和任何其他 IETF 协议相关信息的最好方法就是定向浏览 Internet。但是，如果对 IETF 协议不熟悉，你将花费过多的时间才能找到一条特定的信息。有时候寻找一个关于协议的好的网页很困难。例如，仅仅靠一个单词“SIP”，一般的搜索引擎会返回很多相关信息。这个附录包含了与 SIP 相关的最实用的网页，这些网页能帮助读者更快地找到需要的信息。我们也给出了一些关于 SIP 的很有趣的邮件列表。如前所述，邮件列表是 IETF 协议开发的最重要的工具之一。

在附录的末尾，给出了一个 RFC 实例，它将帮助读者理解一个 IETF 规范的具体面貌。

IETF 网站

寻找任何特定网络协议的最佳地点当然是 IETF 的网站，如图 A-1 所示。

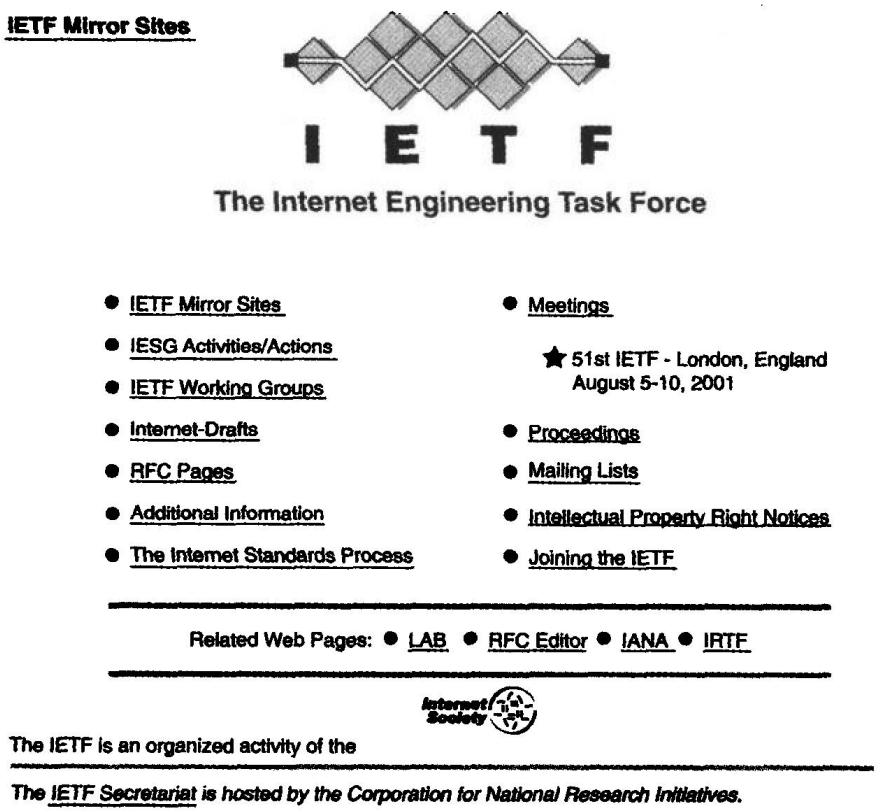


图 A-1　http：//www.ietf.org

IETF 网页（http://www.ietf.org）包括组织本身的一些信息以及过去和将来的一些会话信息，然而，我们更希望找到具体的技术规范。在整本书中，我们已给出了一些有关 Internet 草案以及 RFC 的参考资源，在 IETF 网站可以很容易地找到它们。在“Internet-Drafts”链接下，可以根据作者的姓名或其标题寻找一份特定的草案。也有一个按照草案所属的工作组编排的所有 Internet 草案的索引。

如果要找某个 RFC，则要在“RFC Pages”链接下寻找，在那里可以找到某个 RFC 号码或者浏览 RFC 索引，索引中包含了 IETF 发表的所有 RFC。

习惯于在其它标准组织的网页上查找信息的读者很快就会发现，IETF 不需要任何口令或成员身份就可以下载任何内容，任何人都可以。在 IETF 的网页上查找文件要比在其他组织的网页查找简单得多，那些组织很难记录一个文件的不同版本。

读者还会发现定向浏览“IETF Working Groups”链接非常有用，它可以连到不同的工作组网页。读者尤其会感到 SIP、SIMPL 和 MMUSIC 等小组的章程特别有趣。还有一个读者可能感兴趣的工作组是 IPTEL。

一个工作组的网页还包含如何加入该工作组邮件列表的说明。邮件列表档案向读者提供了一些在 IETF 内部进行的技术讨论的实例。

但是，请注意，工作组的邮件列表是为开发协议的工程师准备的，对 SIP 不熟悉的读者最好不要到邮件列表中问一些有关协议的基本问题。在另一个名为“SIP implementors”邮件列表中提供了一般的信息以及现有的实现。还可以在 SIP 工作组的网页（http://www.ietf org/html.charters/sip-charter.html）中找到如何订阅这个邮件列表的信息。在下一小节，将会介绍到哪里寻找非常有用的常见问题集（FAQ，Frequently Asked Questions）。

Henning Schulzrinne 的 SIP 网页

我们知道，IETF 网站是下载协议规范和订阅邮件列表的最佳地点。然而，如果读者想得到一般的 SIP 信息，那么 Henning Schulzrinne 的网页（如图 A-2 所示）是好的选择之一（http://www.CS.Columbia.edu/sip）。

Schulzrinne 是 SIP 的共同发起者之一，他维护的这个网站毫无疑问是跟踪 SIP 发展的最好网页。“news”栏内包含与 SIP 相关的、诸如新推出的一个 RFC 等重大事件的信息，以及到下一个 SIP 互操作事件组织者的链接。Schulzrinne 的网站还有到各种使用 SIP 的组织以及各种电信会话发表文献的链接。值得花费一定时间浏览该网页并熟悉它。所需要的任何 SIP 信息都可以在这里找到。

