单位代码：10293 密　级：公开

南京郵電大學

专 业 学 位 硕 士 论 文

论文题目：VoIP 紧急呼叫系统及技术实现

学　　号	1210012203
姓　　名	黄 励 博
导　　师	糜正琨 教授
专业学位类别	工程硕士
类　　型	全　　日　　制
专业（领域）	下一代网络
论文提交日期	二零一三年三月

VoIP Emergency Call System and Its Implementation Technology

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Engineering

By

Libo Huang

Supervisor: Prof. Zheng-Kun Mi

March 2013

南京邮电大学学位论文原创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京邮电大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

本人学位论文及涉及相关资料若有不实，愿意承担一切相关的法律责任。

研究生签名：_____ 日期：_____

南京邮电大学学位论文使用授权声明

本人授权南京邮电大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档；允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索；可以采用影印、缩印或扫描等复制手段保存、汇编本学位论文。本文电子文档的内容和纸质论文的内容相一致。论文的公布（包括刊登）授权南京邮电大学研究生院办理。

涉密学位论文在解密后适用本授权书。

研究生签名：_____ 导师签名：_____ 日期：_____

摘要

在 VoIP 业务大量应用的情况下，基于 VoIP 的紧急呼叫已成为通信网亟待解决的问题，它涉及终端、服务器、协议、网络、业务等多个层面的配合，其中最重要的是必须支持基于 SIP 协议的 VoIP 紧急呼叫。

论文在调研分析相关技术发展动态和最新国际标准的基础上，参照 ECRIT 紧急呼叫系统架构，深入研究基于 SIP 的 VoIP 紧急呼叫技术及其实现。

论文首先概述 VoIP 紧急呼叫的研究背景，简要介绍了 ECRIT 紧急呼叫系统架构和 VoIP 紧急呼叫的关键技术。然后，重点研究基于 SIP 的 VoIP 紧急呼叫，在概述 VoIP 技术和 SIP 协议的基础上，给出了 VoIP 紧急呼叫系统设计与实现方案。论文设计了紧急呼叫消息结构和消息编解码方式，提出了 VoIP 紧急呼叫控制系统的实现方案，给出了 SIP 协议扩展功能设计、客户端和服务端各功能模块的详细设计，并完成部分模块的 C 语言编程设计。

论文还研究了 VoIP 紧急呼叫系统实现的若干关键技术，包括用户位置信息配置、紧急呼叫号码的自动获取、PSAP 内部处理、紧急呼叫补充信息、数据紧急呼叫、去活呼叫转移功能、去活呼叫等待等技术。

论文最后对笔者开发的 VoIP 紧急呼叫原型系统软件进行了功能测试，测试结果表明，系统设计合理，功能正确，达到了设计要求。

关键词： 紧急呼叫，VoIP ，定位信息，SIP

Abstract

Among a large number of VoIP service applications,VoIP-based emergency call has become an urgent problem of communication networks.It involves the co-ordination at various levels of the user terminal, network protocols and provider service.The most important concern is the support of SIP-based VoIP emergency call.

Based on the study and analysis of relevant technological advance and the latest international standards,the dissertation makes an in-depth study on software design and implementation technology of SIP-based VoIP emergency call under the ECRIT emergency call system architecture.

First of all, The dissertation outlines the background of VoIP emergency call and briefly introduces the key technologies of the ECRIT emergency call system architecture and VoIP emergency call.Then, focus is devoted on the SIP-based VoIP emergency call. Based on an overview of VoIP technology and SIP protocol, the dissertation gives the design and implementation of VoIP emergency call system.

It designs the emergency call message structure, message coding and parsing functions and proposes the implementation of the control system of VoIP emergency call system. Detailed design is given of the SIP protocol extended functions, functional modules of the client and the server together with the completed C language coding.

The dissertation also studies some key technologies for VoIP emergency call system implementation, including location information configuration, automatic access to the emergency call number, call processing at the PSAP, data-only emergency call， emergency call supplementary information, inactivation of call waiting and call forwarding features during emergency call.

Finally, the dissertation runs the functional testing on the software of the author developed VoIP emergency call prototype system. The testing results have shown that the system meets the the design requirements with reasonable design and correct functions.

Key words: Emergency call , VoIP, Location information, SIP

目录

缩略语表

CCSA	China Communications Standards Association	中国通信标准化协会
DHCP	Dynamic Host Configuration Protocol	动态主机配置协议
E-VoIP	VoIP Emergency Call	VoIP 紧急呼叫
ECRIT	Emergency Context Resolution with Internet Technologies	基于互联网技术的紧急环境解析工作组
GEOPRIV	Geographic Location/Privacy	地理位置/隐私工作组
HELD	HTTP-Enabled Location Delivery	HTTP 使能位置发送
HTTP	Hypertext Transfer Protocol	超文本传输协议
IETF	Internet Engineering Task Force	互联网工程任务组
IM	Instant Messaging	即时消息
LoST	Location-to-Service Translation	位置至服务中心翻译
NENA	National Emergency Number Association	国家紧急呼叫号码协会
PASP	Public Safety Answering Point	公共安全应答点
PIDF	Presence Information Data Format	呈现信息数据格式
PIDF-LO	The Presence Information Data Format Location Object	状态信息数据格式位置对象
PSTN	Public Switched Telephone Network	公众交换电话网
QoS	Quality of Service	服务质量
RAS	Registration， Admission， and Status	登记接纳和状态协议
RTCP	RTP Control Protocol	RTP 控制协议
RTP	Real-time Transport Protocol	实时传送协议
RTSP	Real Time Streaming Protocol	实时流协议
SIP	Session Initiation Protocol	会话启动协议
UAC	User Agent Client	用户代理客户机
UAS	User Agent Server	用户代理服务器
URI	Uniform Resource Identifier	统一资源标识符
URN	Uniform Resource Name	统一资源名
VoIP	Voice over Internet Protocol	IP 语音业务

第一章 引言

1.1 课题背景

VoIP 是指通过 Internet 网络，以 IP 协议作为语音数据的载体，以分组(Packet)的形式传输数字化的语音信号[1]。VoIP 技术由于比传统语音业务更广泛、更便捷，成本更低，在二十世纪九十年代开始，就在互联网上得到迅速地发展与运用。VoIP 技术主要在 IP 网络之间或者 IP 网络和 PSTN 网络之间传输语音数据，由语音编码技术、语音解码技术、信令控制技术、QoS 技术、实时传输技术和语音压缩技术等组成[2][3]。

紧急通信是当前电话网络的一项重要功能。紧急通信是指在出现突发性紧急情况时，为了防止生命财产受到危险，利用通信资源与紧急呼叫中心进行必要通信[4]。反应迅速是其基本要求，此外紧急通信还面临随机性、安全性等问题[5]。传统 PSTN 用户进行紧急呼叫时，PSTN 终端根据 PSTN 号就可以定位用户的物理位置，从而把数据提供给呼叫中心[6]。随着反恐、医疗紧急救助、火警和自然灾害救援等紧急呼叫业务重要性的不断提高，对于 VoIP 紧急呼叫的需求正在不断提升。

在 Internet 上，带宽有限，服务质量(QoS)远远比不上传统 PSTN。通过 VoIP 拨打紧急呼叫也远没有通过 PSTN 呼叫方便、可靠[7][8]。而且在一个 VoIP 系统中，紧急呼叫号码可以在 IP 网络的任何位置使用，在拨打紧急呼叫的同时，系统无法提供其所处的确切位置。正因为如此，研究和实现一种支持基于 VoIP 的语音、文本、实时多媒体通信的紧急呼叫的同时，能够迅速反映物理位置并且保证一定的 QoS 是非常必要的。

1.2 国际研究动态

随着 Internet 上 VoIP 业务的大量开展，IETF 任务组开始日益重视互联网上的应急通信问题，建立了 ECRIT 工作组，专门研究 Internet 的应急通信问题。

2004 年，ECRIT 工作组扩展了 DHCP 协议，加入基于坐标位置的配置信息[9]。

2008 年，ECRIT 工作组为紧急呼叫和其他未知服务确定了统一资源名称（URN）并且标准化了有关位置信息的统一文件类型(PIDF-LO)；设计了客户端位置到服务中心位置的 srsName 协议(LoST)；制定了一个互联网标准追踪协议，通过 DHCP 协议发现 LoST 服务器[10]-[14]。

2009 年，给出了 PIDF-LO 的用途、注意事项和相关建议[15][16]。

2010 年，制定了通过 HTTP 的方式进行位置交付的协议（HELD）。设计了发现本地位置信息服务器的标准[17]-[19]。

2011 年，完成了多媒体紧急呼叫标准的草稿[20]-[23]。

以美国和欧洲交通管理部门为首的相关机构提出了面向全IP时代的下一代紧急通信系统的构想。它的先进性体现在：不仅能够承载语音，还能够用文字、语音和图像等多媒体形式更好地、自动地为紧急救援服务；PSAP 的位置将变得不再重要，呼叫将根据地理坐标自动进行路由，并且系统可以支持转接回退等功能；可以利用 IP 路由技术更好地处理链路拥塞问题。在实现这个目标之前，NENA 协会和 ECRIT 工作组都已经提出了针对已经广泛商用的 VoIP 业务的紧急呼叫服务模型。在这些不同的网络中用来确定终端位置的技术差别很大，例如可能涉及二层和三层 IP 技术、无线网络中的定位技术、甚至包括无线自组织网络中的定位技术。

在没有提出紧急通信标准体系标准之前，我国已经存在了一些实际上与紧急通信相关的技术研究，其中包括定位系统、地理信息系统、卫星通信、传统通信、微波通信、视频会议和视频监控等。CCSA 从 2004 年开始了紧急通信标准的相关研究，在 2009 年成立了紧急通信特设任务组，主要工作包括制定紧急通信标准体系、满足特定业务的紧急呼叫、分析传统通信网对紧急通信的基本要求等等。原有领域那些与紧急呼叫相关的标准都可以应用到现在的紧急通信标准体系中，当紧急通信需要那些技术时，原有技术先需要符合原有领域标准，然后根据紧急通信系统的需求，增加新技术要求[24]。

在中国长途电话领域 VoIP 已经占据了第一的位置，数据显示，2011 年，通过移动电话终端发起的 IP 电话所占比重从上年的 66.2%上升到 70.5%。但我国的 VoIP 技术与传统 PSTN 电话相比只是完成了语音呼叫，而不能实现紧急呼叫系统的要求。如今 VoIP 技术作为一种越来越普遍使用的通信技术，未来要获得更广泛地发展，就应当和 PSTN 技术一样在语音通话的基础之上完成一些基本的附加功能[25]，其中特别重要的功能包括，用户位置的定位、发现紧急呼叫中心的发现、基于位置的紧急呼叫实现和紧急呼叫中心的内部处理等。我国的 VoIP 技术在未来可以通过市场和技术手段满足这些需求，弥补紧急通信方面所存在的欠缺。

国内有关这方面的研究较少，国外推出了紧急呼叫架构，但具体的实现做得较少。

1.3 论文主要内容

紧急通信可以分为三类[2]：

（1） 自下而上。个人用户到政府或者民间组织机构的紧急通信需求

个人用户遇到特定的紧急情况，比如急救、报警等，拨打紧急呼叫号码。

（2） 政府或者民间组织机构之间的紧急通信需求

在各种紧急事件发生时，比如火灾、地震等，政府或者民间组织之间需要完成信息分享，综合调度、实时传输和现场指挥等工作。

（3） 自上而下。政府或者民间组织机构到个人用户的紧急通信需求

政府或者民间组织机构通过移动互联网、短信、语音电话等方式通知个人用户，分享紧急信息，告警安抚，或者是组织机构主动呼叫已发起紧急呼叫的个人用户。

论文主要研究基于 SIP 的第一类 VoIP 紧急通信，即自下而上的方式。论文第二章首先简要介绍用于 VoIP 的 SIP 协议技术，包括 VoIP 的基本结构与主要技术，SIP 协议体系结构和 SIP 呼叫控制机制。第三章介绍 ECRIT 紧急呼叫系统架构，研究其关键技术，包括用户位置描述、用户位置获取、紧急呼叫中心发现。第四章阐述 VoIP 紧急呼叫系统设计，设计了基于 SIP 的紧急呼叫流程、紧急呼叫 SIP 消息结构，给出了 SIP 协议扩展功能设计，并在此基础上研究了 VoIP 紧急呼叫的几个必要技术，包括：用户位置配置的设计、紧急呼叫号码的自动获取、PSAP 内部处理设计、基于数据的紧急呼叫设计，携带补充信息的紧急呼叫设计。第五章主要对 VoIP 紧急呼叫系统进行代码实现，编程实现 SIP 消息的编码与解码，设计了紧急呼叫的客户端和服务端，并完成了位置配置模块、紧急呼叫中心发现模块和 PSAP 内部处理的 C 语言编程设计，最后完成了去活紧急呼叫中的呼叫等待功能，去活紧急呼叫中的呼叫保持功能。

第二章 基于 SIP 的 VoIP 技术

2.1 VoIP 技术

2.1.1 VoIP 基本结构

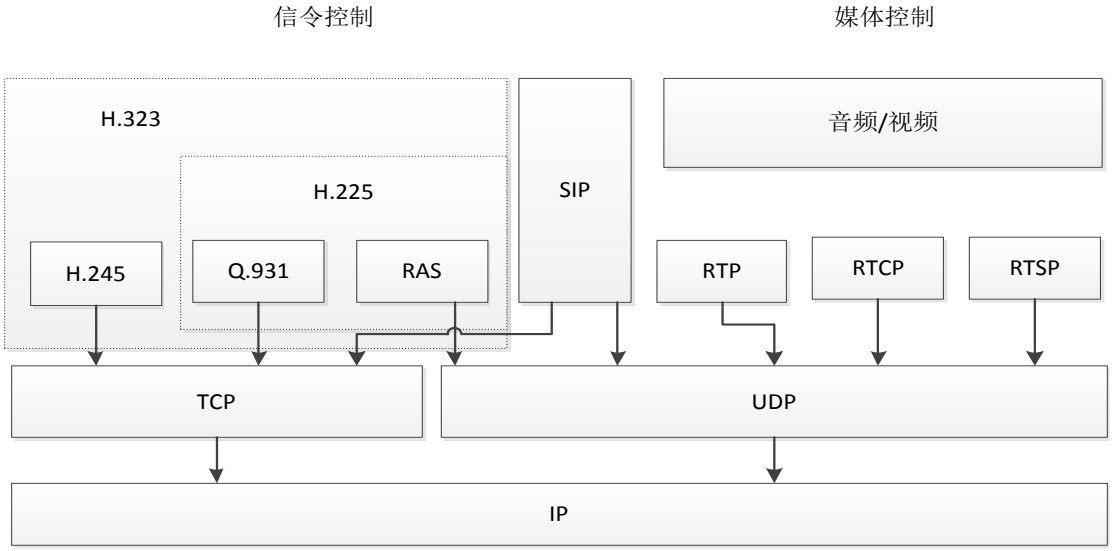


图 2.1 VOIP 协议结构

如图 2.1 所示，VoIP 的支持协议包括两大类：一类是媒体控制协议，另一类是信令控制协议。其中媒体控制协议包括实时传输协议 RTP、实时传输控制协议 RTCP 和实时流传输协议 RTSP，它的传输层承载协议通常是不可靠传送协议，例如 UDP 协议。信令控制协议包括 H.323 协议和 SIP 协议，它的传输层承载协议采用可靠传送协议（例如 TCP 协议）与不可靠传送协议（例如 UDP 协议）均可，但通常情况下，使用可靠传送协议来承载信令控制协议[26]。

2.1.2 VoIP 基本技术

（1） 信令控制协议

（a） H．323 协议簇

H．323 协议簇标准适用业务是包括语音、数据和视频及其组合的多媒体通信[3]。

如表 2.1 所示，H.225.0 协议主要用于呼叫控制，H.245 协议主要用于用于媒体信道控制。两者都是 H.323 协议簇中的核心协议。远程访问控制协议 RAS 主要作用是给网络管理点提供确定端点地址和状态，实施呼叫接纳控制等功能[27]。

表 2.1 H.323 协议结构

<table>
<tr><td colspan="2">声像应用</td><td colspan="3">终端控制和管理</td><td>数据应用</td></tr>
<tr><td>G.7××</td><td>H.26×</td><td rowspan="3">RTCP</td><td rowspan="3">H.225.0 终端至网守信令（RAS）</td><td rowspan="3">H.225.0 呼叫信令</td><td rowspan="3">H.245 媒体信道控制</td><td rowspan="3">T.120 系列</td></tr>
<tr><td colspan="2">加密</td></tr>
<tr><td colspan="2">RTP</td></tr>
<tr><td colspan="4">不可靠传输协议（UDP 协议）</td><td colspan="3">可靠传输协议（TCP 协议）</td></tr>
<tr><td colspan="7">网络层</td></tr>
<tr><td colspan="7">链路层</td></tr>
<tr><td colspan="7">物理层</td></tr>
</table>

（b） SIP（会话初始化协议）

SIP 协议主要负责完成呼叫控制信令的传送，其作用类似于 H.323 协议簇中 H.225.0 协议。但是不同的是，SIP 并不是专门为 IP 电信业务而设计的新协议，而是一个通用的会话控制建立协议。协议设计思想基于文本，语法简单，易于扩展，这点和 Internet 常用协议（例如 HTTP 等）一致。

SDP 协议主要用于描述媒体信道的类型和属性，其作用类似于 H.323 协议簇中 H.245 协议。SDP 媒体描述是基于基于文本的，与 SIP 协议设计思想一致，可以作为消息体结构嵌入 SIP 协议中，采用的是呼叫控制和连接控制结合的方法，有利于呼叫的快速建立。

（2） 媒体控制协议

RTP（实时传输协议）为音频、视频等实时信号数据提供端到端的传递服务，是一种针对多媒体数据流的传输协议。

RTP 协议用于封装实时数据，其头部格式如表 2.2 所示。

表 2.2 RTP 分组固定头部格式

<table>
<tr><td>V</td><td>P</td><td>X</td><td>CC</td><td>M</td><td>PT</td><td>序号</td></tr>
<tr><td colspan="7">时戳</td></tr>
<tr><td colspan="7">同步源（SSRC）标识</td></tr>
<tr><td colspan="7">分信源（CSRC）标识（0～15 个）</td></tr>
</table>

其中 V 是版本号，指示 RTP 版本号。P 为填充指示比特，如果 P 比特为 1 表示分组结尾会有一个或者多个填充字节。X 是扩展指示比特，当它为 1 的时候，则在固定 RTP 头部后还有一个扩展头部。CC 是分信源（CSRC）计数。M 是标识位，标识数据流中的事件。PT 为净荷类型，指明了 RTP 包的负载类型。序号表示每发送一个 RTP 包，它的值加 1。时间戳是数据包的第一个字节的采样时刻，和序号一样，时间戳的初始值也是随机数。同步源标识的值随机选择，保证一个会话中任意两个同步源的标识都不同，这样接收方就可以根据 SSRC 标识来对数据包进行分组。分信源标识用于标识各个组成部分信号的信源。

（3） 语音处理技术

话音激活检测静音抑制技术 VAD 算法，主要对通话过程中的输入信号进行如下分类：语音、静音和背景噪音[28]。语音和背景噪音如果出现误判，要么降低了传输速率（把背景噪声判成语音），要么语音被系统地减短了，降低了话音质量（把语音判成背景噪声）。所以 VAD 算法的好坏直接影响通话的语音质量。其中舒适噪音生成技术 CNG 算法，主要是在通话过程中用来为电话通信产生背景噪声。

回声消除技术。与传统电话相比，IP 电话语音由于延时的存在，造成回声，影响通话质量。其中延时包括以下几个部分：

（a） 累计时延。这是由于编码器对语音进行采样处理而产生的，因此也常称为算法时延。

（b） 处理时延。这是由于编码过程（即编码时延）及发送至整个分组网络时（传输时延）所产生的。

（c） 网络时延。包括数字化语音信号、压缩语音信号、分组、网络排队、网络串行时延等等

2.2 SIP 协议

2.2.1 概述

国际上研究 VoIP 系统有两大系统。一个是 ITU-T 体系，制定了 H.323 标准[29]。另一个是 IETF 体系，它没有指定新的协议标准，而是利用已有的基础协议：SIP 协议。她们都对信令控制提出了一个完整的解决方案，本文采用 SIP 协议。

首先， H.323 采用基于 ASN.1 和压缩编码规则的二进制方法表示其消息，而 SIP 是一个基于文本的协议，它的代码生成和语法解析相比较而言都比较简单。

其次，H.323 采用传统电话信令的模式，实现集中、层次控制，尽管易于管理，易于计费，而 Internet 是一个客户机/服务器模式、水平控制的网络，SIP 采用分布式呼叫模式，这样以现有的 Internet 为基础来构架 IP 电话业务网，适用于多点控制场景。

最后，H.323 补充业务需要定义专门的协议，而 SIP 头部易于识别，易于扩展。

2.2.2 SIP 协议体系结构

SIP 系统采用 Internet 网络中常用的客户机/服务器（C/S）模式，客户机向服务器发送请求而与服务器建立连接，服务器响应客户机发来的请求提供服务，并向客户机回送应答。SIP 协议本身也是一个 C/S 协议[30]。

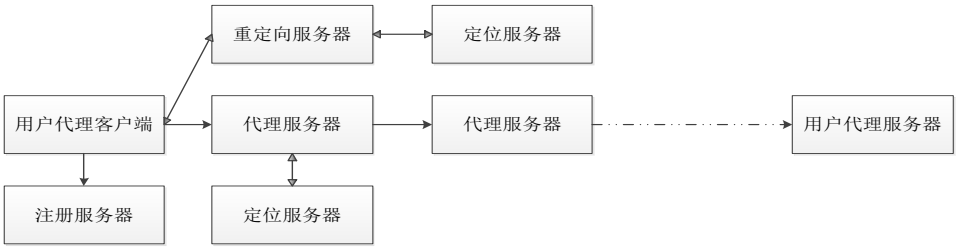


图 2.2 SIP 系统网络结构

SIP 系统网络结构如图 2.2 所示，在 SIP 体系中有以下几类服务器[31]：

（1） 代理服务器（proxy server）：它收到 SIP 请求之后将其转发给下一跳服务器，它在转发之前有可能改写消息的内容，其中 SIP 请求可以经过多跳代理服务器。

（2） 重定向服务器（redirect server）：它负责接收客户机发来的请求，通过响应告知客户机下一跳的服务器地址，然后客户机可以向此服务器地址发送 SIP 请求。

（3） 注册服务器（register server）：它负责接收客户机发来的注册消息，响应注册请求，完成客户机的注册过程

由于用户 SIP 终端既能够发出呼叫也能接受呼叫，因此一个 SIP 端系统，需要包括用户代理客户机 UAC 和用户代理服务器 UAS。

2.2.3 SIP 呼叫控制机制

SIP 协议支持以下三种呼叫方式[32]：

（1） 由呼叫方 UAC 向接收方 UAS 直接呼叫

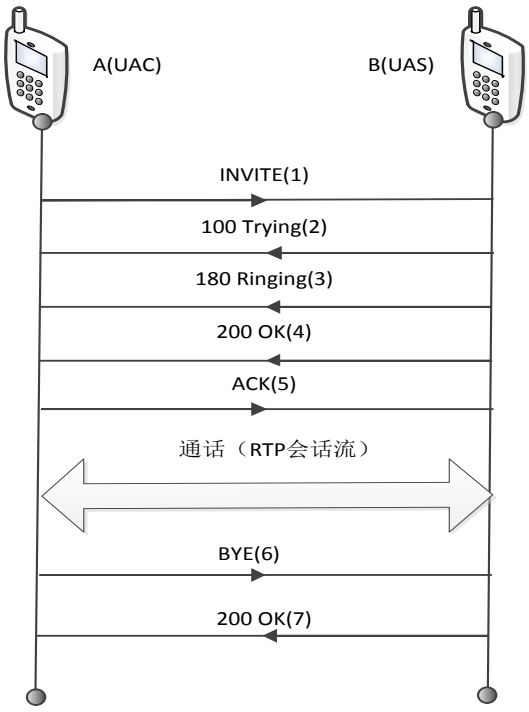


图 2.3 SIP 直接呼叫流程

图 2.3 给出了最基本的两方呼叫建立和释放的 SIP 控制流程（上述流程忽略了代理服务器的寻址转发等）。其主要步骤简要概括如下：

（a） 呼叫方 A(UAC)向被叫方 B(UAS)发送 INVITE 请求，其消息体中包含 SDP 描述，给出了呼叫方 A 要求的编码、媒体类型、接收地址及端口号。

（b） 被叫方 B 回 100 Trying 响应，说明呼叫正在处理

（c） 被叫方 B 回 180 Ringing 响应，说明被叫空闲。此消息体内包含 SDP 描述，给出被叫方 B 要求的编码、媒体类型、接收地址及端口号。从此完成呼叫双方的媒体信道协商。

