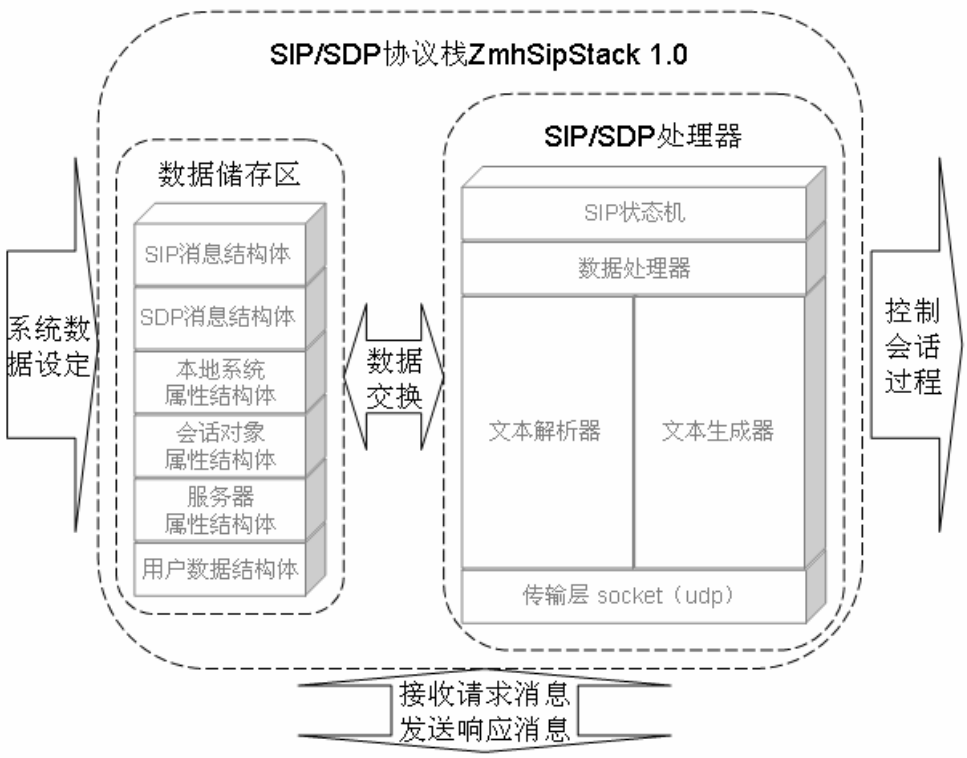
自己动手写 SIP 协议栈



Author：张梦晗
E_mail：enyaxp@student.dlut.edu.cn
2006 年 9 月 13 日

前言

本文由作者的本科毕设论文改写而成。由于时间仓促及作者水平所限，本文中的 SIP 协议栈尚不能实用，仅供初学 SIP 协议的朋友们入个门而已。不足之处还请多多包涵。

第一章　绪论

1.1 SIP 协议简介

SIP（Session Initiation Protocal）称为会话初始化协议，是由 IETF（Internet Engineering Task Force）组织于 1999 年提出的一个基于 IP 网络中实现实时通信应用的一种信令协议。作为一个由 IETF 提出的标准，SIP 协议很大程度上借鉴了其他各种广泛存在的 Internet 协议，采用基于文本的编码方式，这也是 SIP 协议同下一代网络视音频通信领域其他现有标准相比最大的特点之一。

和原有的多媒体会话协议（如 H.323）相比，SIP 具有以下优点：

- 可与 Voice XML、JSP、J2ME 等结合，快速开发增值业务；
- 支持多媒体应用，如语音、视频、图像、音频、文字、数据等业务；
- 可将语音、视频、Presence、短消息、Web 浏览、定位信息、Push、文件共享等业务结合起来，在语音、数据业务结合和互通方面有天然优势；
- 业务网络和承载网络分离，两者可独立发展；
- 业务网络可以融合现有的固网和移动网业务；
- 协议简单，具有公认的扩展潜力。

基于以上优势，SIP 协议被广泛应用于下一代网络的业务开发当中。而在现代软交换网络和未来移动网络中，SIP 协议更是将作为各个功能单元之间互通的基础协议，因此，有人也称 SIP 协议为下一代网络中的“TCP/IP”协议（图 1.1）。

1.2　论文各章内容

本论文的各章内容如下：

⑴　本文第二章介绍 SIP 协议的基本原理、格式和与其他辅

助协议的结合，然后介绍 SIP 系统的工作原理和基本会话过程；

(a) VoIP 软件　(b) 可视电话终端　(c) 多媒体网关　(d) 3G 手机

图 1.1 基于 SIP 协议的通信应用

⑵ 第三章介绍 SIP 系统的核心—SIP 协议栈的编写：首先进行方案论证，然后分析协议栈的工作原理和需要实现的机能，紧接着介绍自主开发的 SIP 协议栈 ZmhSipStack 的软件架构、结构体定义和各具体功能函数；

⑶ 第四章介绍了基于此协议栈编写的应用于 PC 机的用户代理程序 SipUA 和服务器程序 SipProxyServer，并对实现程序各功能的函数进行解释，最后展示程序运行效果；

⑷ 第五章中介绍了如何将 SIP 协议栈 ZmhSipStack 移植到嵌入式 ARM/μCOS-Ⅱ 环境下，并介绍了基于此协议栈编写的应用于嵌入式 ARM 平台的用户代理程序和服务器程序；

⑸ 第六章应用用户代理和服务器程序在 PC 机和 ARM 平台之间进行 SIP 协议会话实验，并给出实验结果。

第二章　SIP 协议基本介绍

本章主要介绍了SIP系统的基本组成和SIP协议的消息格式，并介绍了 SIP 系统的四种基本会话过程。

2.1 SIP 系统基本组成

SIP系统在RFC3261中有详细的定义。为了描述问题方便，RFC3261定义了几种逻辑功能实体，协议对每种实体的功能和行为都进行了详尽描述。实际应用中，一个物理实体可能集成了多个逻辑功能。

在RFC3261看来，SIP系统可按逻辑功能分为4种元素：User Agent（用户代理），Proxy（代理服务器），Redirect Sever（重定向服务器），Registrar（注册服务器）。

图 2.1描述了一个完备SIP系统的网络结构。从中我们可以发现用户代理、注册服务器、代理服务器都在同一个网域中，而重定向服务器可以在这个网域中，也可以在另一个网域中。SIP系统的会话过程请参看本章2.4节。

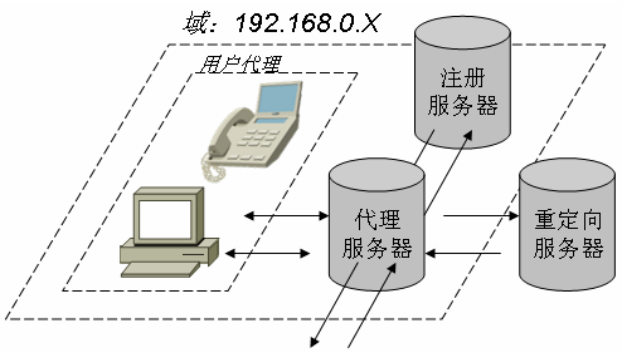


图 2.1 SIP 系统的网络结构

2.2 SIP消息描述

SIP协议是基于文本的通信信令协议。SIP消息以文本形式表示消息的语法、语义和编码，因此相对于二进制的信令，SIP消息

显得简单、易懂。

SIP消息有两种:客户端到服务器的请求消息和服务器到客户端的响应消息。SIP消息由一个起始行（start-line）、一个或多个字段(field)组成的消息头、一个标志消息头结束的空行(CRLF)以及作为可选项的消息体（message body）组成。其中描述消息体的头称为实体头（entity header），其格式如下：

generic-message = start-line
***message-header**
CRLF
[message-body]

下面分别对起始行、消息头及消息体一一进行解释。

2.2.1 起始行（start-line）

起始行分请求行（Request-Line）和状态行（Status-Line）两种，其中请求行是请求消息的起始行，状态行是响应消息的起始行。

1．请求行（Request-Line）

请求行已方法（method）标记开始，后面是统一定位标示符（URI）和协议版本号（SIP-Version），最后以回车换行符结束，各个元素间用空格键字符间隔。

Request-Line = Method SP **Request-URI** SP **SIP-Version** CRLF

方法标记“Method”来对说明部分进行描述。SIP协议在RFC3261中一共定义了六种方法。具体定义如下：

- INVITE：用于邀请用户或服务参加一个会话。INVITE消息中必须包含主叫方和被叫方的信息和双方交换的多媒体信息流类型。除了能够用用于启动双方通信会话外，还具有启动多方会议的能力；
- ACK：用于客户机向服务器证实它已经收到了对INVITE请求的最终响应。ACK消息中的主叫、被叫信息是由前

期媒体协商得来，可以包含消息体描述也可以不包含。ACK消息发送后，双方多媒体会话才真正开始；

- BYE：用于客户端向服务器表明它想释放呼叫。主叫方或被叫方都可以发送该消息。当会话参与者退出会话时，它必须向对方发送BYE消息，表示终止当前会话；
- OPTIONS：用于向服务器查询其能力。按照RFC3261的初衷，实体发起呼叫前可以发起该消息，以确认网络实体是否支持某种消息或某种能力，但从网络运行安全的角度来讲，不希望终端设备能够获知网络能力，因此在终端询问网络能力这一应用上，一般不允许发生。目前OPTIONS指令主要用于主叫方确认被叫方是否存在；
- CANCEL：用于取消正在进行的请求。CANCEL只能由主叫方发起，而且若已接收到最终响应状态200/OK，则该方法无效；
- REGISTER：用于用户向网络注册服务器发送注册消息。

随着应用的增多，仅仅有以上六种请求方法并不能够满足需求。因此IETF在RFC3261的基础上提出了许多扩展方法以满足应用需求。具体扩展定义请参考IEFT RFC2976。

2．状态行（Status-Line）

状态行以协议版本号（SIP-Version）开始，接下来是用数字表示的状态码（Status-Code），然后是相关文字说明（Reason-Phrase），最后以回车换行符结束，各个元素间用空格键字符间隔。

Status-Line = **SIP-Version** SP **Status-Code** SP **Reason-Phrase** CRLF

SIP协议中用三位整数的状态码（Status-Code）和文本形式的原因短语（Reason-Phrase）来表示对请求作出的回答。状态码用于机器识别操作，原因短语用于人工识别操作。

状态码的第一个数字定义响应的类别，在2.0版本中第一个数

字有六个值：

- 1xx（Provisional）：请求已经接收到，正在处理；
- 2xx（Success）：请求已经收到、理解，并接收；
- 3xx（Redirection）：为了完成请求，还需要进行下一步动作，用在重定向场合；
- 4xx（Client Error）：请求有语法错误或不能够被此服务器执行；
- 5xx（Server Error）：服务器不能够执行明显的有效请求；
- 6xx（Global Failure）：网络中所有的服务器不能够执行请求。