FAQ 栏尤其让人感兴趣。任何有疑问或者考虑在一般信息邮件列表发贴子的 SIP 新用户应先看一下 FAQ 栏。在这里有各种最常见的问题及答案。到邮件列表提问题之前先看 FAQ 不仅仅可节省带宽，还可以节省那些为了帮助其他人熟悉此协议而订阅邮件列表的工程师们的时间，他们非常希望不要反复地回答一些相同的问题。

Session Initiation Protocol (SIP)

SIP, the Session Initiation Protocol, is a signaling protocol for Internet conferencing, telepony, presence, events notification and instant messaging. SIP was developed within the IETF MMUSIC (Multiparty Multimedia Session Control) working group, with work proceeding since September 1999 in the IETF SIP working group.

A number of standardization organizations and groups are using or considering SIP:

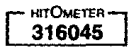
- IETF PINT working group
- 3GPP for third-generation wireless networks
- Softswitch Consortium
- IMTC and ETSI Tiphon are working onSIP-H.323 interworking
- PacketCable DCS (distributed call signaling) specification
- 3GPP (for third-generation wireless)
- SpeechLinks, for moving between speech-enabled sites

News

- May 29, 2001: RFC2543bis (-03) draft
- May 4, 2001: Information about the 8th SIP Interoperability Test Event is now available
- April 14, 2001: search feature added.
- April 11, 2001: The SIP interoperability test event has a new logo, courtesy of Ubiquity.

- April 10, 2001: RFC 3087 (*Control of Service Context using SIP Request-URI*) published
- Feb. 1, 2001: RFC 3050 (*Common Gateway Interface for SIP*) published
- Nov. 30, 2000: Caller preferences draft in WG last call until December 24, 2000
- Nov. 29, 2000: Guidelines for Authors of SIP Extensions draft in WG last call until December 24, 2000
- November 24, 2000: RFC2543bis (-02) draft
- Nov. 17, 2000: CPL in IESG last call.
- RFC 2976 (*The SIP INFO Method*) published
- The sixth SIP interoperability test event took place December 5-8, 2000 at Sylantro and Sun in Silicon Valley, California.
- June 20, 2000: The SIP Forum was founded. "SIP Forun is a non profit association whose mission is to promote awareness and provide information about the benefits and capabilities that are enabled by SIP."
- The fifth SIP interoperability test event took place August 8-10, 2000 at pulver.com in Melville, Long Island.
- June 15, 2000: RFC 2848, *The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services*, published.
- Added SIP internship and job listing.
- The fourth SIP interoperability test event took place April 17-19, 2000 in Rolling Meadows (near Chicago), Illinois, hosted by 3Com.
- February 28, 2000): Draft *The SIP INFO Method* is in IETF last call for Proposed Standard.
- September 1999: A new IETF working group on SIP has been created.
- The third SIP interoperability test event took place December 6th throught 8th, 1999 in Richardson, Texa, hosted by Ericsson.
- The second SIP interoperability test event took place August 5th and 6th, 1999 at pulver.com (Melville, NY).
- The first SIP interoperability test event (known as "bake off") took place April 8th and 9th, 1999 at Columbia University, New York.
- SIP is a Proposed Standard (Feb. 2, 1999) published as RFC 2543 (March 17, 1999).
- New list of public SIP servers.

[Overview] [Where is SIP being discussed?] [Search] [What SIP extensions are being planned?] [Drafts] [SIP grammar] [Implementations] [SIP service providers] [mailing list] [Public SIP Servers] [Papers] [Talks] [Related Drafts and Documents] [Draft History] [Frequently Asked Questions (FAQ)] [Port Assignments and DNS] [Compact headers] [SIP-related events] [SIP interoperability testing events] [Press Coverage] [Emergency Calling] [Non-Internet-related SIP sightings] [Status and Schedule] [Internships and Jobs] [Other SIP sites]



Last updated Friday, June 15, 2001 21:30:07 by Hennig Schulzrinne

图 A-2 http://www.CS.Columbia.edu/sip

Dean Willis 网页

在 IETF 的 SIP 工作小组网页上可以找到 Dean Willis 网页的链接。这个附加的 SIP 网页（http://www.softarmor.com/sipwg）由 Dean Willis 维护，他是 SIP 工作组的主席之一。这个网页（如图 A-3 所示）处理 SIP 工作组的管理问题，包含有关开发小组、解决特定问题的工作小组以及最终调用日程等信息。

Session Initiation Protocol (SIP) Working Group Supplemental Home Page

[Drafts] [Morgue] [Last Call Info] [Meetings] [Design teams] [Issues] Process] [References] [SIPPING]

Important Notes

14-Jun-01: the Last Call Calendar has been updated.

1-Jun-01: This site has been heavily modified.

1-Jun-01: See the SIPPING Site here.

General Notes

Volunteer to conduct nits and last call reviews by contacting Rakesh Shah (rshah@dynamicsoft.com).

Menu

- Drafts – Text and status information.
- Morgue – Archive of expired drafts.
- Last Call Info – Schedule, reviewers and reviewers references.
- Meetings – IETF, Interim Meetings, etc.
- Design Teams – Teams working on specific issues.
- Issues – Stuff we need to take care of.
- Processes – Ok, everybody has to have a few rules.
- References – Some useful information that we don't have.
- SIPPING – SIPPING Working Group Supplemental Home Page.

Brought to you as a personal effort (that's no corporate sponsorship implied, ok?) of Dean Willis, who can be reached at or dean.willis@softarmor.com and serveral other unlikely places. This is a personal server. Please don't thrash it. Have a nice day!