（d） 被叫方 B 回 200 OK 响应，说明被叫已经应答

（e） 呼叫方 A 回 ACK 响应，说明主叫方收到被叫方的 200 OK 响应。呼叫双方开始正式通信。

（f） 呼叫方 A 决定结束通信，向被叫方 B 发送 BYE 请求。

（g） 被叫方 B 回 200 OK 响应，呼叫正式结束

（2） 由代理服务器代表呼叫方 UAC 向接收方 UAS 发起呼叫

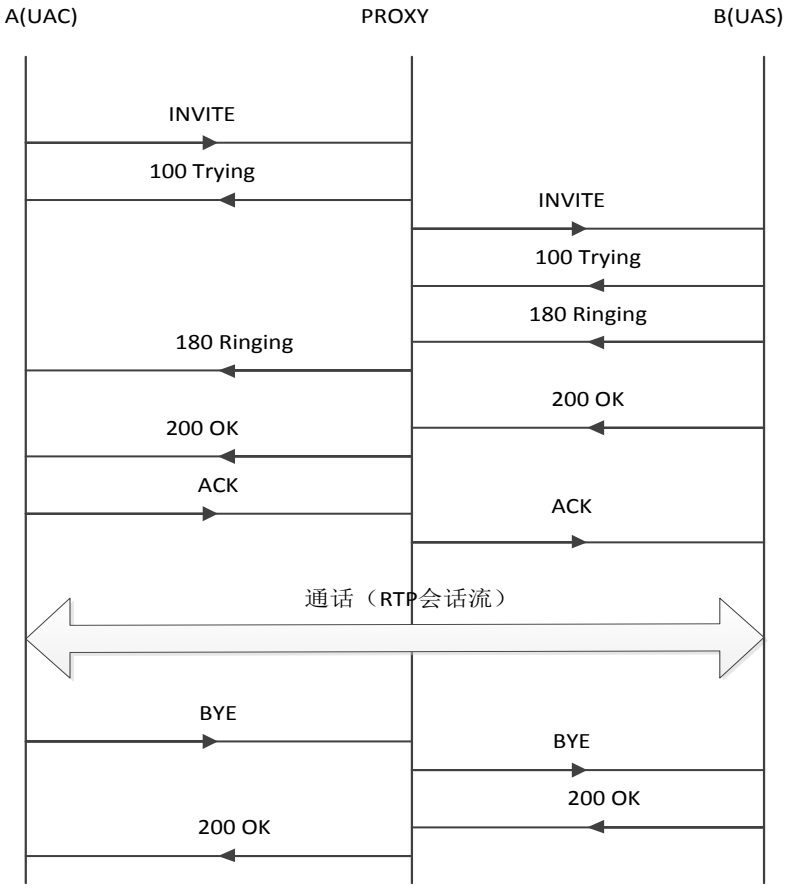


图 2.4 通过 proxy 的 SIP 呼叫控制流程

如图 2.4 所示，其主要步骤简要概括如下：

（a） 呼叫方 A(UAC)向 SIP 代理服务器发送 INVITE 请求

（b） SIP 代理服务器回 100 Trying 响应，说明呼叫正在处理

（c） SIP 代理服务器向被叫方 B（UAS）发送呼叫，建立请求

（d） 被叫方 B 回 100 Trying 响应，说明呼叫正在处理

（e） 被叫方 B 回 180 Ringing 响应，说明被叫空闲。

（f） SIP 代理服务器向呼叫方 A 发送 180 Ringing 响应

（g） 被叫方 B 回 200 OK 响应，说明被叫已经应答

（h） SIP 代理服务器向呼叫方 A 发送 200 OK 响应

（i） 呼叫方 A 回 ACK 响应，说明主叫方收到被叫方的 200 OK 响应。

（j） SIP 代理服务器向向被叫方 B（UAS）发送 ACK 响应。呼叫双方开始正式通信。

（k） 呼叫方 A 决定结束通信，向 SIP 代理服务器发送 BYE 请求。

（l） SIP 代理服务器向被叫方 B（UAS）发送 BYE 请求，请求结束。

（m） 被叫方 B 向 SIP 代理服务器回 200 OK 响应

（n） SIP 代理服务器向回呼叫方 A200 OK，呼叫正式结束

（3） 在重定向服务器的帮助下，呼叫方 UAC 进行重定向呼叫

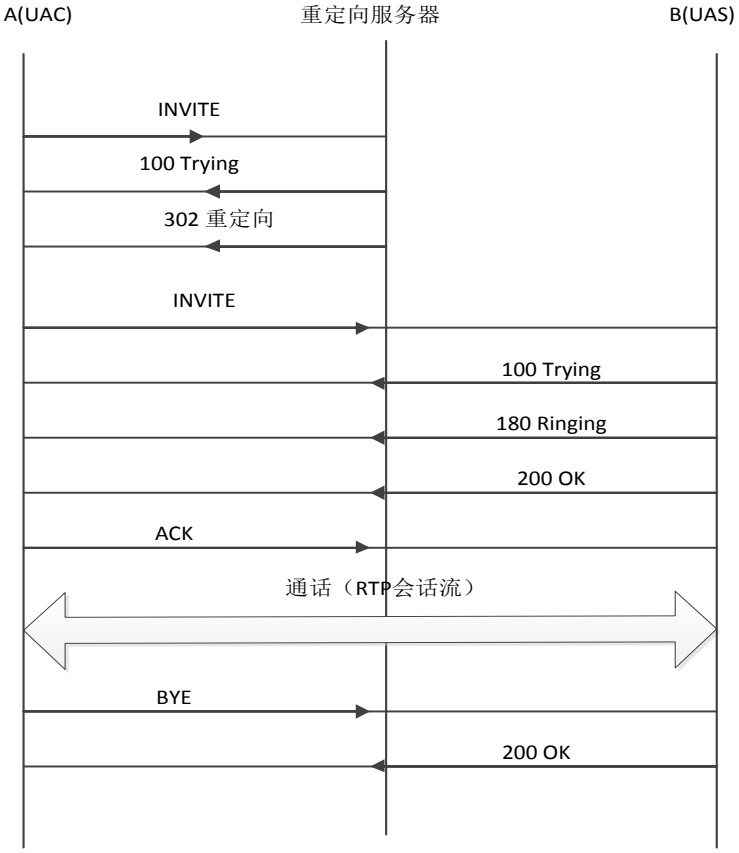


图 2.5 SIP 重定向呼叫控制流程

如图 2.5 所示，其主要步骤简要概括如下：

（a） 呼叫方 A 向 SIP 重定向服务器发送呼叫，建立请求（INVITE）。

（b） SIP 重定向服务器返回重定向消息

（c） 呼叫方 A 从重定向消息中获取被叫方 B 信息，并向其发送呼叫，建立请求（INVITE）

（d） 接下去的步骤和第一种类型的呼叫方式相同，这里就不在赘述。

论文中只实现第一种类型的呼叫方式来实现 SIP 的紧急呼叫。

2.3 本章小结

本章第一部分主要介绍了 VoIP 技术，包括 VoIP 的基本原理和技术，其中有 VoIP 协议结构，VoIP 信令协议标准，媒体控制协议标准和语音处理技术。本章第二部分主要介绍了 SIP 协议的体系结构，重点描述了 SIP 呼叫控制的三种呼叫方式，并给出了具体的控制流程。

第三章 ECRIT 紧急呼叫系统

经过多年的研究和协同工作，ECRIT 工作组对 VoIP 紧急呼叫的必要组件都做了标准化，还有一些细节还没有确定下来。在 IETF 中，一些不同的工作组（ECRIT、GEOPRIV 和在更小范围内的 SIPCODE）围绕紧急呼叫的主题开发了面向下一代紧急呼叫的框架[33]。因为 IETF 提出的 VoIP 技术都是采用 SIP 作为控制协议的，因此 ECRIT 架构主要关注的是 SIP，但是对于其他协议（H.323、XMPP），该架构仍然适用。

本章基于 SIP 系统技术介绍 ECRIT 紧急呼叫系统。

3.1 E-VoIP 系统架构

在互联网上拨打紧急电话，以下信息是必不可少的：

（1） 定位信息(Location Information)

定位信息是紧急呼叫的基本信息。没有定位信息，在互联网上的紧急呼叫是不可能实现的。由定位信息才能决定应该对该紧急呼叫做出响应的紧急呼叫中心（PSAP），并为呼叫中心提供用户的当前地址[34]

（2） 本地紧急呼叫号码（Local Emergency dial strings）

紧急呼叫是通过用户呼出的特定的号码标识的，所以从支持全球通信的角度出发，必须知道各国所有的紧急呼叫号码。

（3） 响应 PASP 的联络信息（Contact information of the responsible PSAP）

为了使紧急呼叫连接到 PSAP，PSAP 必须公开它的联络信息（比如，SIP 通用资源标识号）。在 ECRIT 架构中 VoIP 服务提供商并不是必须的，也就是说，紧急呼叫可以通过任何方式（如互联网）接入 PSAP，并不一定要通过 VoIP 服务提供商呼叫 PSAP。

一个完整的 ECRIT 紧急呼叫系统包括如下参与方：

（1） SIP 呼叫方：发起 VoIP 紧急呼叫

（2） DHCP 服务器：动态配置 IP 地址

（3） VoIP 服务提供商：SIP 呼叫方注册后，提供 VoIP 服务

（4） 位置信息服务器：存储 SIP 呼叫方的位置信息

（5） 映射服务器：存储位置信息与 PSAP 的映射信息

（6） 呼叫中心 PSAP：响应 SIP 呼叫方的请求

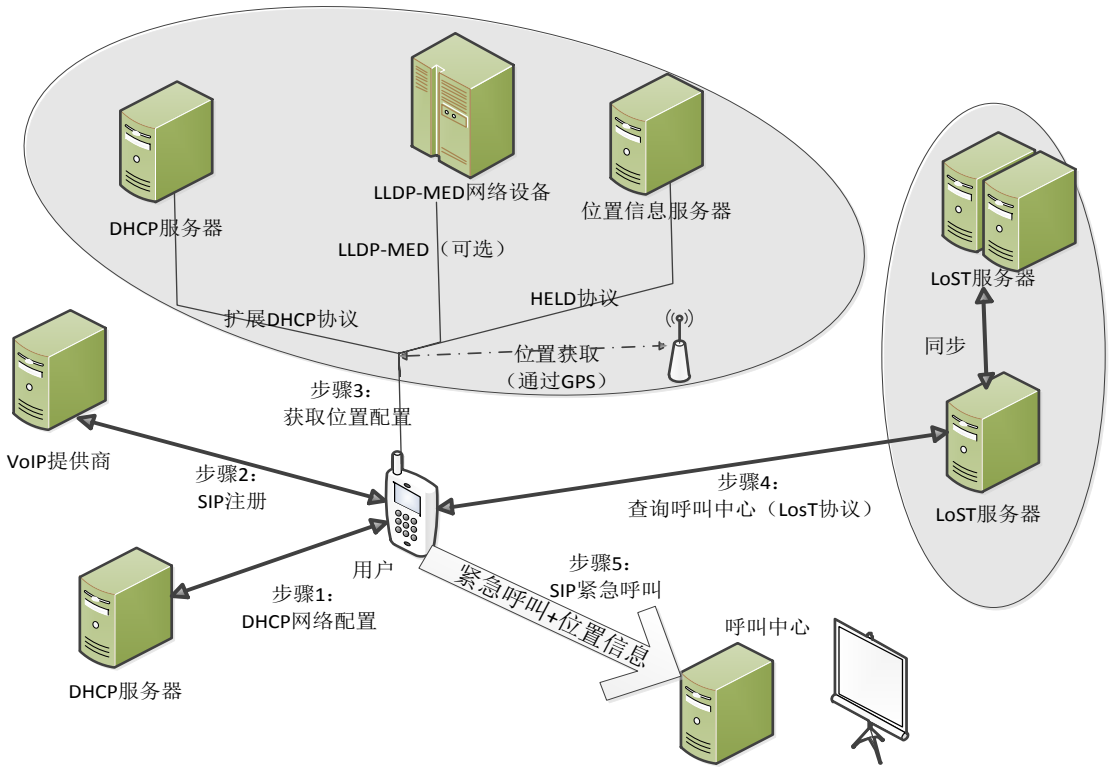


图 3.1 E-VoIP 系统架构

图 3.1 显示了 E-VoIP 紧急呼叫的系统架构[35]，其中涉及一系列独立的步骤和协议。

VoIP 设备开启后，进行以下步骤，即可进行紧急呼叫

步骤 1：网络配置。在大多数的网络中，网络配置通过 DHCP 服务器来完成。DHCP 服务器通常负责给设备分配一个 IP 地址，同时还负责给设备配置 DHCP 服务器的位置信息（或者一个映射服务器地址）。

步骤 2：向 VoIP 服务提供商注册。此为可选步骤，VoIP 服务提供商也可以给设备配置 DHCP 服务器的位置信息

步骤 3：获取定位信息。通过 HELD 或者 DHCP 协议可以完成这一目标，或者直接通过 GPS 等手段获取定位信息。这是 E-VoIP 系统架构中至关重要的部分。没有位置信息，后续步骤都无法开展。

步骤 4：获取呼叫中心的连接信息。通过 LoST(定位位置到服务中心的翻译)协议查询映射服务器，根据呼叫用户的定位信息与 PSAP 的映射关系，找到需要连接的 PSAP。映射服务器返回 PSAP 的连接信息（例如，SIP URI）和紧急呼叫号码（例如，110，119 等）。

一旦映射完成，呼叫用户就已经收集了所有必要信息，可随时拨打紧急呼叫。虽然在通话过程中，也会不断更新呼叫用户的位置信息，并可能对 PSAP 再次进行映射，但以上所有工作都必须在紧急呼叫拨号前完成。

由于客户端必须在拨号前知道用户拨出的号码是否为紧急呼叫号码，而这个信息只有 LoST 才可以提供。所以即便在不拨打紧急呼叫的情况下，呼叫用户在拨号前也需要完成以上步骤。

步骤 5：SIP 紧急呼叫。把位置信息通过 SIP 发送给呼叫中心，开始紧急呼叫。

3.2 E-VoIP 关键技术

下面介绍两个 E-VoIP 的关键技术：用户位置信息获取与描述、紧急呼叫中心发现[36]

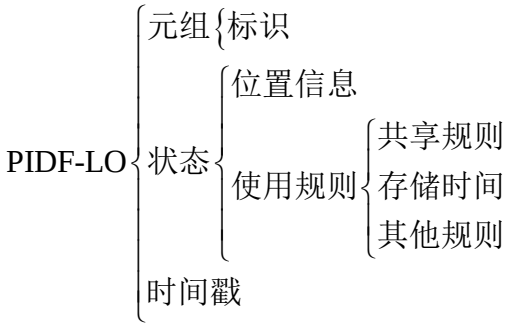
3.2.1 用户位置信息描述

如果紧急呼叫方可以足够清晰的方式描述他的位置，那么也许自动定位就不那么重要了。但在许多实际情况下，紧急呼叫方由于各种原因无法提供足够的信息。

传统的 PSTN 拨打紧急电话系统并不在呼叫信令中传送呼叫者的位置，因为主叫号码是和主叫用户的物理位置绑定的。但是在 VoIP 中，用户物理位置和其 SIP URI 标识并无确定的对应关系，因此给出用户当前位置信息十分必要。而且位置信息在紧急呼叫发起之前就应该获取，以便确定用户所在的本地紧急呼叫号码。

GEOPRIV 工作组对位置信息的描述制定了标准格式：PIDF-LO（基于 XML 格式）。

PIDF-LO 格式：




```
<tuple id="0123">   （一个唯一的随机 ID）
 <status>
  <gp:geopriv>
   <gp:location-info>必备信息元，包含实际位置信息
   <!-- location information is inserted here -->
   </gp:location-info>
   <gp:usage-rules>                usage-rules：必备信息
```

<gp:retransmission-allowed> 是否允许接受方重发，以便与第三方共享位置信息

no

</gp:retransmission-allowed>

<gp:retention-expiry> 位置信息的有效时间

2013-02-00T00:12:34+23:45

</gp:retention-expiry>

</gp:usage-rules>

</gp:geopriv>

< gp:method> 可选。例如，“GPS”、“RFID”、“LLDP-MED”、“Wiremap”

</gp:method>

<provided-by>

</provided-by>

</status>

<timestamp>2013-02-00T00:12:34+23:45</timestamp>

</tuple>

其中位置信息（location-info）主要分两种类型

（1）地理位置

（a） 点（纬度，经度）或者 点（纬度，经度，高度）

（b） 多边形：一系列 2D 点或 3D 点（同一高度）

（c） 圆：2D 中心点+半径

（d） 圆弧带：2D 中心点+内外径+圆弧带角度

（e） 球体：3D 中心点+半径

（f） 棱柱：多边形+高度

（g） 其他形式

以点（纬度，经度）为例，其表示形式为：

<gp:location-info>

<gml:location>

<gml:Point gml:id="点"

srsName="abc:def:ghi:jks::1234">

<gml:pos>100 200</gml:pos>

</gml:Point>

```
    </gml:location>
  </gp:location-info>
```

（2）地址

地址：国家，省，市，街道

```
  <gp:location-info>
      <cl:civicAddress>
          <cl:country>中国</cl:country>          //国家
          <cl:A1>江苏省</cl:A1>                  //省
          <cl:A2>城市</cl:A2>                    //市
          <cl:RD>街道</cl:RD>                    //街道
          <cl:HNO>1</cl:HNO>                     //门牌号
      </cl:civicAddress>
  </gp:location-info>
```

坐标全球可以统一格式，但是地址每个国家各有不同。

3.2.2 用户位置信息获取

位置定位信息可以通过（实际的 PIDF-LO）值或者（PIDF-LO 的 URI）引用两种描述形式来获取。

（1）用值的形式来传递定位信息（Location by Value）

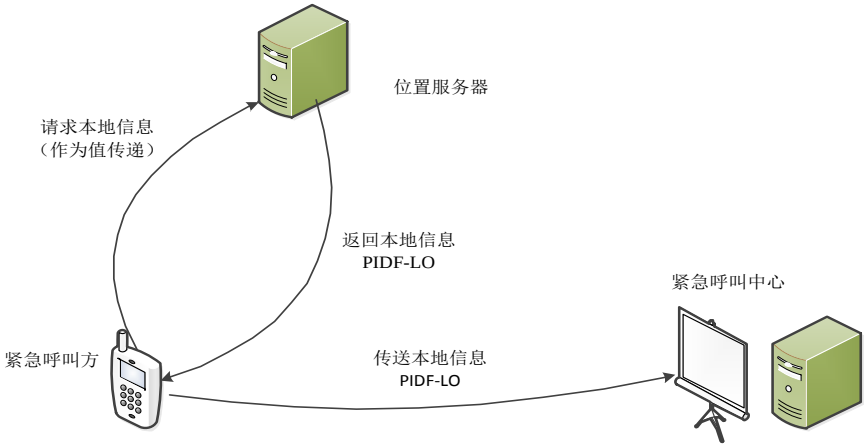


图 3.2 定位信息的值传递

如图 3.2 所示，呼叫方首先请求位置服务器，以获取定位信息，位置服务器响应该请求，返回定位信息，整个 PIDF-LO 都发送给呼叫方，最后再由呼叫方把定位信息（整个 PIDF-LO）

发送给呼叫中心。

（a） 呼叫必须发送整个 PIDF-LO

（b） 当位置更新后，客户必须发送位置更新

（c） 适合自主的位置信息（例如，GPS），可以不需要位置服务器

（d） 连接断开后，没有位置更新

（e） 位置信息直接发送给接收方

（f） 支持任何位置配置协议

（2）用引用的形式来传递定位信息（Location by Reference）

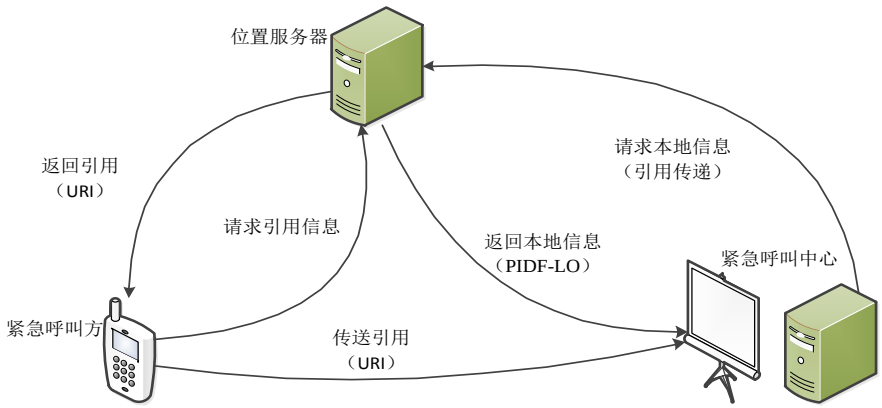


图 3.3 定位信息的引用传递

如图 3.3 所示，呼叫方首先请求位置服务器，以获取定位信息的引用（URI），位置服务器响应该请求，返回引用（URI），接着再由呼叫方把引用（URI）发送给呼叫中心。当呼叫中心需要呼叫方的位置信息时，呼叫中心需要通过呼叫方的引用（URI）请求位置服务器，最后位置服务器返回整个 PIDF-LO 给呼叫中心。

（a） 客户只需要发送一个引用（URI）

（b） 接收方能够通过 URI 自动更新位置信息

（c） 位置服务器是必须的，自主的位置信息必须提供给位置服务器（不是标准位置配置协议的一部分）或者手机自己成为一个位置服务器

（d） 甚至当客户手机挂断后，位置信息仍然可以从位置服务器获取

（e） 支持规定的位置配置协议

（f） 发送者不知道接收方能否解决引用问题

3.2.3 紧急呼叫中心发现

常用的紧急呼叫有救护、消防和公安呼叫，除此以外，很多国家还有其他一些紧急服务，比如山地救援、气体泄漏、毒物控制、海岸警卫队、森林火警等。所以，在目前的紧急呼叫系统中，每个国家都有自己的一套紧急服务。为了确保这些服务以一个统一的方式呈现，IETF 在 2008 年创立了一个“服务 URN”的注册表。这些统一资源名称（URN）是短文本字符串，比如消防队在这里确定为 urn:service:sos。

以下是一组已经定义的 URN，包括紧急求救服务 URN 和咨询服务 URN。

表 3.1 紧急求救服务 URN

统一资源名称 URN	服务
urn:service:sos	一般紧急求救服务
urn:service:sos.ambulance	救护服务
urn:service:sos.animal-control	动物控制服务
urn:service:sos.fire	消防服务
urn:service:sos.gas	煤气泄漏和煤气紧急事故服务
urn:service:sos.marine	海上搜救服务
urn:service:sos.mountain	山地救援服务
urn:service:sos.physician	医疗信息服务
urn:service:sos.poison	中毒控制服务
urn:service:sos.police	警方或其他执法服务

表 3.2 咨询服务 URN

统一资源名称 URN	服务
urn:service:counseling	一般咨询辅导服务
urn:service:counseling.children	辅导孩子服务
urn:service:counseling.mental-health	健康心理健康辅导服务
urn:service:counseling.suicide	预防自杀热线服务

为了确定一个负责处理的 PSAP，需提供由位置信息到 PSAP 的映射信息。IETF 开发的 LoST（位置到服务中心翻译协议）是一个基于 XML 文本的协议。它不仅能告诉客户正确的 PSAP，也可以返回有效服务，提供这个映射有效的区域，甚至验证地址。

