VOCAL sip 协议栈分析及其应用

翟奇
北京邮电大学计算机科学与技术学院，北京（100876）
E-mail：zhaiqi163@gmail.com

摘　要：本文研究了一种嵌入式电话终端解决方案中所使用的 VOCAL sip 协议栈，并且对其的结构和重要的代码进行了深入的分析，并且在最后给出了该嵌入式电话智能终端解决方案的简略描述。

关键词：sip，H.323，VOCAL，电话智能终端
中图分类号：TP393

1. 引言

SIP (Session Initiation Protocol)称为会话初始协议[1][4]，是一个与HTTP和SMTP类似的、基于文本的协议，SIP独立于传输层协议和其它会话控制协议，可以与其他协议（如RSVP，RTSP等）一起构建多媒体通信系统如智能家居网络、视频会议[2]等。

H.323 是分组网内与SIP 相互竞争的协议，H.323 协议为ITU-T 制定的标准，SIP协议为IETF 制定的标准，两者均可以完成呼叫建立、释放、补充业务和能力交换等功能。H.323协议采用ISDN 的设计思想，使用Q.931 协议完成呼叫的建立和释放，明显带有电信网可管理性和集中的特征。目前，H.323协议已经在网上得到广泛应用。与SIP 相比，H.323 更为成熟，因此目前我国各运营商的IP 电话网均选用该协议。而SIP协议具有简单、扩展性好以及与现有Internet 应用紧密联系的特点。许多人认为该协议较易实施，近期在世界范围内快速发展，同时SIP 将在第三代移动通信核心网、智能网中得到广泛应用。

本文首先介绍了sip协议的相关内容，然后通过对VOCAL sip协议栈的实现的代码分析，对sip协议作出详细的描述和研究，进而提出了一种嵌入式IP-PHONE智能终端的实现，并对其整体的架构进行了描述。

2. sip 协议简介

SIP协议是由IETF提出的一种用于IP网络多媒体通信的应用层控制协议，其主要功能是建立、修改、终止和管理多媒体会话或呼叫[3][5]。

SIP协议的语法和语义很大程度上借鉴了SMTP和HTTP的实现机制，使用客户/服务器通讯模式,其消息报文基于文本格式。该协议开放灵活，对下层传输平台没有过多要求，既可以采用TCP，也可以采用UDP或其他协议。

SIP协议系统的功能实体有两类:用户代理和网络服务器。用户代理通常指呼叫终端设备,网络服务器则用来提供地址解析、路由寻址和用户定位等功能。Sip系统的组成示意图如图1所示：

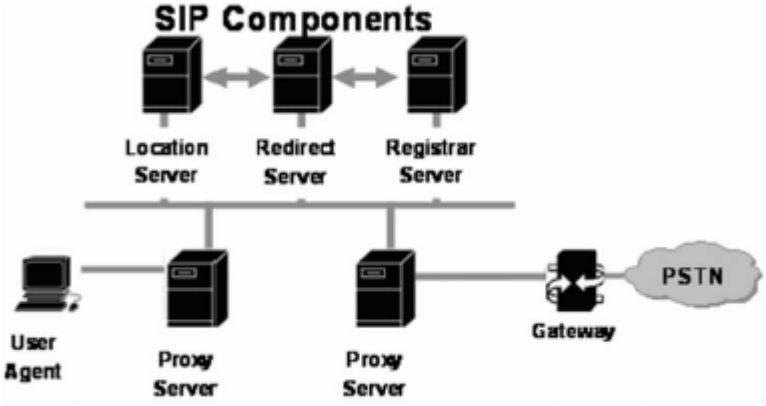

图 1 sip 系统组成

2.1 互操作性和开放性好

SIP的设计思想是对会话进行管理和控制，而不对会话内容进行限制。它支持多种服务且具有媒体协商能力，能够在不同设备之间通过SIP服务器或其他网络服务器进行交互,以提高嵌入式设备的互操作性。同时由于设备遵循的是统一的信令控制标准，因此也有助于提高设备的开放性。

SIP摒弃了传统上为每项业务指定标准的做法,通过扩充协议消息的方法扩展其能力。由于SIP是基于文本方式的,它功能扩展十分方便,用户可根据需要,通过在SIP终端定义新的消息体、新的消息头部或新的消息类型来实现各项业务。同时,SIP很好的模块化设计使得SIP可以方便地和其他负责身份验证、位置信息、语音质量等现有的Internet协议整合在一起以提供不同的服务。