Updated June 06, 2001 14:08 UST

图 A-3 http://www.softarmor.com/sipwg

最终调用日程非常重要，通过它可以获知一份 Internet 草案何时将成为一个 RFC。这样，可以很好地检验各种不同扩展的成熟程度。当一份草案进入最终调用日程，在它被送给 IESG 之前，工作组有最后的机会对其进行评审。一旦草案被呈递到 IESG，该领域的主席们将决定它是否可以成为 RFC 或者工作组是否需要继续对其进行研究。

Dean Willis 还维护另外一个网页（如图 A-4 所示），网址是http://www.softarmer.

om/sipping。该网页包括使框架具体化的工作和新的以 SIP 为基础的应用要求。所有相关 Internet 草案的情况也都有最终调用日程。

Session Initiation Protocol INvestiGation (SIPPING) Working Group
Supplemental Home Page

[Drafts] [Morgue] [Last Call Info] [Meetings] [Design Teams] [Issues] [Process] [References] [SIP]

Important Notes

14-Jun-01: The Last Call Calendar has been updated.

1-Jun-01: This site is all new.

1-Jun-01: See the SIP Site here

General Notes

Volunteer to conduct nits and last call reviews by contacting Rakesh Shah (rshah@dynamicsoft.com).

Menu

- Draft charter for SIPPING Working Group.
- Drafts -- Text and status information.
- Morgue -- Archive of expired drafts.
- Last Call Info -- Schedule, reviewers and reviewers references.
- Meetings -- IETF, Interim Meetings, etc.
- Design Teams -- Teams working on specific issues.
- Issues -- Stuff we need to take care of.
- Processes -- Ok, everybody has to have a few rules.
- References -- Some useful information that we don't have.
- SIP WG -- SIP Working Group Supplemental Home Page.

Brought to you as a personal effort (that's no corporate sponsorship implied, ok?) of Dean Willis, who can be reached at or dean.willis@softarmor.com and several other unlikely places. This is a personal server. Please don't thrash it. Have a nice day!

Updated June 06, 2001 14:08 UST

图 A-4 http://www.softarmer.com/sipping

SIP 论坛

尽管 SIP 论坛（如图 A-5 所示）不是一个技术性组织，但它还是值得一提的。SIP 论坛（http://www.sipforum.org）的宗旨是传播 SIP 相关的信息。一般是通过论文、会话和邮件列表完成。值得注意的是，SIP 论坛不产生任何技术规范，它只传播 SIP 信息。

RFC 实例

本节给出一个 RFC 实例，它是专为那些不喜欢到 Internet 上浏览却想知道 RFC 是什么内容的读者准备的。鉴于此部分的目的只是展示一个 IETF 规范的实例，我们选择了目前发布