LoST 规范要求 XML 消息通过 HTTP 或者 HTTPS 协议来传送。

如图 3.4 所示，发现 PSAP 需要如下三步：

图 3.4 LoST 发现 PSAP 流程

（1） 配置 LoST 服务器

LoST 信息和 DNS 信息类似，存在于一个全球分布式数据库中

LoST 的基本设备包括：

搜索器（Seeker）：使用 LoST 服务器的客户端在搜索器的帮助下，获得映射服务。

解析器（Resolver）：帮助客户端解析获得的映射信息。

树（Tree）：映射服务器的一个层次结构，常以国家为单位建立。

树林导向（Forest guide）：用来发现哪棵树映射特定服务、特定地区，其间不包含具体的映射信息。

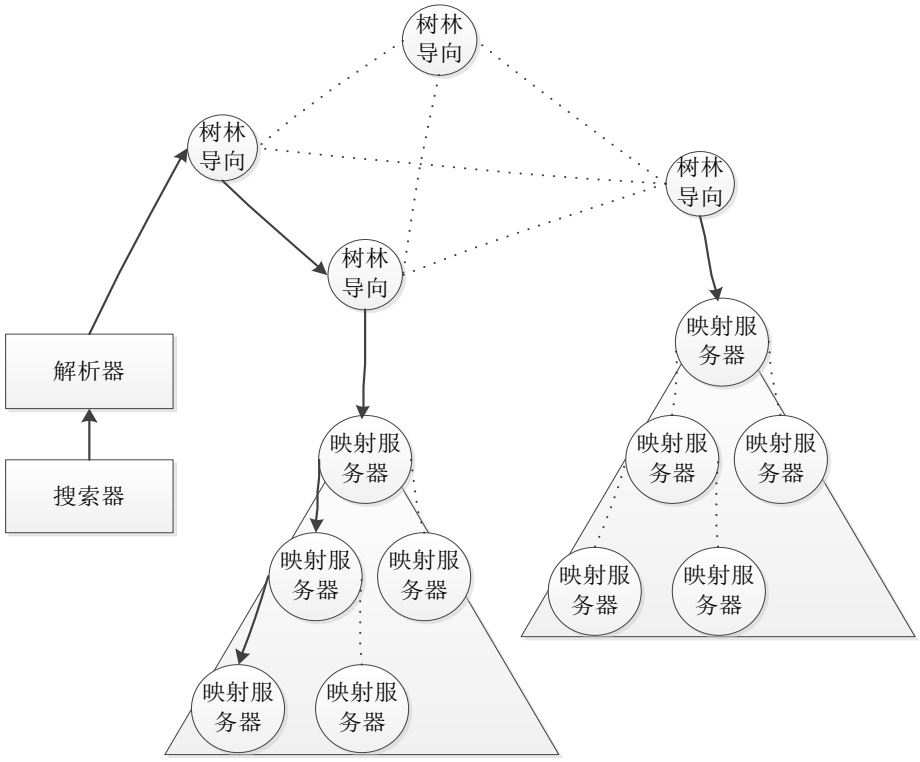


图 3.5 LoST 的架构

（2） 检索服务中心列表

检索服务中心列表获取紧急呼叫服务的 URN

查询：

```
<listServicesByLocation>
    <location id="0123" profile="geodetic-2d">              //位置信息
        <gml:Point gml:id="点"
            srsName="abc:def:ghi:jks::1234">
             <gml:pos>100 200</gml:pos>
        </gml:Point>
    </location>
    <service>urn:service:sos</service>                      //查询服务中心列表
</listServicesByLocation>
```

响应：

```
<listServicesByLocationResponse>
    <serviceList>                                           //紧急呼叫服务的 URN 列表
        urn:service:sos.ambulance
        urn:service:sos.fire
    </serviceList>
</listServicesByLocationResponse>
```

（3） 查找服务中心

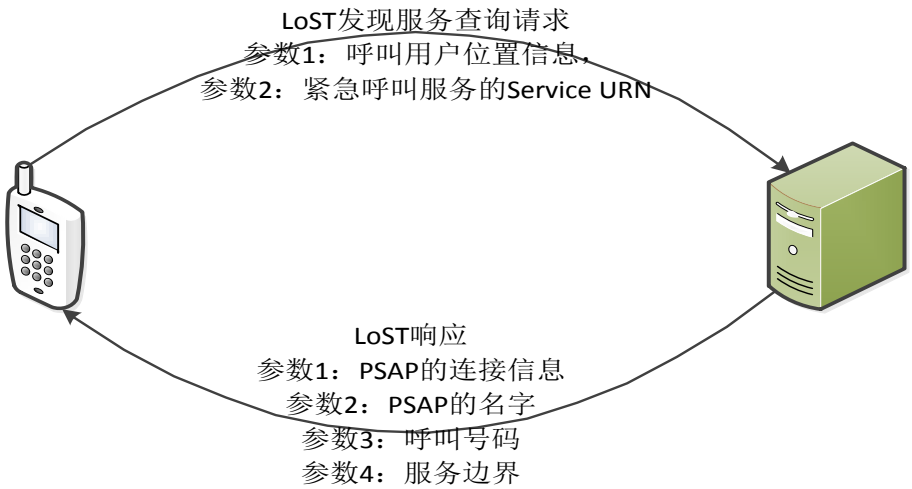


图 3.6 查询服务中心

如图 3.6 所示，呼叫方发送 LoST 发现服务查询请求给 LoST 服务器，LoST 服务器收到查询请求回送响应给呼叫方

（a） LoST 发现服务

```
<findService serviceBoundary="reference" recursive="true">
  <location id="98765" profile="geodetic-2d">   //用户位置信息编号
    <gml:Point gml:id="点"                      //呼叫用户位置信息
      srsName="abc:def:ghi:jks::1234">
      <gml:pos>100 200</gml:pos>
    </gml:Point>
  </location>
  <service>urn:service:sos.fire </service>          //紧急呼叫服务的 Service URN
</findService>
```

（b） LoST 响应

```
<findServiceResponse>
  <mapping source=" abc.def " sourceId="1234567890">
    <displayName xml:lang="zh">
    南京火警                                                    //显示名字
    </displayName>
    <service>urn:service:sos.fire </service>
    <serviceBoundaryReference source="abc.def "  key="0123456789" />
      <uri>sip:nanjingfire@def.cn</uri>                        // PSAP 的联系信息
        <serviceNumber>119</serviceNumber>                     //呼叫号码
  </mapping>
  <locationUsed id="98765"/>                                   //用户位置信息编号
</findServiceResponse>
```

因为提供持续的位置信息更新会给 LoST 产生额外的负荷，并过快地耗尽设备的电能，所以提出了 serviceBoundary（责任区域）的概念，当只有小范围的位置更新时，可认为服务中心不变，这个小范围就是该服务中心的责任区域。

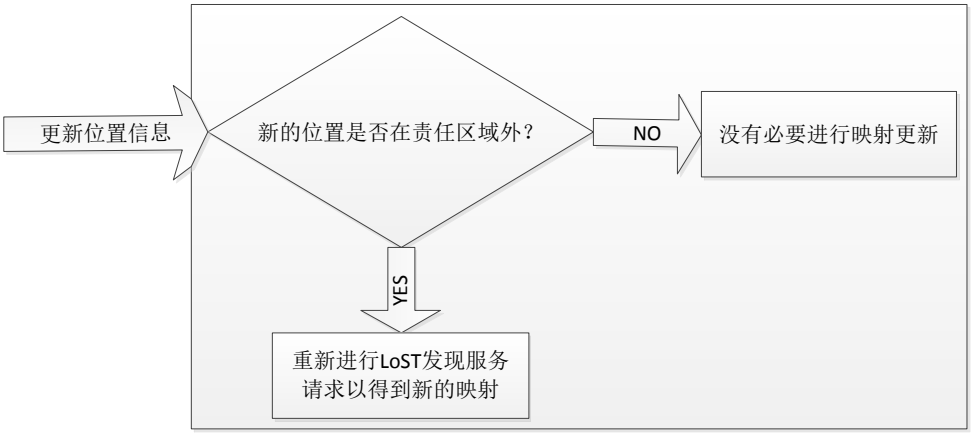


图 3.7 查询责任区域

如图 3.7 所示，每当发生位置更新时，首先判断新的位置是不是在当前的 PSAP 的责任区域外，如果是，则重新进行 LoST 发现服务以便得到新的映射，反之，则没有必要进行 LoST 发现服务，以节省能耗。

3.3 本章小结

本章首先对 ECRIT 工作组制定的紧急呼叫架构进行了详细的研究，主要关注用户位置信息的描述方式与传递方法，提出了基于地理位置和基于地址位置两种位置信息描述方式，分析了位置信息作为值传递与位置信息作为引用传递的异同，最后描述了紧急呼叫中心发现的协议与流程。

第四章 VoIP 紧急呼叫系统设计

4.1 VoIP 紧急呼叫设计要求分析

4.1.1 VoIP 紧急呼叫功能分析

第三章叙述了在紧急呼叫之前需要进行的几个步骤，位置信息获取，紧急呼叫中心发现等等。这些步骤确保如果发生紧急情况，用户可以正常拨打紧急呼叫号码进行通信[37]-[38]。

在紧急呼叫正在进行时，呼叫方需要执行一些相应的功能，以确保能接受呼叫中心可能的回叫

（1） 停用呼叫转移和免打扰功能（以便 PSAP 可能的回叫）

（2） 防止意外挂断（比如，用户界面上挂断键的误操作），对于紧急呼叫而言，结束通话的应该是紧急呼叫中心，而不是呼叫者。

（3） 停用呼叫保持功能，使用户不会错误地使紧急呼叫置于通话保持状态

客户端必需区分呼叫方发起的是否为紧急呼叫。当呼叫方发起紧急呼叫时，呼叫请求的 Request-URI 和 TO 头部是 sip:sos@abc.def。其中用户名为 sos，由此识别为紧急呼叫。

一个基于 SIP 的 E-VoIP 呼叫总是以 SIP INVITE 消息开始的。和一般的 SIP INVITE 消息不同的是，此消息需要包含如下信息

（1） 紧急呼叫服务 URN

（2） PSAP 的 SIP URI

（3） 呼叫用户的 SIP URI

（4） 位置信息

最后，SIP 呼叫方把 SIP INVITE 消息通过图 4.1 所示的结构发送给呼叫中心，完成呼叫控制。

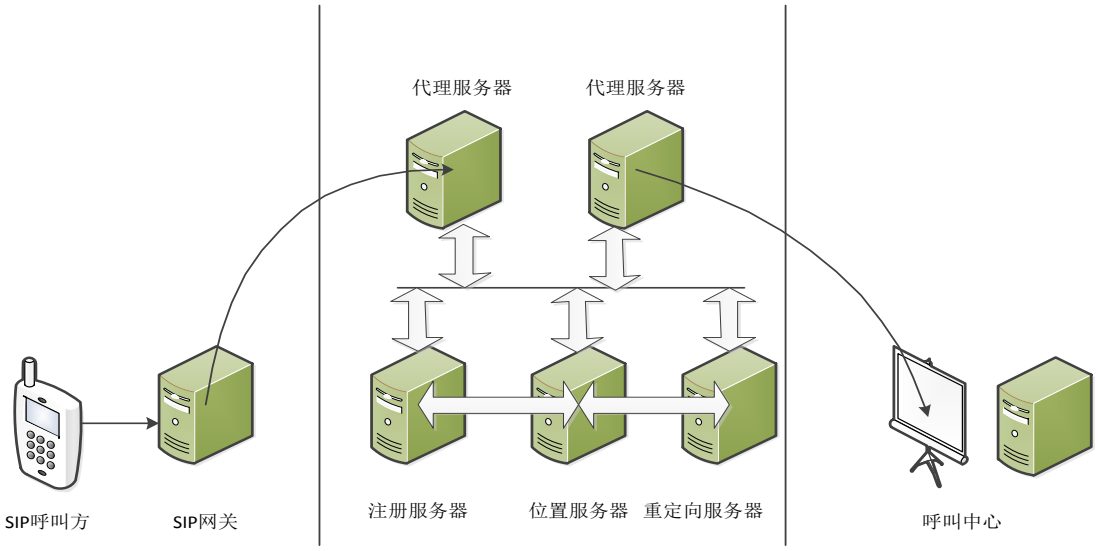


图 4.1 基于 SIP 的 E-VoIP 呼叫系统结构图

4.1.2 SIP 协议栈实现

论文采用 oSIP 实现 SIP 协议的基本功能，并在充分消化掌握 oSIP 的基础上，根据 VoIP 紧急呼叫的功能要求对 SIP 协议栈进行了扩充。oSIP 是广为使用的 SIP 协议栈实现的开放源代码，使用标准 C 编写，协议功能遵循 IETF 标准，论文采用的是 oSIP V4.0.0 版本[39-40]。

oSIP 提供了 SIP 消息的构造、解析、事务状态机、SDP 消息体的 API。但并没有高层的会话控制[41]。

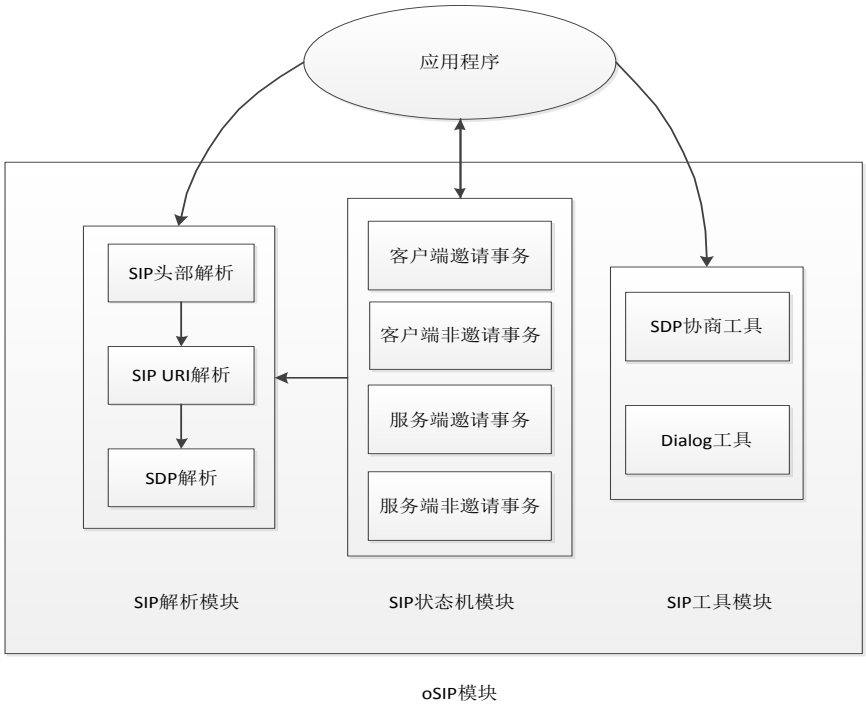


图 4.2 oSIP 体系结构

图 4.2 是 oSIP 的体系结构。

oSIP 的构成主要有：SIP 解析模块、SIP 状态机模块和 SIP 工具模块

（1） SIP 解析模块

SIP 解析模块主要有三部分组成：SIP 头部解析、SIP URI 解析和 SDP 解析

SIP 头部解析主要通过以下函数实现： 函数 osip_message_parse 、函数 _osip_message_parse、函数 osip_util_replace_all_lws、函数__osip_message_startline_parse、函数 msg_headers_parse、函数 msg_osip_body_parse。当然还包括对头部的读写操作，对不同头部有不同的处理操作函数。

SIP URI 解析主要通过以下函数实现，函数 url_init、函数 url_free、函数 url_setscheme、函数 url_setusername、函数 url_sethost、函数 url_setport。函数 url_init 主要负责分配内存，并对 url_t 结构做初始化操作。函数 url_free 主要负责释放 url_t 结构，并对 url_t 结构的变量清空。函数 url_setscheme 设置 url_t 结构中的摘要，函数 url_setusername 设置 url_t 结构中的用户名，函数 url_sethost 设置 url_t 结构中的主机、函数 url_setport 设置 url_t 结构中的端口。

在论文的后续编程中，我们会用到这些 API 函数

（2） SIP 状态机模块

oSIP 协议栈中状态机分四种类型：请求客户端事物状态机 ICT、非请求客户端事物状态机 NICT、请求服务器端事物状态机 IST、非请求服务器端事物状态机 NIST。

客户端事务状态机如图 4.3 和图 4.4 所示。

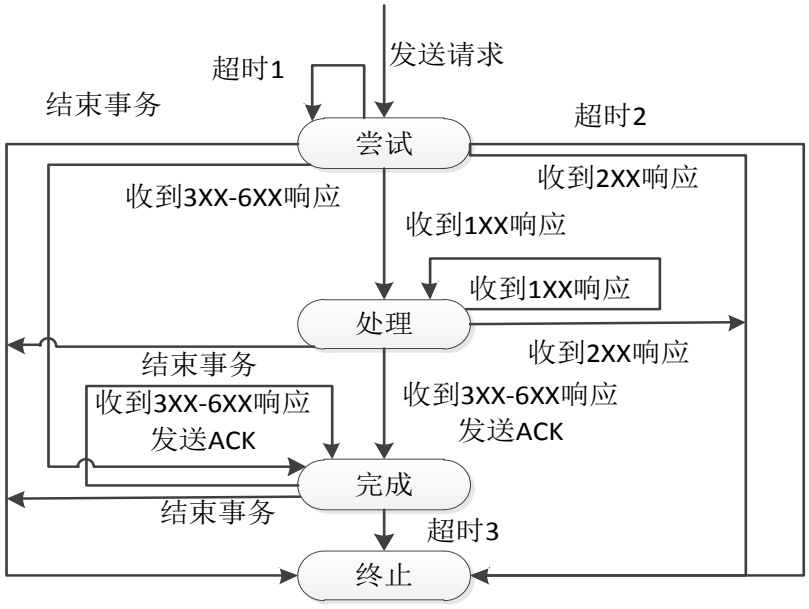


图 4.3 请求客户端事务状态机 ICT

请求客户端事务状态机 ICT 如图 4.3 所示

请求客户端事务状态机 ICT 一共有四个状态：“尝试”（Trying）状态，“处理”（Proceding）状态，“完成”（Comleted）状态，“终止”（Terminate）状态。

首先当发送一个 INVITE 请求时，事务初始化为“尝试”状态，然后将 INVITE 请求传送给 UDP 传输层，同时包括目的地址的 IP 地址、端口号。如果目的地成功收到 INVITE 消息，立即向呼叫方发生响应消息。如果超时 1，表示在规定的时间（定时器 A 溢出时间每次增加一倍）内，呼叫方没有收到被叫方响应的消息，这时请求呼叫方重传。如果超时 2，表示在规定的时间内（定时器 B 溢出时间），呼叫方仍然没有收到被叫方发来的响应，则表示 INVITE 请求失败，结束“尝试”状态直接置为“终止”状态。而如果在规定时间内收到服务端的事务响应，请求客户端事务状态机 ICT 状态发生相应变化。如果收到 1XX 响应，无论呼叫方处于“尝试”状态还是“过程”状态都需要切换到“过程”状态。如果收到 2XX-6XX 响应，则无论呼叫方处于“尝试”状态还是“过程”状态都需要切换到“完成”状态。在“完成”状态呼叫方必须生成 ACK 消息，并将它传送给 UDP 传输层，同时必须保证目的地址的 IP 地址、端口号和 INVITE 请求时完全相同。另外，ACK 消息必须和 INVITE 消息有相同的 URI、From、Call-ID 和 Via。最后如果定时器 D 溢出，呼叫方从“完成”状态切换到“终止”状态，如果成功收到 OK 响应消息，同时也从“完成”状态切换到“终止”状态。

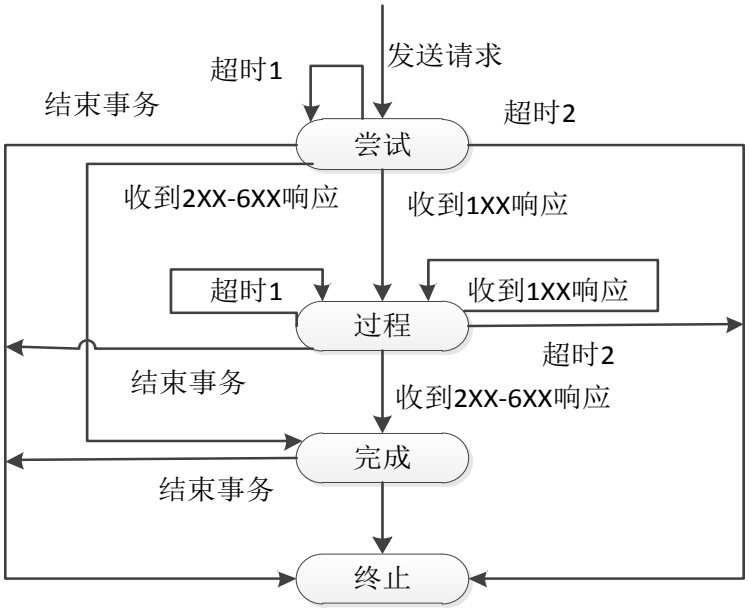


图 4.4 非请求客户端事务状态机 NICT

非请求客户端事务状态机 NICT 如图 4.4 所示，由于实现过程和请求客户端事务状态机 ICT 类似，这里就不再赘述。

服务端事务状态转换如图 4.5 和图 4.6 所示。

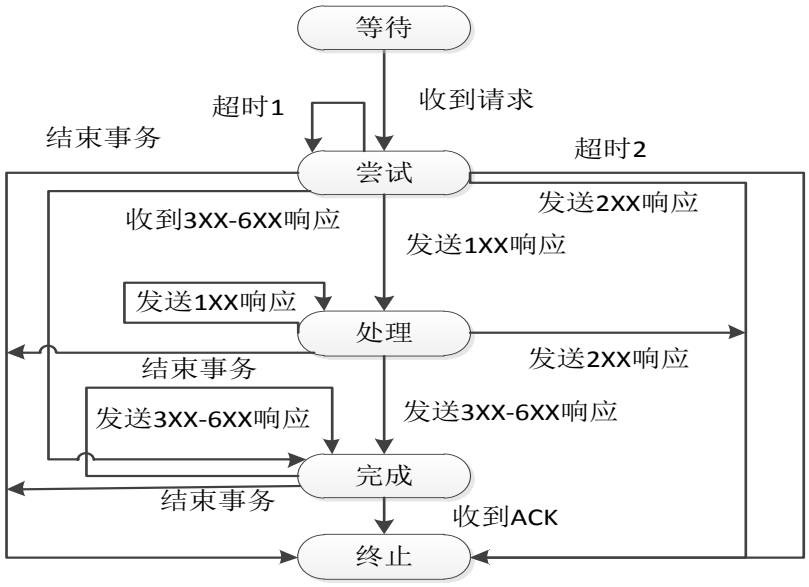


图 4.5 请求服务器事务状态机 IST

请求服务器端事物状态机 IST 有五个状态：“等待”状态、“尝试”（Trying）状态，“处理”（Proceding）状态，“完成”（Comleted）状态，“终止”（Terminate）状态。首先服务端事务进入“等待”状态等待客户端的 INVITE 请求，收到 INVITE 请求后进入“尝试”状态，进行简单的处理，并发送 100 Trying 给客户端，响应消息的头部和 INVITE 请求消息具有相同的 URI、From、Call-ID 和 Cscq。处理完成后如果发生 1XX 响应，则跳转到“处理”状态，如果发生 2XX 响应，则跳转到“终止”状态，如果发生 3XX-6XX 响应，则跳转到“完成”状态。当收到客户端发来 ACK 请求，服务器由“完成”状态跳转到“终止”状态。

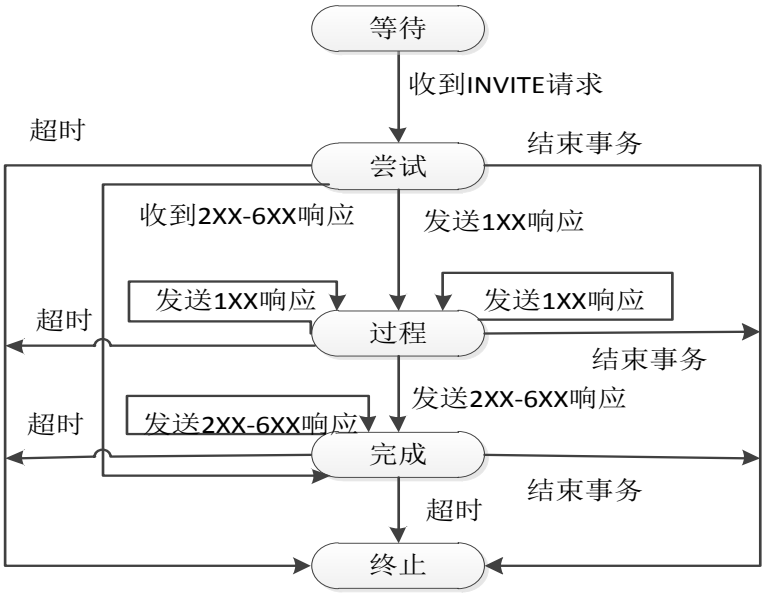


图 4.6 非请求服务器事务状态机 NIST

非请求服务端事务状态机 NIST 如图 4.6 所示，由于实现过程和请求服务端事务状态机 IST 类似，这里就不再赘述。

（3） 回调函数 Callback

oSIP 结构如下

```
struct osip {
  void *application_context;
  void *ict_fastmutex;              /*ICT事务的互斥元*/
  void *ist_fastmutex;              /*IST事务的互斥元*/
  void *nict_fastmutex;             /*NICT事务的互斥元*/
  void *nist_fastmutex;             /*NIST事务的互斥元*/
  void *ixt_fastmutex;
  void *id_mutex;
  ……
  osip_list_t osip_ict_transactions;          /* ICT事务转换表*/
  osip_list_t osip_ist_transactions;          /* IST事务转换表*/
  osip_list_t osip_nict_transactions;         /* NICT事务转换表*/
  osip_list_t osip_nist_transactions;         /* NIST事务转换表*/
  ……
  osip_message_cb_t msg_callbacks[];          /*消息回调函数*/
  osip_kill_transaction_cb_t kill_callbacks[]; /*结束事务的回调函数*/
  ……
  /*事务出错的回调函数*/
  ……
  int (*cb_send_message) ();                  /*发送消息的回调函数*/
    };
```

在 oSIP 协议栈中，对各类事件的响应都是以回调函数 callback 的方式实现的。回调函数是一个函数指针的形式，在状态机内部状态跳转时被事务处理函数调用。回调函数是 oSIP 开源协议栈留给用户的接口。