利用SIP的易于扩展的特性,可在嵌入式系统中提供多项服务。如：

①Presence服务　即提供某个用户与其他用户通信的意愿和能力的一种服务。

②即时信息服务　可实现使用不同设备的用户间的相互交流，在嵌入式终端设备中集成该服务，可以方便用户利用单一的硬件工具享受多种通信服务，增强设备对用户的吸引力。

③实时在线应用　随着嵌入式术的发展,实时应用逐渐深入到嵌入式领域，而嵌入式小型灵活的特点又能促使该应用需求的发展。在嵌入式设备上实现在线实时游戏、在线共享应用程序、共享文本编辑、共享白板这些PC上常见的服务,是对嵌入式需求提出的一种新的理念。

2.2 支持移动性

随着Internet和移动通信的发展,提供移动性服务已是嵌入式网络环境下小型设备的研究热点之一。SIP协议中通过代理服务器和重定向请求提供了对个人移动性的支持,通过对其进行合理扩充,还能提供对终端移动性、服务移动性的支持。这完全可满足多媒体通信系统移动性的要求,使得嵌入式系统应用可以集成更多的移动性服务。

SIP沿用了Internet协议的设计原则,采用简单的请求/响应模式和文本消息的形式,可读性强,设计实现简单,有利于开发低成本的多媒体终端。而且对于这种基于文本的、有语义内容的描述行的词法、句法分析比较简单,计算量较小,适合于嵌入式这类计算能力十分宝贵的应用环境。通过以上对SIP在嵌入式应用中的优势分析,可以得出,SIP在嵌入式环境中具有较大的应用潜力。

3. 协议栈结构

这套协议栈的实现不只是包括对sip消息的处理，还有TSP层和UI层，但这些都不是本篇论文的重点，所以我们这里的协议栈的结构只是介绍其中的对sip消息处理的部分。但这里还是把这三层的关系展现出来，见图2：

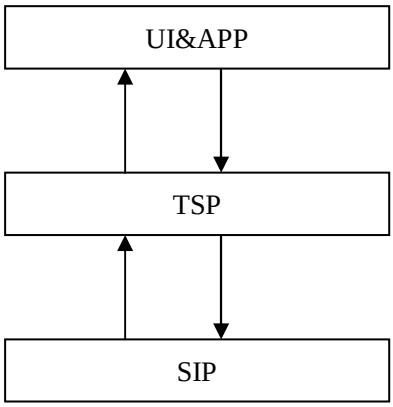

图 2 协议栈层次

UI&APP层是最上层，这里实现了用户操作和终端显示的部分，当APP接收到底下层次的部分送上来的消息，对其进行解析和判断，然后根据不同的信息作出相应的显示和，同样APP也可以接收到来自用户操作的信息，进而将其信息的内容传送到底下的TSP层。这一层是没有状态的（stateless）。

TSP和SIP层都是有状态的层次（stateful），在这两层中有各有一个状态机，分别处理不同状态下收到的不同消息。

我们将要重点介绍的就是最底下的一层-----SIP层。这套代码是用C++实现的，所以其中不论是sip消息的存储还是对消息的处理都是封装在类中的，所以下面的介绍中的名称没有特别说明都是类的名字。

4. 重点代码分析

首先看一个sip协议栈的管理的类，与sip消息相关的处理都是从这个类开始的HeartLessProxy，在介绍这个类之前先看一下相关的内容：

CallContainer----用来记录callInfo的信息，是callInfo的一个容器

Builder----是对收到的sip消息进行建设，也就是说，判断这个消息的类型然后将其放入合适的状态，如果是一个新的call则进行一通电话的状态信息的构建

Event queue----则是一个队列，在其中存储了接收到的消息，起到缓冲的作用

Transceiver----是进行传输的通道，从UDP（TCP）接收和向UDP（TCP）发送sip消息都是通过这个类来进行的

WorkerThread----从event queue中取出event然后处理，处理则是通过builder来进行“建设”的

SipThread----利用transceiver从UDP中取出数据，生成相对应的event然后put入队列

这个类中的private变量包括：

```
Sptr<CallContainer>                    itsCallContainer;
Sptr<Builder>                          itsBuilder;
Sptr<Fifo<Sptr< SipProxyEvent> > >     itsCallProcessingQueue;
Sptr<VovidaSipStack>                   itsSipStack;
Sptr<WorkerThread>                     itsWorkerThread;
Sptr<SipThread>                        itsSipThread;
```