SIP协议将1xx响应定义为临时响应（Provisional Response），而其他五类则定义为最终响应（Final Response）。当服务器接收到临时响应消息时，表示对当前请求的处理并没有完结；当接收到最终响应消息时，表示对当前请求的处理已经完结。

2.2.2 消息头（message-header）

SIP协议的消息头是由多个消息头字段（header-field）构成的。消息头携带了SIP会话的一切必要信息，是SIP协议的主体。每一个头字段都遵循以下格式：首先是字段名（Field Name），后面是冒号，然后是字段值，不同的字段其所带字段值的格式及意义都不同。

Message-header = field-name ":" **[field-value]** CRLF

对于不同的请求消息或响应消息，有些消息头是必须的，有些消息头是可选的，而有些消息头是不允许出现的。关于请求消息、响应消息与消息头之间的映射关系请参考**附录。**

消息头根据使用的方法不同可以分为通用头（General-header）、实体头字段（Entity-header）、请求头（Request-header）以及响应头（Response-header）四类。这四种头在消息头域中混合使用，使得代理服务器或用户代理服务器能

够更好的对消息进行处理。表2.1中分类列出了所有消息头。

表 2.1 四类消息头

通用头	实体头	请求头	响应头
Accept	Content-Disposition	Authorization	Allow
Accept-Encoding	Content-Encoding	Hide	Proxy-Authenticate
Accept-Language	Content-Language	In-Reply-To	Retry-After
Call-ID	Content-Length	Max-Forwards	Server
Contact	Content-Type	Priority	Supported
CSeq		Proxy-Authorization	Warning
Date		Proxy-Require	WWW-Authenticate
Encryption		Route	
Expires		Require	
From		Response-Key	
Organization		Subject	
Record-Route			
Timestamp			
To			
User-Agent			
Via			
MIME-Version			

2.3 SIP基本会话过程

基本的SIP会话可分为直接会话、有服务器的间接会话和注册注销会话等。下面对各会话过程进行解释。

两个用户代理客户端可以不通过代理服务器进行直接通话，因为SIP网络强调了用户端的智能性，提高了通话的灵活程度，在一定意义上增强了通讯类型的多样化。直接会话的过程如下图2.2所示。

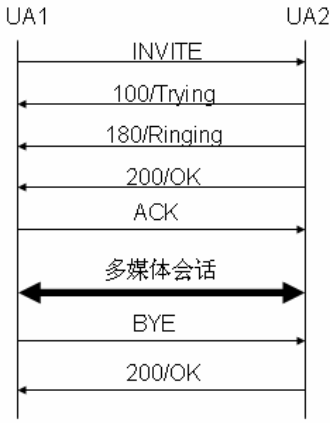


图 2.2 直接会话过程

直接会话开始后，由用户代理UA1向UA2发送携带本地媒体信息的INVITE请求并在接收响应前每隔一定时间重发该消息。UA2接收到后先返回一个100临时响应，告诉UA1正在进行会话处理，UA1接收后停止重发INVITE消息。UA2经过处理后返回180响应，表示处理完毕并等待用户接通。用户摘机后再返回最终响应200/OK并附带自身媒体信息。UA1接收200响应后发送ACK请求确认会话，然后两者可以直接进行多媒体通信。结束通话时由其中一方发送BYE请求，另一方接受后返回200/OK响应，完成会话的中断。

图 2.3 带服务器的多媒体间接通信

相对直接会话而言，间接SIP会话需要在两台用户代理之间加一台或多台服务器进行转接，两者的区别在于通过服务器的间接通信可以进行跨网段的多媒体会话。图2.3为间接会话的过程。

通信过程和直接通信类似，只是中间的服务器在接收到的消息体的Route头字段中加入了自己本身的信息。下图2.4为具体添加过程。

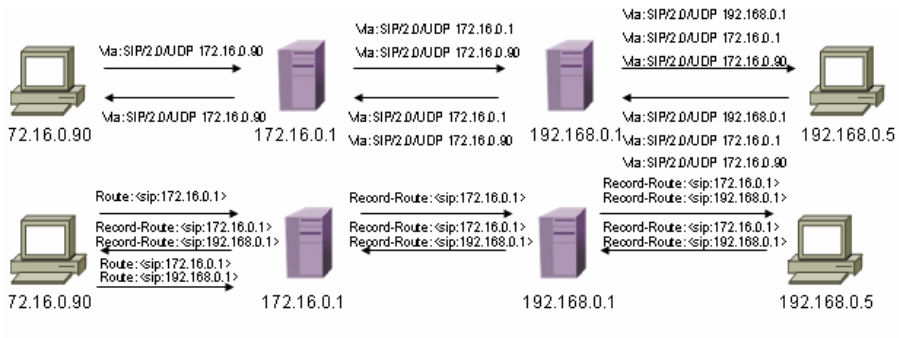


图 2.4 服务器添加消息头的过程

注册注销过程为一种特殊的会话过程，如图 2.5，用户代理向服务器发送 REGISTER 请求，并携带注册信息如用户名、注册有效期等，服务器接收该请求后将请求中的信息加入注册服务器中的用户数据库中，并返回 200/OK 消息提示注册成功。注销过程和注册过程类似，只是在 Expires 头字段中添零即可。

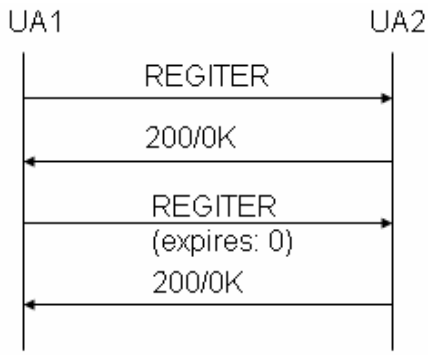


图 2.5 注册/注销过程

第三章　SIP 协议栈 ZmhSipStack 的实现

本章介绍了由作者自主开发的 SIP 协议栈 ZmhSipStack。协议栈实现了对 SIP 协议中常用消息头、常用状态码以及 SDP 协议常用字段的解析，并能够生成符合 RFC3261/2327 格式的 SIP/SDP 请求及响应消息，结合传输层函数及由版本决定的事务状态机机制，完成直接或间接的 SIP 多媒体会话过程。

3.1 协议栈工作原理

图 3.1 为 SIP 协议栈 ZmhSipStack 的概念图。从图中我们可以归纳出一个 SIP 协议栈应该实现的功能：

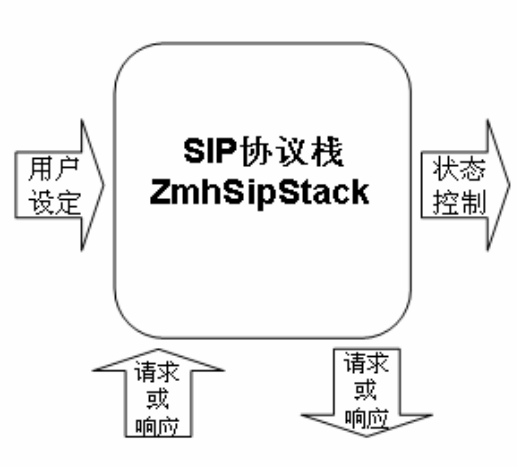


图 3.1 SIP 协议栈

- 能够解析并生成符合 SIP 及其扩展协议格式的数据报；
- 能够应用状态机控制消息的发送接收以及多媒体通信的开始和结束；
- 具有独立的数据空间用于存储解析出的 SIP/SDP 参数；
- 具有简单便捷的函数接口，利于开发应用程序；

根据以上要求，本文设计出了 SIP 协议栈 ZmhSipStack。图 3.2 为其结构图。从图中可以看出，ZmhSipStack 由协议处理器和

数据存储区两大部分构成。协议处理器用于解析、处理接收到的SIP 消息，生成相应的回复消息，并通过状态机设定协议栈的工作状态，进而调度多媒体会话过程，协议处理器由底向上依次为传输层模块、文本解析器、文本生成器、数据处理器以及 SIP 状态机，具体内容请参看本章 3.2 节；数据存储区用于存放用户设定数据、解析出的 SIP/SDP 消息参数以及用户和系统信息。该区域由许多根据协议格式定义的结构体构成，具体内容请参看本章 3.3 节。

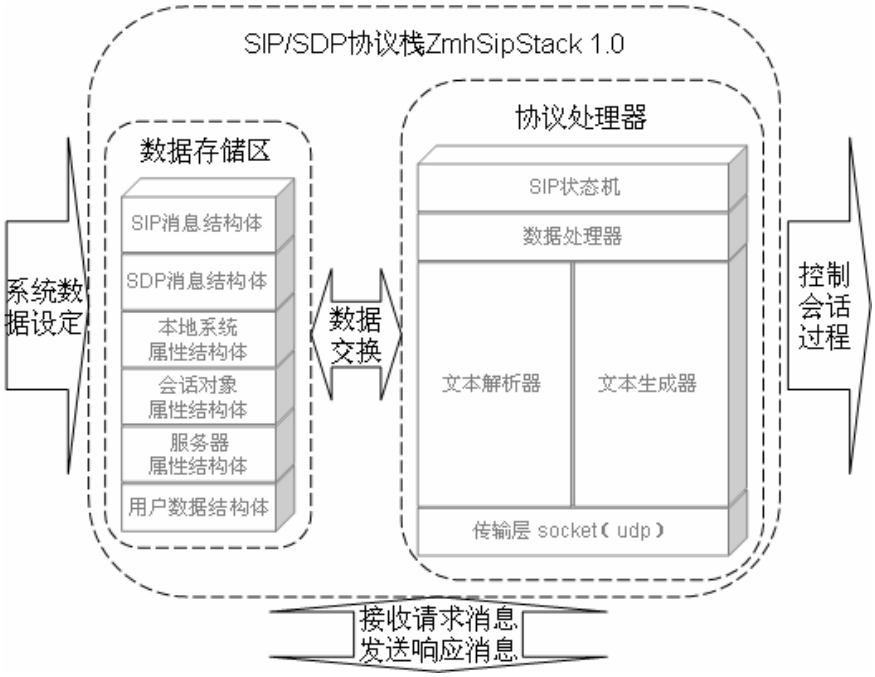


图 3.2 ZmhSipStack 结构图

图 3.3 显示了协议栈的工作原理。如图所示，协议栈以一个事务周期为其工作周期。协议栈工作后首先打开传输层监听，当本地为主叫端时，文本生成器调用数据存储区中由用户输入的被叫方信息，生成会话请求信息后经由传输层发送出去，当接收到对方的响应消息后，首先用文本解析器对该消息进行解析，并将

解析出的数据送入存储区，接着数据处理器访问存储区处理这些数据，处理完毕后文本生成器根据处理结果生成回应消息，再通过传输层发送出去，结束一个事务周期，下一个周期从解析响应消息开始；当本地为被叫端时，事务周期直接从解析请求消息开始，没有生成并发送请求消息的过程。协议栈中的状态机在每个事务周期完成后改变自身的状态，下个周期开始后根据接收到的消息类型确定采用的数据处理方法和文本生成方式，并在特定的状态下开启多媒体会话进程，使整个 SIP 会话能够正常进行。