比如客户端内部状态跳转时，根据不同的状况，调用不同的回调函数，常用的回调函数如下：回调函数 Cb_ict_invite_sent2、回调函数 Cb_ict_transport_error、回调函数 Cb_ict_1xx_received、回调函数 Cb_ict_2xx_received、回调函数 Cb_ict_3456xx_received。

（4） 事件 Event

状态机的状态跳转都是由 Event 触发的。应用程序与状态机之间通过 Event 来通信。

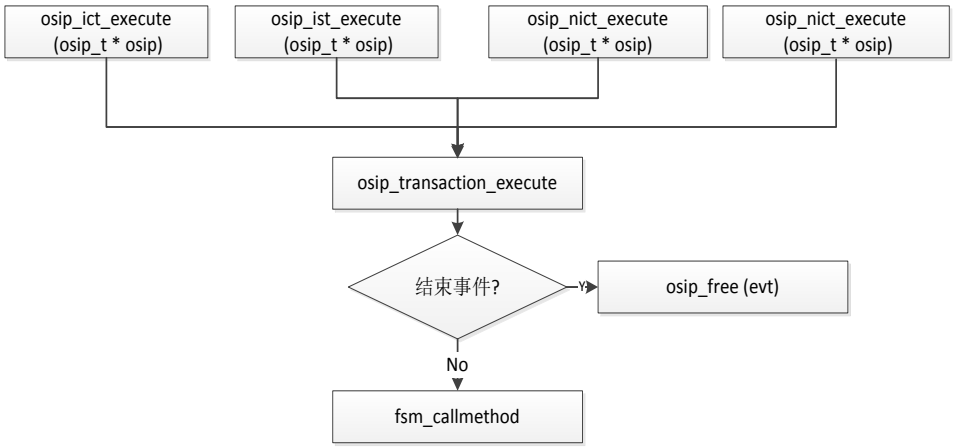


图 4.7 Event 整体处理流程

如图 4.7 所示，是事件整体处理流程。状态机中有 3 种事件：计时器、接收 SIP 消息、发送 SIP 消息。由此构造事件类型，如下所示：

```
typedef enum type_t {
    /*ICT的计时器*/
    TIMEOUT_A,                  /*计时器A*/
    TIMEOUT_B,                  /*计时器B*/
    TIMEOUT_D,                  /*计时器D*/
    /*NICT 的计时器*/
    TIMEOUT_E,                  /*计时器E*/
    TIMEOUT_F,                  /*计时器F*/
    TIMEOUT_K,                  /*计时器K*/
    /*IST的计时器*/
    TIMEOUT_G,                  /*计时器G*/
    TIMEOUT_H,                  /*计时器H*/
    TIMEOUT_I,                  /*计时器I*/
    /*NIST的计时器*/
    TIMEOUT_J,                  /*计时器J*/
    /*接收SIP消息*/
    RCV_REQINVITE,              /*收到INVITE请求*/
    RCV_REQACK,                 /*收到ACK请求*/
    RCV_REQUEST,                /*收到NON-INVITE and NON-ACK请求 */
    RCV_STATUS_1XX,             /*收到1XX响应*/
    RCV_STATUS_2XX,             /*收到2XX响应*/
    RCV_STATUS_3456XX,          /*收到其他响应*/
    /*发送SIP消息*/
    SND_REQINVITE,              /*发送INVITE请求*/
    SND_REQACK,                 /*发送ACK请求*/
    SND_REQUEST,                /*发送NON-INVITE and NON-ACK请求 */
    SND_STATUS_1XX,             /*发送1XX响应*/
    SND_STATUS_2XX,             /*发送2XX响应*/
    SND_STATUS_3456XX,          /*发送其他响应*/
    KILL_TRANSACTION,
    UNKNOWN_EVT
```

} type_t;

ICT 使用计时器 A、B、D，NICT 使用计时器 E、F、K，IST 使用计时器 J、H、I，NIST 使用计时器 J。

以计时器 B 为例，在客户端发送 Invite 请求而需要创建新的 ICT 事务时，调用计时器 B，TIMEOUT_B 数值为整个食物的生命周期。

（5） SIP 工具模块

SIP 工具模块分为 SDP 协商模块，Dialog 工具模块。

SDP 协商模块实现了 SDP 消息结构体的定义，SDP 字段的添加、SDP 字段的设置、SIP 结构体提供/应答模式和整个 SDP 媒体协商机制。

Dialog 工具模块使得用户可以按 SIP 协议的规范对 Dialog 进行各种操作，包裹建立、添加、查询、删除 Dialog 结构体。

4.2 用户位置配置的设计

用户位置确定可以由代理服务器确定，例如根据 UA 的 MAC 地址。也可以由用户自行配置确定位置，比如，可以通过扩展 DHCP 协议来实现。

在 DHCP 协议中加入扩展字段，方法如表 4.1 所示：

表 4.1 通过坐标配置的 DHCP 选项

GEOCONF_GEO	8bit
Length	8bit
LaRes	6bit
Latitude	34bit
LoRes	6bit
Longitude	34bit
AT	4bit
AltRes	6bit
Altitude	30bit
Datum	8bit

在 DHCP 选项中加入经度纬度高度信息，当然也可以加入地址信息来实现定位信息的配置。一个客户端以此可以在获得 IP 地址时获取位置信息。

在论文中，由于没有使用 DHCP 服务器配置客户端，所以采用一个特定的位置配置协议

——HELD 协议来实现位置的自动配置。

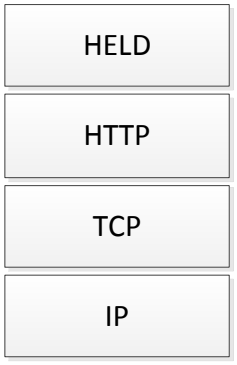


图 4.8 HELD 协议栈

如图 4.8 所示，HELD 协议是一个基于 HTTP 的应用层位置配置协议。HELD 协议通过 XML 文件来传输位置信息以实现客户端与定位服务器之间的通信。定位服务器给用户提供一个位置配置服务，当它接收到一个位置信息请求时（使用 locationRequest 消息），服务器根据到达消息的源 IP 地址来确定用户的位置信息。

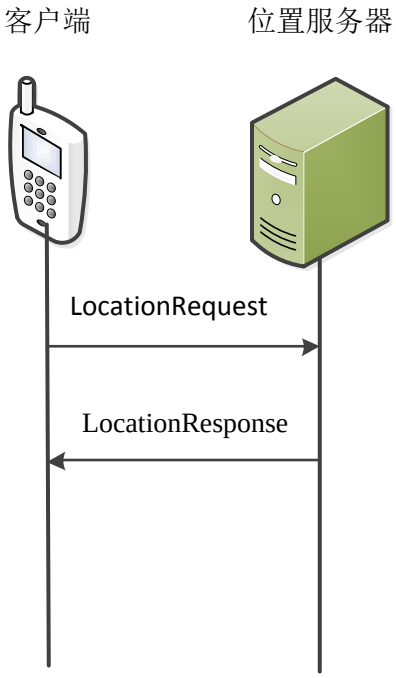


图 4.9 HELD 请求的简易流程

HELD 请求的简易流程如图 4.9 所示，客户端发送 LocationRequest 请求给位置服务器，

POST /location HTTP/1.1

Host: lis.example.com

Accept: application/held+xml,

application/xml;q=0.8,

text/xml;q=0.7

```
Accept-Charset: UTF-8,*
Content-Type: application/held+xml
<?xml version="1.0"?>
<locationRequest xmlns="urn:ietf:params:xml:ns:geopriv:held">
    <locationType exact="true">
        geodetic
        locationURI
    </locationType>
</locationRequest>
```

其中“exact”表示，服务器必须返回“locationType”元素中的位置类型，如果没有此位置类型，那么服务器不必遵循客户端的参数选择，可以直接返回其他有效的信息类型。

响应消息 LocationResponse 如果成功处理，则返回有关位置信息的统一文件类型 PIDF-LO。位置信息可以作为实际的 PIDF-LO 值或者 PIDF-LO 的 URI 来传递。

反之，如果处理失败，返回信息如下：

```
HTTP/1.x 200 OK
Server: Example LIS
Date: Tue, 29 Jan 2011 11:20:00 GMT
Expires: Tue, 30 Jan 2011 11:20:00 GMT
Cache-control: private
Content-Type: application/held+xml
<?xml version="1.0"?>
<error xmlns="urn:ietf:params:xml:ns:geopriv:held"
code="locationUnknown"
message="Unable to determine location"/>
```

此信息表明此位置服务器无法给用户提供定位信息。

此实现方法需要特别注意的是，当一个用户通过 NAT 接入网络的时候，需要确保 IP 地址的唯一（以精确定位），或者当一个用户通过 VPN 接入网络的时候，用户必须在 VPN 连接建立之前，完成 HELD 协议的相关步骤进行紧急呼叫发现。

4.3 紧急呼叫消息设计

4.3.1 SIP 消息通用结构

SIP 根据其基本功能要求，定义了 6 个请求消息和 6 个响应消息。

（1） 6 个请求消息是

（a） INVITE（邀请）。INVITE 消息用来邀请用户加入某个呼叫处理，它是呼叫方发送的第一条消息。

（b） ACK（确证）。ACK 消息用于证实代理或者呼叫方已收到 INVITE 请求的最终响应。

（c） BYE（再见）。主被叫用 BYE 消息来释放呼叫。

（d） CANCEL（取消）。CANCEL 消息用于取消正在进行中的请求。对于已经建立的呼叫没有作用。

（e） OPTIONS（选择）。OPTIONS 消息用于查询服务器的容量。

（f） REGISTER（登记）。用户用 REGISTER 消息向服务器注册。

SIP 请求消息的格式：

请求消息 = 请求开始行（包括：方法 请求 URI SIP 版本号）

*消息头部 1，消息头部 2，……

[消息内容]

（2） 6 个响应消息所对应的状态码是

（a） 1×× 消息响应，是中间响应

（b） 2×× 成功响应，表明消息已经被处理。

（c） 3×× 重定向响应，表明当前被叫方地址不可用

（d） 4×× 客户出错，请求失败，方法错误

（e） 5×× 服务器出错，服务器不支持或无法完成此请求

（f） 6×× 全局故障，服务器无法处理的错误

SIP 响应消息的格式：

响应消息 = 状态行（包括：SIP 版本号 状态码 SIP 理由描述）

*消息头部 1，消息头部 2，……

[消息内容]

一个基本的 INVITE 请求消息的例子：

INVITE urn:service:sos.fire SIP/2.0

Route: <sip:police@example.at;lr>

Via: SIP/2.0/TCP 192.0.2.1:5060;rport;branch=z9hG4bK4r1id0

To: "Vienna Police" < urn:service:sos.fire> （To 字段指示被叫方的地址）

From: HELLO<sip:hello @def.cn >;tag=1nexm5g （From 字段指示请求方的地址）

CSeq: 2 INVITE

Contact: HELLOsip:hello @10.10.134.39;grid=74500e587

4.3.2 紧急呼叫 SIP 消息的编码设计

分析 SIP INVITE 消息在整个紧急呼叫过程中的结构。

如前所述，SIP 紧急呼叫的 INVITE 消息需要包含如下信息：

（1）紧急呼叫服务 URN

（2）PSAP 的 SIP URI

（3）呼叫用户的 SIP URI。

（4）位置信息

对 SIP INVITE 的头部信息设计如下

Request URI ：紧急呼叫服务 URN

To 头部 ：紧急呼叫服务 URN

From 头部 ：用户的 SIP 地址

Route 头部 ：PSAP 的 SIP URI

Contact 头部 ：连接信息（用来支持回叫）

Supported 头部 ：包含“geolocation”值

Geolocation 头部 ：位置信息的必要参数

SIP INVITE 扩展实例如下：

INVITE urn:service:sos.fire SIP/2.0

Route: <sip:police@example.at;lr>

Via: SIP/2.0/TCP 192.0.2.1:5060;rport;branch=z9hG4bK4r1id0

To: "Fire Service" < urn:service:sos.fire>

From: HELLO<sip:hello @def.cn >;tag=1nexm5g

CSeq: 2 INVITE

Contact: HELLO<sip:hello @10.10.134.39;grid=74500e587>

Supported: geolocation

Geolocation: <cid:pidflo@zap>;

inserted-by=192.0.2.1;recipient=endpoint;used-for-

4.4 紧急呼叫号码自动获取

在紧急呼叫号码的自动获取一般有两种方式，一是客户端直接提供 SOS 按钮，这样客户端只需要根据位置信息完成紧急呼叫中心的发现即可；另一种是用户直接呼叫 911、110 等熟悉的紧急呼叫号码，由系统来识别它是否属于紧急呼叫，并转接到适合的紧急呼叫中心。

在两种情况下，首先需要完成的是，基于位置信息发现紧急呼叫中心。根据第三章紧急呼叫中心发现所述，我们采用 LoST 协议来完成紧急呼叫发现工作。

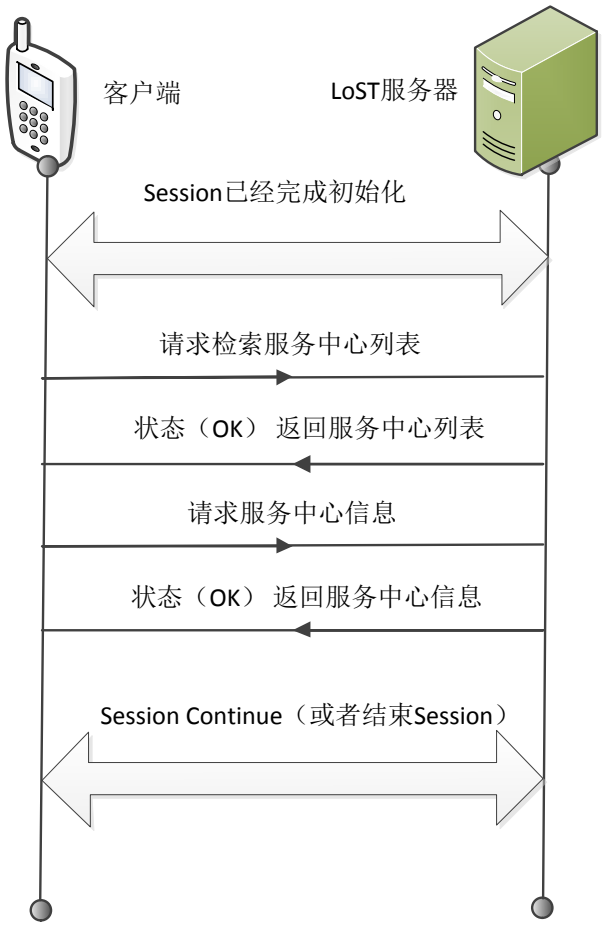


图 4.10 LoST 协议流程图

LoST 协议流程如图 4.10 所示，接下来，当用户直接呼叫 911、110 等熟悉的紧急呼叫号码时，论文采用紧急呼叫通信录的方式在客户端实现自动切换。

紧急呼叫通信录中包含如下信息

（1） 紧急呼叫服务的 URN（例如，urn:service:sos.fire）

（2） 紧急呼叫服务名称（例如，南京 XX 消防局）

（3） PSAP 的联系信息（例如，sip:nanjingfire@def.cn）

（4） 紧急呼叫号码（例如，02587654321）

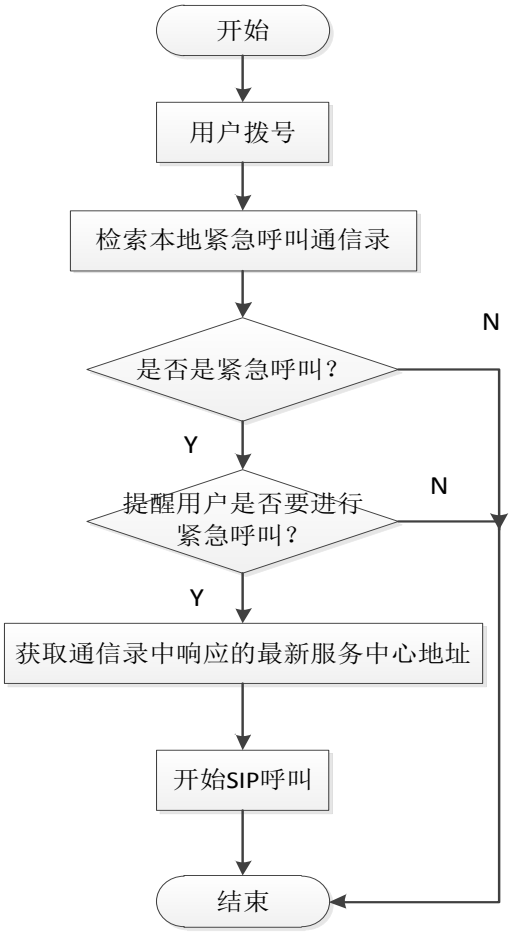


图 4.11 紧急号码自动获取流程图

紧急号码自动获取流程如图 4.11 所示，在紧急呼叫通信录中，把常用的紧急呼叫号码按 URN 进行统一分类，每当通过 LoST 协议发现一个服务中心的联系信息和呼叫号码的时候，客户端会按 URN 把这些信息写入紧急呼叫通信录。每次用户直接呼叫号码时，客户端首先进入紧急呼叫通信录进行搜索，查询该号码是否紧急呼叫号码，如果是紧急呼叫号码，则询问用户是否需要进行紧急呼叫，如果需要，则根据该紧急呼叫号码在紧急呼叫通讯录中搜索相对应的最新通过 LoST 协议发现的紧急呼叫中心，开始紧急呼叫。

4.5 PSAP 内部处理

PSAP 是一个呼叫中心，负责接听报警、火警、急救等紧急呼叫。PSAP 的功能包括在地图上显示紧急呼叫用户的位置，给用户分配紧急呼叫接话员，监视当前紧急呼叫，负责存储

整个紧急呼叫过程的详细信息，对媒体数据进行归类归档，并根据通话记录生成统计信息。

在 PSAP 内部采用一个紧急呼叫控制器来处理所有来话紧急呼叫，它负责给来话呼叫分配一个适合的紧急呼叫接话员，并且记录下所有紧急呼叫的详细信息。每次接到紧急呼叫时，呼叫控制器开启一个电话会议，负责管理和记录电话会议，并允许多方加入电话会议，其中包括紧急呼叫接话员、警察、医院、消防队等等。

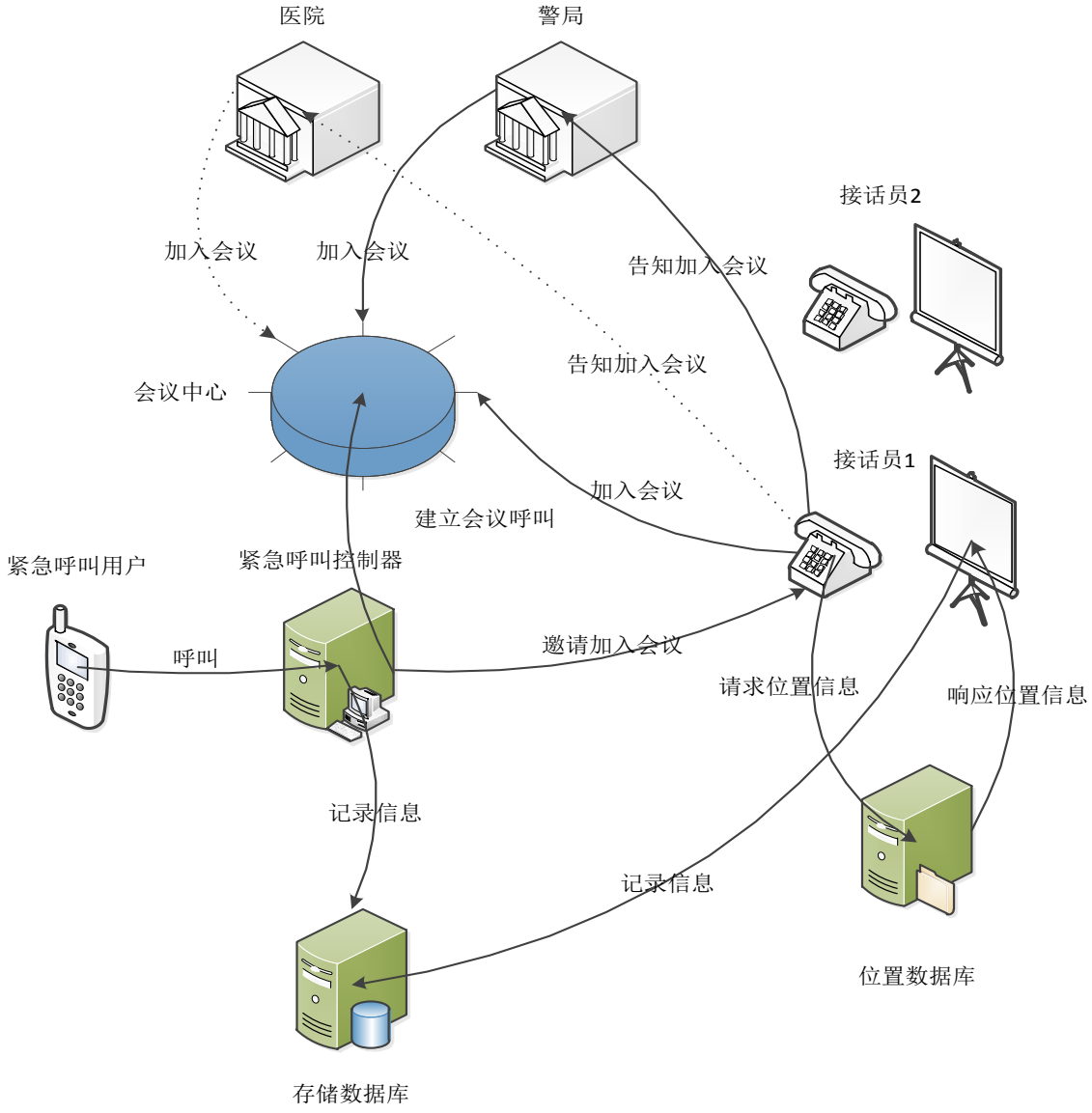


图 4.12 PSAP 内部处理流程图

PSAP 内部处理流程如图 4.12 所示。

（1） 紧急呼叫用户向 PSAP 发起紧急呼叫

（2） 紧急呼叫控制器接到紧急呼叫后，开始向存储数据库记录关于紧急呼叫的详细信息，并开始建立一个紧急呼叫会议

（3） 紧急呼叫控制器选择一个适当的紧急呼叫接话员，并邀请它加入紧急呼叫会议

（4）紧急呼叫接话员接到紧急呼叫会议邀请时，加入紧急呼叫会议，并向位置服务器请求定位信息，当收到响应的定位信息后，把定位信息更新到紧急呼叫接话员的地图软件上

（5）紧急呼叫接话员接通与用户的会议电话后，开始向存储数据库记录关于紧急呼叫的详细信息，如果发现有必要紧急呼叫处理方（比如警局、医院）的协作，则通告紧急呼叫处理方加入此会议