整个协议的实现其实很简单，就是两个线程的启用：itsSipThread和itsWorkerThread。在第一个线程中利用itsSipStack从socket中取出sip消息转化成相对应的sip事件然后放入sip事件的队列（itsCallProcessingQueue）当中，这个队列我们简单说成是sip消息队列；同时在第二个线程中不断的从消息队列中取出sip消息，利用itsBuilder来对sip消息进行处理，如果是一个建立call的消息则生成相应的Call Info信息存储在itsCallContainer里面，如果是在一通call中则要判断是否是已经有的Call Info。

下面具体分析一下sip消息的解析和处理的流程：

状态机的实现------State Machine

Sip文本消息的解析-------Data类

UaBuilder类中的process是用来专门进行sip消息的处理的（包括发送和接收两种消息），在process中我们将从sip消息队列中取得的sip消息首先进行分析进而进行分门别类的分析。对于register的相关的处理和其他的消息处理是不一样的，对于register的相关的消息是不用把它放入到状态机中进行处理的，因为只有在register成功过后，我们的sip电话才能进行正常的sip协议之间的联系。所以，在UaBuilder中我们的成员变量只有两个，一个就是sipStack—用来发送一些status（等功能）另外一个就是register Management----用来进行对register相关消息的处理

一个状态机是和一个call息息相关的，所以每一个sip call要和一个state machine相关联，在sip call中保留有state machine的信息

process中处理两类消息（incoming call和outgoing call），但是包括三类event（sip event，tsp event & timer event），sipEvent---incoming call（因为收到的信息都是从别的UA那里得的sip消息，所以incoming call都是sipEvent）tspEvent & timerEvent---outgoing call（而从UI和APP那里进入sip stack的都要进过TSP所以一般是tspEvent而有些需要计时事件，所以也会有timerEvent）

而无论是什么消息，都要最终进入到状态机中进行处理，每个状态机在每个状态都会有不同的操作来针对不同的消息进行处理。状态机的示意图见图3

下面我们就看看状态机的实现过程：

每一个call都会和一个状态机相关联的，那么用一个类来对状态机进行描述和记录，这个类就是UaStateMachine，这个类继承自Feature类，在这个状态机中用一个链表来存储有限状态机的各个状态，也在其中存储一个指向状态链表中元素的指针。而在这个类中的主要的两个函数就是初始化函数还有就是状态机的process函数，因为第一个函数能够把这个状态机所具有的状态添加进去，而因为第二个函数，则能够使得这个状态机运行良好。

状态机有八大状态在其中，

State----Call Ended

State----Fast Busy

State----Idle

State----In Call

State----Ringing
State----Trying
State----Proceeding
State----Await Ack

前面提到过，每个call是与一个状态机相关联的，所以一个call info中必然要记录一个状态机的当前状态，这样才能够使这个状态机成为有限状态机，因为它是可控的。