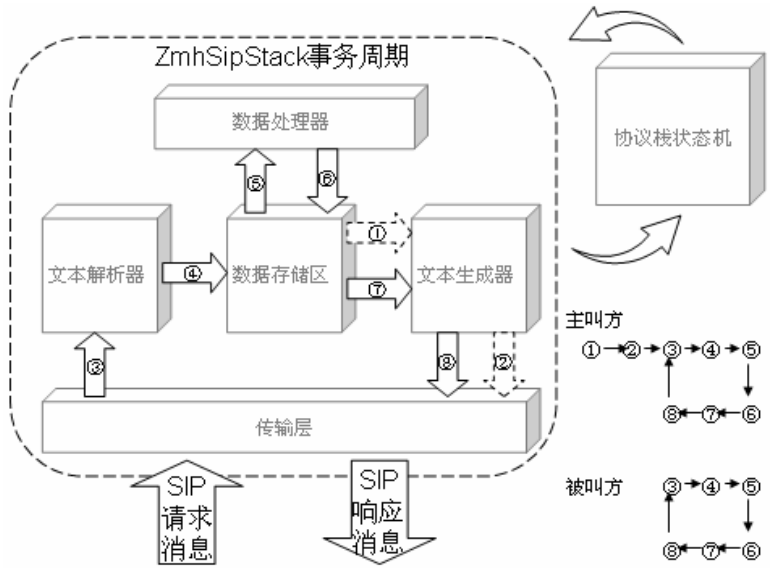


图 3.3 ZmhSipStack 工作原理图

3.2 协议栈处理器的实现

协议栈处理器是协议栈的工作核心，用于解析、处理接收到的 SIP 消息，生成相应的回复消息，并通过状态机设定协议栈的工作状态，进而调度多媒体会话过程。协议处理器由底向上依次为传输层模块、文本解析器、文本生成器、数据处理器以及 SIP 状态机。下面以 PC 环境下的 UA 版协议栈为例对各部分进行讨论。

3.2.1 传输层模块

负责接收和发送 SIP 消息，是 ZmhSipStack 的最底层模块。在 PC 上，这部分模块基于 Winsock 编程，是处于应用层的 SIP 协议栈与传输层 TCP/IP 协议栈的接口。在 UA 版中，传输层模块被封装为 CSipUAManager 类。表 3.1 是类函数列表。

Set 类函数用于向传输层输入会话目的信息以便开启接收和接收 Socket；Get 类函数用于获得已设定的目的地信息；状态机类函数用于状态机状态码的设定和获得；发送 Socket 类函数负责发送 Socket 的建立、注销，以及发送 SIP 消息；接收 Socket 类函数负责接收 Socket 的建立、注销，监听线程的开启、停止。

表 3.1 CSipUAManager 类函数列表（UA 版）

分类	函数名称		
Set 类	SetTargetIP SetDialog	SetTargetPort	SetLocalPort
Get 类	GetTargetIP	GetTargetPort	GetLocalPort
发送 Socket 类	CreateSender SendTo	DeleteSender	Send
接收 Socket 类	CreateReceiver DeleteReceiver	StartReceiving StopReceiving	ReceivingThrd ReceivingLoop

在具体函数的应用方面，在协议栈工作前，首先调用 CreateSender 和 CreateReceiver 类函数创建发送接收 Socket，再调用 StartReceiving、ReceivingThrd 类函数创建监听线程。发送 SIP 请求消息时，CSipManager 类通过 Set 类函数读入被叫方信息并设定发送 Socket，再调用 Send 类函数发送 SIP 请求消息；接收到响应消息时，ReceivingLoop 函数负责进行处理，其主体为一个 while 循环，循环体内部接收对方传递过来的 SIP 消息，调用文本解析、文本生成以及数据处理函数对数据进行处理，实现了一个

事务周期，其工作流程如图 3.4。

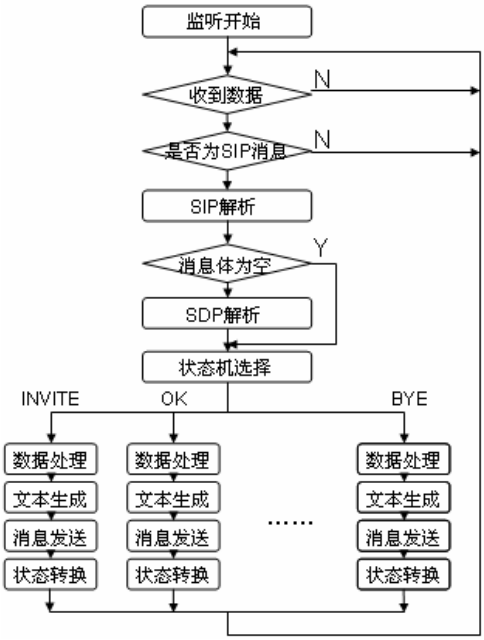


图 3.4 类函数 ReceivingLoop 工作流程图

3.2.2 文本解析器

为了便于理解 ZmhSipStack 文本解析器的工作原理，请先看下面的 INVITE 请求消息实例：

INVITE sip:172.16.0.90 SIP/2.0
Via: SIP/2.0/UDP 172.16.0.15:6000
Max-Forwards: 90
From: "zmhxp"<sip:No>;tag=abcd
To: <sip:172.16.0.90>
Call-ID: bcde@172.16.0.15
CSeq: 1 INVITE
Contact: <sip:172.16.0.15:6000>
User-Agent: ENYAXP1.3
Content-Type: application/sdp
Content-Length: 384

v=0
o=- 0 0 IN IP4 172.16.0.15
s=session
c=IN IP4 172.16.0.15
b=CT:1000
t=0 0
m=audio 37572 RTP/AVP 112
k=base64:fplsa
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=encryption:optional
m=video 33232 RTP/AVP 34
k=base64:SdtjN
a=rtpmap:34 H263/9000
a=encryption:optional
m=application 1503 udp text

a=sendonly | **a=encryption:optional**

从中我们可以看出 SIP、SDP 协议同 HTTP 协议类似，都是按行解析的文本协议。从实例中可以归纳出 SIP 和 SDP 协议格式的一些基本特点：

- 第一行为请求行或状态行，标识消息的方法、要到达的最终目的地址以及 SIP 协议版本号；
- 从第二行开始直到空行前为消息头域，其中每一行为一个消息头字段。每一个消息头都遵循以下格式：首先是头字段名（如 Via、From 等），后面是冒号；然后紧接为一个获多个前导空格，后面为字段值，其格式依字段名而定。消息头语法请参看第二章 2.2.2 小节；
- 消息头域以一个空行为结束标志，接下来即为消息体。消息体按照 SDP 协议格式填充；
- SDP 协议格式特点和 SIP 协议类似。每一行都遵循以下格式：首先是类型名，为一个字母，后面是等号；然后是类型值，其格式依类型名而定，具体语法请参看第二章 2.3.1 小节。

根据以上特点，本文设计了一套四级流水线式的文本解析流程（如图 3.5）：第一级：解析出空行，并以此为界将 SIP 消息分割为消息头与消息体两部分；第二级：解析出消息头和消息体中的回车换行符，并以此为界将消息头和消息体分割成一行行的字段；第三级：消息头部分根据 SIP 格式解析各行第一个冒号前的字段名，消息体部分则根据 SDP 格式解析各行第一个等号前的字段名，然后按照不同的字段名采取不同的解析函数；第四级：各解析函数按照协议规定的格式，以”:”、”@”等标志为界解析头字段，并最终解析出各个头字段中的参数值。

在 UA 版中，文本解析器包含 CSipMsgAnalysis 和 CSdpMsgAnalysis 两种类。分别用于 SIP 格式消息头和 SDP 格式消息体的解析。表 3.2、3.3 是类函数列表。

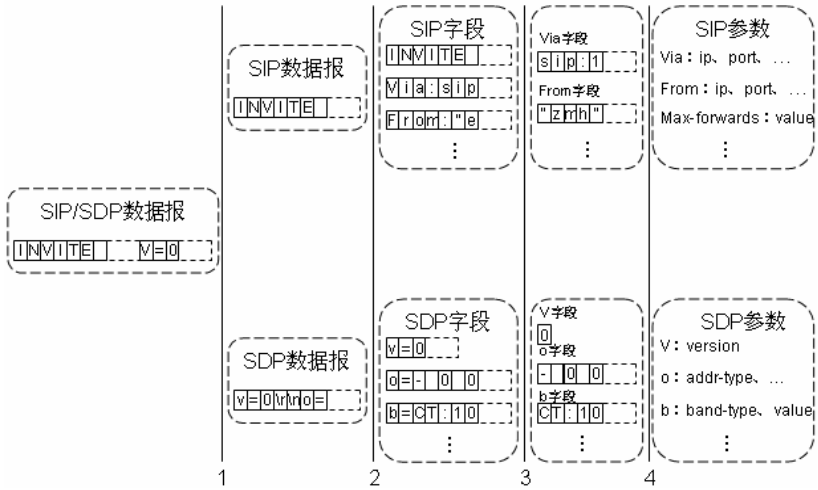


图 3.5 四级流水线式文本解析流程示意图

表 3.2 CSipMsgAnalysis 类函数列表（UA 版）

分类	函数名称		
第二层流水线	SipMsgAnalysis		
第三层流水线	SipLinerAnalysis	CompareSipLinerName	ProtoJudgement
第四层流水线	SipViaAnalysis	SipMaxForwardsAnalysis	SipRouteAnalysis
	SipFromAnalysis	SipRecordRouteAnalysis	SipToAnalysis
	SipCallIDAnalysis	SipContactAnalysis	SipCSeqAnalysis
	SipCtypeAnalysis	SipUserAgentAnalysis	SipClengthAnalysis
	SipExpiresAnalysis		

表 3.3 CSdpMsgAnalysis 类函数列表（UA 版）

分类	函数名称		
第二层流水线	SdpMsgAnalysis		
第三层流水线	SdpLinerAnalysis	CompareSdpLinerName	
第四层流水线	SdpOriginAnalysis	SdpConnectAnalysis	SdpBandwidthAnalysis
	SdpTimesAnalysis	SdpMediaAnalysis	SdpKeyAnalysis
	SdpAttributeAnalysis		

下面描述流水线的具体编程实现。第一级流水线是由CSipMsgAnalysis 中的类函数 SipMsgAnalysis 实现的，该函数一方面将消息头分段实现第二级流水线，另一方面也检测消息头域的界限，当其检测到连续两次出现的回车换行符（一个空行）即返回一个指向消息体首地址的指针，供 CSdpMsgAnalysis 类解析消息体之用（如图 3.6）。