（6）警局或医院一旦接到紧急呼叫会议通告，则加入会议呼叫

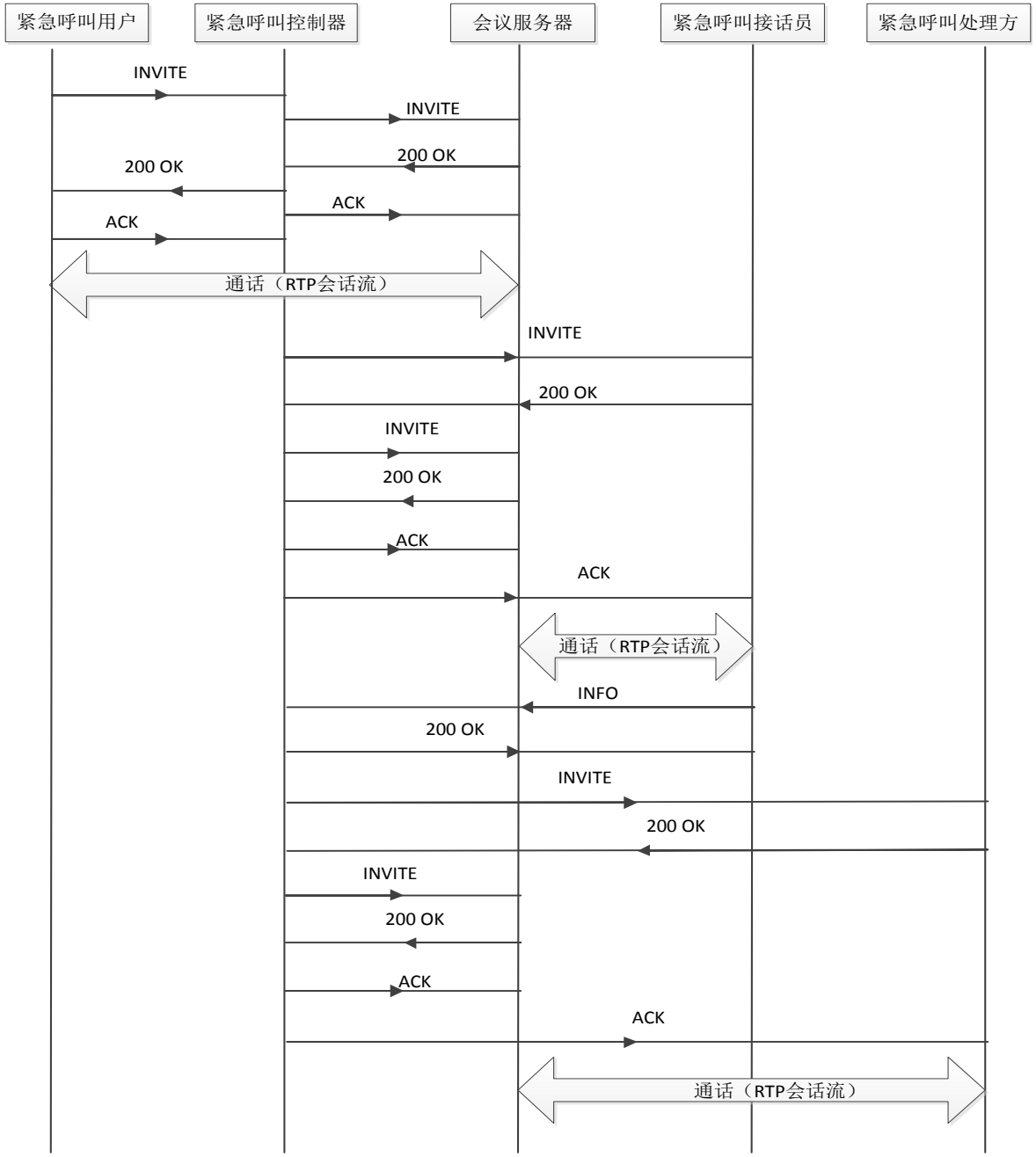


图 4.13 PSAP 内部处理消息流程图

PSAP 内部处理消息流程如图 4.13 所示。

（1） 紧急呼叫用户发起 SIP INVITE 请求，其中包含定位信息（PIDF-LO 或 PIDF-LO 的引用）

（2） 紧急呼叫控制器收到用户发来的 SIP INVITE 请求后，向会议服务器请求建立会议

（3） 会议服务器接受邀请，返回 200 OK 给紧急呼叫控制器

（4） 紧急呼叫控制器转发 200 OK 消息给用户，并向会议服务器发送 ACK 包

（5） 紧急呼叫用户发送 ACK 消息给控制器，用户加入会话，传输媒体消息

（6） 紧急呼叫控制器发送 SIP INVITE 请求给接话员，邀请其加入此会议会话

（7） 紧急呼叫接话员通过控制器与会议服务器进行 SDP 媒体协商，最后加入此会议会话，传输媒体消息

（8） 紧急呼叫接话员如果发现需要请求紧急呼叫处理方的帮助，则以 SIP INFO 消息的形式通知紧急呼叫控制器

（9） 紧急呼叫控制器接收到 SIP INFO 消息后，邀请相应的紧急呼叫处理方加入此会话会议

（10） 紧急呼叫处理方接受并响应 200 OK，通过控制器与会话服务器进行 SDP 媒体协商，最后加入此会话会议，传输媒体消息

（11） 最后紧急呼叫用户、接话员、处理方三方开始通信

4.6 数据紧急呼叫

4.6.1 数据紧急呼叫概述

在某些情况下，紧急呼叫涉及的是客户端上传应用数据，例如：由温度传感器上传警报数据，由车辆传感器上传撞车数据。这些告警数据常常是一次性的数据传输，在少数情况下，可能需要建立一个数据会话。这些数据交互类型的紧急呼叫称为“只有数据的紧急呼叫”（data-only emergency calls）

和 VoIP 紧急呼叫相同，只有数据的紧急呼叫也需要实现紧急标识、紧急呼叫路由功能和客户端位置定位，不同的是，紧急呼叫过程中的 RTP 包传送的不是语音信息，而是视频数据或者实时文本。。

根据是否有伴随的会话，数据紧急呼叫有如下两种使用模型：

（1） 需要接收方的响应：

客户端向 PSAP 发送一个只包含数据的紧急警报，同时建立一个由第三方发起的紧急呼叫（该呼叫包括语音、视频和/或数据信息）

（2） 不需要接收方的响应。

此时，SIP 消息中只需包含处理 IP 紧急呼叫服务的 URN，接收者对于发起者而言是未知的。这类警报型紧急呼叫不支持语音和/或视频通信，只支持由事件触发的紧急数据上传

4.6.2 扩展 SIP 协议消息实现方法

为了实现事件触发的紧急数据的上传，论文采用 MESSAGE 方法扩展基本 SIP 协议的功能，以即时消息（Instant Messaging）的方式主动向服务器端发送紧急数据。

SIP MESSAGE 可以用以下三种模式发送：

（1） 寻呼（pager）模式：不涉及会话的概念，每条 MESSAGE 消息互相独立，消息之间没有明确的联系。消息的关联完全由服务器自行判断。

（2） 会话（session）模式：在该模式下，即时消息是以 INVITE 请求建立，用 BYE 请求结束的一个会话。会话中可发送多个 MESSAGE 消息，这些 MESSAGE 消息共同组成一个即时消息，传递紧急数据。

（3） 大消息（Large Message Mode）模式：此模式用于含有多媒体内容的大型消息的传递。

无论采用哪一种模式，关键是 MESSAGE 方法的实现。论文采用 oSIP 予以实现，具体方法描述如下。

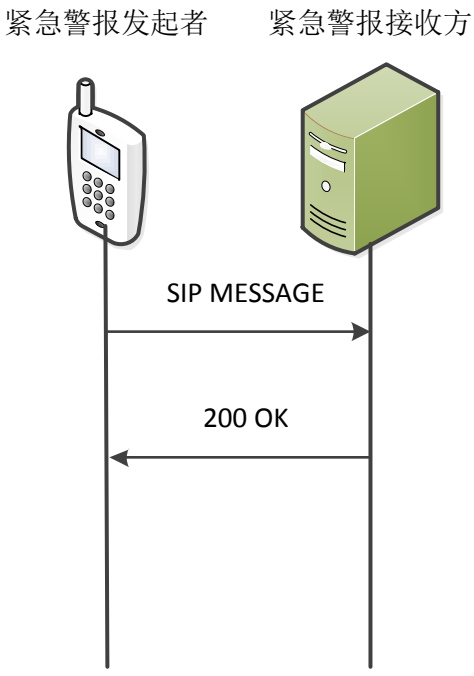


图 4.14 SIP MESSAGE 模式

如图 4.14 所示是 SIP MESSGE 模式，其中的 SIP MESSAGE 消息构成如下：

```
MESSAGE sip:aggregator@emergencycall.com SIP/2.0
Via: SIP/2.0/TCP sensor. emergencycall.com;branch=abcdefghijk
From: sip:sensor@ emergencycall.com;tag=12345
To: sip:aggregator@ emergencycall.com
Call-ID: 123456789@a.b.c.d
Geolocation: <cid:abcdef@ emergencycall.com>;routing-allowed=yes
Supported: geolocation
Accept: application/pidf+xml, application/cap+xml
CSeq: 1 MESSAGE
Content-Type: multipart/mixed; boundary=boundary1
Content-Length: ...
      --boundary1
Content-Type: cap+xml
Content-ID: <abcdef2@ emergencycall.com>
<?xml version="1.0" encoding="UTF-8"?>
    <alert xmlns="urn:oasis:names:tc:emergency:cap:1.1">
       <identifier>abc</identifier>
       <sender>sip:sensor@ emergency.com</sender>
       <sent>2013-01-31T13:28:00-20:00</sent>
       <status>Actual</status>
       <msgType>Alert</msgType>
       <scope>Private</scope>
       <incidents>1234567890</incidents>
       <info>
            <category>Security</category>
            <event>BURGLARY</event>
            <urgency>Expected</urgency>
            <certainty>Likely</certainty>
            <severity>Moderate</severity>
            <senderName>SENSOR</senderName>
            <parameter>
```

```
            <valueName>SENSOR-DATA-NAMESPACE1</valueName>
            <value>123</value>
          </parameter>
        </info>
      </alert>
      --boundary1
```

4.7 紧急呼叫补充信息

4.7.1 补充信息提供流程

与呼叫相关的数据（Data Associated with a Call）：有些补充信息可以作为 SIP 头部或 SIP 消息的组成部分，在 SIP 呼叫建立过程中携带，比如呼叫用户或者服务提供商，包括联络数据、订阅数据、服务数据和设备数据。

与定位相关的数据（Data Associated with a Location）：如第三章所述，定位信息可以通过 PIDF-LO 而获得，其中包括居住位置和地理位置。然而，还可能存在一些补充信息没有包含在 PIDF-LO 中，但与定位信息密切相关，比如提供位置信息的载体，获取定位信息的途径，还有建筑平面图信息，承租人和业主的联络数据等等。

与呼叫用户相关的数据（Data Associated with a Caller）：有些补充信息是用户的私人数据，例如，呼叫救护服务时，用户可以提供自身的医药信息。这些信息也可以在保证紧急呼叫正常进行的情况下，提供给紧急呼叫方。

除此以外，接入网络也有类似的信息对于 PSAP 有用，信息不提供位置定位信息，但是提供关于环境的描述性信息。

4.7.2 基于 SIP 协议扩展的实现方法

补充信息是一系列的信息块，每块都是一个 MIME 类型。

SIP 协议可以提供两种机制来传输数据：

（1） 引用。提供额外数据的 URI，它可以嵌入到 SIP INVITE 或者 SIP MESSAGE 消息头部，比如 Call-Info 头部中包含“emergencyCallData”。如果数据以引用的形式出现，可以以一个 HTTPS Get 从 URI 中获取额外数据。

（2） 数值。SIP INVITE 或者 SIP MESSAGE 消息体内加入额外数据来实现传输。

INVITE sips:abc@xyz.emergencycall.com SIP/2.0

Via: SIPS/2.0/TLS pc33. abcdef. emergencycall.com;branch=z9hG4bK74bf9

To: XYZ <sips: abc@xyz.emergencycall.com >

From: DET <sips: def@det.emergencycall.com >;tag=dfsfe3tf57g

Call-ID: 123456789@ det.emergencycall.com

Call-Info: < http://www. emergencycall.com/ det /> ;purpose=emergencyCallData

Geolocation: <cid:target123@abcdef. emergencycall.com>

Geolocation-Routing: no

Accept: application/sdp, application/pidf+xml

CSeq: 2 INVITE

Contact: <sips:alice@ abcdef. emergencycall.com>

Content-Type: multipart/mixed; boundary=boundary1

Content-Length: ...

--boundary1

Content-Type: application/sdp

...SDP 的信息

--boundary1

Content-Type: application/pidf+xml

Content-ID: <target123@ abcdef. emergencycall.com>

...PIDF-LO 的信息

--boundary1--

4.8 本章小结

本章主要是设计与实现 VoIP 紧急呼叫中的 SIP 流程，分析了开源协议栈 oSIP 的 SIP 解析模块、SIP 状态机模块和 SIP 事件与回调函数，在此基础上设计了紧急呼叫 SIP 消息的编码。除此之外，还讨论了 VoIP 紧急呼叫技术的几个基础应用。一个是通过 HELD 协议确定用户位置的设计。一个是基于 LoST 发现服务中心服务完成紧急呼叫号码自动获取的设计。一个是 PSAP 内部处理设计。一个是只有数据的紧急呼叫，通过扩展 SIP MESSAGE 的消息结构来实现此功能。一个是紧急呼叫中补充信息的携带，通过扩展 SIP INVITE 的消息结构来实现此功能。

第五章 VoIP 紧急呼叫系统软件实现

本章在实验室环境下，对基于 VoIP 的紧急呼叫系统原型进行主要用户功能的实现及测试，对测试结果进行了分析与总结。软件环境是安装了 SIP 协议栈、HTTP 协议栈的实验室电脑。运行环境是 win7，无防火墙。

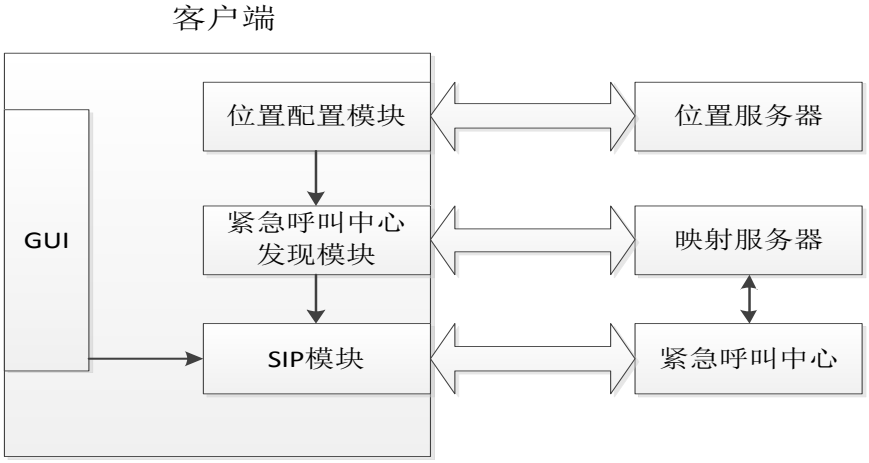


图 5.1 E-VoIP 紧急呼叫模块关系图

如图 5.1 所示，软件实现的 VoIP 紧急呼叫系统中客户端包含如下三个主要模块：位置配置模块，紧急呼叫中心发现模块和 SIP 模块。位置配置模块通过 HELD 协议与位置服务器进行交互，紧急呼叫中心发现模块通过 LoST 协议与映射服务器进行交互，SIP 模块通过扩展的 SIP 模块与呼叫中心进行交互。

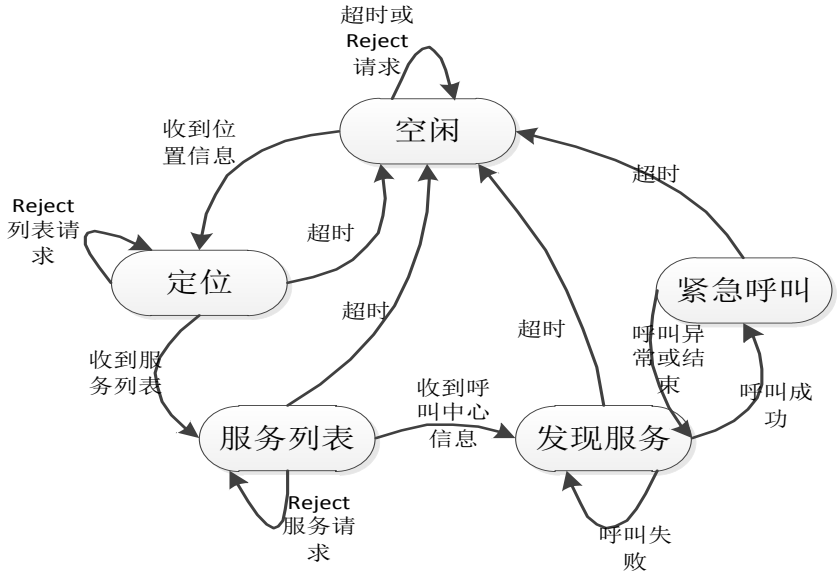


图 5.2 E-VoIP 紧急呼叫客户端状态机

E-VoIP 紧急呼叫客户端状态机如图 5.2 所示。

此客户端状态机一共有五个状态：“空闲”状态，“定位”状态，“服务列表”状态，“发现服务”状态和“紧急呼叫”状态。

（1） 首先当客户端开机时，状态机初始化为“空闲”状态，然后将位置信息请求 LocationRequest 发送给位置服务器，如果超时或者收到 Reject 请求，则重发 LocationRequest 请求，如果正常收到位置信息，则状态机跳转到为“定位”状态，如果在。

（2） 客户端处于“定位”状态时，发送 ListServicesByLocationRequest 请求，如果正常收到 ListServicesByLocationResponse 响应，则跳转到“服务列表”状态，如果定时器超时（说明需要重新定位），状态机跳转到“空闲状态”，否则如果收到 Reject 请求，则重发 ListServicesByLocationRequest 请求，状态机保持在“定位”状态。

（3） 客户端处于“服务列表”状态时，发送 FindServiceRequest 请求，如果正常收到 FindServiceResponse 响应，则跳转到“发现服务”状态，如果定时器超时（说明需要重新定位），状态机跳转到“空闲状态”，否则如果收到 Reject 请求，则重发 FindServiceRequest 请求，状态机保持在“服务列表”状态。

（4） 客户端处于“发现服务”状态时，通过紧急呼叫号码自动获取等过程，如果发起 SIP 呼叫，并呼叫连接成功建立，则跳转到“紧急呼叫”状态，如果定时器超时（说明需要重新定位），状态机跳转到“空闲”状态，否则如果收到 Reject 请求，则重发 SIP INVITE 请求，状态机保持在“发现服务”状态。

（5） 客户端处于“紧急呼叫”状态时，如果出现呼叫异常，则状态机跳转回到“发现服务”状态，重新发起呼叫，直到成功接入，如果定时器超时，再从“空闲”状态重新发起定位请求，最后如果呼叫正常结束，则状态机回到“发现服务”状态，等待下一次紧急呼叫。

5.1 位置配置模块

位置配置模块，采用 HELD 协议完成 E-VoIP 紧急呼叫客户端 “空闲”状态与“定位”状态的编程切换。

HELD 协议的流程实现如下：

本实现基于 HTTP 开源代码，在此基础上，实现 HELD 协议的基本流程设计。

（1） 创建 HTTP 栈

使用 thttp_stack_create (thttp_stack_callback_f callback, ...)创建一个 HTTP 栈。

第一个参数指向一个 callback 函数，用于处理到达传输层的新消息。后面跟着的参数是配置参数，可以使用 THTTP_STACK_SET_*() 宏来配置它们，参数列表必须以 THTTP_STACK_SET_NULL()结尾。

（2） 启动 HTTP 栈

使用 thttp_stack_start(thttp_stack_handle_t *self)启动一个 HTTP 栈。

因为 HTTP 栈是 C 类，最后可以使用 TSK_OBJECT_SAFE_FREE()宏来删除。

（3） 创建和配置 Session

使用 thttp_session_create()来创建和配置 Session。使用如下：

```
thttp_session_handle_t * session = thttp_session_create(stack,
    THTTP_SESSION_SET_CRED("abcde", "open same"),
    THTTP_SESSION_SET_OPTION(THTTP_SESSION_OPTION_TIMEOUT, "6000"),
    THTTP_SESSION_SET_HEADER("Accept: ", " application/held+xml"),
    THTTP_SESSION_SET_HEADER("Connection", "Keep-Alive"),
    THTTP_SESSION_SET_HEADER("Content-Type ", " application/held+xml "),
    THTTP_SESSION_SET_NULL());
```

（4） HTTP 请求

HTTP 共有 9 中请求，分别是 CONNET, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT 和 TRACE。本次实现采用 POST 方法实现 HTTP 请求

```
thttp_action_POST(session, "http://127.0.0.1:6000",
    THTTP_ACTION_SET_OPTION(THTTP_ACTION_OPTION_TIMEOUT, "2500"),
    THTTP_ACTION_SET_HEADER("Accept ", " application/held+xml "),
    THTTP_ACTION_SET_HEADER("Connection", "Keep-Alive"),
    THTTP_ACTION_SET_HEADER("Content-Type", " application/held+xml "),
    THTTP_ACTION_SET_PAYLOAD(buff0,strlen(buff0)),
    THTTP_ACTION_SET_NULL());
```

下面进行位置配置模块的测试：

No	Protocol	Src MAC	Dest MAC	Src IP	Dest IP	Src Port	Dest Port	Time
1	IP/UDP	00:00:00:00:0...	00:00:00:00:...	117.89.254.93	127.254.89.11...	netbios-...	netbios-...	00:...
2	IP/UDP	00:00:00:00:0...	00:00:00:00:...	192.168.137.1	192.168.137.255	netbios-...	netbios-...	00:...
3	IP/UDP	00:00:00:00:0...	00:00:00:00:...	117.89.254.93	127.254.89.11...	netbios-...	netbios-...	00:...
4	IP/TCP	00:00:00:00:0...	00:00:00:00:...	HUANG	HUANG	6561	6000	00:...
5	IP/TCP	00:00:00:00:0...	00:00:00:00:...	HUANG	HUANG	6000	6561	00:...
6	IP/TCP	00:00:00:00:0...	00:00:00:00:...	HUANG	HUANG	6561	6000	00:...
7	IP/TCP	00:00:00:00:0...	00:00:00:00:...	HUANG	HUANG	6561	6000	00:...
8	IP/TCP	00:00:00:00:0...	00:00:00:00:...	HUANG	HUANG	6000	6561	00:...
9	IP/TCP	00:00:00:00:0...	00:00:00:00:...	HUANG	HUANG	6000	6561	00:...
10	IP/TCP	00:00:00:00:0...	00:00:00:00:...	HUANG	HUANG	6561	6000	00:...

图 5.3　HELD 协议流程抓包图

HELD 协议流程的抓包结果如图 5.3 所示，其中 SrcIP 和 DestIP 均为 HUANG 是回环数据包，是传输相关流程的数据包。

在此过程中，客户端占用端口 6544，服务器占用端口 6000，总共传输了两个包，一个 594byte，一个 279byte。

提取出来分析可以得到 TCP 流程如下

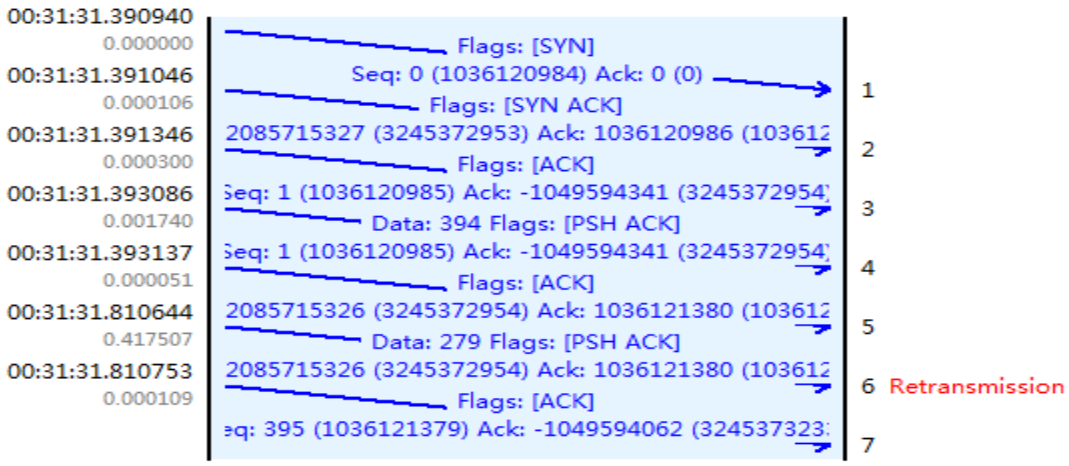


图 5.4　HLED 协议的 TCP 流程

如图 5.4 所示，首先是 TCP 三次握手，然后传输携带 HLED 协议数据的两个数据包交互。

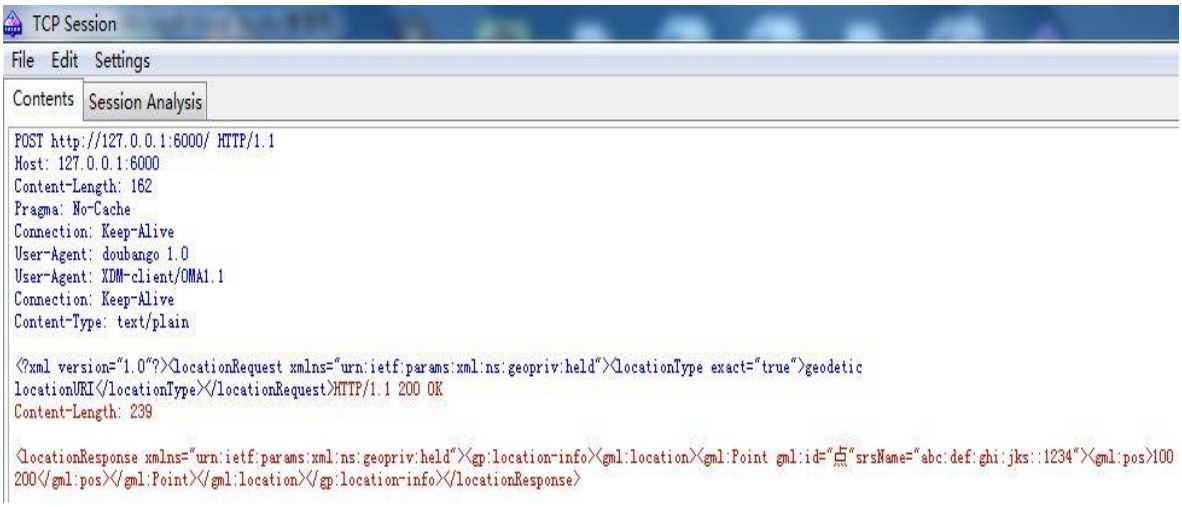


图 5.5　HELD 协议流程中所传输数据

由图5.5可以看到，携带HLED协议数据的两个数据包内容，都是基于HTTP协议的HELD流程中所传输的数据。第一个是 HTTP POST 请求，正文中是携带 XML 格式的数据LocationRequest，第二个是 HTTP 200 OK 的响应，正文中是携带 XML 格式的数据LocationResponse。