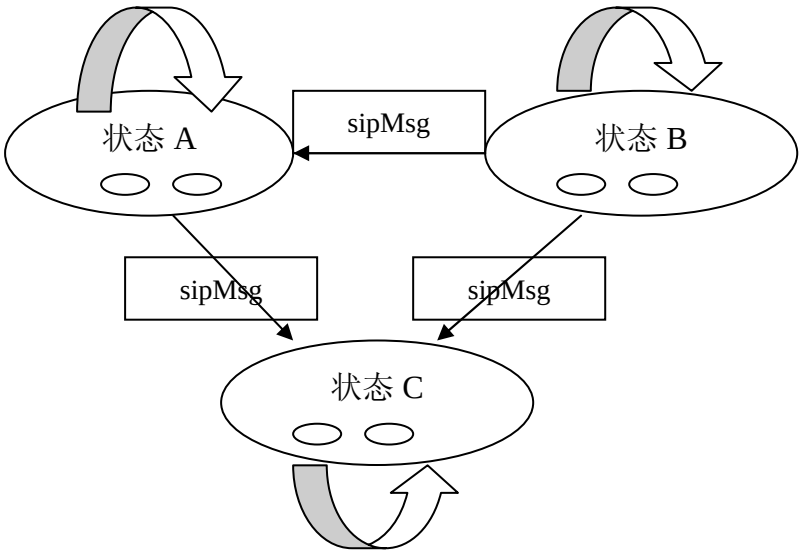

图 3　sip 状态机

大椭圆表示是我们状态机中的各种不同状态，这里为了简化只画出其三个作为代表，大椭圆中的小椭圆则是各个状态中处理不同消息的操作，A, B, C中的操作可能有交集，也就是说可能有相同的操作，因为不同的状态会处理相同的消息。

这样的话，在call中会得到它的当前的状态，然后利用当前状态的process函数，就可以处理得到的sip消息了，

上面介绍的只是一个状态机的总体的架构，而状态机中每个状态对sip消息的处理则是最重要也是最复杂的。下面来详细的介绍状态中的操作：

每个状态有三个链表：1，基本操作的链表，2，在这个状态进入之前必须进行的操作的链表，3，这个状态的基本操作都完成后进行的操作的链表

链表1就是普通的操作链表，链表2则是在前一个状态的处理循环结束后进行的操作，这样处理是为了在下一个状态能够都执行这个链表中的操作，而链表3中的操作则是在链表1中的操作完成后进行的操作。

具体的操作通过以下最简单的通话过程来进行描述（示意见图4）：（我们这一方A, 通话的对方是B）

- B----→A

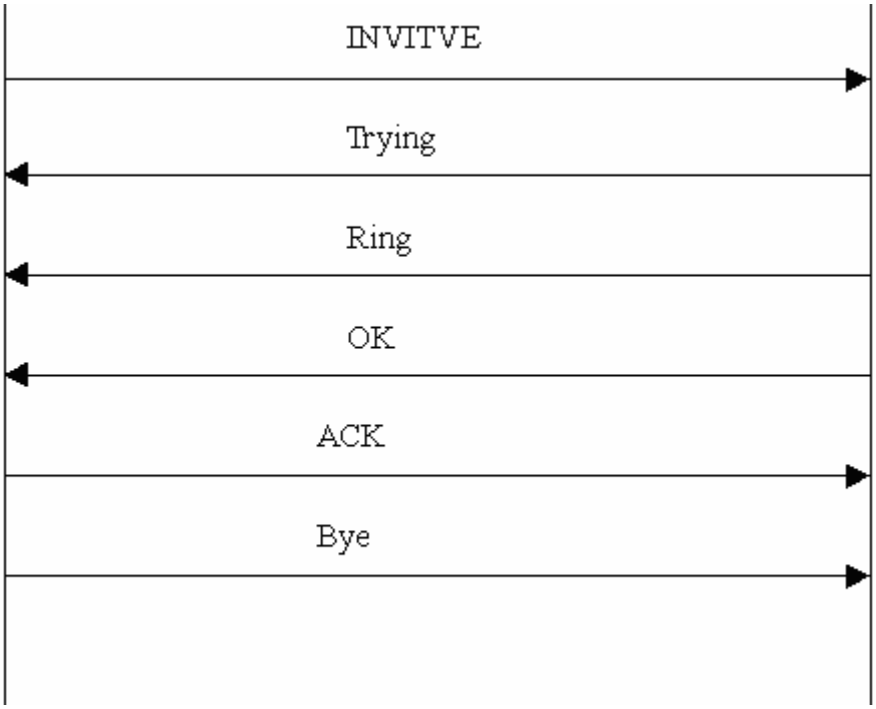

图 4 简单通话建立及结束过程

两个UA都注册到proxy上后进行的操作，还有一种情况就是可以进行peer←-→peer的操作不需要进行注册服务器进行注册操作，

socket中收到了INVITE消息后，在对这个消息进行存储和解析后，放入消息队列，然后worker thread取出这个消息，在UaBuilder的process中进行处理。

然后将这个消息放入状态机中进行相应的处理，这时候如果我们的电话终端处于空闲状态，也就是说状态机处于IDLE状态的话，在这个状态中我们会有以下几种相对于不同sip消息的操作：