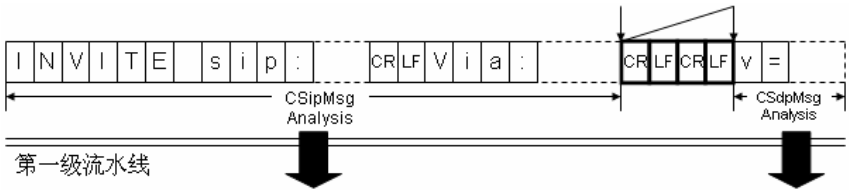


图 3.6 第一级流水线的实现

SipMsgAnalysis 与 SdpMsgAnalysis 类函数实现了第二级流水线。函数首先保存储存 SIP 和 SDP 数据的数组首地址，之后将地址指针循环加一，当发现回车换行符后将首地址到现地址的数据传入下一级流水线，并将现地址指针加 2 后（跳过回车换行符）得到新的数据首地址并进行下一轮解析，这样就可将 SIP 消息头或 SDP 消息体解析为一行行的字段后送入下一级流水线（如图 3.7）。

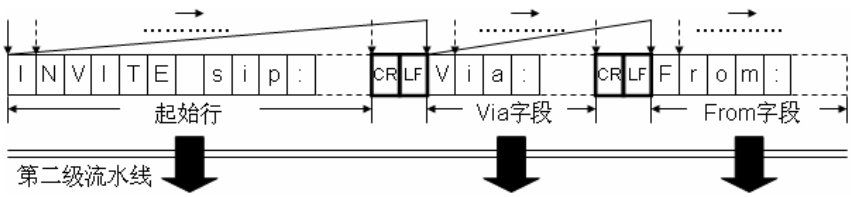


图 3.7 第二级流水线的实现

SipLinerAnalysis 和 CompareSipLinerName 联合实现了第三级流水线。SipLinerAnalysis 函数首先提取第一个冒号前的字段名输入 CompareSipLinerName，根据返回的数值确定字段的名称，并

输入下一级流水线应用符合该字段格式的解析函数进行解析（如图 3.8）。

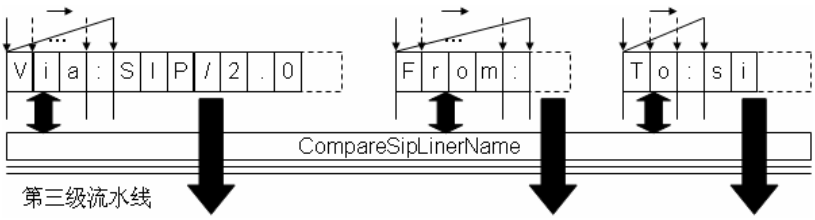


图 3.8 第三级流水线的实现

其余的十三个类函数实现了第四级流水线，即对十三个常用消息头字段的解析。这十三个函数的解析过程比较相似，都是以“@”、“：”等特殊字符为界将字段分割，并按照协议格式保存到数据存储区。下面以 Via 头字段为例讨论第四级流水线的解析过程：首先保存字段数据首地址，之后将地址指针循环加一，当发现空格符后将首地址到现地址的数据存入 ipacket.via.protocol，并将现地址指针加 1 后（跳过空格符）得到新的数据首地址并继续解析，再发现“：”符号时将新首地址到现地址的数据存入 ipacket.via.hostaddr 并将现地址指针加 1 后（跳过“：”）再次得到新的数据首地址，最后将以此地址开始的字符串转换为无符号整数后存入 ipacket.via.hostport，完成 Via 字段的解析（如图 3.9）。程序流程图如图 3.10。

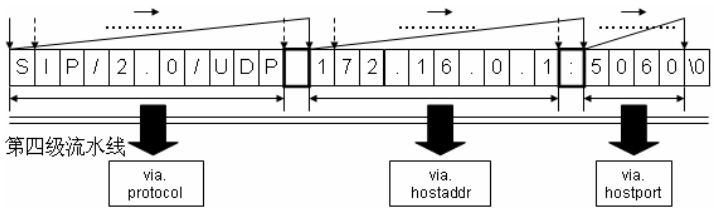


图 3.9 第四级流水线的实现

SERVER 版的文本解析器不需要对消息体进行解析，所以其

中不包含类 CSdpMsgAnalysis，并且为了加快文本生成的速度，还增加了一个名为 SipGetFromPoint 的类函数，用于找到不需要服务器修改的数据的首地址，供生成转发消息使用。

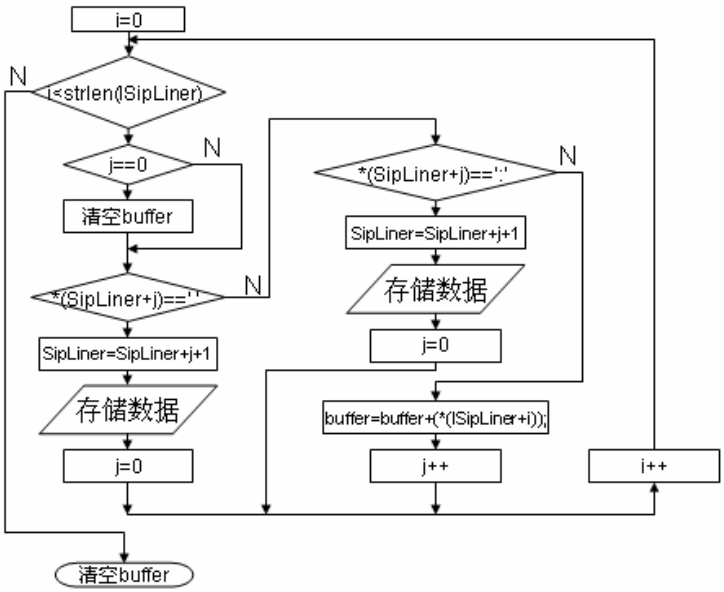


图 3.10 类函数 SipViaAnalysis 工作流程图

3.2.3 文本生成器

文本生成器用于生成符合协议格式的消息文本。文本解析器将解析出的数据经过数据处理器处理后输入文本生成器，文本生成器按照消息格式生成起始行和消息头，若消息体不为空则根据 SDP 协议格式生成消息体，组合起来经过传输层发送出去。

本协议栈支持 INVITE、ACK、BYE、REGISTER 等四种请求消息和 100/Trying、180/Ringing、200/OK 等三种响应消息的生成，并能够在 INVITE、200/OK 消息中加入 SDP 格式的消息体用于媒体描述。在 UA 版本中，文本生成器包含 CSipMsgBuilding 和 CSdpMsgBuilding 两种类，分别用于 SIP 消息和 SDP 消息的生成。表 3.4、3.5 为类函数列表

每一个类函数对应一种消息的生成，类函数外部输入储存数

据的结构体指针，内部主要采用多个 sprintf 函数将输入的数据按照 SIP/SDP 协议格式写入发送数组。

表 3.4 CSipMsgBuilding 类函数列表（UA 版）

分类	函数名称		
SIP 消息生成	SipInviteBuilding	SipByeBuilding	SipOKBuilding
	SipRingingBuilding	SipAckBuilding	SipTryingBuilding
	SipRegisterBuilding		

表 3.5 CSdpMsgBuilding 类函数列表（UA 版）

分类	函数名称	
SDP 消息生成	SdpInviteBuilding	SdpOKBuilding

作为例子，下面列出 SipRegisterBuilding 函数中的部分代码：

```
j=sprintf(SendMsg,"%s sip:%s %s\r\n","REGISTER",
          UAinfo->serverip,UAinfo->sipversion);
j+=sprintf(SendMsg+j,"%s: %s/%s %s:%d\r\n","Via",UAinfo->sipversion,
          UAinfo->baseproto,UAinfo->hostip,UAinfo->hostport);
j+=sprintf(SendMsg+j,"%s: sip:%s@%s\r\n","From",
          UAinfo->username,UAinfo->hostip);
j+=sprintf(SendMsg+j,"%s: sip:%s@%s\r\n","To",
          UAinfo->username,UAinfo->hostip);
……
```

SERVER 版的文本生成器不需要生成消息体，并且消息头部分也只是对 Via、Route 头字段部分进行修改，所以其中不包含 CSdpMsgBuilding 类。文本生成器生成转发消息时只需利用文本解析器中 SipGetFromPoint 函数提供的数据首地址，将原有的 From 字段一下的消息头部分和消息体粘贴到修改后的 Via、Route 字段后即可。

3.2.4 数据处理器

数据处理器用于对文本解析出的数据进行处理，以供文本生

成器生成响应消息时使用。数据处理器的处理办法和协议栈的版本有关：用户代理版的数据处理器侧重媒体类型协商；服务器版的用户处理则更侧重于用户的注册和鉴权还有消息头的修改。

在UA版本中，数据处理器包含CSipMsgProcessing类，用于对文本解析器解析出的数据进行处理。表3.6为类函数列表。

表 3.6 CSipMsgProcessing类函数列表（UA版）

分类	函数名称		
SIP/SDP 数据处理	SipInviteProcessing	SipByeProcessing	SipOKProcessing
	SipRingingProcessing	SipAckProcessing	SipTryingProcessing

每一个类函数对应一种接收消息的数据处理。SipInviteProcessing 和 SipOKProcessing 类函数对接收到的INVITE、200/OK 消息的消息体进行解析，并将解析出的对方所能支持的媒体格式与本地机能够支持的媒体格式作比较后选出其中的交集作为通信用音视频编解码格式；SipTryingProcessing 和SipRingingProcessing 处理的是消息体为空的临时响应消息，所以只是简单延时即可；SipAckProcessing 最终确认媒体格式，并开始接收音视频数据；SipByeProcessing 终止音视频数据的传输。

SERVER 版的数据处理器与 UA 版数据处理器有较大不同，下面表3.7为SERVER版的CSipMsgProcessing类函数列表。

表 3.7 CSipMsgProcessing类函数列表（SERVER版）

分类	函数名称	
鉴权类	CheckUserInfo	
数据处理	SipInvite_Ack_Bye_Processing	SipRegisterProcessing
	SipOK_Ringing_Processing	

函数 CheckUserInfo 完成鉴权功能，在任何非 REGISTER 消

息处理之前服务器必须首先使用该函数对消息的发起者和接受者进行鉴权，只有注册过的用户才允许通信。函数SipRegisterProcessing完成对注册请求的处理：若发送请求的用户代理未注册，则将请求中携带的用户代理信息写入服务器自带的用户列表中；若已经注册且 Expires 字段值不为零，则重新注册用户代理，并更新用户使用时限；若已经注册且 Expires 字段值为零，则注销该用户代理端，将其信息从用户列表中移除。函数SipInvite_Ack_Bye_Processing完成对除REGISTER消息以外的请求消息的处理：函数首先鉴权，然后判断 Route 域的第一个值是否和本地地址一致，若一致且目的地域为本地域，则将用户列表中保存的目的地信息写入数据存储区以供生成转发消息使用，否则按 Via 列表发回出错消息。函数 SipOK_Ringing_Processing 完成对响应消息的处理，处理过程与 SipInvite_Ack_Bye_Processing 函数相似。