5.2 紧急呼叫中心发现模块

紧急呼叫中心发现模块，采用 LoST 协议完成 E-VoIP 紧急呼叫客户端 “空闲”状态、“服务列表”状态、“发现服务”状态之间的编程切换。

与 HELD 协议类似，LoST 协议同样基于 HTTP 开源代码。LoST 协议客户端的基本流程实现如下：

```
int fsm(thttp_event_t* evt）//LoST 消息的处理状态机部分（接收 LoST 消息）{
    int reval=-1;
    switch(cur_state){
    case 0: {                       //“空闲”状态
    //不是 LoST 需要处理的操作
        break;
    }
    case 1: {                       //“定位”状态
        if(TimeOutEvent）{
            cur_state = 0; //“空闲”状态
        }
        else if(RecivedListServicesByLocationResponseEvent){
            lost_fsm_find_service_request(evt);
            cur_state = 2;      //“服务列表”状态
        }
        else if(RejectListServicesByLocationRequestEvent)
        {//请求失败，再次请求或者结束
            cur_state = 1;      //“定位”状态
        }
        else{
```

```
                lost_fsm_list_services_by_location_request(evt);
                cur_state = 1;//“定位”状态
            }
            break;
        }
        case 2:{                    //“服务列表”状态
            if(TimeOutEvent）{
                cur_state = 0; //“空闲”状态
            }
          else if(RecivedFindServiceResponseEvent)
          {//分析 Response
                cur_state = 3;     //“发现服务”状态
            }
            else if(RejectFindServiceRequestEvent)
            {//请求失败，再次请求或者结束
                cur_state = 2;     //“服务列表”状态
            }
            else{
                lost_fsm_find_service_request_request(evt);
                cur_state = 2;     //“服务列表”状态
            }
            break;
        }
        //其他状态
}
```

服务器的状态机实现与客户端类似。

下面进行紧急呼叫中心发现模块的测试。

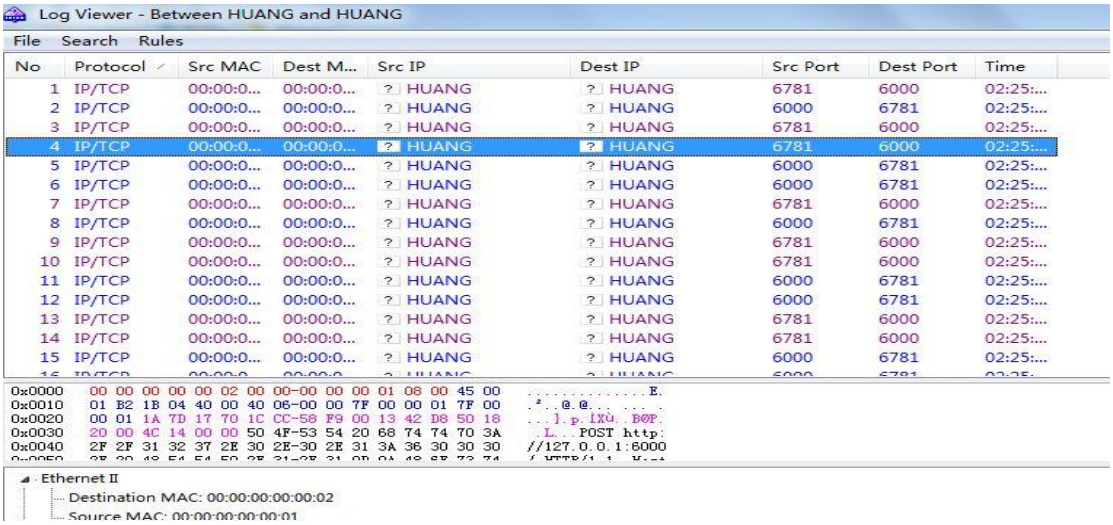


图 5.6　LoST 协议流程抓包图

LoST 协议流程的抓包结果如图 5.6 所示，其中 SrcIP 和 DestIP 均为 HUANG 是回环数据包，是传输相关流程的数据包。

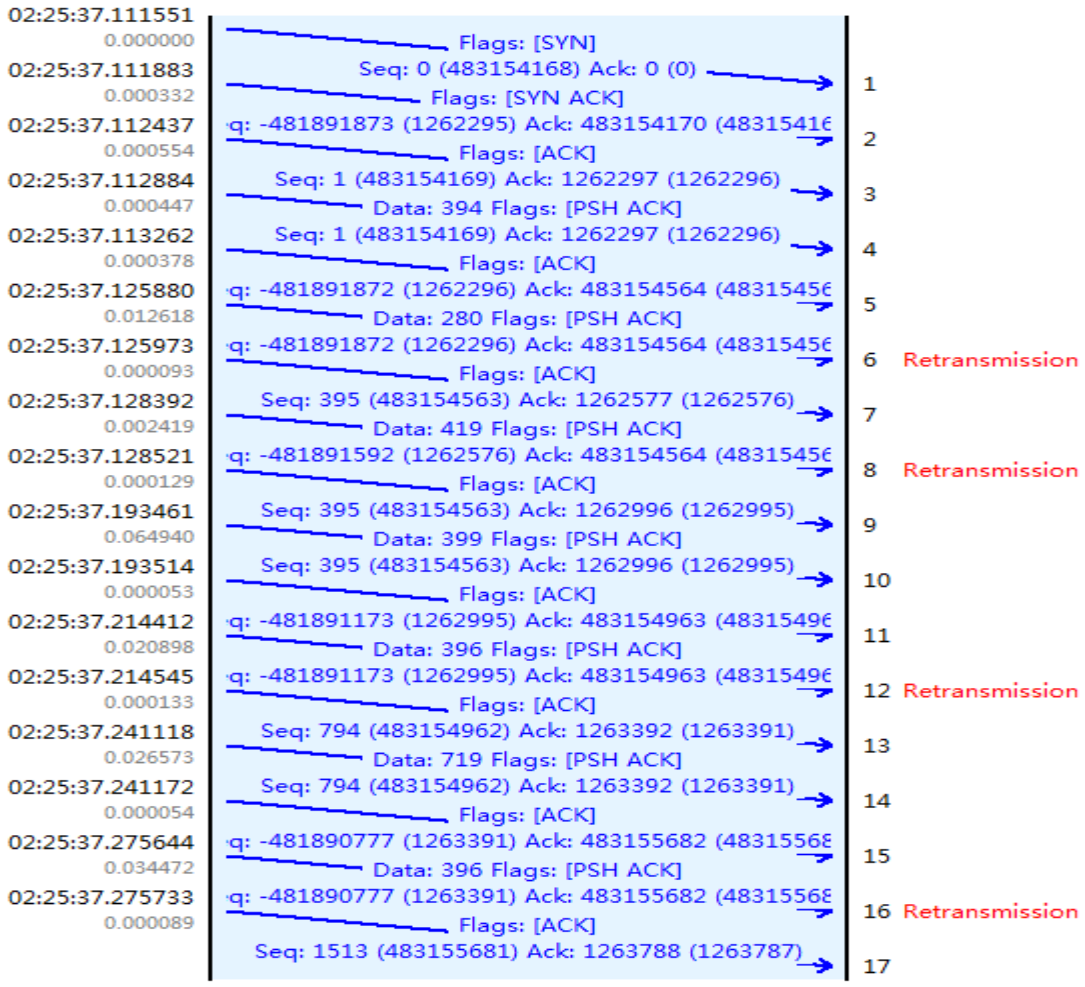


图 5.7　客户端的 TCP 流程

如图 5.7 所示，首先是 TCP 三次握手，然后传输携带 HLED 协议数据的 2 个数据包交互

和 LoST 协议数据的 4 个数据包交互。

下面主要分析 LoST 协议的 4 个数据包。

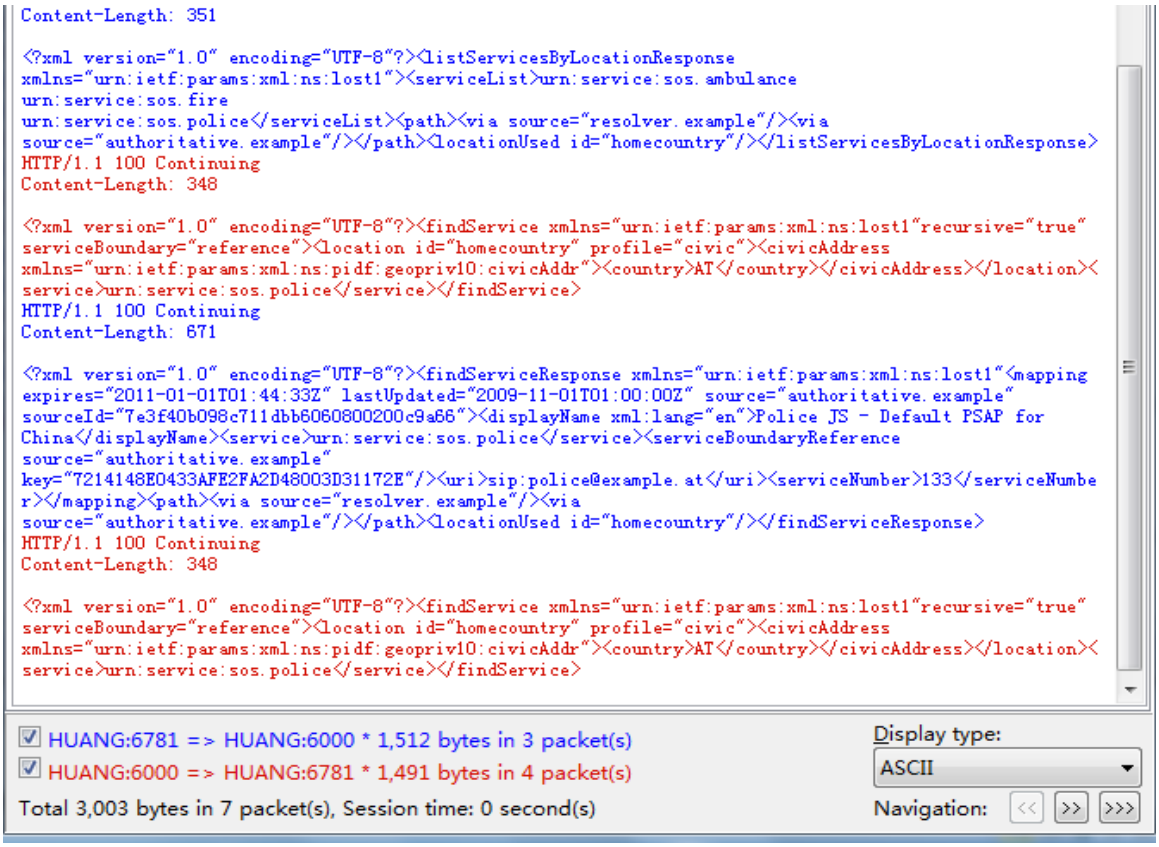


图 5.8 LoST 协议流程中所传输数据

由图 5.8 可以看到，携带 LoST 协议数据的四个数据包内容，都是基于 HTTP 协议的 LoST 流程中所传输的数据。

（1） 第一个是 HTTP POST 请求，正文中是携带 XML 格式的数据 ListServicesByLocation

（2） 第二个是 HTTP 100 continue，正文中是携带 XML 格式的数据 ListServicesByLocationResponse

（3） 第三个同样是 HTTP 100 continue，正文中是携带 XML 格式的数据 FindService

（4） 第四个还是 HTTP 100 continue，HTTP 的 session 还没有结束，正文中是携带 XML 格式的数据 FindServiceResponse

5.3 SIP 模块

SIP 模块，采用扩展的 SIP 协议完成 E-VoIP 紧急呼叫客户端 “空闲”状态、“紧急呼叫”状态、“发现服务”状态之间的编程切换。

5.3.1 紧急呼叫 SIP 消息的编码实现

SIP INVITE 消息的编码设计

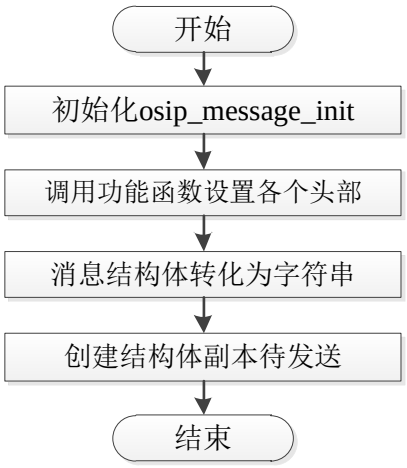


图 5.9 SIP 消息的构造

如图 5.9 所示，SIP 消息的构造首先是通过函数 osip_header_init、函数 osip_message_init 等对起始行、头部等结构体进行初始化，再通过调用各个不同的功能函数 osip_header_set_name、函数 osip_header_set_value、函数 osip_message_set_reason_phrase、函数 osip_message_set_status_code 、函数 osip_message_set_method 、函数 osip_message_set_version、函数 osip_message_set_uri 等对 SIP 消息中的各项内容进行添加，调用函数 osip_message_to_str、函数 osip_header_to_str 等对 SIP 消息结构体中的信息转化为可以发送的字符串，调用函数 osip_message_clone、函数 osip_header_clone 等为 SIP 消息结构体创建副本（以防重新发送），最后调用函数 osip_message_free、、函数 osip_header_free 等对 SIP 消息结构体进行释放。

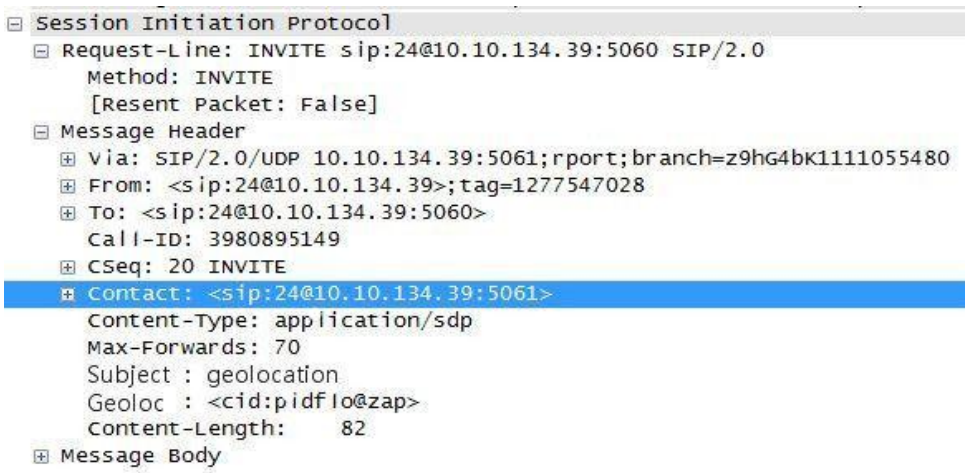


图 5.10 SIP 协议的消息编码

如图 5.10 所示是基于位置信息扩展 SIP 协议消息编码

5.3.2 紧急呼叫 SIP 消息的解码实现

SIP INVITE 消息的解析设计

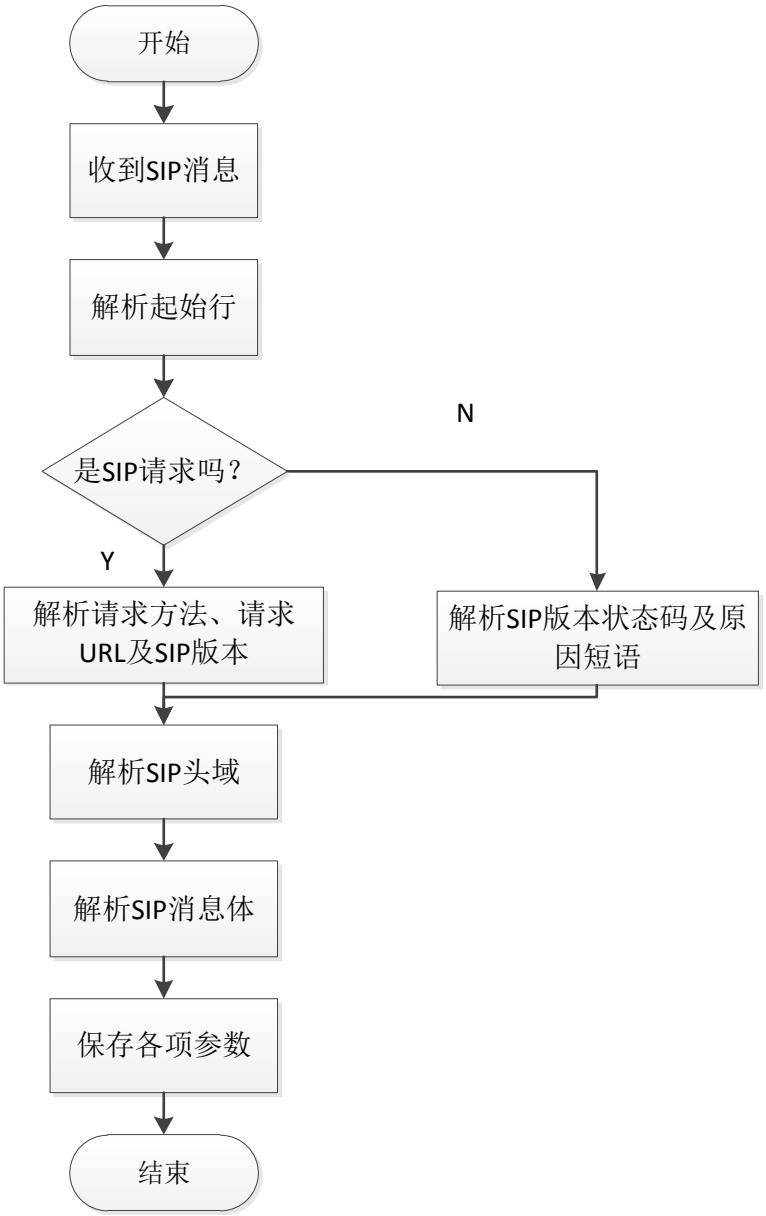


图 5.11 SIP 消息解析设计

如图 5.11 所示，在接收到一个 message 时，调用函数 osip_message_parse 进行 message 的解析

（1） 函数 MyosipMessageExchange

替换掉 message 中的连续出现的 ‘\r\n\t’、 ‘\r\t’、 ‘\n\t’、 ‘\r\n ’、‘\r ’、‘\n ’为空格

message 的 headers 和 body 之间的分界是以‘\r\n\r\n’为标志的

每个 headers 都是起新行，而且新行的头一个字符不为空格或‘\t’，所以两个 header 之间的‘\r\n’不会被替换掉

例如：

INVITE urn:service:sos.fire SIP/2.0

Route: <sip:police@example.at;lr>

Via: SIP/2.0/TCP 192.0.2.1:5060;rport;

branch=z9hG4bK4r1id0

Via 允许有多个值出现，而且允许分行，但是分行必须以空格或\t 开头，而 INVITE 和 Route 行必需顶格开始

（2） 函数 MyosipMessageStartline

一个 message 由三部分组成，首先是 message 的起始行部分，该行指明这是一个 sip 的 message，包括 sip 标志，请求或应答说明，状态值，然后以‘\r\n’做为和 headers 的分隔符

用函数 MyosipMessageStartline 对起始行进行解析。

解析得到 message 的类型，message 的 sipversion 以及 message 的 status_code，当 status_code 为初始化值 0 时，该 message 为一个请求，否则为应答

如果是请求，调用函数 MyosipMessageStartlineParsereq

如果是应答，调用函数 MyosipMessageStartlineParseresp

解析结果保存在 osip_message 的结构成员变量中

```
struct osip_message {
/*SIP 请求*/
    char *sip_version;          /*版本号*/
    osip_uri_t *req_uri;        /*请求 URI */
    char *sip_method;           /*方法*/
/*SIP 响应*/
    int status_code;            /*状态码  */
……
//起始行结束
……
    osip_call_id_t *call_id;            /*Call-ID */
    osip_list_t contacts;               /*Contacts */
    osip_content_length_t *content_length;
    osip_content_type_t *content_type;
    osip_cseq_t *cseq;                  /*CSeq*/
```

```
    osip_from_t *from;              /*From */
    osip_list_t routes;             /* Route */
    osip_to_t *to;                  /* To */
    osip_list_t vias;               /*Vias */
    osip_list_t supported;          /*新加入的字段*/
    osip_list_t geolocation;        /*新加入的字段*/
    osip_list_t headers;            /*其他*/
    osip_list_t bodies;
  };
```

（3）解析 messge 的 headers 部分，调用函数 MyosipHeadersParse

头部采用通用格式，结构定义如下：

```
struct osip_header
{
   char *hname;        /* Name of header */
   char *hvalue;       /*Value for header */
 };
```

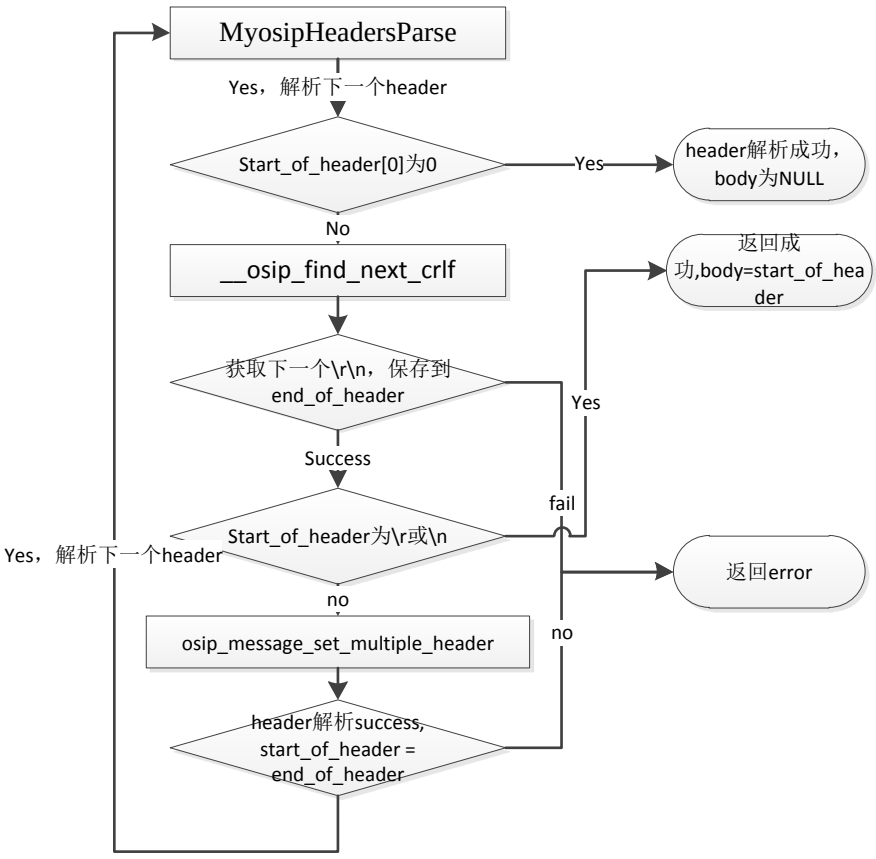


图 5.12 MyosipHeadersParse 函数流程

如图 5.12 所示，字符串开头保存在 start_of_header 中。调用函数__osip_find_next_crlf 找到这个 header 的结束字符并保存在 end_of_header 中。根据 header 内部分隔符“：”，取出 header 的 hname 和 hvalue。

osip_message_set_multiple_header 来解析该 header 的 hvalue 字符串。

在 osip_message_set_multiple_header 中，将 headers 分为两类，一类如上面例子中的 Subject，只允许一个值，则直接调用 osip_message_set__header 进行解析；一类如上面例子中的 Via，允许多个值，根据 sip 协议，每个值之间以“，”进行分隔，所以需要查询整个 hvalue 字符串，根据”,”将 hvalue 分隔成多个值，每个值调用 osip_message_set__header 进行解析并保存解析结果到 osip_message 的数据成员变量中。

解析成功后，置 start_of_header 为已经解析完的 header 的 end_of_header，开始解析下一个 header。

header 头解析所使用到的全局管理变量__osip_message_config_t 的结构定义如下：

```
typedef struct ___osip_message_config_t
{
   char *hname;
   int (*setheader) (osip_message_t *, const char *);
   int ignored_when_invalid;
}__osip_message_config_t;
```

（4） 函数 msg_osip_body_parse

如果 message 中在 headers 之后不是结束符’\0’，则继续解析 message 的负载部分，调用函数 msg_osip_body_parse 进行解析