的与 SIP 相关的篇幅最短（在页数方面）的 RFC：SIP 信息方法[RFC 2976]。

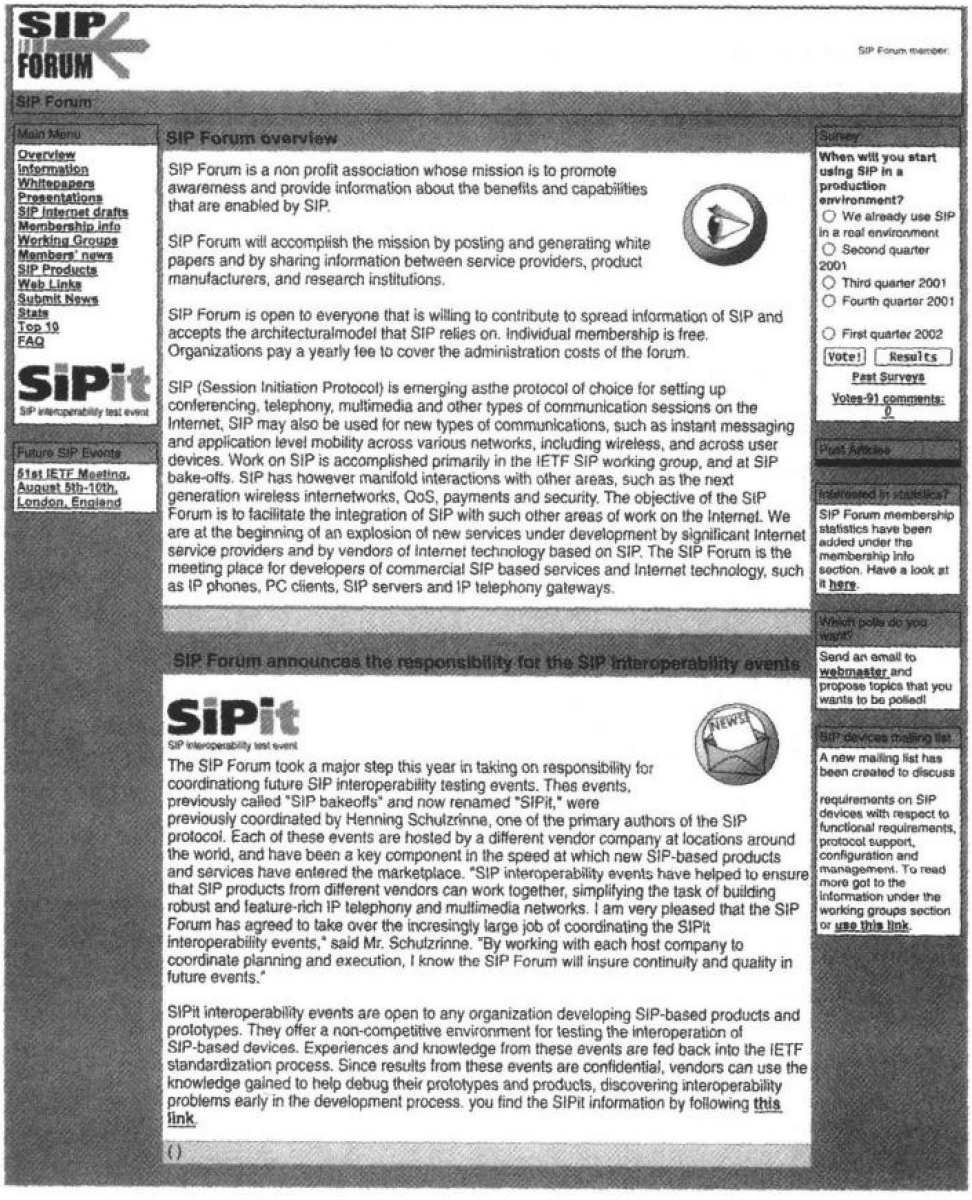


图 A-5 http://www.sipforum.org

读者看到此 RFC 最后一页详尽的版权声明时，可能会感到非常有趣。每一个 RFC 末尾都有这样的声明。RFC 的另一个有趣的现象是：它们都是按文本形式书写的，因此，如果 RFC 有插图，这些图也是用文本字符绘制的。

该 RFC 可从 http://www.ietf.org/rfc/rfc 2976.txt 下载。

RFC

Network Working Group
Request for Comments: 2976
Category: Standards Track

S. Donovan
dynamicsoft
October 2000

SIP 信息方法
（RFC 2976　The SIP INFO Method）

本备忘录的状态

本文档讲述了一种 Internet 社团的 Internet 标准跟踪协议，它还需要进行讨论和听取各方面的建议以进一步得到改进。请参考最新版的“Internet 正式协议标准”（STD1）获得本协议的标准化程度和状态。本备忘录的发布不受任何限制。

版权声明

摘要

本文档提出了会话初始化协议（SIP）的一种扩展机制。这一扩展为 SIP 增加了 INFO 方法。INFO 方法的目的是允许传送会话相关的控制信息，这些控制信息是在会话中生成的。这种会话控制信息的一个例子是用于控制电话业务的 ISUP 和 ISDN 信令消息。

其他的关于 INFO 方法的例子将在随后进行标准化。

目录

1. 简介

参考资料[1]中描述的 SIP 协议定义了用在 SIP 控制的会话建立和拆除阶段的会话控制信息。

另外，SIP re-INVITE 能够用于在一个会话的过程中改变其特性。它通常是与会话相关的媒体流量属性或者更新 SIP 会话计时器。

然而，没有通用的目标机制在会话过程中沿着 SIP 信令通路承载会话控制信息。

INFO 消息的作用就是沿着 SIP 信令通路携带应用层消息。

INFO 方法并不是用来改变 SIP 呼叫的状态，或会话 SIP 的初始化状态参数，它仅用于发送通常与会话有关的应用层的可选信息。会话中的信令信息必须穿过会话后的 SIP 信令通路来建立。这个通路是 SIP 的 re-INVITEs、BYE 和其他与一个独立会话相联系的 SIP 请求所采用的通路。它允许 SIP 代理服务器接收并潜在地对会话中的信令信息产生影响。

此文档通过定义新的 INFO 方法提供 SIP 的一个扩展机制。INFO 方法将被用于沿着会话信令通路传送呼叫中信号信息。

1.1 用法举例

以下是一些 INFO 消息的可能应用：

— 在 PSTN 网关之间传送呼叫中的 PSTN 信令消息；
— 传送 SIP 会话中生成的 DTMF 数字；
— 传送无线信号强度信息以支持无线移动应用；
— 传送计算平衡信息；
— 在会话的参加者之间传送影像或其它的非流信息。

这些只是可能的应用；本文档并不指定这些应用，也不一定必须介绍它们。

也可以想象 INFO 方法还有其他的电话和非电话方面的应用。

2. INFO 方法

INFO 方法用于沿着呼叫的信令通路进行会话中信令消息间的通信。

INFO 方法并不是用于改变 SIP 呼叫的状态，也不是用于改变 SIP 初始化的会话状态。然而，它所提供的增加的选项信息可以进一步加强 SIP 应用程序的功能。

INFO 方法的信令通路是呼叫建立之后建立的信令通路。这可以是呼叫方和被呼叫方用户代理之间的直接信令，也可以是牵涉到呼叫建立和自己增加到初始 INVITE 信息记录路由头部的 SIP 代理服务器的信令通路。

会话中的信息能够在 INFO 信息头部或作为一个消息体的一部分进行通信。消息体和/或消息头部被定义用来传送会话中的信息这部分内容不属于本文档讨论的范围。

没有与 INFO 相关的特别语法语义。语义是从定义给 INFO 用的头部或协议体那里继承来的。

2.1 INFO 方法的头域支持

表 1 和表 2 是在参考资料[1]中的表 3 和表 4 的基础上增加了一列。请参考该参考资料中的第 6 节对表中内容的描述。 请注意在附录里定义的规则和参考资料[1]中表 3 和表 4 的 e-e 列同样引用了在 INFO 请求和回应 INFO 请求中的头部的应用。

2.2 INFO 请求方法的响应

如果服务器收到一个 INFO 请求，它必须发出一个最后的回应。

如果 INFO 请求被现有呼叫成功地接收到，UAS 必须发送一个没有消息体的 200 OK 回应给该 INFO 请求。除此之外，不需要其他操作。

表 1　　头域的概括，A-O

头部 Header	地方 Where	信息 INFO
------	-----	----
接收 Accept	R	o
接收－编码 Accept-Encoding	R	o
接收－语言 Accept-Language	R	o
允许 Allow	200	-
允许 Allow	405	o
认可 Authorization	R	o
呼叫号 Call-ID	gc	m
连接 Contact	R	o
连接 Contact	1xx	-
连接 Contact	2xx	-
连接 Contact	3xx	-
连接 Contact	485	-
内容－编码 Content-Encoding	e	o
内容－长度 Content-Length	e	o
内容－类型 Content-Type	e	*