3.2.5 SIP 状态机

SIP 状态机是整个协议栈的控制核心，总体调度协议栈的运行。状态机机制存在用户代理部分及有状态的服务器中，这样可以提高终端的智能性，减少网络中继设备的成本。在用户代理版协议栈中，SIP 状态机由 CSipUAManager 中的 SetStatusCode 和 GetStatusCode 类函数控制，所有的状态值都用方法名或状态描述符表示，这也体现了一次状态的转化就是一次事务的完成。用户代理端的事务状态机如图 3.12。

事务状态机启动后，网络监听开启，状态机设定为 WAITING，若无会话请求则保持此状态。这时状态机会拥有两条走向：

(1) 若本地机发起会话，则状态机由 WAITING 转为 INVITE，并等待接收 100/Trying 响应消息；接收到 100/Trying 消息后，本地机停止重发 INVITE 消息，状态机由 INVITE 转为 TRYING，并等待接收 180/Ringing 消息；接收到 180/Ringing 消息后，状态

机由 TRYING 转为 RINGING，并等待 200/OK 消息；若未收到 200/OK 消息，状态机由 RINGING 转为 WAITING，若收到则状态机转为 ACK 并发出 ACK 请求，同时打开会话开始多媒体传输；

(2) 若本地机收到会话请求，则状态机由 WAITING 转为 RINGING，并等待用户选择接收请求或放弃请求；若用户不接受请求，状态机由 RINGING 转为 WAITING，若接受请求则转为 OK 并发送 200/OK 消息；接收到对方最终请求 ACK 后状态机由 OK 转为 ACK，同时打开会话开始多媒体传输。

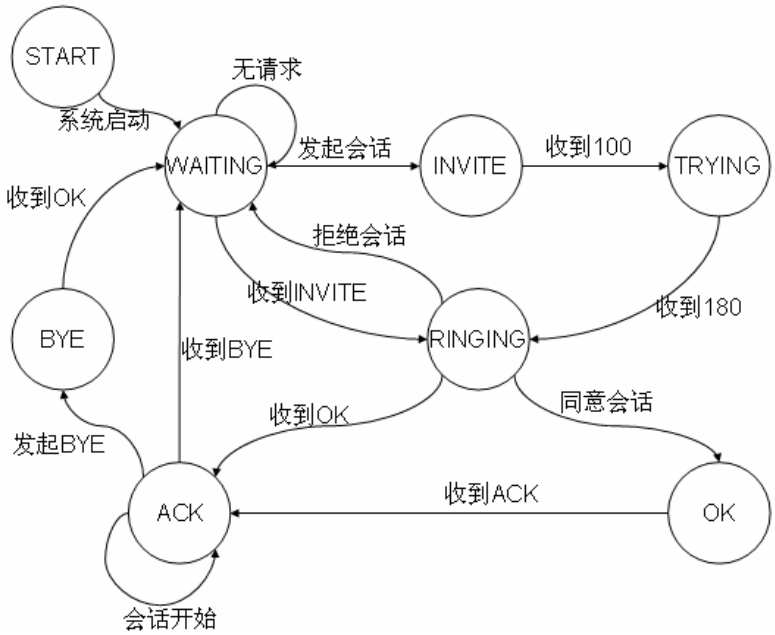


图 3.12 ZmhSipStack 事务状态机（UA 版）

终止会话时，一方发送 BYE 请求，状态机由 ACK 转为 BYE，并等待 200/OK 消息；另一方接受 BYE 请求后，状态机由 ACK 转为 WAITING，发送 200/OK 响应并释放会话；发送 BYE 请求的一方接收到 200/OK 消息后，状态机由 BYE 转为 WAITING 并释放会话，此时一个会话过程完成。

在服务器方面，因为本文中构建的服务器为无状态服务器，

所以在 SERVER 版协议栈中未引入状态机机制。若将其改造为有状态服务器，则需要保存每个通过该服务器节点的会话的状态，这时就需要引入状态机机制了，且服务器的状态机是将主叫方与被叫方两者的状态机结合在一起设定的。

3.3 数据存储区的实现

SIP 协议是一种文本格式的信令协议，会话双方的信息通过请求和响应的形式进行交互。每一次消息的解析和会话协商都会产生大量的数据，并且大部分数据是需要在整个会话过程中使用的，这就要求协议栈能够单独开辟一个存储空间用于存放这些数据，并且使协议栈处理器可以对其进行方便的访问。本文根据 SIP/SDP 协议的有关规定，定义了一系列结构体用于保存 SIP/SDP 文本数据。表 3.8 为结构体列表。

表 3.8 数据存储区结构体列表

分类	用户代理版		服务器版	
总体结构	ua_info caller_info	sip_packet sdp_packet	users_info	sip_packet
SIP 字段	sip_message_l sip_maxforwards_l sip_record_route_l sip_content_type_l sip_cseq_l sip_user_agent_l sip_content_length_l	sip_via_l sip_route_l sip_from_l sip_call_id_l sip_contact_l sip_expires_l sip_to_l	sip_message_l sip_maxforwards_l sip_record_route_l sip_content_type_l sip_cseq_l sip_user_agent_l sip_content_length_l	sip_via_l sip_route_l sip_from_l sip_call_id_l sip_contact_l sip_expires_l sip_to_l
SDP 字段	sdp_version_l sdp_session_l sdp_bandwidth_l sdp_attributes_l sdp_media_l	sdp_origin_l sdp_connect_l sdp_times_l sdp_key_l	无	

其中结构体 ua_info 和 caller_info 只存在于用户代理版协议栈

中，分别用于存储本地用户代理和对方用户代理的信息；结构体 user_info 只存在于服务器版协议栈中，用于存储在服务器上登记的用户代理信息，具体定义请参看 SipStructure.h 头文件。

结构体 sip_packet 和 sdp_packet 用于存储由文本解析器解析出的 SIP 和 SDP 数据。它们内部是由表中各个表示字段的结构体构成的。结构体 sip_packet 的定义如下：

```
struct sip_packet
{
    LPTSTR sip_stack[MAX_SIP_STACK];      sip_from_l from;
    int via_number;                       sip_to_l to;
    int route_number;                     sip_call_id_l call_id;
    int record_route_number;              sip_cseq_l cseq;
    sip_message_l message;                sip_contact_l contact;
    sip_via_l via[MAX_SERVER];            sip_user_agent_l user_agent;
    sip_maxforwards_l maxforwards;        sip_content_type_l content_type;
    sip_route_l route;                    sip_content_length_l content_length;
    sip_record_route_l record_route;      sip_expires_l expires;
};
```

字段结构体一般定义为 sip_字段名_l 或 sdp_段名_l，其内部参数按照 RFC 3261 中的规定进行定义，具体内容请参看 SipStructure.h 和 SdpStructure.h 头文件。

协议栈运行前，首先划出独立的存储空间用于存放结构体，这部分存储空间就构成了数据存储区，用户代理的属性、呼叫方的属性和文本解析结果等数据就存放在这个区域内。文本解析器、文本生成器以及数据处理器按照声明的结构体格式对这个存储区进行读写。

第四章　基于 ZmhSipStack 的应用程序编写

本章主要介绍如何基于 ZmhSipStack 编写应用程序。首先介绍基于 ZmhSipStack 构建应用程序的基本原理和方法，然后分别介绍两个应用实例：用户代理程序 SipUA 和服务器程序 SipProxyServer。

4.1 基于 ZmhSipStack 的应用程序开发

作为应用层协议栈，ZmhSipStack 一方面需要针对传输层协议栈具有良好的接口，一方面也需要对更上层的用户界面有很好的支持。同时协议栈也要协调好与应用功能模块之间的调度关系。一个简单的 SIP 应用程序结构如图 4.1。

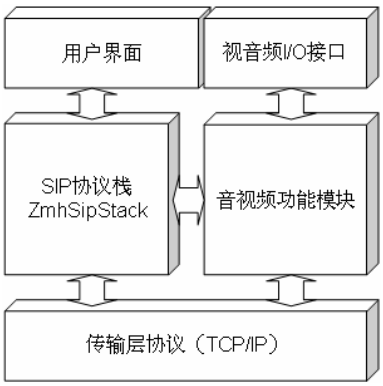


图 4.1 一个简单的 SIP 应用程序结构图

如上图所示，一个简单的 SIP 应用程序除了应具备一个 SIP 协议栈外，还需要传输层协议栈、友好的用户界面、一个或多个功能模块以及与之对应的输入输出接口。传输层协议栈负责传递 SIP 消息和音视频数据；用户界面负责设定协议栈的属性以及会话的触发；功能模块实现在 SIP 协议栈状态机控制下的各种应用功能（如音视频、文本聊天、白板、共享文件等等）；而 I/O 口则负责显示应用功能的处理结果以及向功能模块内输入外部数据。

开发基于 ZmhSipStack 的应用程序时，可以将协议栈和功能模块实体化为一个个的类，并在对话框类中调用这些类并完成其功能，即以用户界面类为框架构建整个系统。

基于以上原理，本文在 ZmhSipStack 协议栈的基础上构建了用户代理程序 SipUA 和服务器程序 SipProxyServer，下面将对这两个应用程序进行剖析。

4.2 用户代理程序 SipUA 的实现

用户代理程序 SipUA 基于 UA 版 ZmhSipStack 协议栈，内含音视频采集输出模块、编解码模块及 RTP 协议模块，实现 SIP 协议控制下的端到端音视频 RTP 通信。音频采集与输出模块应用 Microsoft 提供的 Waveform Audio 函数库；视频采集与输出模块则采用 VFW 软件包中的 Video Capture 函数库与 DrawDib 函数库；音频编解码器采用开源 G.729a 库；视频编解码器采用 K-liteCodecPack2.24，并应用 VFW 中的 Video Compression Manager 函数库调用；RTP 协议模块则是参考 RFC 1889 中的格式规定编写而成的。

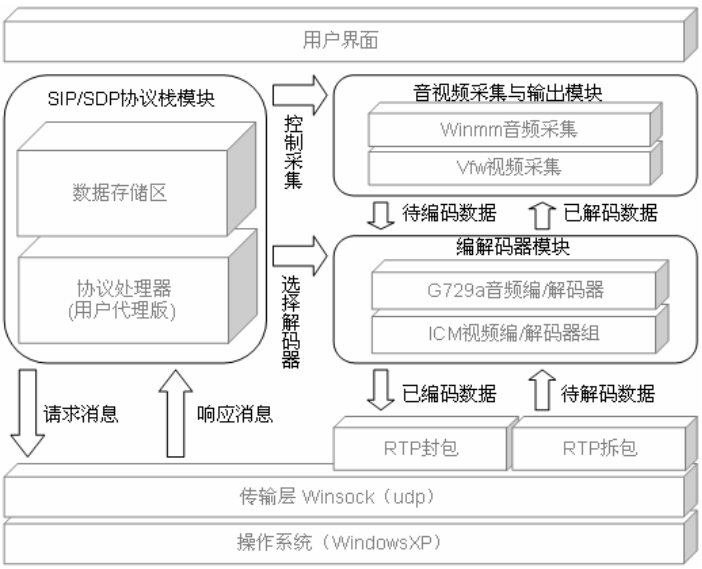


图 4.2 用户代理程序 SipUA 的结构图

图 4.2 为用户代理程序 SipUA 的结构图。用户代理工作时，协议栈既通过状态机控制音视频采集传输的开始和结束，又通过数据处理器的处理结果选择音视频编解码器。音视频采集模块采集到数据后将这些数据输入到编解码器模块，编解码器模块根据媒体协商的结果采用指定的编解码器对输入数据进行编码后输出至 RTP 模块，RTP 模块对输入的已编码数据进行封包，最后通过传输层协议栈传输出去；传输层接收到对方传来的 RTP 包后，首先进行 RTP 拆包，然后将音视频数据输入音视频解码器进行解码，解码后的数据送入音视频采集输出模块，最后显示视频信号显示在用户界面上，音频数据输出至扬声器。下面分别介绍各功能模块的实现。