根据编码类型（content_type）和编码内容长度（content_length）两个标题的值来判断消息体 body 是否有值。

Message 的 body 解析首先查询 headers 头解析中保存的 content——即 body－－的属性：content_type，如果 content_type 中的 type 不为 multipart，即不支持多种 mime 方式的 content，说明 body 中就一个编码方式，直接将整个 body 解析为一个内容；如果 type 为 multitype，说明有多个编码方式的 body 组合在一起形成一个整体的 body，则以”--”为分隔符解析 body，将 body 分为多个 mime 编码方式的字符串，每个解析后的 body 内容保存在 osip_message 结构中的 bodies 结构成员中。

下面是紧急呼叫 SIP 消息编解码测试结果。

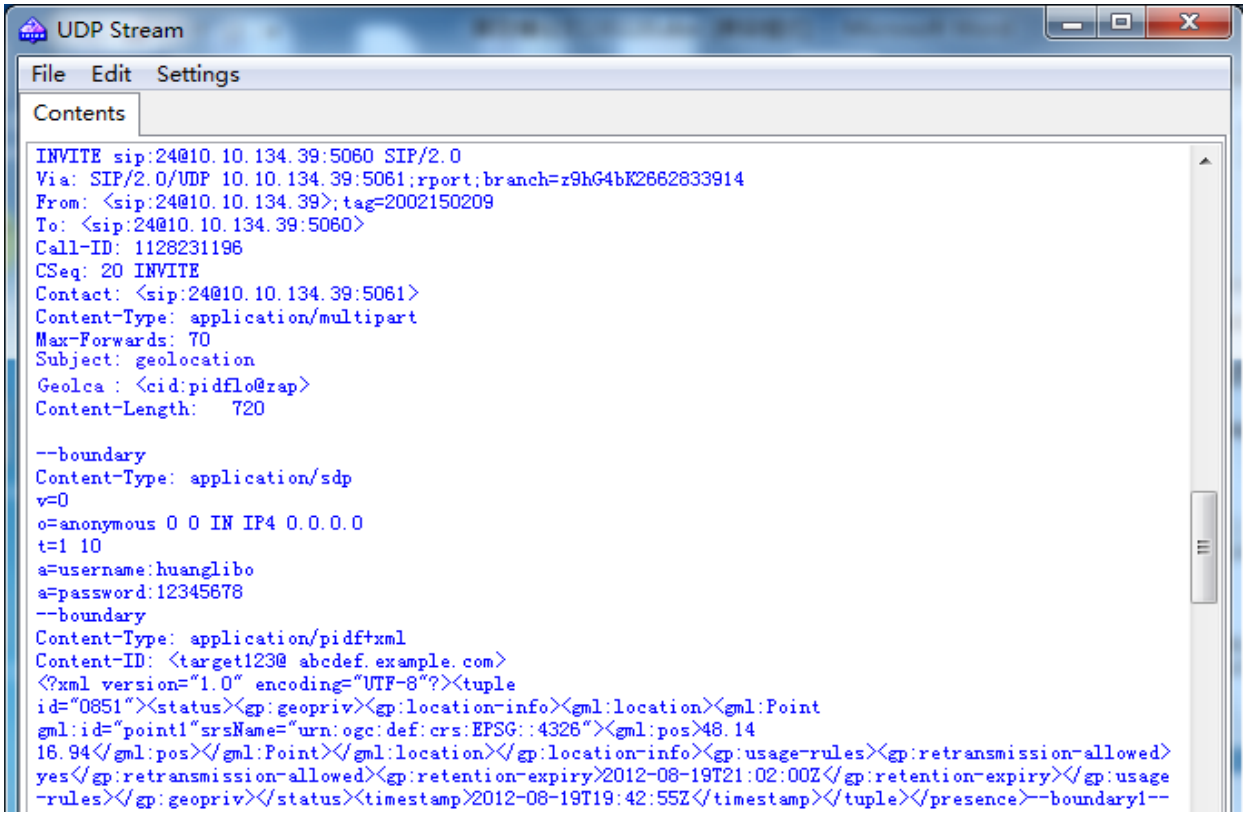


图 5.13 SIP INVITE 消息

如图 5.13 所示是 SIP INVITE 消息内容，消息解析的 XML 文件如下所示

图 5.14 PIDF-LO 的 XML 文件

5.3.3 数据紧急呼叫 SIP 消息实现

构建 MESSAGE 消息的步骤：

（1） 函数 MyMessageBuildReq ();

此函数只需调用函数 generating_request_out_of_dialog()

（2） 函数 osip_message_set_body ();函数 osip_message_set_content_type ()

用来设置内容与编码类型

（3） 函数 MyMessageSendReq ();

事务 transaction 初始化，在创建完 transaction 之后，生成一个 transaction 上的 event，将该 event 添加到该 transaction 的 event 队列中。

服务器中也需要对函数 MyMessageBuildAnswer ()和函数 MyMessageSendAnswer ()进行相同操作。

下面是数据紧急呼叫 SIP 消息的软件测试结果及分析。

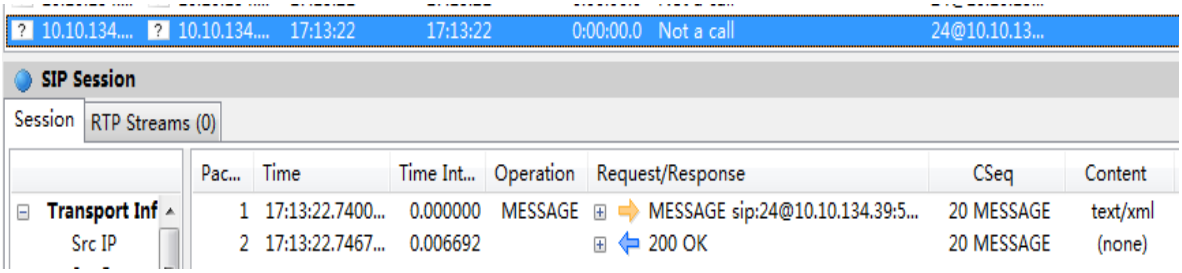


图 5.15 扩展的 SIP MESSAGE 流程实现

如图 5.15 可知是呼叫方 UAC 向接收方 UAS 直接呼叫的完整 SIP MESSAGE 流程，包含 MESSAGE 和 200OK。

SIP MESSAGE 流程的具体消息结构如下图 5.16 所示：

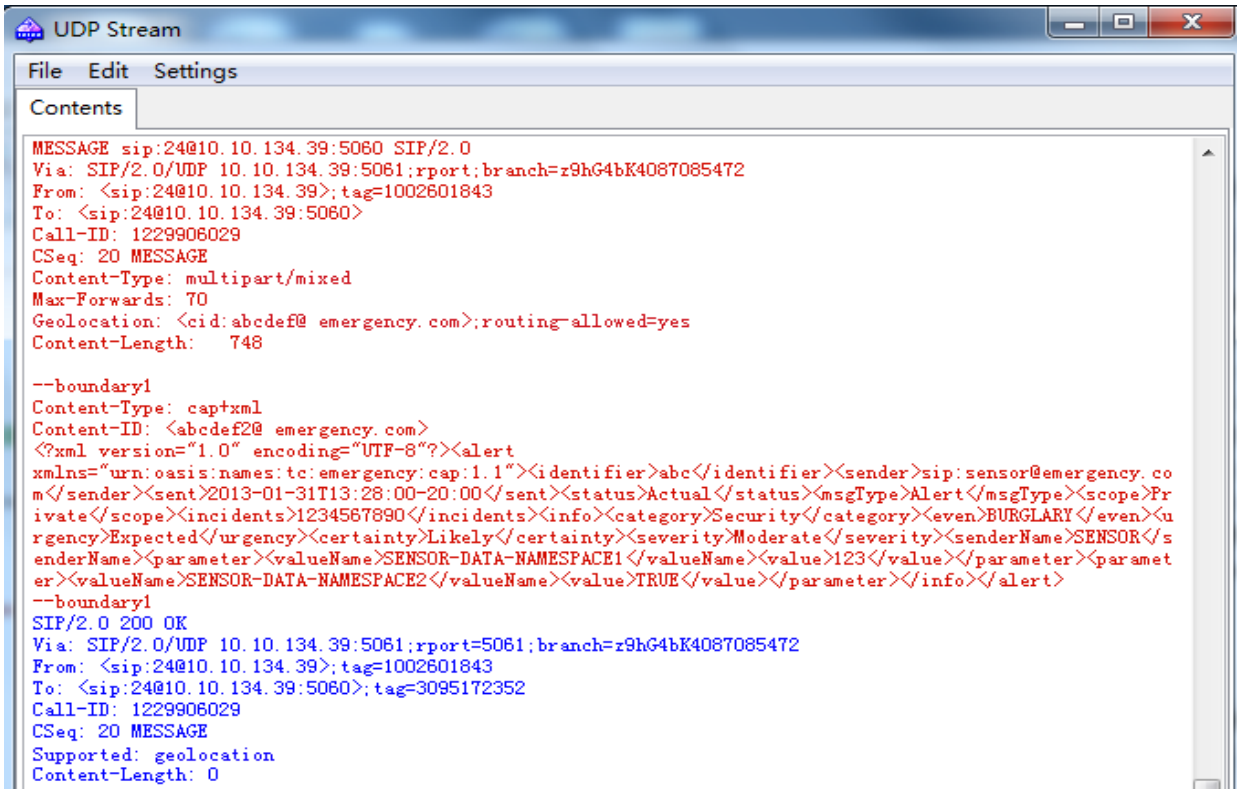


图 5.16 扩展的 SIP MESSAGE 消息分析

5.3.4 VoIP 紧急呼叫的 SIP 呼叫控制实现

论文采用 UDP socket 套接字实现底层 SIP 协议的接收和发送[42]。

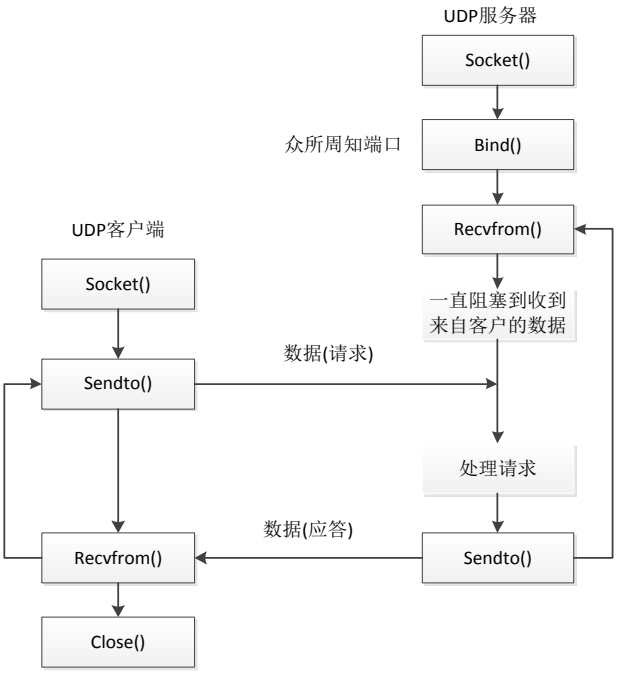


图 5.17 UDP 客户端服务器所用套接字函数

如图 5.17 所示，UDP 客户端首先调用 socket()函数，接着调用 sendto()函数把数据发往 UDP 服务器，并调用 recvfrom( )函数准备接受 UDP 服务器的响应。

UDP 服务器同样也调用 socket( )函数，接着调用 bind()函数指定本地地址/端口，调用 recvfrom( )函数，一直阻塞到接收到 UDP 客户端发来的请求，处理请求，最后通过 sendto()函数把响应发往 UDP 客户端。

下面基于 UDP socket 建立最基本的两方呼叫建立和释放的 SIP 呼叫控制流程。

在客户端发送 INVITE 请求前，需要完成如下操作：

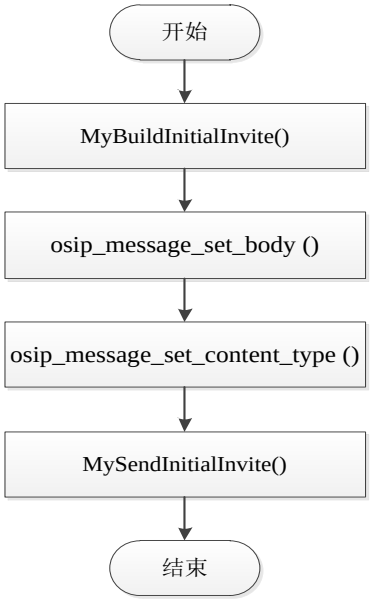


图 5.18 发送 INVITE 请求前初始化流程

（1） 函数 MyBuildInitialInvite，初始化一个 INVITE 请求呼叫。

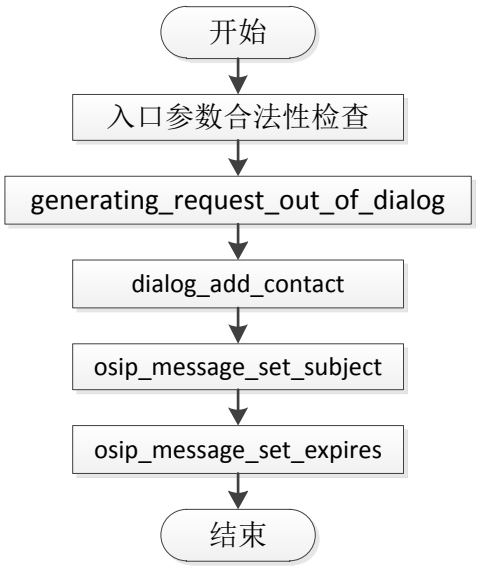


图 5.19 创建 Invite 初始化流程

如图 5.19 所示，在创建一个新的 INVITE 时，而且该 INVITE 是 call 的第一个 INVITE，则需要检查必要参数目的端地址 to 的合法性。

在检查合法性之后，调用通用的请求构造函数 generating_request_out_of_dialog 构造生成一个 request message，在构造参数中指定要构建的是一个 INVITE。因为是在 dialog 创建之前构建 INVITE 请求，所以调用的接口为 out_of_dialog 的，即不需要从 dialog 中获取信息。

在创建成功通用的 dialog 外的请求后，添加 INVITE 相关部分字段，包括 contact，subject 和超时时间 expires。其中 contact 之间使用本端内部设置的 IP 地址。

（2） 设置 INVITE 请求的内容与编码类型，分别调用函数 eXosip_call_build_ack ()与函数

eXosip_call_send_ack ()

（3） 函数 MySendInitialInvite

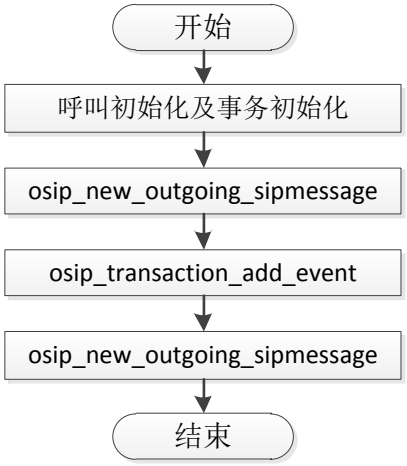


图 5.20 呼叫和事务初始化

INVITE 请求呼叫发送前进行呼叫初始化与事务初始化，在创建完 call 和 transaction 之后，根据要发送的 INVITE 生成一个 transaction 上的 event，将该 event 添加到该 transaction 的 event 队列中。

最后调用 socket 套接字，发送接收数据。

同样服务端方面需要进行如下操作

（1） 初始化 osip 函数库，并打开信令传输端口。

（2） 接收到请求程序后，解析 SIP 消息

（3） 调用 MyBuildAnswer 函数对方的请求建立相应的响应，根据 transaction id 查询得到 call、dialog、transaction 的结构。

（4） 调用 MySendAnswer 函数

首先进行参数的合法性检查，再根据 transaction id 查询得到 call、dialog、transaction 的结构，如果没有找到，则返回错误。最后如果没有错误，把建立的响应发送给客户端。

下面是 VoIP 紧急呼叫的 SIP 呼叫流程的软件测试结果及分析。

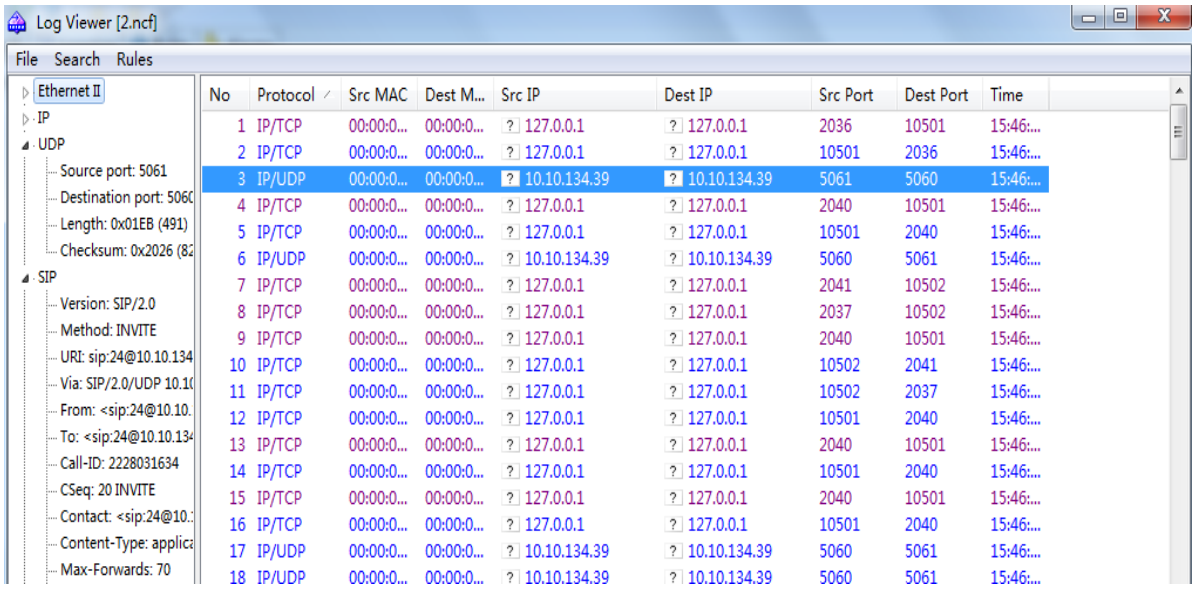


图 5.21　　VoIP 紧急呼叫的 SIP 呼叫流程抓包图

如图 5.21 所示，是 VoIP 紧急呼叫的 SIP 呼叫流程抓包图，其中 SrcIP 和 DestIP 都是 10.10.134.39 的是 SIP 呼叫的相关流程，把它们提取如下图 5.22 所示：

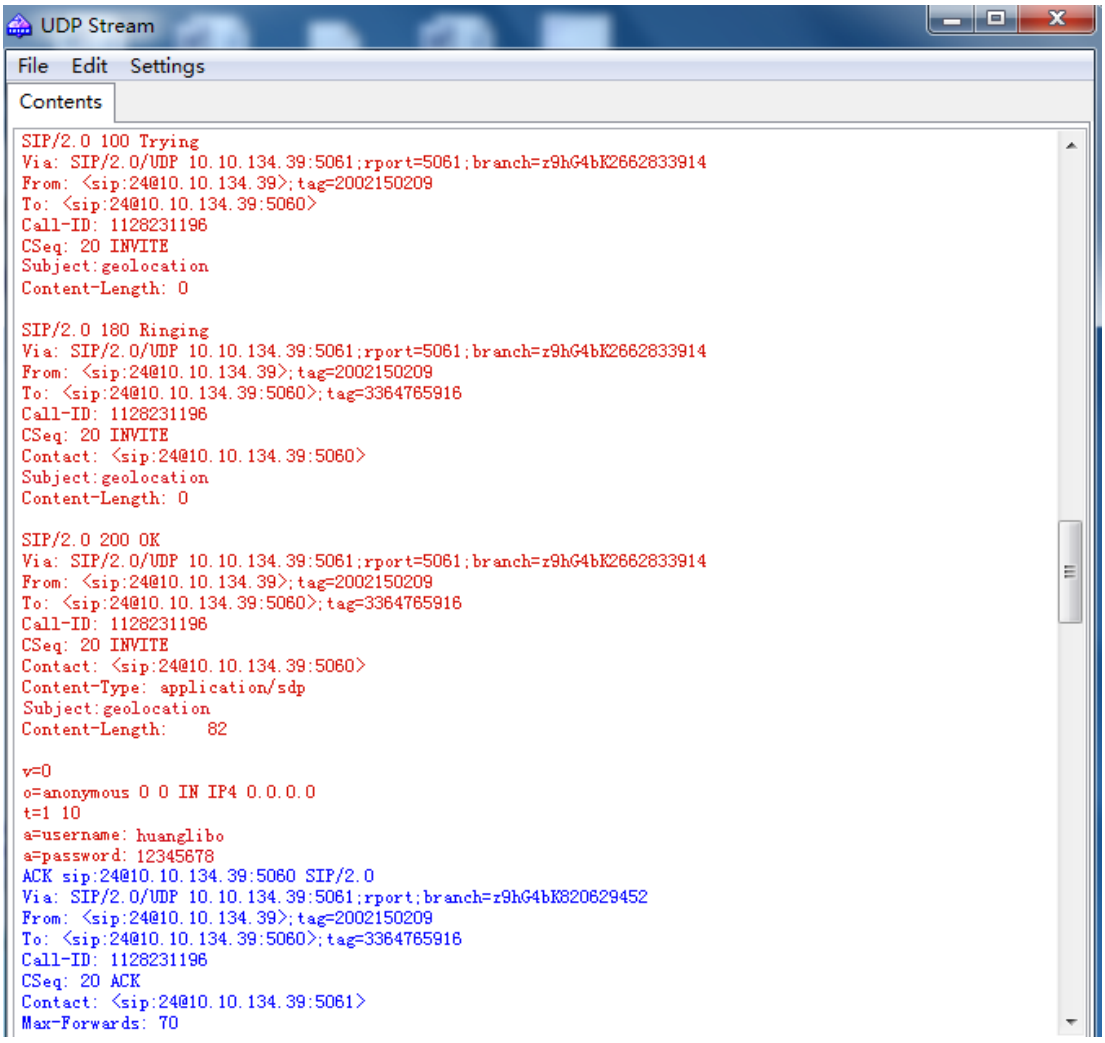


图 5.22　　SIP INVITE 流程的消息分析

Pac...	Time	Time Int...	Operation	Request/Response	CSeq	Content
1	19:52:47.3149...	0.000000	INVITE	INVITE sip:24@10.10.134.39:5060	20 INVITE	applicatio...
				Header		
				Content		
2	19:52:47.3264...	0.011481		100 Trying	20 INVITE	(none)
3	19:52:47.3299...	0.003528		180 Ringing	20 INVITE	(none)
4	19:52:47.3301...	0.000112		200 OK	20 INVITE	SDP
5	19:52:47.3327...	0.002592		ACK sip:24@10.10.134.39:5060	20 ACK	(none)

ACK sip:24@10.10.134.39:5060

图 5.23　　SIP 呼叫流程实现

如图 5.23 可知是呼叫方 UAC 向接收方 UAS 直接呼叫的完整 SIP 呼叫控制流程，包含 INVITE、100 Trying、180 Ringing、200 OK 及 ACK。

5.4 PSAP 内部处理模块

在 PSAP 内部，论文实现采用紧急呼叫接线员的方式来集中处理所有的紧急呼叫请求。如果用户需要特定的服务，可以转给特定紧急处理方。

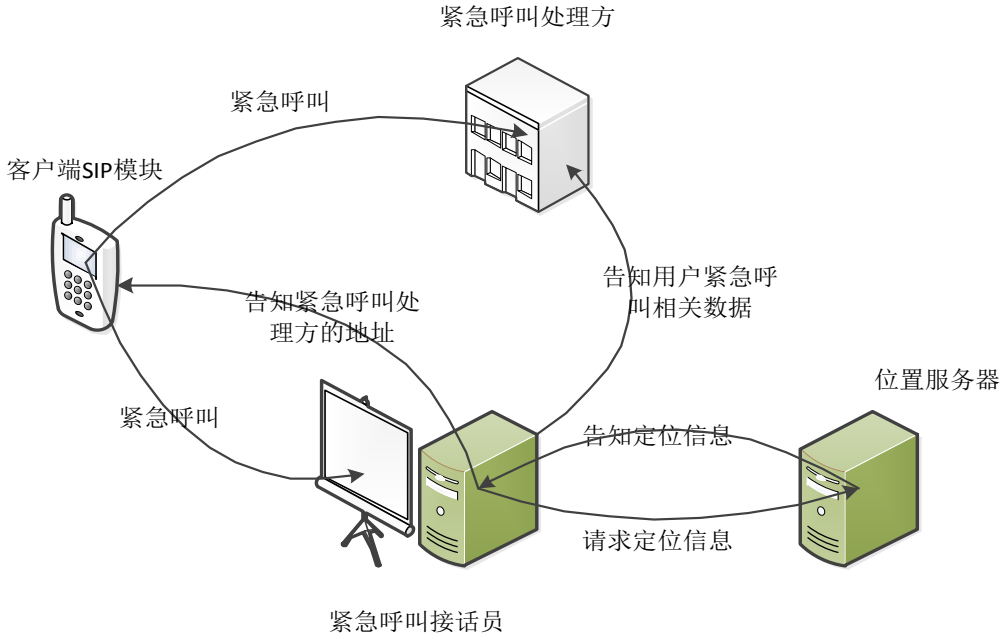


图 5.24　　PSAP 内部处理架构

图 5.24 是实现的 PSAP 内部处理架构，其中包含客户端的 SIP 模块、紧急呼叫接话员、位置服务器和紧急呼叫处理方。客户端 SIP 模块发起紧急呼叫，接话员接到紧急呼叫后可以把相应处理工作转给紧急呼叫处理方，并提供呼叫处理方相关数据。