续表

头部 Header	地方 Where	信息 INFO
CSeq	gc	m
数据 Date	g	o
加密 Encryption	g	o
期满 Expires	g	o
From	gc	m
隐藏 Hide	R	o
最大一向前流 Max-Forwards	R	o
组织 Organization	g	o

包括消息体的 INFO 消息不在本文档的讨论范围之内。本文档消息体的定义将同样需要引用 SIP 中的那些消息体的定义。

如果 INFO 请求与任何现存的呼叫 leg 不匹配，那么必须在一个 UAS 中发送 481 呼叫 Leg/Transaction 不存在消息。

如果服务器收到一个它能理解消息体的 INFO 请求，但是它又对与 INFO 过程有关的消息体规则没有一点了解，那么这个消息体可能被翻译并显示给用户。这个 INFO 被一个 200 OK 回应。

如果 INFO 请求包括服务器当时不能理解的消息体，则在缺乏 INFO 相关消息体的进程规则缺乏时，服务器必须回应一个 415 不支持的媒体类型消息（Unsupported Media Type）。

表 2　　头域的概括，P-Z

头　部	地　方	信　息
------	-----	----
优先权	R	o
Proxy 验证	407	o
Proxy 验证	R	o
Proxy-需求	R	o
请求	R	o
重试-之后	R	-
重试-之后	404，480，486	o
重试-之后	503	o
重试-之后	600，603	o
回应-关键字	R	o
记录-路由	R	o
记录-路由	2xx	o
路由	R	o

续表

头　部	地　方	信　息
服务器	r	o
主体	R	o
时间戳	g	o
To 到	gc（1）	m
不支持的	420	o
用户代理	g	o
Via	gc（2）	m
告警	r	o
WWW-验证	401	o

那些在SIP呼叫状态中或被SIP初始化后的会话中有作了改变的消息体不能被放在INFO消息中发送。

其他请求失败（4xx）、服务器失败（5xx）和全局失败 （6xx）回应将被送给 INFO 请求。

2.3 消息体的内容

INFO 请求可能包含一个消息体。

2.4 SIP 用户代理的行为

除非已经进行过申明，INFO 请求的协议规则控制着标记（Tags）的用法。路由和记录—路由（Route and Record-Route）的重传及可靠性、CSeq 自增以及消息的格式都遵从参考资料[1]中类似于 BYE 请求的定义。

一个 INFO 请求可能被取消。如果一个最终的回应没有送给 INFO 并且其行为与该请求从未被接收类似，那么，UAS 接收一个给 INFO 请求的取消（CANCEL）将用“487 请求已取消”回应给 INFO。

然而， INFO 消息决不许改变 SIP 呼叫的状态或 SIP 初始化的会话。

2.5 SIP 代理和重定向服务器的行为

2.5.1 代理服务器

除非已经进行过申明，在一个代理服务器上的 INFO 请求协议规定与那些在参考资料[1]中说明的 BYE 请求协议规定相一致。

2.5.2 分支代理服务器

除非已经进行过申明，在一个代理服务器上的 INFO 请求协议规定与那些在参考资料[1]中说明的 BYE 请求协议规定相一致。

2.5.3 重定向服务器

除非已经进行过申明，在一个代理服务器上的 INFO 请求协议规定与那些在参考资料[1]中说明的 BYE 请求协议规定相一致。

3. INFO 消息体

INFO 消息的作用是在 SIP 用户代理间传送会话中的信息。尽管这种信息能够通过 INFO 信息头传送，但是它一般被放在消息体中传送。

对于消息体的定义或其他任何为 INFO 方法产生的新头部不属于本文档讨论的范围。期望能够产生以定义这些实体为目标的独立文档。

另外，INFO 方法并不定义确保按顺序传送的附加机制。当 CSeq 头部在传送新的 INFO 消息时自增，就不能被用来决定 INFO 信息的顺序。这是因为，用户代理发送有 re-INVITES 或其他的 SIP 消息时，在 INFO 消息的 CSeq 计数中将会引起差异。

4．利用 INFO 扩展的指导方针

以下是定义利用 INFO 方法的 SIP 扩展时必须考虑的几方面问题：

— 必须考虑被 INFO 消息传送的消息体的大小。由于消息将可能在 UDP 上传送并且可能重组一个大的消息，消息体应该保持为较小的状态；

— 有一种潜在的可能是 INFO 消息被 SIP 代理服务器创建。需要考虑完成这一 INFO 消息中的信息的创建过程；

— 当定义被 INFO 消息传送的消息体时，多部分（Multi-part）消息体的应用将会很有用；

— 用 INFO 消息的扩展不允许依靠 INFO 消息产生那些影响 SIP 呼叫状态或相关会话状态的行为；

— 本文档定义的 INFO 扩展机制不依赖于请求或代理请求头部的应用。用 INFO 消息的扩展名可能需要应用这些机制。然而，如果有可能，应避免使用请求或代理请求，以便在 SIP 实体间实现互操作。

5．安全性考虑

如果消息体的内容是私有的，那么对端到端的消息体进行加密能够阻止未授权的访问。

INFO 方法没有其他特定的安全问题。在 SIP 说明中所列出的安全考虑同样适用于 INFO 方法。

6．参考资料

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

7．致谢

作者需要感谢 Matthew Cannon 对这一文档作出的贡献。另外，作者想感谢 MMUSIC 和 SIP 工作组的成员，特别是 Jonathan Rosenberg，他对本文档进行过评论并提出修改的建议以提高本文档的质量。