4.2.1 RTP模块

RTP模块用于将等待传输的音视频数据封装为RTP数据包，同时也用于将接收到的RTP数据包拆包。RTP报头格式定义在RtpConstant.h中，具体格式如下：

```
struct RtpHeaderStruct
{
u_int32 count : 4;                    u_int32 marker : 1;
u_int32 extension : 1;                u_int32 sequence : 16;
u_int32 padding : 1;                  RtpTime timestamp;
u_int32 version : 2;                  RtpSrc ssrc;
u_int32 type : 7;                     RtpSrc startOfCsrc;
} RTP_HEADER;
```

在SipUA程序中，RTP模块被定义为CRtpStructure类， RtpInit类函数用于初始化RTP报头，随机生成初始序列号和时间戳；ReadRtpHeader类函数用于读取接收到RTP包中的数据并解包；WriteRtpHeader类函数用于在待发送的音视频数据前加RTP报头。SipUA程序在传输和接收音视频数据时调用这些类函数，实现RTP流媒体通信。

4.2.2 编解码器模块

编解码器用于处理视音频数据。采集模块采集到的音视频数据根据媒体协商确定的编码器进行编码，传输层接收到的音视频数据根据媒体协商确定的解码器进行解码。音频编解码器采用开源库 G729a.lib，视频编解码器采用 K-liteCodecPack2.24，并应用 VFW 中的 Video Compression Manager 函数库调用。

在 SipUA 中，音频编解码器被定义为 CSipAudioCodec 类，视频编解码器被定义为 CSipVideoCodec 类，在音频编解码器类中，Compress 类函数内部调用了 G729.lib 中的 va_g729a_encoder 函数，实现了一路音频数据的编码，频率为 8KHz，每帧 160 个采样点；UnCompress 类函数内部则调用了 va_g729a_decoder 函数，实现了一路音频数据的解码；在视频编解码器中，Init 类函数用于初始化 COMPVARS 结构体，然后调用 IC 系列函数打开设置视频编解码器；EncodeVideoData 类函数内部调用 ICSeqCompressFrame 函数，用于对 COMPVARS 结构体中设置好的视频流进行按帧编码；DecodeVideoData 类函数内部则调用了 ICDecompress 函数，用于对视频流按帧解码；SetCodecType 类函数用于选择视频编解码格式，这个格式是由 SIP 媒体协商得到的。

4.2.3 音视频采集与输出模块

音视频采集模块用于操纵声卡及摄像头采集音视频数据，并将数据传送入音视频编解码器中；音视频播放模块用于将音频和视频数据输出到扬声器和显示屏幕上。在SipUA中，音频采集和播放部分应用了Microsoft提供的waveform Audio系列API函数，并封装在CSipAudioRecv和CSipAudioPlay两个CWinThread子类中；视频采集和播放部分应用了VFW函数库中的Video Capture系列函数及DrawDib系列函数，并封装在CSipVideoManager类中。

在 CsipAudioRecv 中，OnStartRecording 类函数中调用了 waveInOpen、waveInStart等函数，用于接收声卡采集出的音频数据；OnStopRecording类函数用于停止音频采集；OnEndThread是

从父类CWndThread继承来的类函数，用于终止线程。OnSoundData类函数中一方面发送音频数据到远端，一方面加载用过的音频缓冲以便使缓冲区循环使用。

在CsipAudioPlay类中，OnStartPlaying类函数中调用了waveOutOpen等函数，用于初始化扬声器；OnStopPlaying类函数用于停止扬声器设备；OnEndThread是从父类CWndThread继承来的类函数，用于终止线程。OnWriteSoundData类函数用于将音频数据写入输出缓冲区以播放音频数据；OnEndPlaySoundData用于暂停音频播放。

在CSipVideoManage类中，Initialize类函数用于初始化视频设备，并调用capPreview函数播放本地视频；SetCapturePara类函数设定状态字CAPTUREPARMS用于设定捕捉图像的属性；OnCaptureVideo类函数获得本地视频，并开启传输线程；StartCapture类函数开启摄像头视频捕捉；StopCapture类函数停止视频捕捉。

4.2.4 用户对话框设计以及程序运行效果

SipUA的最外层为用户界面，这部分负责接收用户输入的对方地址服务器地址等信息，并负责会话的发起结束及显示视频图像等功能。在SipUA中，有关用户界面对话框的定义都包含在CSipUADlg类中。其中OnInitDialog类函数是对话框初始化函数，也是整个应用程序的初始化函数，在该函数中开启ZmhSipStack协议栈监听线程，开启音视频运行线程函数并初始化了一些数组和指针；OnRingDlg类函数当用户代理程序收到会话请求时创建Ringing对话框，并由用户选择是否同意会话，同意则发送200/OK相应消息；InitCap和InitDib分别用于摄像头和视频显示的开启；DestroyCap和InitDib则用于摄像头和视频显示的关闭；OnSipInvite、OnSipRegister和OnSipBye等三个类函数都是按钮控件消息响应函数，分别用于发起INVITE、REGISTER、BYE三种

请求消息，函数内部调用协议中的文本生成器和传输层函数用于请求消息的生成和发出，并在程序的最后设定协议栈状态机；OnStartConference类函数由协议栈状态机控制，当状态机设定为ACK时，视音频通信开始，这时协议栈发出WM_SIPUA_STARTCALL消息，启动这个类函数开始点对点视音频会话；DestroyConference类函数也由协议状态机控制，当状态机由ACK转为WAITING时，意味着结束会话，这时协议栈发出WM_SIPUA_STOPCALL消息，启动该函数结束会话。

构建好的SipUA程序在奔腾4处理器、256M内存及10M 以太网环境下运行状态良好。图4.3为程序运行界面。

图 4.3 SipUA 运行界面

(a) MPEG-4　　(b)H.263　　(c)H.261

图 4.4 视频协商效果图

远程用户代理与本地用户代理之间通过 SDP 媒体协商确定使用的视音频编解码格式。在 SipUA 中，音频编解码器包括 G711 和 G729a 两种（可以自行添加），视频编解码器包括 H.263、H.261、

MPEG-4 等多种。媒体协商时，首先由协议栈中的数据处理器处理最终响应消息的消息体，获得对方用户代理的音视频编解码器列表，并与本地用户代理的音视频编解码器列表进行比对，最终取其交集中码率符合网络条件的。图 4.4 为分别采用 MPEG-4，H.263 和 H.261 的视频效果。

4.3 无状态服务器程序 SipProxyServer 的实现

无状态服务器程序 SipProxyServer 同样基于 ZmhSipStack 协议栈。与 SipUA 不同，服务器程序中不需要音视频解码器、状态机等模块，因为其只需对接收到的消息数据进行转发并修改消息头域中的路由部分即可。服务器工作时，接收用户代理发送来的请求或响应消息，首先对发送消息的用户代理进行鉴权，若此用户代理已在改服务器上进行注册，则通过消息头域中的 Via、Route、Record-Route 等头字段判断下一跳的传输地址，具体原理请参考本文第二章的有关内容。图 4.5 为 SipProxyServer 的结构图。

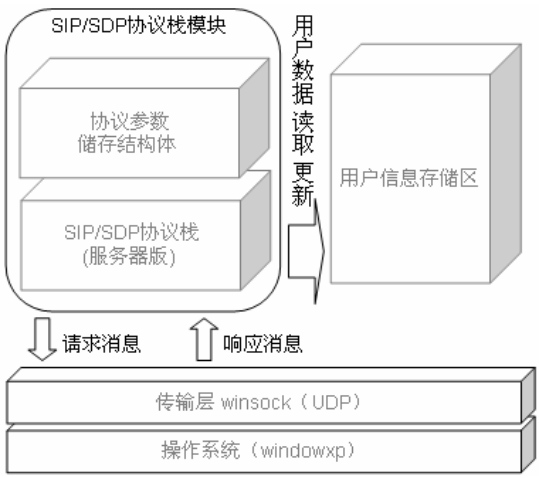


图 4.5 服务器应用程序 SipProxyServer 结构图

从图中可以看出，SipProxyServer 相对 SipUA 来说除了去除

了一些功能模块外，还要增加一块用于存储注册用户数据的存储区，在实际应用中这部分相当于注册服务器使用的数据库，服务器通过对用户数据的读取和更新维护网络用户的使用权限。本应用程序只需将接收到的请求消息进行转发进行转发即可，所以SIP 状态机也可去除，使得服务器程序能够适应嵌入式环境下的运行。

程序协议栈代码和 SipUA 程序类似，只是在数据处理器和文本生成器方面做出适当修改以适应服务器处理过程。

第五章 ZmhSipStack 在嵌入式环境下的应用

本章介绍 SIP 协议栈 ZmhSipStack 在嵌入式 ARM/μCOS-Ⅱ环境下的应用。首先介绍嵌入式 ARM 环境，然后讲解协议栈的嵌入式移植过程，最后介绍嵌入式应用程序 ArmSipUA 和 ArmSipProxy 的构建。

5.1 ARM/μCOS-Ⅱ嵌入式开发环境介绍

本文采用的嵌入式硬件开发环境为周立功公司生产的 SmartARM2200 开发实验板，此开发板核心为 PHILIPS 公司基于 ARM7TDMI 核的 LPC2210 芯片；软件操作系统采用的是 μC/OS-Ⅱ小型实时操作系统；开发工具采用 ADS 1.2 IDE。工作平台和开发环境如图 5.1 所示。