图 5.25 是基于此设计架构的 PSAP 内部处理流程。

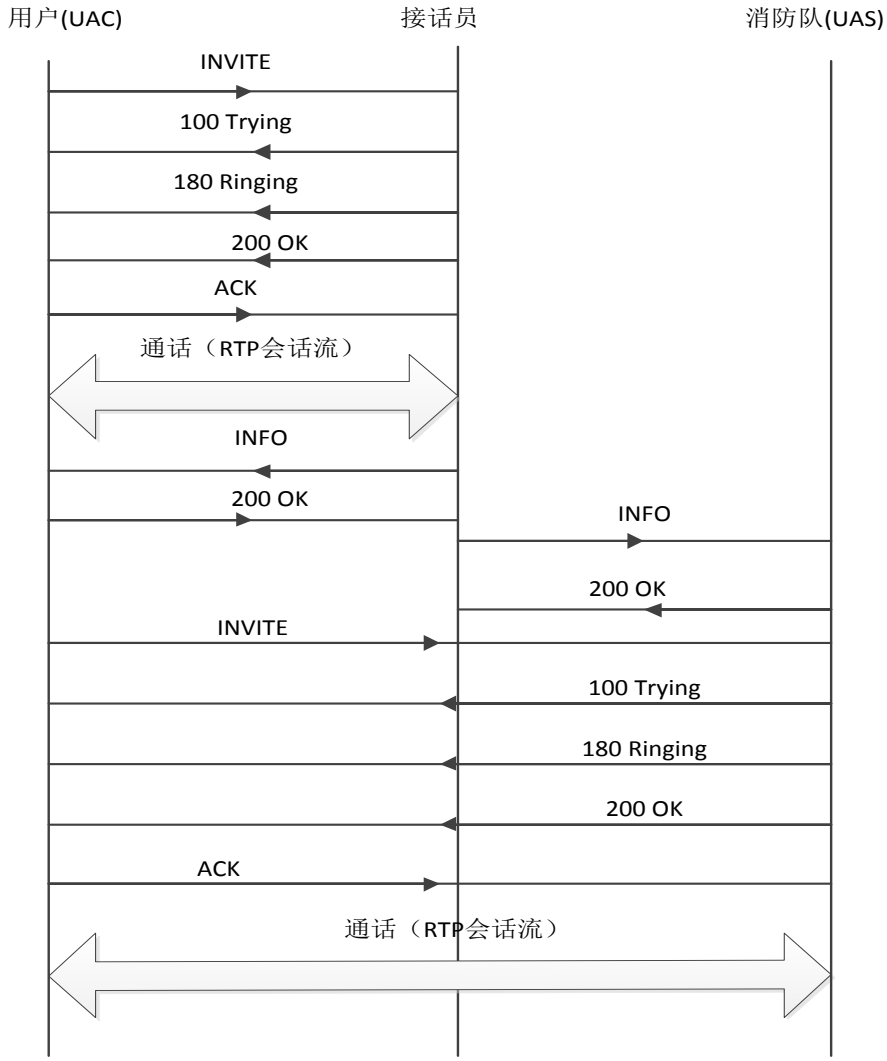


图 5.25 PSAP 内部处理消息流程

（1） 用户发起紧急呼叫 SIP INVITE 消息给紧急呼叫接话员

（2） 紧急呼叫接话员正常响应用户的 INVITE 请求

（3） 当接话员需要把紧急呼叫转接给特定紧急处理方（如图中的消防队）时，接话员发送 SIP INFO 消息把紧急处理方的 SIP URI（例如："Fire Service" < urn:service:sos.fire>）发送给用户，同时接话员发送 SIP INFO 消息把用户的相应紧急呼叫数据发送给紧急处理方

（4） 用户收到紧急处理方的 SIP URI 后，向它发起紧急呼叫 SIP INVITE 请求

（5） 紧急处理方正常响应用户的 INVITE 请求

论文在实现中采用服务器同时模拟接话员和消防队两者。软件测试后，收到两个完整的 SIP 呼叫流程。

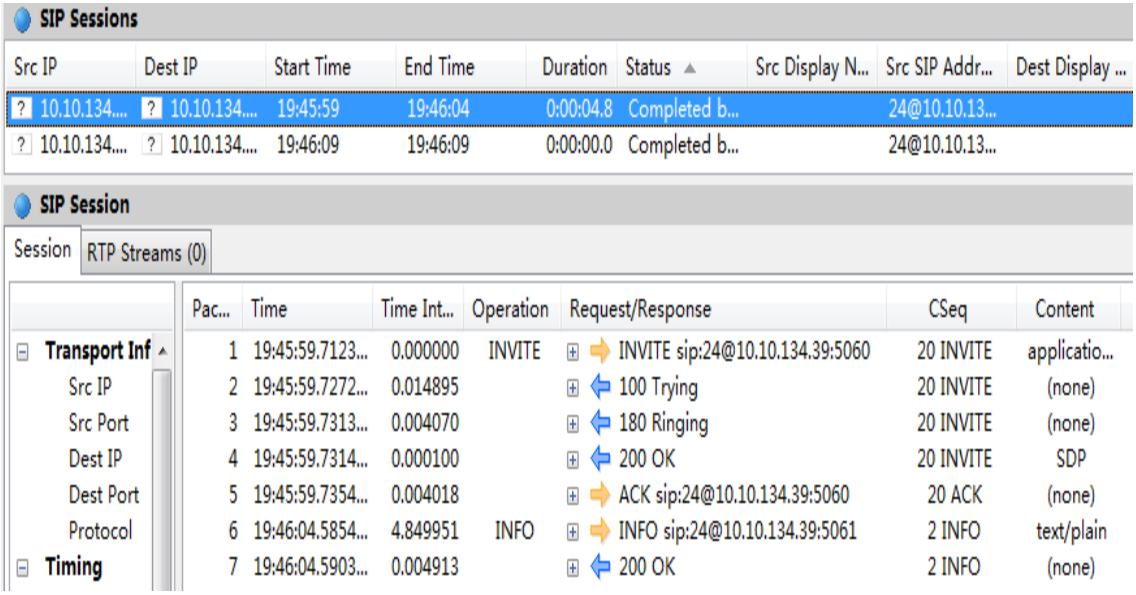


图 5.26 用户与接话员的通信实现

第一个 SIP 流程如图 5.26 所示，是用户与接话员的通信过程，包括 SIP INVITE 和 SIP INFO。

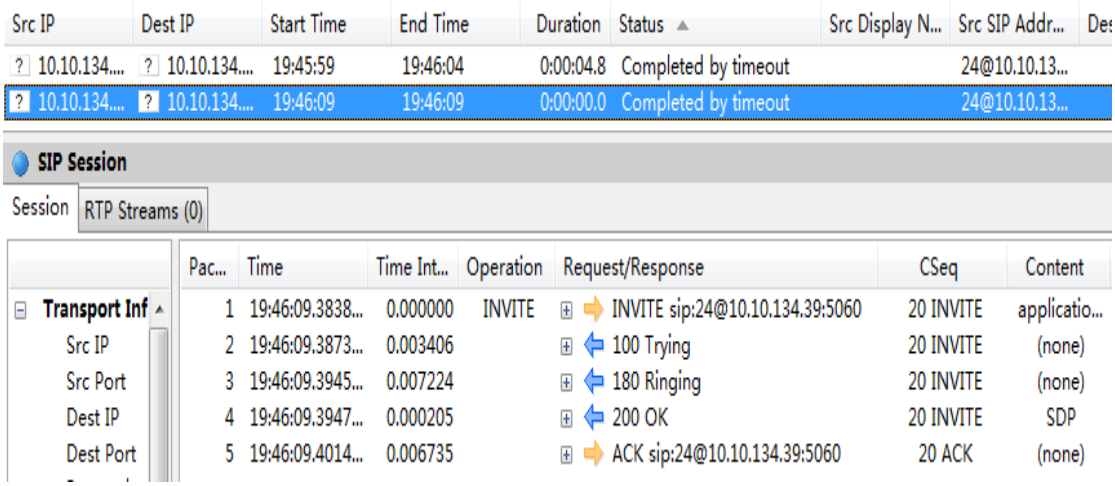


图 5.27 用户与消防队的通信实现

第二个 SIP 流程如图 5.27 所示，是用户与消防队的通信过程，包括 SIP INVITE 流程。

5.5 去活呼叫保持

呼叫保持是指在呼叫进行时终止语音数据的传输，却并没有因此而结束呼叫，即可以继续进行信令的传输，而呼叫保持者可以随时恢复语音数据的传输。呼叫保持采用 REINVITE 请求，即对已经成功建立通话的双方继续发送 INVITE 消息。

呼叫保持的信令流程如下：

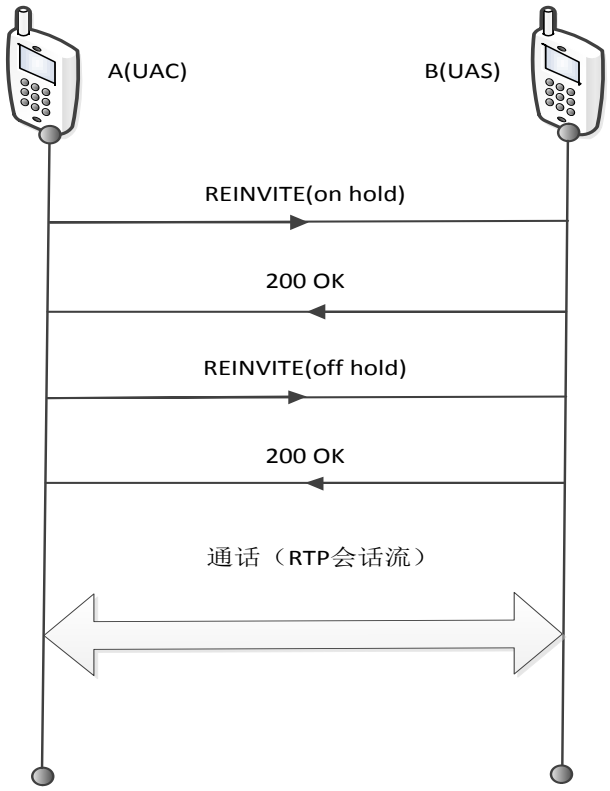


图 5.28 保持呼叫的 SIP REINVITE 流程

如图 5.28 所示，呼叫方 A 与被叫方 B 已经建立了一个呼叫流程，此时如果呼叫双方希望保持这路通话，则发送 REINVITE（on hold）请求，如果呼叫双发希望解除之前的呼叫保持则调用 REINVITE（off hold）请求。

REINVITE（on hold）方法调用以下函数：函数 osip_lock()；函数 osip_on_hold_call()；函数 osip_unlock()。

REINVITE（off hold）方法调用以下函数：函数 osip_lock()；函数 osip_off_hold_call()；函数 osip_unlock()。

在 VoIP 紧急呼叫中，必须屏蔽呼叫保持功能，以防用户无意识的开启此功能。

程序实现方法是，在用户发起紧急呼叫时，检测呼叫功能配置，若用户开启了呼叫保持功能，则程序不再调用 REINVITE（on hold）方法的函数 osip_on_hold_call()或者 REINVITE（off hold）方法的函数 osip_off_hold_call()，而改为调用一个空函数。

5.6 去活呼叫转移

呼叫转移是指当被叫方不能接听电话时，把来话转移到之前预先设定的号码上的业务。在 SIP 协议中，采用 REFER 和 NOTIFY 方法

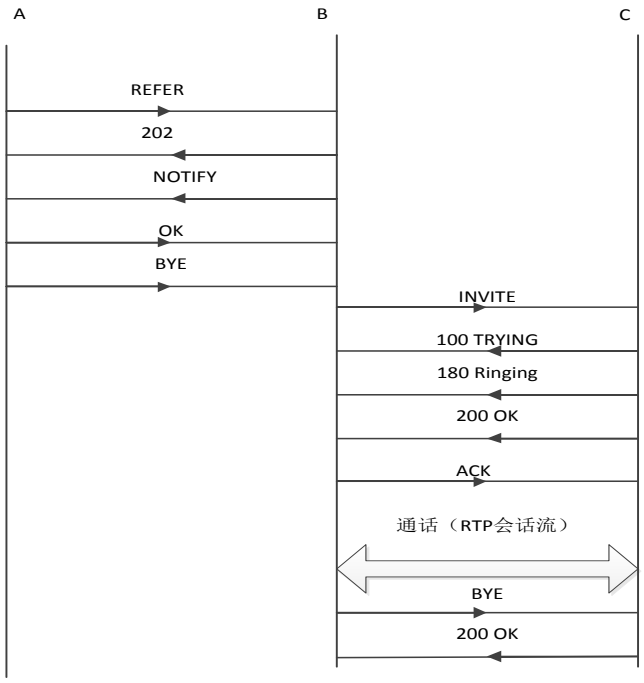


图 5.29 呼叫转移的 SIP 控制流程

如图 5.29 所示，A 与 B 已经建立了一个呼叫流程，此时如果 A 希望把此通话呼叫转移到 C，则发送 REFER 请求，如果同意此请求，则回送 NOTIFY 响应。此后 B 再发送 INVITE 请求给 C，请求建立通话。

REFER 方法调用以下函数：函数 osip_lock()；函数 osip_transfer_call ()；函数 osip_unlock()。

NOTIF 方法调用以下函数：函数 osip_lock()；函数 osip_transfer_send_notify ()；函数 osip_unlock()。

在 VoIP 紧急呼叫中，必须屏蔽呼叫转移功能，以防用户无意识的开启此功能。

程序实现方法是，在用户发起紧急呼叫时，检测呼叫功能配置，若用户开启了呼叫转移功能，则程序不再调用 REFER 方法的函数 osip_transfer_call ()或者 NOTIF 方法的函数 osip_transfer_send_notify ()，而改为调用一个空函数。

5.7 本章小结

本章主要是软件实现了 VoIP 紧急呼叫中的主要流程，包括三大模块位置配置模块、紧急呼叫中心模块和 SIP 模块。位置配置模块主要是通过 HELD 协议来实现位置配置的基本流程，紧急呼叫中心模块主要是通过 LoST 协议来实现呼叫中心发现的基本流程，最后是 SIP 模块，主要软件实现了 SIP 消息的编解码流程，紧急数据呼叫的 SIP 实现和相应的客户端服务器呼叫控制以及设计实现了 PSAP 内部的基本处理流程。另外还去除了在紧急呼叫中的呼叫保持和去活呼叫转移功能。

第六章 总结与展望

相对于传统的 PSTN 紧急呼叫，VoIP 紧急呼叫基于 IP 网络，可移动，便捷，成本低，具有更广泛的通用性与扩展性，改变了以往只能由紧急呼叫方在语音通信过程中单方面告知呼叫中心所在确切位置的情况，提高了紧急呼叫的全面性与广泛性，增强了紧急呼叫的适应能力。

本论文在 ECRIT 工作组对紧急呼叫系统标准化的基础上，给出了 VoIP 紧急呼叫中有关 SIP 呼叫的消息构成和流程，并完成了相关功能的软件实现，并进行了软件测试。

本文主要完成以下主要工作：

（1） 研究基于 VoIP 的紧急呼叫系统架构和关键技术。

（2） 研究 VoIP 的 SIP 协议技术。

（3） 分析了 oSIP 协议栈结构，设计了紧急呼叫 SIP 消息，完成了 SIP 消息的编解码，并给予 C 语言编程实现

（4） 设计了 VoIP 紧急呼叫客户端与服务器，给出了通信流程，同时给予 C 语言编程实现

（5） 分析了只有数据的紧急呼叫与携带补充信息的紧急呼叫，设计了响应 SIP 消息，完成了编解码操作

（6） 设计并实现了 PSAP 内部的基本处理流程

（7） 讨论紧急呼叫中去活呼叫保持和去活呼叫转移功能

（8） C 语言编程实现了紧急呼叫中的位置配置模块，紧急呼叫中心发现模块

（9） 对以上软件结构进行测试，测试结果说明了软件设计和软件实现的可行性。

在未来的无线互联网环境中，紧急呼叫问题的研究会具有重要意义。携带位置信息（数据信息、补充信息）的 SIP 呼叫是其中一个至关重要的功能。论文通过研究，肯定了方案设计的可行性，实现了紧急呼叫的 SIP 呼叫应用。

论文在 VoIP 紧急呼叫方面的下一步研究方向：

（1）进一步分析用户位置的获取方式，并在条件许可的情况下，给予相应实现

（2）进一步分析紧急呼叫中心的发现方式，并在条件许可的情况下，给予相应实现

（3）SIP 呼叫现在还只是给出了最基本呼叫流程的实现方案，以后需要对其他一些复杂呼叫情况给予分析与实现

参考文献

[1] UYLESS B.VOIP：IP语音技术[M]. 机械工业出版社,2000.5.

[2] 深圳市无线电管理局. 合理配置应急通信系统切实提高应急通信的指挥能力.网络,2011.04 http://www.szradio.gov.cn/ztlm/jsjd/200809/t20080902_942319.htm.

[3] 刘文成,杨丰瑞. 浅谈VoIP网络及其安全性分析[J].电信快报.2007.8.

[4] Karpen J, Jean BS, Roe G, McCarthy J.Emergency call handling system[J]. Google Patents.US Patent App. 2005.

[5] Kim JY, Song W, Schulzrinne H.An enhanced VoIP emergency services prototype[J]. ISCRAM, Newark, NJ, 2006.

[6] Gende MF.Emergency call identification, location and routing method and system[J].Google Patents.US Patent App,2006.

[7] Muehleisen TC, Muehleisen ST.A Simplified Second Generation Enhanced Emergency Communications System SSGE-911[J].US Patent App, 2004.

[8] Gende MF.Emergency call identification, location and routing method and system.US Patent App, 2006

[9] Polk J, et al. DHCP Option for Coordinate-based Location Configuration Information[EB/OL].IETF ECRIT. RFC 3825 , July 2004. http://www.ietf.org/rfc/rfc3825.txt.

[10] Peterson J, et al. A Presence-based GEOPRIV Location Object Format[EB/OL].IETF ECRIT.RFC 4119, December 2005. http://www.ietf.org/rfc/rfc4119.txt.

[11] Schulzrinne H, et al. A Uniform Resource Name (URN) for Emergency and Other Well-Known Services[EB/OL]. IETF ECRIT. RFC5031, January 2008. http://www.ietf.org/rfc/rfc5031.txt.

[12] Thomson M, Winterbottom J, et al. PIDF-LO[EB/OL].IETF ECRIT. RFC5139, February 2008. http://www.ietf.org/rfc/rfc5139.txt

[13] Hardie T, et al. LoST: A Location-to-Service Translation Protocol[EB/OL]. IETF ECRIT.RFC5222,August 2008. http://www.ietf.org/rfc/rfc5222.txt.

[14] Schulzrinne H, et al. Discovering LoST Servers using DHCP[EB/OL]. IETF ECRIT.RFC5223,August 2008. http://www.ietf.org/rfc/rfc5223.txt.

[15] Winterbottom J, et al. PIDF-LO Usage Clarification, Considerations, and Recommendations[EB/OL]. IETF ECRIT .RFC5491, March 2009. http://www.ietf.org/rfc/rfc5491.txt.

[16] Schulzrinne H, et al. Location-to-URL Mapping Architecture and Framework[EB/OL]. IETF ECRIT. RFC5582, September 2009. http://www.ietf.org/rfc/rfc5582.txt.

[17] Wolf K, et al. Considerations for Civic Addresses in the PIDF-LO[EB/OL]. IETF ECRIT.RFC5774, March 2010. http://www.ietf.org/rfc/rfc5774.txt.

[18] Barnes M, Ed. , et al. HTTP-Enabled Location Delivery (HELD) [EB/OL]. IETF ECRIT.RFC5985, September 2010. http://www.ietf.org/rfc/rfc5985.txt.

[19] Thomson M, et al. Discovering the Local Location Information Server[EB/OL]. IETF ECRIT.RFC5986, September 2010. http://www.ietf.org/rfc/rfc5986.txt.

[20] Polk J, et al. Dynamic Host Configuration Protocol Options for Coordinate-Based Location Configuration Information [EB/OL]. IETF ECRIT.RFC6225, July 2011. http://www.ietf.org/rfc/rfc6225.txt.

[21] Rosen B, et al.Framework for Emergency Calling using Internet Multimedia [EB/OL]. IETF ECRIT.Internet-Draft, September 8, 2011.http://tools.ietf.org/html/draft-ietf-ecrit-framework-13.

[22] Schulzrinne H, et al. Location Hiding: Problem Statement and Requirements[EB/OL]. IETF ECRIT. Internet-Draft, February 21, 2010. https://tools.ietf.org/html/draft-ietf-ecrit-location-hiding-req.

[23] Rosen B, et al.Best Current Practice for Communications Services in support of Emergency

Calling[EB/OL]. IETF ECRIT. Internet-Draft, September 7, 2011. http://tools.ietf.org/html/draft-ietf-ecrit-phonebcp-20.

[24] 张雪丽. 应急通信标准化和技术热点分析[J].电信网技术. 2007.11

[25] 林密.VoIP市场现状分析报告[J].数字通信世界 VoIP论坛,2006.01

[26] Daniel C, Carrier G.Voice over IP[M].北京:人民邮电出版社，2001

[27] 赖亚寒. 电信下一代网络(NGN)技术及应用探讨[C]. 四川大学硕士论文,2005.4

[28] Voice over IP.From Wikipedia, the free encyclopedia.http://en.wikipedia.org/wiki/Voice_over_IP

[29] ITU-T Recommendation H.323.Packet based multimedia communications systems.1999

[30] 陈燕，龚建荣．SIP 协议的内容及其基本网络结构[J]．广东通信技术，2005(4)

[31] 糜正琨.交换技术[M]. 清华大学出版社发行部,2006.7

[32] 沈波,张顺颐,沈苏彬.会话发起协议SIP的分析与研究[J].数据通信,2001,4

[33] Karl HW.Richard B. VoIP Emergency Calling Foundations and Practice[M]. John Wiley & Sons Ltd.,2011

[34] Matthew M, Anshuman R, Henning S, and Xiaotao Wu. “A VoIP Emergency Services Architecture and Prototype” .Oct. 2005

[35] Wolf K.H and Barnes R. VoIP Emergence Calling[M] .Wiley, 2011

[36] Mintz-Habib M, Rawat A, Schulzrinne H, Wu X. A VoIP emergency services architecture and prototype [J]. Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on , vol., no., pp. 523- 528, 17-19 Oct. 2005

[37] Roach AB.Session Initiation Protocol (SIP)-Specific Event Notification[J]..IETF ECRIT. RFC 3265 , June 2002

[38] Rosenberg J, Schulzrinne H,Columbia U..SIP: Session Initiation Protocol[J]..IETF ECRIT. RFC 3261 , June 2002

[39] The GNU oSIP library.GNU Project.Free Software Foundation (FSF).http://www.gnu.org/software/osip/

[40] The eXtended osip library.Summary [Savannah]. http://savannah.nongnu.org/projects/exosip/

[41] oSIP UAER MANUAL. Aymeric Moizard GNU Free Documention[S],1999.

[42] Richard WS.UNIX网络编程（第2版）第1卷_套接口API和XOpen.传输接口API[M]. 施振川, 周利民, 孙宏晖等. 清华大学出版社.1997,9

致谢

论文撰写工作即将结束，借此机会，我由衷感谢曾经为我的论文提供宝贵意见的所有老师和同学们。

首先我要感谢我的导师糜正琨教授，感谢糜老师这三年研究生阶段的给我在学习和生活上的帮助和指导，这是我一生中最宝贵的学习阶段。糜老师严谨的治学态度、渊博的学术知识、不懈的科研精神深深地感染着我。这将成为我终身受益的宝贵财富，并激励我在今后工作和学习的路上勤勤恳恳不断努力。

同时感谢曹申会、陈杰、奚晓华、梁倩、荣涛及 405 教研室的其他同学们，他们在项目开发和撰写论文方面给我提出了许多意见和建议，并陪我一起度过了这段美好的学习时光。

最后感谢在百忙之中审阅我的论文并参加我的论文答辩的各位专家和老师。