8．作者联系方法

Steve Donovan

Dynamicsoft

5100 Tennyson Parkway， Suite 200

Plano， Texas 75024
Email: sdonovan@dynamicsoft.com

9．完全版权声明

致谢

目前，RFC 编者的活动基金由 Internet 社团提供。

缩 写 词

2G（Second generation of mobile systems）：第二代移动通信系统

3G（Third generation of mobile systems）：第三代移动通信系统

3GPP（Third Generation Partnership Project）：第三代移动通信伙伴项目

3GPP2（Third Generation Partnership Project 2）：第三代移动通信伙伴项目 2

AC（Alternating Current）：交流电

ANSI（American National Standards Institute）：美国国家标准协会

APEX（Application Exchange）：应用交换

ARIB（Association of Radio Industries and Businesses）：日本无线工业和商业协会

ASCII（American Standard Code for Information Interchange）：美国信息交换标准代码

ASN-1（Abstract Syntax Notation 1）：抽象语法符号 1

ATM（Asynchronous Transfer Mode）：异步传输模式

AVP（Audio/Video Profile）：音频/视频轮廓

BCP（Best Current Practice）：最优当前实现

BOF（Birds of Feathers）：同行会议，尤指计算机业内从事相同科技领域，但在不同公司内任职者所举行的会议，藉此交换信息和经验，常缩写为 BOF meeting

BTS（Base Transceiver Station）：基站收发机

CAS（Channel Associated Signalling）：信道相关信令

CCP（Connection Control Protocol）：连接控制协议

CMS（Call Management Server）：呼叫管理服务器

CMSS（Call Management Server Signalling）：呼叫管理服务器信令

CSCF（Call/Session Control Function）：呼叫/会话控制功能

CSS（Common Channel Signalling）：公共信道信令

CWTS（China Wireless Telecommunications Standard）：中国无线通信标准（研究组）

DC（Direct current）：直流电

DHCP（Dynamic Host Configuration Protocol）：动态主机配置协议

DiffServ（Differentiated Services）：区分服务

DMP（Device Messaging Protocol）：设备消息协议

DSS-1（Digital Subscriber Line No. 1 ）：1 号数字用户线

DTMF（Dual Tone Multi-Frequency）：双音多频音

EIA（Electronic Industries Alliance）：电子工业联盟

ETSI（European Telecommunication Standard Institute）：欧洲电信标准协会

FDM（Frequency Division Multiplexing）：频分多路复用

GSM（Global System for Mobile communications）：全球移动通信系统

HSS（Home Subscriber Server）：家庭预定服务器

HTTP（Hypertext Transfer Protocol）：超文本传输协议

IAB（Internet Architecture Board ）：Internet 活动委员会

ICB（Internet Cooperation Board ）：Internet 协作委员会

ICCB（Internet Configuration Control Board）：Internet 配置控制委员会

I-CSCF（Interrogating-Call/Session Control Function）：询问-呼叫/会话控制功能

ID（Identification）：身份标识

I-D（Internet Draft）：Internet 草案

IETF（Internet Engineering Task Force）：Internet 工程任务组

IMAP（Internet Message Access Protocol）：Internet 消息访问协议

IP（Internet Protocol ）：Internet 协议

IPSec（Internet Protocol Security ）：Internet 协议安全

IPTEL（IP Telephony ）：IP 电话

IMPP（Instant Messaging and Presence Protocol）：即时消息和存在协议

IMT-2000（International Mobile Telecommunications 2000）：国际移动通信-2000

IN（Intelligent Network）：智能网

INAP（Intelligent Network Application Protocol）：智能网络应用协议

INRIA（Institut National de Recherche en Informatiqueet en Automatique）：法国国家计算机科学与控制学会

IRG（Internet Research Group）：Internet 研究小组

ISDN（Integrated Services Digital Network）：综合业务数字网

ISOC（Internet Society）：Internet 协会

ISUP（ISDN User Part）：ISDN 用户部分

ITU（International Telecommunication Union）：国际电信联盟

ITU-T（International Telecommunication Union Telecommunication Standardization Sector）：国际电信联盟电信标准化组织

IVS（INRIA Videoconferencing System）：INRIA 视频会议系统

Kbps（Kilobits per second）：千比特每秒（kbit/s）

LAN（Local Area Network）：局域网

LDAP（Lightweight Directory Access Protocol）：轻量目录访问协议

MCU（Multipoint Control Unit）：多点控制单元

MIME（Multipurpose Internet Mail Extensions）：多用途 Internet 邮件扩展

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
MMCC（Multimedia Conference Control）：多媒体会议控制
MMUSIC（Multiparty Multimedia Session Control）：多方多媒体会话控制
MSC（Mobile Switching Center）：移动交换中心
MTA（Multimedia Terminal Adapter）：多媒体终端适配器
NCP（Network Control Protocol）：网络控制协议
NCS（Network Call Signalling）：网络呼叫信令
PBX（Private Branch Exchange）：专用小交换机
PCM（Pulse Code Modulation）：脉冲编码调制
PDF（Portable Document Format）：可移植文档格式
P-CSCF（Proxy-Call/Session Control Function）：代理-呼叫/会话控制功能
PHB（Per Hop Behavior）：逐跳行为
PINT（PSTN and Internet Interworking）：公用电话交换网和 Internet 互通
PRIM（Presence and Instant Messaging）：存在和即时消息
PSTN（Public Switched Telephone Network）：公用交换电话网络
PUA（Presence User Agent）：存在用户代理
QoS（Quality of Service）：服务质量
RFC（Request for Comments）：请求评论
RFI（Request for Information）：请求信息
RFP（Request for Products）：请求产品
RSVP（ReSerVation Protocol）：资源预留协议
RTCP（Real-time Transport Control Protocol）：实时传输控制协议
RTP（Real-time Transport Protocol）：实时传输协议
RTSP（Real-time Streaming Protocol）：实时流协议
SAP（Session Announcement Protocol）：会话通告协议
SCIP（Simple Conference Invitation Protocol）：简单会议邀请协议
SCP（Service Control Point）：服务控制点
S-CSCF（Serving-Call/Session Control Function）：服务-呼叫/会话控制功能
SCTP（Stream Control Transmission Protocol）：流控制传输协议
SDP（Session Description Protocol）：会话描述协议
SDPng（SDP next generation）：下一代会话描述协议
SG（Signalling Gateway）：信令网关
SIMPLE（SIP for Instant Messaging and Presence Leveraging Extensions）：用于即时消息