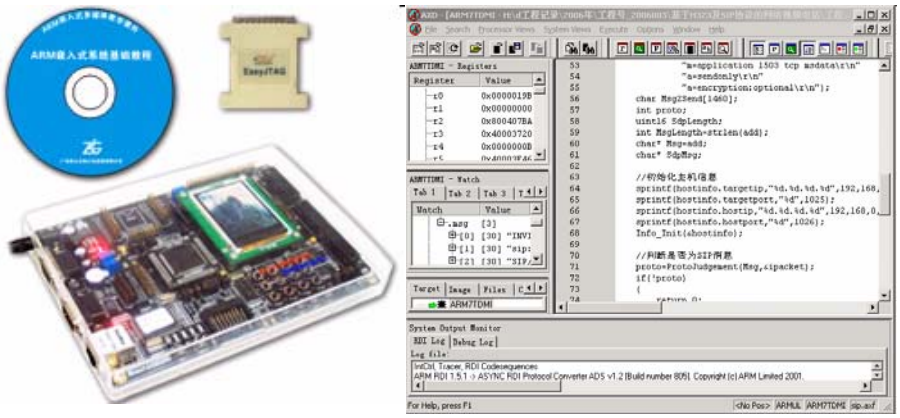


图 5.1 嵌入式开发环境

SmartARM2200 开发实验板上标配 2M字节NOR FLASH、8M字节PSRAM、16M字节NAND FLASH和 256 字节E^2PROM。外设接口丰富，本文主要应用其上标配的 10M以太网接口（采用RTL8019AS芯片）。

μC/OS-Ⅱ实时操作系统以其小巧、灵活、资源占用少等优点在低成本嵌入式产品中获得了广泛应用。操作系统本身可以看成是一种任务控制程序，所有功能函数必须定义为含有一个死循环的任务并独占一定的堆栈资源。任务执行时通过操作系统根据创

建任务时规定的优先级从高到低依次执行，低优先级的任务只有当高优先级任务挂起或注销后才会执行。任务和任务之间通过消息邮箱或消息队列进行通信。

ADS 1.2IDE 采用了文本编译和调试分离的用户界面，支持软件仿真和联机 JTAG 单步调试。本 IDE 支持多种类型的 ARM 芯片，用户可以灵活的定义需要的编译器生成相应得汇编代码。

下面介绍 SIP 协议栈 ZmhSipStack 在嵌入式环境下的移植过程并在此基础上构建嵌入式 SIP 应用程序。

5.2 ZmhSipStack 的嵌入式移植

在协议栈移植前，需要将 VC++环境下的协议栈代码改写为纯 C 代码，改写过程中主要注意以下几个问题：

- 所有宏定义，所有结构体定义都按照所属协议不同分为 SIP 和 SDP 两类分别定义在头文件 sip.h 和 sdp.h 中；
- 所有的类都需要展开，类函数根据处理协议的不同分为 SIP 和 SDP 两类分别在头文件 sip.h 和 sdp.h 中声明为外部函数；
- 遵从操作系统的定义，将 char 定义为 uint8，int 定义为 uint16；
- 源程序中用到的 CString 类都需要改换为 uint8 格式的数组，清空字符串将.Empty()改用 memset 函数；
- 传输层接口按照嵌入式 TCP/IP 协议栈修改，本文采用了开发板自带的 ZLG_IP 协议栈。

修改好后的 ZmhSipStack 为由两个头文件(sip.h 和 sdp.h)、两个 c 文件(sip.c 和 sdp.c)构成。下面简述协议栈的应用过程。

首先在 ADS1.2 环境下建立 ARM for lpc2200 工程并设置软硬件环境，再将 ZLG_IP 文件夹放入工程目录下，并将其中所有文件包含进工程中。然后将修改好的 sip.h 和 sdp.h 加入 ZLG_IP 协议栈目录下的 Include 文件夹中，并在 cfg_net.h 文件中包含，然

后在协议栈根目录下建两个名为 sip 和 sdp 的文件夹，再将 sip.c 和 sdp.c 放入这两个文件夹中，这样就使 ZmhSipStack 协议栈加入到了 ZLG_IP 里，效果如图 5.2 所示。

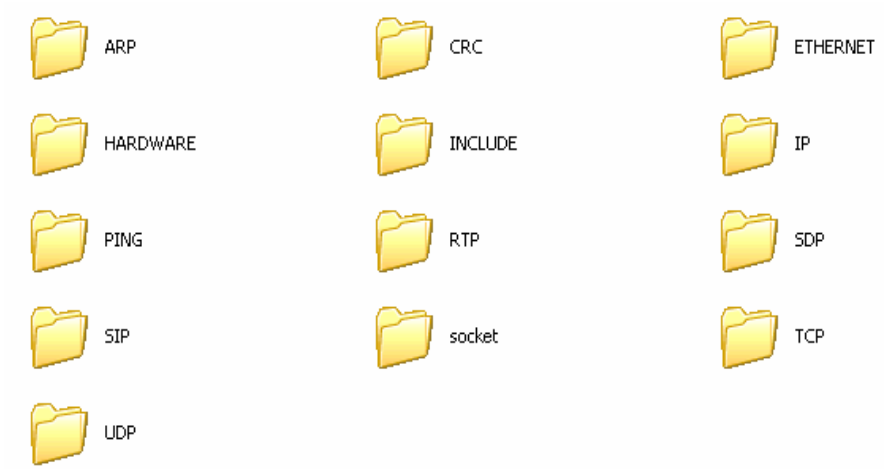


图 5.2 ZLG_IP 根目录

下一步将 μC/OS-Ⅱ系统源代码导入工程中，并设置 include.h 文件，将协议栈的 cfg_net.h 头文件包含进来，并在工程中的 user 文件夹下新建 main.c 文件，准备开始构建嵌入式应用程序。在移植过程中需要调试时可以采用脱机仿真的方式，但是之前注意需要将 config.h 中的一处中断查询循环注释掉。

5.3 基于 ZmhSipStack 的嵌入式应用程序构建

μC/OS-Ⅱ下的应用程序构建与 Windows 相比有很大不同。在 μC/OS-Ⅱ下的编程是以任务块为单位的，开发者需要自己设定任务的优先级和其所占用的堆栈大小。一般来说，应用程序首先须调用 OSInit 初始化系统，然后调用 OSTaskCreateExt 或 OSTaskCreate 创建任务，最后调用 OSStart 运行系统任务调度函数，该函数按照以创建任务的优先级首先运行优先级最高的任务，并只有当该任务挂起、注销或中断时才开始运行低优先级的任务。

一个任务中必须包含一个死循环，并在循环体中实现该任务的功能，高优先级任务的循环内部还需要包含 OSTimeDly 函数用

于挂起任务，这样才能使得低优先级的任务得以执行。

基于以上原理，本文在 μC/OS-Ⅱ环境下基于 ZmhSipStack 协议栈构建了用户代理应用程序 ArmSipUA 和服务器应用程序 ArmSipProxy。

下面依次说明 ArmSipUA 和 ArmSipProxy 的构建。

5.3.1 ArmSipUA 构建

嵌入式用户代理程序是基于嵌入式版的 ZmhSipStack，并仿照 PC 环境下运行的 SipUA 得到的。

在 μCOS-Ⅱ环境下，应用程序是以任务为单位构成的。在 ArmSipUA 中一共有四个任务：任务 A 优先级最高为 0，负责定时器初始化等基本功能的实现；任务 B 优先级次之为 1，负责 UDP 协议初始化；任务 C 优先级为 2，负责实现 ZmhSipStack 的运行和调度；任务 D 优先级最小，用于实现嵌入式 RTP 通信。任务 A 由主函数创建并首先开始运行，而在任务 A 中又将创建任务 B、C、D，并通过挂起任务 A 使得这三个任务得以依次进行。这四个任务的基本属性参数如下表所示。

表 5.1 ArmSipUA 任务属性表

任务名称	堆栈大小(B)	实现功能	优先级
TaskA	200	定时器初始化	最高（0）
TaskB	1000	UDP 协议	1
TaskC	3000	SIP 协议栈	2
TaskD	1000	RTP 通信(应用)	3

图 5.3 为 ArmSipUA 的状态转换图。

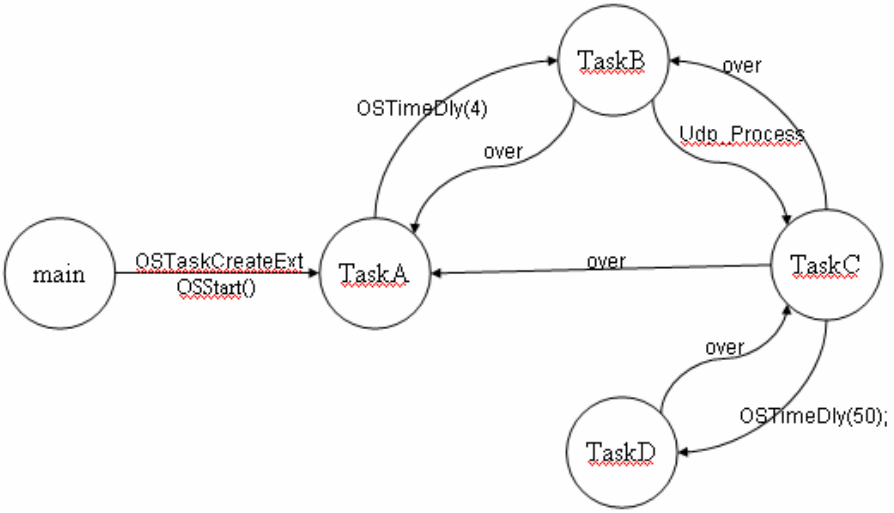


图 5.3 ArmSipUA 状态转换图

5.3.2 ArmSipProxy 构建

与 ArmSipUA 类似，服务器程序 ArmSipProxy 是由 PC 版服务器程序 SipProxyServer 改写而来。在 ArmSipProxy 中共有三个任务，前两个任务的功能与 ArmSipUA 完全一致，只是在最后的任务中加入了注册消息处理及响应程序。表 5.2 为其任务属性列表

表 5.2 ArmSipProxy 任务属性表

任务名称	堆栈大小(B)	实现功能	优先级
TaskA	200	定时器初始化	最高（0）
TaskB	2000	UDP 协议	1
TaskC	4000	SIP 协议栈（含注册）	2

第六章 局域网内的 SIP 通信实验

本章介绍基于局域网环境内 PC 端到 PC 端的 SIP 客户端直接通信以及嵌入式 SIP 服务器与 PC 端用户代理之间的间接通信实验，并对实验结果给予分析。

6.1 实验条件介绍

硬件条件：10M 带宽局域网络，两台 PC 机，一台嵌入式开发板，两部摄像头，麦克耳机，网线若干；

软件条件：基于 PC 机的 SIP 用户代理程序 SipUA，基于嵌入式 ARM 环境的服务器程序 ArmSipProxy，网络抓包工具 Ethereal（如图 6.1）。