OpStartInvite ---用来向对方发起通话请求

OpStartIncoming ----用来接受对方发起的通话请求

操作中处理的其实都是各种事件，sip消息对应的是sip事件，而从TSP向sip stack中放入的事件就叫做TSP事件，在OpStartInvite的process中，只是用来向对方发起通话请求，那么处理的都是TSP事件，所以如果是sip事件的话则会结束处理。所以在这个状态中，就只有OpStartIncoming能够处理sip的事件，而事实上也确实是在这个类的process中处理的INVITE消息，在这个处理中我们保存INVITE消息中携带的sdp信息，以及为这个call保存第一个INVITE消息，也就是这个消息本身，然后向对方发送100 tying status msg告诉对方收到INVITE防止对方重发INVITE

然后状态转入到Proceeding状态，处理的原则同上，在这个状态中我们处理的是上层给协议栈发送的事件，如果我们的终端振铃那么我们收到的就是ring request的消息事件，则进行相应处理的就是OpRingCall，在这个操作中我们向对方发送我们的振铃信息180 ring status msg，然后状态转入Ringing，之后如果我们进行摘机的动作那么就会在这个状态中进行OpAnswerCall操作，向对方发送200 OK status msg，然后等待对方发送ACK确认信息。如果收到相应的ACK信息那么这个通话就确定建立，然后双方通过RTP等协议进行语音或是视频的交流。

5. 协议栈的应用

本文介绍的协议栈已经被成功的运用到了一种嵌入式智能终端的应用当中，在这个应用中除了能够实现普通电话的功能外，还能实现一些普通电话所不具有的特性和服务，如：BLF当然这需要PBX的支持。

6. 小结

本文除了对sip协议的前景和发展进行了分析，还对VOCAL sip协议栈进行了分析和研究，主要是对sip消息的解析和处理进行了详细的描述和介绍。

随着SIP 扩展协议对SIP 核心的逐渐完善, SIP 将会发挥越来越重要的作用。同时3GPP(第三代移动通信伙伴项目)、PacketCable(便携线缆设备)等研究机构以及微软、3Com、Cisco、朗讯等一些大型国际企业将SIP 作为工作协议极大地促进了SIP 标准的进一步发展, 而VOIP、多媒体会议、push—to—talk(按键通话)、定位服务、在线信息和IM服务等领域逐步采用SIP 协议进行实现, 标志着SIP 已经成为一个应用范围广泛、逐步走向成熟的协议。目前, SIP 协议已经成为下一代网络中软交换的核心协议之一, 相信随着SIP 相关标准的进一步完善以及国内外对SIP 应用研究的进一步深入, SIP 的发展前景将无限广阔。

本文介绍的协议栈的实现已经成功运用到了实践中，文中最后提到的嵌入式智能终端就是其实现的代表。

参考文献

[1] 储泰山，基于 SIP 的服务器的研究与实现[D]，浙江大学硕士学位论文，2004.3。
[2] 叶德谦，基于 SIP 集中式多媒体视频会议系统中对私下会议问题的研究[J]，微计算机信息 2006.1-3P78-79、P268。Douglas C.Schmidt，Stephen D.Huston．C++网络编程[M]．武汉：华中科技大学出版社，2003．
[3] Gonzalo Camarillo． SIP 揭秘[M]． 北京：人民邮电出版社，2003．
[4] Garcia-Martin M, Henrikson E, Mills D. Private header (P-Header) extensions to the session initiation protocol (SIP) for the 3rd-generation partnership project(3GPP)[S]. Internet RFC3455, 2003.
[5] RFC3261．SIP: Session Initiation Protocol[S]．www.ietf.org：IETF，2002．

Description and Application of VOCAL Sip Stack

Zhai Qi
Beijing University of Posts and Telecommunications，Beijing (100876)

Abstract

This article describes the VOCAL Sip Stack and instructs it’s one application. The Vovida Open Communication Application Library (VOCAL) is an open source project targeted at facilitating the adoption of VoIP in the marketplace. VOCAL provides the development community with software and tools needed to build new and exciting VoIP features, applications and services. The software in VOCAL includes a SIP based Redirect Server, Feature Server, Provisioning Server and Marshal Proxy.

Keywords：sip，H.323，VOCAL，Intelligent Terminal