和存在杠杆的 SIP 扩展

SIP（Session Initiation Protocol）：会话初始化协议

SLP（Service Location Protocol）：服务定位协议

S/MIME（Secure/Multipurpose Internet Mail Extensions）：安全/多用途 Internet 邮件扩展

SMTP（Simple Mail Transport Protocol）：简单邮件传输协议

SS6（Signalling System no. 6）：6 号信令系统

SS7（Signalling System no. 7）：7 号信令系统

SSP（Service Switching Point）：服务交换点

STD（Standard）：标准

TCP（Transmission Control Protocol）：传输控制协议

TDM（Time Division Multiplexing）：时分多路复用

TIA（Telecommunications Industry Association）：（美国）电信工业协会

TLS（Transport Layer Security）：传输层安全

TN（Telephone Network）：电话网络

TTA（Telecommunications Technology Association）：（韩国）电信科技协会

TTC（Telecommunications Technology Committee）：（日本）电信技术委员会

TUP（Telephone User Part）：电视用户部分

TV（Television）：电视

UA（User Agent）：用户代理

UAC（User Agent Client）：用户代理客户端

UAS（User Agent Server）：用户代理服务器

UDP（User Datagram Protocol）：用户数据报协议

URI（Universal Resource Identifier）：统一资源标识符

URL（Uniform Resource Locator）：统一资源定位器

US（United States）：美国

VCR（Video Cassette Recorder）：录像机

VoIP（Voice over IP）：IP 电话

参 考 文 献

[draft-ietf-bgmp-spec] D. Thaler, D. Estrin, D. Meyer. "Border Gateway Multicart Protocol (BGMP)," IETF.Work in progress

[draft-ietf-impp-cpim] D. Crocker, A. Diacakis, F. Mazzoldi, C. Huitema, G. Klyne,M. Rose, J. Rosenberg, R. Sparks, H. Sugano. "A Common Profile for Instant Messaging (CPIM)," IETF.Work in progress

[draft-ietf-mmusic-confarch] M. Handley, J. Crowcroft, C. Bormann, J. Ott. "The Internet Multimedia Conferencing Architecture," IETF.Work in progress

[draft-ietf-mmusic-sdpng] D. Kutscher, J. Ott, C. Bormann. "Session Description and Capability Negotiation," IETF.Work in progress

[draft-ietf-sip-100rel] J. Rosenberg, H. Schulzrinne. "Reliability of Provisional Responses in SIP," IETF.Work in progress

[draft-ietf-sip-callerprefs] H. Schulzrinne, J. Rosenberg. "SIP Caller Preferences and Callee Capabilities," IETF.Work in progress

[draft-ietf-sip-cc-transfer] R. Sparks. "SIP Call Control—Transfer," IETF. Work in progress

[draft-ietf-sip-dhcp] G. Nair, H. Schulzrinne. "DHCP Option for SIP Servers," IETF.Work in progress

[draft-ietf-sip-events] A. Roach, "Event Notification in SIP," IETF.Work in progress

[draft-ietf-sip-guidelines] J. Rosenberg, H. Schulzrinne. "Guidelines for Authors of SIP Extensions," IETF.Work in progress

[draft-ietf-sip-isup] G. Camarillo, A. Roach, J. Peterson, L. Ong. "ISUP to SIP Mapping," IETF.Work in progress

[draft-ietf-sip-isup-mime] E. Zimmerer, J. Peterson, A. Vemuri, L. Ong,M. Watson, M. Zonoun. "MIME media types for ISUP and QSIG Objects," IETF.Work in progress

[draft-ietf-sip-manyfolks-resource] W. Marshall, K. Ramakrishnan, E. Miller, G. Russell, B. Beser,M.Mannette, K. Steinbrenner, D. Oran, F.Andreasen, M. Ramalho, J. Pickens, P. Lalwaney, J. Fellows, D. Evans, K. Kelly, A. Roach, J. Rosenberg, D.Willis, S. Donovan, H. Schulzrinne. "Integration of Resource Management and SIP. SIP Extensions for Resource Management," IETF.Work in progress

[draft-ietf-sip-rfc2543bis] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. "SIP: Session Initiation Protocol," IETF.Work in progress

[draft-ietf-sip-serverfeatures] J. Rosenberg, H. Schulzrinne. "The SIP Supported Header," IETF.Work in progress

[draft-kempf-sip-findsrv] J. Kempf, J. Rosenberg. "Finding a SIP Server with SLP," IETF.Work in progress

[draft-kutscher-mmusic-sdpng-req] D. Kutscher, J. Ott, C. Bormann. "Requirements for Session Description and Capability Negotiation," IETF.Work in progress

[draft-moyer-sip-appliances-framework] S. Moyer, D. Marples, S. Tsang, J. Katz, P. Gurung, T. Cheng, A. Dutta, H. Schulzrinne, A. Roychowdhury. "Framework Draft for Networked Appliances Using the Session Initiation Protocol," IETF.Work in progress

[draft-rosenberg-impp-im] J. Rosenberg, D.Willis, R. Sparks, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, D. Gurle, D. Oran. "SIP Extensions for Instant Messaging," IETF.Work in progress