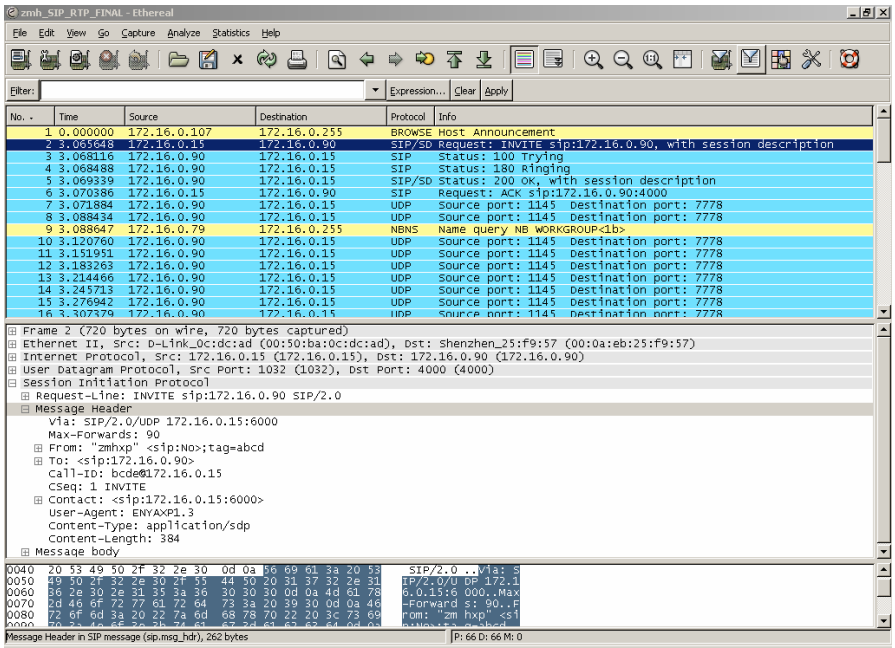


图 6.1 Ethereal 软件抓包环境示意图

6.2 实验步骤说明

1） PC 到 PC 直接通信

- 在两台 PC 机上安装 SipUA 所需的 K-LiteCodec 编解码器集合，并运行 SipUA；
- 点击“INVITE”，在会话属性栏中选择“直接会话”，并填入对方 IP 地址（172.16.0.100）；
- 其中一台运行 Ethereal，开始监听整个网络的数据发送情况；
- 出现对方图像，听到对方声音以后按“BYE”发送 BYE 请求，结束多媒体会话；
- 停止 Ethereal 抓包并将抓包结果保存为.pdf 文件。

实验结果如下图所示：

No.	Time	Source	Destination	Protocol	Info
3	0.002677	172.16.0.100	172.16.0.90	SIP/SD	Request: INVITE sip:172.16.0.90, with session description
4	0.024890	172.16.0.90	172.16.0.100	SIP	Status: 180 Ringing
6	3.480684	172.16.0.90	172.16.0.100	SIP/SD	Status: 200 OK, with session description
7	3.487500	172.16.0.100	172.16.0.90	SIP	Request: ACK sip:\325\305\303\316\352\317@172.16.0.90:0
26	6.302922	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28756, Time=48H263 pa
29	6.488504	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28757, Time=140H263 p
30	6.492186	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28758, Time=141H263 p
31	6.499132	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28759, Time=144H263 p
32	6.553279	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28760, Time=171H263 p
34	6.587047	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28761, Time=187H263 p
35	6.617526	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28762, Time=202H263 p
36	6.682655	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28763, Time=234H263 p
37	6.713383	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28764, Time=249H263 p
39	6.778877	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28765, Time=281H263 p
40	6.809316	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28766, Time=296H263 p
41	6.842291	172.16.0.90	172.16.0.100	H.263	Payload type=ITU-T H.263, SSRC=5519, Seq=28767, Time=312H263 p

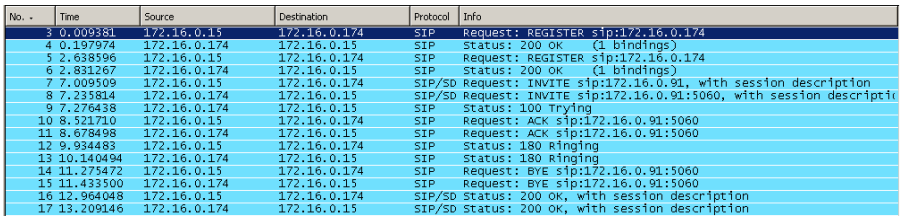
图 6.2 直接会话结果

2） PC 到 ARM 的间接通信

- SmartARM2200 开发板与 PC 机联机，并写入 ArmSipProxy 后脱机运行，使用以太网接口接入网络，开发板 IP 地址设定为 172.16.0.174；
- 在两台 PC 机上安装 SipUA 所需的 K-LiteCodec 编解码器集合，并运行 SipUA；
- 首先点击“REGISTER”，填写本地机信息并以 REGISTER 请求的形式发送给嵌入式服务器，服务器收到后将返回 200/OK 响应，并将用户代理的信息写入用户列表中；
- 两台 PC 机注册后，其中一台运行 Ethereal，开始监听整个网络的数据发送情况；
- 点击其中一个 SipUA 的“INVITE”开始发起会话，将被

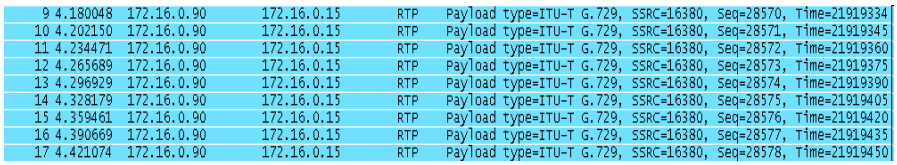
叫方 IP 地址设为另一用户代理所在主机的 IP 地址；

- 出现对方图像，听到对方声音以后按“BYE”发送 BYE 请求，结束多媒体会话；
- 停止 Ethereal 抓包并将抓包结果保存为.pdf 文件。



No.	Time	Source	Destination	Protocol	Info
3	0.009381	172.16.0.15	172.16.0.174	SIP	Request: REGISTER sip:172.16.0.174
4	0.197974	172.16.0.174	172.16.0.15	SIP	Status: 200 OK (1 bindings)
5	2.638596	172.16.0.15	172.16.0.174	SIP	Request: REGISTER sip:172.16.0.174
6	2.831267	172.16.0.174	172.16.0.15	SIP	Status: 200 OK (1 bindings)
7	7.009509	172.16.0.15	172.16.0.174	SIP/SD	Request: INVITE sip:172.16.0.91, with session description
8	7.235814	172.16.0.174	172.16.0.15	SIP/SD	Request: INVITE sip:172.16.0.91:5060, with session descriptic
9	7.276438	172.16.0.174	172.16.0.15	SIP	Status: 100 Trying
10	8.521710	172.16.0.15	172.16.0.174	SIP	Request: ACK sip:172.16.0.91:5060
11	8.678498	172.16.0.174	172.16.0.15	SIP	Request: ACK sip:172.16.0.91:5060
12	9.934483	172.16.0.15	172.16.0.174	SIP	Status: 180 Ringing
13	10.140494	172.16.0.174	172.16.0.15	SIP	Status: 180 Ringing
14	11.275472	172.16.0.15	172.16.0.174	SIP	Request: BYE sip:172.16.0.91:5060
15	11.433500	172.16.0.174	172.16.0.15	SIP	Request: BYE sip:172.16.0.91:5060
16	12.964048	172.16.0.15	172.16.0.174	SIP/SD	Status: 200 OK, with session description
17	13.209146	172.16.0.174	172.16.0.15	SIP/SD	Status: 200 OK, with session description

(a)SIP 会话过程（加 SIP 过滤）



9	4.180048	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28570, Time=21919334
10	4.202150	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28571, Time=21919345
11	4.234471	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28572, Time=21919360
12	4.265689	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28573, Time=21919375
13	4.296929	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28574, Time=21919390
14	4.328179	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28575, Time=21919405
15	4.359461	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28576, Time=21919420
16	4.390669	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28577, Time=21919435
17	4.421074	172.16.0.90	172.16.0.15	RTP	Payload type=ITU-T G.729, SSRC=16380, Seq=28578, Time=21919450

(b)多媒体会话过程（加 RTP 过滤）

图 6.3 带有无状态服务器的间接会话结果

如实验结果所示，SipUA 和 ArmSipProxy 能够正常工作并完成了 PC 到 PC 端得直接会话和通过无状态服务器的 SIP 用户代理间接会话。具体实验数据请参看附录。

结　论

本文介绍了现今最流行的多媒体会话协议—SIP 协议，并论述了其基本结构及工作原理。接下来本文重点介绍了由作者自主研发的 SIP 协议栈 ZmhSipStack，并介绍其设计思想、工作原理及内部结构。然后又给出两个运用 ZmhSipStack 构建 SIP 终端和服务器的实例。最后介绍 ZmhSipStack 的嵌入式移植及其应用。

最终 ZmhSipStack 协议栈实现了以下功能：

- 对常用五种请求消息及三种响应消息进行解析和生成，支持 13 种消息头格式的处理；
- 支持 SDP 格式的消息体解释及生成；
- 支持用户代理状态机，实现通话状态转换；
- 具有 UA 和 SERVER 两种版本，可以运行在 PC 和嵌入式环境下。

应用 ZmhSipStack 可构建如下应用程序：

- 支持多线程状态机的用户代理程序，实现 SIP 直接通信，支持多种音视频编解码格式，并支持媒体数据 RTP 格式打包传输；
- 无状态服务器程序，实现代理服务器和注册服务器功能，实现 SIP 间接通信；
- 嵌入式环境下的用户代理和无状态机服务器程序。

同时，此协议栈还存在以下几点不足：

- 对响应消息尤其是报错消息支持不够；
- 服务器版本尚不支持重定向和状态机功能；
- 嵌入式版本用户代理程序尚未实现音视频传输。

所以，今后工作的重点将集中在增加支持的消息类型，服务器功能的提高以及嵌入式应用程序性能提升上，期望在不久的将来能够使 ZmhSipStack 协议栈真正投入实用。

参考文献

[1] 张友波,张焕强,孙利民. 基于 SIP 的视频会议系统设计与实现.计算机工程.2005 年 11 月,第 31 卷(第 21 期):167~169

[2] 薛宁.SIP 服务器设备技术要求介绍.电信网技术.2006 年 2 月,(第 2 期):21~23

[3] 李军,谢赞福,崔怀林.基于 SIP 的语音通信程序设计与实现.计算机工程.2006 年 1 月,第 32 卷(第 2 期):117~119

[4] 薛晓舟,刘玉璋,尹泽名.在 SIP 会话中重用底层连接.现代电信科技.2005 年 11 月,(第 11 期):46~52

[5] 叶德谦,张树国,孟庆吉.状态服务在 SIP 协议中的应用研究.微计算机信息.2005 年,第 21 卷(第 12-3 期):136~138

[6] 张睿琳,戴青,范作栋,武清芳.基于 SIP 的视频会议中发言权管理的研究与实现微计算机信息.2006 年,第 22 卷(第 1-3 期):65~67

[7] 张俊九.基于 SIP 的视频会议系统研究.邮电设计技术.2006 年,(第 1 期):42~46

[8] Rosenberg J,Schulzrinne H. SIP:Session Initiation Protocol.RFC3261,2002-06

[9] 张智江,张云勇,刘韵洁.SIP 协议及其应用.北京:电子工业出版社,2005 年

[10] 丁展,刘海英.Visual C++网络通信编程实用案例精选.北京:人民邮电出版社,2004 年