[draft-rosenberg-sip-3pcc] J. Rosenberg, J. Peterson, H.Schulzrinne, G. Camarillo. "Third Party Call Control in SIP," IETF.Work in progress

[draft-rosenberg-sip-app-components] J. Rosenberg, P. Mataga, H. Schulzrinne. "An Application Server Component Architecture for SIP," IETF.Work in progress

[draft-rosenberg-sip-conferencing-models] J. Rosenberg, H. Schulzrinne. "Models for Multi-Party Conferencing in SIP," IETF.Work in progress

[draft-tsang-appliances-reqs] S. Tsang, S. Moyer, D. Marples, H. Schulzrinne, A. Roychowdhury. "Requirements for Networked Appliances:Wide-Area Access, Control, and Interworking," IETF.Work in progress

[RFC 768] J. Postel. "User Datagram Protocol," IETF. August 1980

[RFC 791] J. Postel. "INTERNET PROTOCOL," IETF. September 1981

[RFC 793] J. Postel. "Transmission Control Protocol," IETF. September 1981

[RFC 821] J. Postel. "Simple Mail Transfer Protocol," IETF. August 1982

[RFC 854] J. Postel, J.K. Reynolds. "Telnet Protocol Specification," IETF. May 1983

[RFC 1633] R. Braden, D. Clark, S. Shenker. "Integrated Services in the Internet Architecture: An Overview," IETF. June 1994

[RFC 1777] W. Yeong, T. Howes, S. Kille. "Lightweight Directory Access Protocol," IETF. March 1995

[RFC 1827] R. Atkinson. "IP Encapsulating Security Payload (ESP)," IETF. August 1995

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

[RFC 1958] B. Carpenter. "Architectural Principles of the Internet," IETF. June 1996

[RFC 2026] S. Bradner. "The Internet Standards Process —Revision 3," IETF. October 1996

[RFC 2045] N. Freed, N. Borenstein. "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," IETF. November 1996

[RFC 2060] M. Crispin. "Internet Message Access Protocol—Version 4rev1," IETF. December 1996

[RFC 2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. "Hypertext Transfer Protocol—HTTP/1.1," IETF. January 1997

[RFC 2131] R. Droms. "Dynamic Host Configuration Protocol," IETF. March 1997

[RFC 2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin. "Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification," IETF. September 1997

[RFC 2246] T. Dierks, C. Allen. "The TLS Protocol Version 1.0," IETF. January 1999

[RFC 2326] H. Schulzrinne, A. Rao, R. Lanphier. "Real Time Streaming Protocol (RTSP)," IETF. April 1998

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

[RFC 2475] S. Blake, D. Black, M. Carlson, E. Davies, Z.Wang,W.Weiss. "An Architecture for Differentiated Service," IETF. December 1998

[RFC 2543] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. "SIP: Session initiation protocol," IETF. March 1999

[RFC 2597] J. Heinanen, F. Baker,W.Weiss, J. Wroclawski. "Assured Forwarding PHB Group," IETF. June 1999

[RFC 2598] V. Jacobson, K. Nichols, K. Poduri. "An Expedited Forwarding PHB," IETF. June 1999

[RFC 2608] E. Guttman, C. Perkins, J. Veizades. "Service Location Protocol, Version 2," IETF. June 1999

[RFC 2633] B. Ramsdell, "S/MIME Version 3 Message Specification," IETF. June 1999

[RFC 2705] M. Arango, A. Dugan, I. Elliott, C. Huitema, S. Pickett. "Media Gateway Control Protocol (MGCP) Version 1.0," IETF. October 1999

[RFC 2779] M. Day, S. Aggarwal, G. Mohr, J. Vincent. "Instant Messaging/Presence Protocol Requirements," IETF. February 2000

[RFC 2848] S. Petrack, L. Conroy. "The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services," IETF. June 2000

[RFC 2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. "Stream Control Transmission Protocol," IETF. October 2000

[RFC 2974] M. Handley, C. Perkins, E. Whelan. "Session Announcement Protocol," IETF.

October 2000

[RFC 2976] S. Donovan. "The SIP INFO Method," IETF. October 2000

作 者 简 介

Gonzalo Camarillo 是位于芬兰首都赫尔辛基的爱立信公司高级信令研究实验室（Advanced Signaling Research Lab）的首席系统专家。从 IETF 的 SIP 工作组一建立，他就是该工作组的积极参与者。他与他人共同起草了几项关于 SIP 的文件，他也是 VoIP 会议的经常的发言人以及 SIP 论坛中爱立信公司的代言人。他在 Universidad Politecnica de Madrid 获得了机电工程硕士学位，在斯德哥尔摩皇家技术学院获得了另一个机电工程硕士学位。现在，他正在芬兰的赫尔辛基科技大学攻读博士学位。

{
  "filename": "U0lQ5o+t5a+GXzExMDk2NjI1LnppcA==",
  "filename_decoded": "SIP\u63ed\u5bc6_11096625.zip",
  "filesize": 39780038,
  "md5": "5494e8e8c6b58515b875d186db65daad",
  "header_md5": "c5e1df5524cfdd3913a5c1c3221a7e08",
  "sha1": "ddad0ec8742bb815954fd17126f1109ef42d1526",
  "sha256": "be4e28ce963777b6b428f383fe39a360dc6d7f587542fdbc49b7d918c921e000",
  "crc32": 1783082418,
  "zip_password": "",
  "uncompressed_size": 45687128,
  "pdg_dir_name": "SIP\u255c\u2565\u251c\u2584_11096625",
  "pdg_main_pages_found": 184,
  "pdg_main_pages_max": 184,
  "total_pages": 201,
  "total_pixels": 1241947436,
  "pdf_generation_missing_pages": false
}